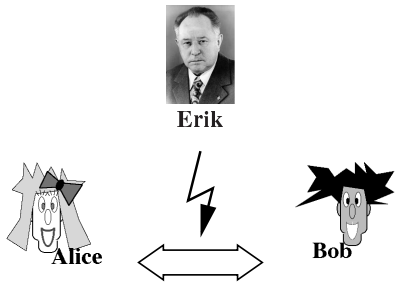
3. Kriptográfia (Jörg Rothe)

Ebben a fejezetben a kriptográfiában használatos protokollokat, valamint alapvető problémákat és algoritmusokat mutatunk be. A kriptográfiában jellemző egyik alaphelyzetet láthatjuk a 3.1. ábrán. Aliz és Bob híreket cserélnek valamilyen nem biztonságos csatornán, például nyilvános telefonvonalon, vagy két hálózati kapcsolatban lévő számítógép közötti elektronikus posta segítségével. Erik megcsapolja a vezetéket, és lehallgatja a kommunikációt. Mivel Aliz és Bob tudják ezt, a küldeményeiket rejtjelezik.



3.1. ábra. A kriptográfiában tipikus alaphelyzet.

A 3.1. alfejezetben olyan szimmetrikus kriptográfiai rendszereket mutatunk be, amelyek az illetéktelen rejtjelfejtést hivatottak akadályozni. Szimmetrikusnak akkor nevezünk egy kriptográfiai rendszert, ha ugyanaz a kulcs használható rejtjelezésre (sifrírozásra) és visszafejtésre (desifrírozásra) is. Vajon hogyan kell Aliznak és Bobnak egy közös kulcsban megegyeznie, ha csupán egy megbízhatatlan csatornán tudnak kommunikálni egymással? Ha például Aliz választ egy kulcsot és rejtjelezi (mondjuk egy szimmetrikus rejtjelrendszerrel), majd elküldi Bobnak, azonnal felmerül a kérdés, milyen kulcsot használjanak ezeknek a kulcsoknak a rejtjelezéséhez.

Ez a paradoxnak tűnő kérdés – amely egy kicsit emlékeztet arra a másikra, hogy vajon a tojás, vagy a tyúk volt-e előbb – ***kulcscsere-problémaként*** ismert. A kriptográfia kezdetétől megoldhatatlannak tartották ezt a problémát. Annál nagyobb volt a meglepetés, amikor Whitfield Diffie és Martin Hellman 1976-ban megoldották. Egy protokollt javasoltak, amelyben Aliz és Bob információkat cserélnek, s ennek segítségével végül ki tudják számítani a közös kulcsukat. Az őket lehallgató Eriknek ugyanakkor fogalma sincs a kulcsukról, még akkor sem, ha az adatcserében résztvevő minden egyes bitet el tudja fogni. A 3.2. alfejezet bemutatja a Diffie–Hellman-protokollt.

A történet némileg ironikus része az, hogy éppen ez a protokoll, amely a szimmetrikus kriptográfiában oly fontos kulcscsere megoldhatatlannak tartott problémáját megoldotta, egy másik protokoll felfedezése előtt egyengette az utat, s ez utóbbiban a megbízhatatlan csatornán történő titkos kulcscsere már semmilyen szerepet nem játszik. Diffie és Hellman 1976-ban megjelent munkájukkal kaput nyitottak a modern ***nyilvános kulcsú kriptográfiának,*** és már két évvel később, 1978-ban, Rivest, Shamir és Adleman szélesre tárták ezt a kaput, amikor megalkották jól ismert RSA-rendszerükkel az első ***nyilvános kulcsú kriptorendszert.*** A 3.3. alfejezetben szerepel az RSA-rendszer, valamint egy, az RSA-n alapuló digitális aláírásra vonatkozó protokoll. Egy ilyen protokollal Aliz el tudja látni kézjegyével Bobnak küldött üzeneteit úgy, hogy Bob az üzenet küldőjének azonosságát ellenőrizni tudja. A digitális aláírások célja annak megakadályozása, hogy Erik Aliz üzenetét meghamisítva úgy tegyen, mintha azt Aliz küldte volna.

A Diffie–Hellman-protokoll biztonsága azon a feltételezésen alapszik, hogy a diszkrét logaritmus nem számítható ki hatékonyan. A moduláris hatványozás – amelynek inverz függvénye a diszkrét logaritmus – egy lehetséges egyirányú függvény. Az RSA biztonsága is egy probléma feltételezett nehézségén nyugszik, nevezetesen azon a feltételezésen, hogy nagy számok nem bonthatók hatékonyan prímtényezők szorzatára. A jogos címzett – Bob – ugyanakkor képes hatékonyan visszafejteni a rejtjelezett üzenetet, miközben egy általa választott szám faktorizációját saját „csapóajtó” információjaként használja fel.

A 3.4. alfejezetben bemutatunk egy Rivest–Sherman-protokollt, amelyik az úgynevezett erősen nem invertálható, asszociatív egyirányú függvényeken nyugszik, és a Diffie–Hellman-protokollhoz hasonlóan titkos kulcscserére szolgál. Ez a protokoll is módosítható oly módon, hogy digitális aláírás számára használható legyen.

Végül a 3.5. alfejezet az interaktív bizonyításrendszerek és a zéró-ismeretű protokollok érdekes területére vezet, amely a kriptográfiában gyakorlati felhasználásra kerül, nevezetesen a hitelesség problémájának megoldása során. Bemutatunk egy zéró-ismeretű protokollt a gráfizomorfizmus problémára. Másrészt ez a terület a bonyolultságelmélethez is kapcsolódik, és ezért a 4. fejezetben újból előkerül, ismét a gráfizomorfizmus problémával kapcsolatban.

3.1. Alapok

A ***kriptográfia*** évezredek óta létező művészet és tudomány, írások és üzenetek titkos anyagba történő olyan rejtjelezésével foglalkozik, amely illetéktelen személyek számára akadályozza a visszafejtést. Ebben az alfejezetben két klasszikus szimmetrikus kriptorendszert mutatunk be, a következő alfejezetekben pedig a manapság használatos legfontosabb protokollok és aszimmetrikus kriptorendszerek közül foglalkozunk néhánnyal. Protokollon két vagy több résztvevő közötti párbeszédet értünk, miközben egy résztvevő lehet egy ember, de lehet egy számítógép is. A kriptográfiai protokollok szövegek rejtjelezésére szolgálnak, oly módon, hogy a jogosult fogadó képes legyen egyszerűen és hatékonyan visszafejteni a rejtett szöveget. Egy protokoll algoritmusként is felfogható, melyet több résztvevő hajt végre.

A ***kriptoanalízis*** szintén több évezredes művészet és tudomány is, amely rejtjelezett üzenetek (illetéktelen) visszafejtésével és a létező kriptorendszerek feltörésével foglalkozik. A ***kriptológia*** ezt a két területet öleli fel, a kriptográfiát és a kriptoanalízist. Ebben a fejezetben főként olyan ***kriptografikus*** algoritmusokra koncentrálunk, amelyek lehetővé

teszik a biztonságos rejtjelezést. A kriptoanalízis algoritmusait – amelyekkel az ember próbálkozik, hogy kriptografikus protokollokat és rendszereket feltörjön –, megemlítjük, de nem vizsgáljuk teljes részletességgel.

3.1.1. Kriptográfia

A kriptográfia egyik jellemző alaphelyzete a 3.1. ábrán látható: Aliz és Bob olyan nem biztonságos csatornán kommunikálnak egymással, amelyet Erik lehallgat, és ezért az üzenetüket rejtjelezik egy kriptorendszer segítségével.

3.1. definíció (kriptorendszer). *A **kriptorendszer** egy $(\mathcal{P}, C, \mathcal{K}, \mathcal{E}, \mathcal{D})$ ötös a következő tulajdonságokkal:*

1. *$\mathcal{P}$, C és $\mathcal{K}$ véges halmazok, $\mathcal{P}$ a **nyílt szöveg tér**, C a **rejtett szöveg tér** és $\mathcal{K}$ a **kulcstér**. $\mathcal{P}$ elemeit **nyílt szövegnek** C elemeit pedig **rejtett szövegnek** nevezzük. Egy **üzenet** a nyílt szöveg szimbólumaiból álló szó.*
2. *$\mathcal{E} = \{E_k \mid k \in \mathcal{K}\}$ azoknak az $E_k : \mathcal{P} \to C$ függvényeknek a családja, amelyeket a rejtjelezéshez használunk. $\mathcal{D} = \{D_k \mid k \in \mathcal{K}\}$ azoknak a $D_k : C \to \mathcal{P}$ függvényeknek a családja, amelyeket a visszafejtéshez használunk.*
3. *Mindegyik $e \in \mathcal{K}$ kulcshoz van egy $d \in \mathcal{K}$ kulcs, melyekre minden $p \in \mathcal{P}$ nyílt szöveg esetén*

$$D_d(E_e(p)) = p \ . \tag{3.1}$$

*Egy kriptorendszerre azt mondjuk, hogy **szimmetrikus** (vagy titkos kulcsú), ha $d = e$ vagy legalábbis d „könnyen" kiszámítható e-ből. Egy **kriptorendszerre** azt mondjuk, hogy **aszimmetrikus** (vagy nyilvános kulcsú, „public-key"), ha $d \neq e$, és „gyakorlatilag nem végezhető el" az a feladat, hogy a d kulcsot e-ből kiszámítjuk. Ekkor e-t **nyilvános kulcsnak** nevezzük, és d az **e-hez tartozó titkos kulcs**.*

Néha különböző kulcstereket használnak a rejtjelezéshez és a visszafejtéshez, ami a fenti definíció megfelelő módosítását vonja maga után.

A klasszikus kriptorendszerek bemutatására példaként legyen a $\Sigma = \{A, B, \ldots, Z\}$ ábécé a nyílt szöveg tér és ugyanakkor a rejtett szöveg tér is. Abból a célból, hogy a betűkkel úgy tudjunk számolni, mintha számok lennének, feleltessük meg Σ-nak $\mathbb{Z}_{26} = \{0, 1, \ldots, 25\}$-öt. A 0 szám tehát az A betűnek felel meg, az 1 a B betűnek stb. A nyílt szöveg szimbólumainak ez a természetes számokkal való kódolása természetesen nem tartozik hozzá a tényleges rejtjelezéshez, illetve visszafejtéshez.

Az üzenetek Σ^* elemei, ahol Σ^* a Σ fölötti szavak halmazát jelöli. Ha valamely $m \in \Sigma^*$ üzenet n hosszúságú blokkokra van felosztva, és blokkonként van rejtjelezve – ahogyan az sok kriptorendszerben szokásos –, akkor m egyes blokkjait $\mathbb{Z}_{26}^n$ elemeiként értelmezhetjük.

3.1. példa. Eltolásos rejtjelező. Az első példa egy monoalfabetikus szimmetrikus kriptorendszer. Legyen $\mathcal{K} = \mathcal{P} = C = \mathbb{Z}_{26}$. Az ***eltolásos rejtjel*** az a rejtett üzenet, amelyben a nyílt szöveg minden jelének az ábécében a k számú betűvel eltolt jel felel meg (modulo 26), ahol k rögzített szám. Az itt szereplő $k \in \mathbb{Z}_{26}$ a kulcs. Ha a rejtett szöveg minden jelét ugyanennek a k kulcsnak a felhasználásával visszatoljuk, akkor feltárul előttünk a nyílt szöveg. Az E_k rejtjelező függvényt és a D_k visszafejtő

m	B U D A P E S T K E L E T P Á R I Z S A
c	E X G D S H V W N H O H W S D U L C V D

3.2. ábra. Példa a Caesar-rejtjelezővel történő sifrírozásra.

függvényt mindegyik $k \in \mathbb{Z}_{26}$ kulcs esetén az alábbi módon definiáljuk:

$$\begin{aligned} E_k(m) &= (m+k) \bmod 26 \\ D_k(c) &= (c-k) \bmod 26\,, \end{aligned}$$

ahol a k-val való összeadást és kivonást jelenként, modulo 26 kell elvégezni.

A 3.2. ábrán látható egy magyar m üzenetnek a c kóddal történő rejtjelezése $k = 3$ esetén.[1] Az eltolással való rejtjelezés ezzel a speciális $k = 3$ kulccsal ***Caesar-rejtjelező*** néven is ismeretes, mert feltehetően a római császár ezt használta háborúiban a katonai üzenetek titkosítására.[2] Ez egy nagyon egyszerű helyettesítéses rejtjelező, amelyikben a nyílt szöveg minden betűjét a rejtjel ábécé egy meghatározott betűjével helyettesítenek.

Mivel a kulcstér nagyon kicsi, az eltolásos rejtjelet igen könnyű feltörni.

Ez a rejtjel már arra a támadásra is érzékeny, amelynek során a támadó csupán a rejtett szöveget ismeri (***rejtett szövegű támadás.*** Ha egyszerűen a 26 lehetséges kulcs mindegyikét végigpróbáljuk, kiderül, hogy melyik kulcs esetén adódik értelmes nyílt szöveg, ha pedig a rejtett szöveg elég hosszú, csak egy megfelelő rejtjelezés adódik.

A Caesar-rejtjelező ***monoalfabetikus kriptorendszer***, mert a nyílt szöveg minden betűjének a rejtett szövegben mindig ugyanaz a betű felel meg. Ezzel szemben egy ***polialfabetikus kriptorendszerben*** lehetséges, hogy ugyanannak a nyílt szövegbeli szimbólumnak különböző rejtett szövegbeli szimbólumok felelnek meg, attól függően, hogy a szöveg melyik helyén állnak. Egy ilyen polialfabetikus kriptorendszert, amelyik az eltolásos rejtjelezésre épül, de jóval nehezebb feltörni, Blaise de Vigenère (1523–1596) francia diplomata javasolt. Rendszere Leon Battista Alberti (szül. 1404) itáliai matematikus, Johannes Trithemius (szül. 1492) német apát és Giovanni Porta (szül. 1535) itáliai természettudós előmunkáira épül. Ez a rejtjelező úgy működik, mint az eltolásos rejtjelező, azzal a különbséggel, hogy a betűk, amelyekkel a nyílt szöveg egy szimbóluma sifrírozva van, most még a szövegbeli helyüknek megfelelően is változnak.

3.2. példa. *Vigenère-rejtjelező.* Ez a szimmetrikus polialfabetikus kriptorendszer egy úgynevezett ***Vigenère-négyzetet*** használ (lásd a 3.3. ábrát). Ez egy 26 sorból és 26 oszlopból álló mátrix. Mindegyik sorban az ábécé 26 betűje szerepel, sorról sorra mindig egy pozícióval balra tolva. Ez azt jelenti, hogy az egyes sorok felfoghatók egy-egy eltolásos rejtjelezőként, ahol a kulcsok sorban $0, 1, \ldots, 25$. A szimbólum szövegbeli helyétől függ az, hogy a Vigenère-négyzet melyik sorát használja az ember egy nyílt szövegbeli szimbólum rejtésére.

Az üzeneteket n állandó hosszúságú blokkra osztjuk és blokkonként rejtjelezzük, vagyis $\mathcal{K} = \mathcal{P} = \mathcal{C} = \mathbb{Z}_{26}^n$. Az n blokkhosszat a rendszer ***periódusának*** is nevezik. Egy w szóban az i-edik betűt w_i-vel jelöljük.

---

[1] Az ékezetes betűk kódja ugyanaz, mint a megfelelő ékezet nélkülieké. *A fordító.*

[2] Történelmi megjegyzés: Gaius Julius Caesar *De Bello Gallico* című művében tudósít arról, hogy a gall háborúban (58–50 Kr.e.) hogyan küldött egy rejtjelezett üzenetet Q. Tullius Ciceronak (a híres szónok testvéröccsének), aki légiójával ostrom alatt volt. Az alkalmazott rendszer monoalfabetikus volt, és a latin betűket göröggel helyettesítette, Caesar írásaiból nem derül ki azonban az, hogy valóban a $k = 3$ kulcsú eltolásos rejtjelezőről volt-e szó. Ezt az információt később Suetonius adta.

0	A	B	C	D	E	F	G	**H**	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
1	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
2	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
3	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
4	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
5	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
6	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
7	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
8	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
9	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
10	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
11	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
12	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
13	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
14	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
15	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
16	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
17	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
18	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
19	**T**	U	V	W	X	Y	Z	**A**	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
20	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
21	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
22	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
23	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
24	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
25	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

3.3. ábra. Vigenère-négyzet: a „H” nyílt szöveget a „T” kulccsal rejtjelezve „A”-t kapunk.

Az E_k rejtjelező függvényt és a D_k visszafejtő függvényt, amelyek mindketten $\mathbb{Z}_{26}^n$-ből $\mathbb{Z}_{26}^n$-be képeznek, mindegyik $k \in \mathbb{Z}_{26}^n$ kulcs esetén a következőképpen definiáljuk:

$$\begin{aligned} E_k(b) &= (b + k) \bmod 26 \\ D_k(c) &= (c - k) \bmod 26\,, \end{aligned}$$

ahol a k-val való összeadást és kivonást megint jelenként kell elvégezni modulo 26. Ez azt jelenti, hogy a megbeszélt $k \in \mathbb{Z}_{26}^n$ kulcsot betűről betűre a $b \in \mathbb{Z}_{26}^n$ blokk szimbólumai fölé írjuk. Ha a nyílt szöveg utolsó blokkjának kevesebb szimbóluma van, mint n, akkor a kulcsnak csak annyi jelét használjuk fel, amennyi szükséges. Amikor a nyílt szöveg i-edik b_i szimbólumát rejtjelezzük, fölötte a k_i kulcsszimbólum áll, és a Vigenère-négyzet k_i-edik sorát használjuk eltolásos rejtjelezőként.

Válasszuk például az $n = 4$ blokkhosszat, és a k = TONY kulcsot. A 3.4. ábra egy angol nyelvű m nyílt szöveg rejtjelezését mutatja, amelyik hét blokkból áll, és a c rejtett szövegben a Vigenère-rejtjelezőt alkalmaztuk a k kulccsal. A nyílt szöveg első betűjéhez, a „H”-hoz a „T” kulcsszimbólumot rendeljük hozzá. A Vigenère-négyzet „H”-oszlopának és a “„T”-sorának kereszteződésében „A” található, ez lesz a rejtett szöveg első szimbóluma, amint azt a 3.3. ábrán láthatjuk.

k	T O N Y T O N Y T O N Y T O N Y T O N Y T O N Y
m	H U N G A R I A N I S A L L G R E E K T O G E R M A N S
c	A I A E T F V Y G W F Y E Z T P X S X R H U R P F O A Q

3.4. ábra. Példa a Vigenère-rejtjelezőt használó sifrírozásra.

Még számos további klasszikus kriptorendszer létezik, amelyeket azonban itt nem vizsgálunk meg közelebbről. Több lehetőség van arra is, hogy a kriptorendszereket jellegük és tulajdonságaik alapján osztályozzuk. A 3.1. definíció megvilágítja a ***szimmetrikus*** és az ***aszimmetrikus*** kriptorendszerek közötti különbséget. A két bemutatott szimmetrikus rendszer (az eltolásos rejtjelező és a Vigenère-rejtjelező) rámutat a ***monoalfabetikus*** és a ***polialfabetikus*** rendszerek közötti eltérésekre. Mindkettő ***helyettesítéses rejtjelező***. Ezek szembeállíthatók a ***permutációs rejtjelezőkkel*** (más néven ***transzpozíciós rejtjelezőkkel***), amelyeknél a nyílt szöveg betűit nem a rejtjelábécé bizonyos betűi helyettesítenek, hanem csupán a szövegbeli helyük változik, egyébként azonban nem változnak meg.

Ha blokkonként n periódussal rejtjelezünk és Σ^n összes permutációinak halmazát alkalmazzuk kulcstérként, ahol Σ m betűs ábécé, akkor $m^n!$ különböző lehetőségünk van arra, hogy egy kulcsot kiválasszunk. Továbbá a ***blokkrejtjelezőt*** – amelyben, mint a Vigenère-rendszer esetén a nyílt szöveget blokkokra osztjuk, blokkonként rejtjelezzük –, egybevethetjük a folyamrejtjelekkel, amelyek egy folyamatos kulcsfolyamot hoznak létre. A blokkrejtjelezők különböző változatait vizsgálhatjuk. Egy fontos típust képviselnek az ***affin blokkrejtjelezők.*** Ezeket a következőképpen definiáljuk: az $E_{(A,\vec{b})}$ rejtjelező függvények, és a $D_{(A^{-1},\vec{b})}$ visszafejtő függvények $\mathbb{Z}_m^n$-ből $\mathbb{Z}_m^n$-be képeznek, és affin leképezések, ami azt jelenti, hogy az alábbi alakúak:

$$\begin{aligned} E_{(A,\vec{b})}(\vec{x}) &= A\vec{x}+\vec{b} \bmod m\,, \\ D_{(A^{-1},\vec{b})}(\vec{y}) &= A^{-1}(\vec{y}-\vec{b}) \bmod m\,. \end{aligned} \tag{3.2}$$

Itt $(A,\vec{b})$ és $(A^{-1},\vec{b})$ a rejtjelezés illetve visszafejtés kulcsai; A $(n\times n)$-es mátrix $\mathbb{Z}_m$-beli elemekkel. A^{-1} az A inverz mátrixa; $\vec{x}$, $\vec{y}$ és $\vec{b}$ mind $\mathbb{Z}_m^n$-beli vektorok, a műveleteket pedig modulo m végezzük. Nézzünk néhány matematikai kiegészítést (lásd a 3.1.3. pontban a 3.2. definíciót is): egy – a $\mathbb{Z}_m$ gyűrű fölötti – $(n\times n)$-es A mátrixnak pontosan akkor van multiplikatív inverze, ha $\mathrm{lnko}(\det A, m)=1$. A ***inverz mátrixát*** $A^{-1}=(\det A)^{-1}A_{\mathtt{adj}}$ segítségével definiáljuk, ahol $\det A$ az A determinánsa és $A_{\mathtt{adj}}=((-1)^{i+j}\det A_{j,i})$ az A ***adjungált mátrixa***. A $\det A$ ***determinánsát*** rekurzív módon definiáljuk: $n=1$ és $A=(a)$ esetén $\det A=a$; $n>1$ esetén minden i-re, ahol $i\in\{1,2,\ldots,n\}$, $\det A=\sum_{j=1}^{n}(-1)^{i+j}a_{i,j}\det A_{i,j}$, ahol $a_{i,j}$ az A-nak az (i,j) indexű eleme és az $(n-1)\times(n-1)$-es $A_{i,j}$ mátrixokat az A-ból úgy kapjuk, hogy az i-edik sort és a j-edik oszlopot töröljük. Egy mátrix determinánsát hatékonyan ki lehet számítani (lásd a 3-3. feladatot).

Példaként megemlítjük, hogy a Vigenère-rejtjelező affin rejtjelező, amelynek kulcstere, $\mathcal{K}=\mathbb{Z}_m^n$ pontosan m^n elemű (lásd a 3.2. példát). Ha (3.2)-ben a $\vec{b}$ nullvektor, akkor ***lineáris blokkrejtjelezőről*** van szó. Erre klasszikus példa a ***Hill-rejtjelező,*** amit 1929-ben Lester Hill talált ki. Ennél a rejtjelezőnél a kulcstér az összes olyan $(n\times n)$-es $\mathbb{Z}_m$-beli elemeket tartalmazó A mátrix, melyre $\mathrm{lnko}(\det A, m)=1$. Emiatt a kulcsként megengedett mátrixok invertálhatók, s az A^{-1} inverz mátrixszal fejthetjük vissza az A mátrixszal rejtjelezett üzenetet. A Hill-rejtjelezőt az $E_A(\vec{x})=A\vec{x} \bmod m$ rejtjelező függvénnyel és a $D_{A^{-1}}(\vec{y})=A\vec{y} \bmod m$

visszafejtő függvénnyel definiáljuk. Ez a legáltalánosabb lineáris rejtjelező. A permutációs rejtjelezők szintén lineáris rejtjelezők, és így a Hill-rejtjelezők speciális esetei.

3.1.2. Kriptoanalízis

A kriptoanalízis feladata az, hogy adott kriptorendszereket feltörjön, elsősorban azért, hogy a szükséges kulcsot a visszafejtéshez megállapítsa. Attól függően, hogy milyen információk állnak a kriptoanalízis rendelkezésére, a támadások több típusát lehet megkülönböztetni, és ezzel a vizsgált kriptorendszer biztonságát, illetve sérülékenységét jellemezni. Az eltolásos rejtjelezővel kapcsolatban már említettük a *rejtett szövegű támadást*. Ez a leggyengébb formája a támadásnak, és az a kriptorendszer, amelyik ennek a támadásnak sem áll ellen, nem sokat ér.

Az affin blokkrejtjelezők, mint például a Vigenère- és a Hill-rejtjelezők érzékenyek az olyan támadásokra, amelyeknél a támadó egy elcsípett rejtjelezett szöveget és a hozzátartozó nyílt szöveget is ismeri ***(ismert nyílt szövegű támadás),*** és ebből az alkalmazott kulcsra következtetni tud. Még inkább érzékenyek az olyan támadásokra, amelyeknél a támadó saját maga ki tud választani egy nyílt szöveget, és utána látja, hogy milyen rejtett szöveg tartozik hozzá ***(választott nyílt szövegű támadás.*** A negyedik fajta támadás főként az aszimmetrikus kriptorendszerekre hatásos. Az ilyen ***rejtjelező kulcs támadás***, „encryption-key-attacks" esetén a támadó csupán a nyilvánosságra hozott kulcsot ismeri, de nem ismer rejtett üzenetet, és egyedül ebből az információból próbálja meghatározni a titkos kulcsot. A különbség az, hogy a támadónak van ideje számításokat végezni, a többi támadás esetén azonban sietnie kell, mert az üzenetet már elküldték. Ezért kell aszimmetrikus kriptorendszerek esetén a kulcsteret nagynak választani, s ezzel szavatolni a rendszer biztonságát. Ennek következménye az, hogy a gyakorlatban sokszor kevéssé hatékonyak az aszimmetrikus rendszerek.

Az ilyen támadások gyakran a rejtjelezett szövegben előforduló betűk gyakoriságát vizsgálják. Emellett a nyílt szöveg számára alkalmazott természetes nyelv redundanciáját is kihasználják. Például sok természetes nyelvben az „E" betű statisztikusan szignifikáns módon a leggyakrabban fordul elő. Hosszú „tipikus" szövegek vizsgálata szerint az „E" gyakorisága az angolban 12.31%, a franciában 15.87%, a németben pedig 18.46%. Más nyelvekben más betűk léphetnek fel a legnagyobb gyakorisággal. „Tipikus" finn szövegekben például 12.06%-kal az „A" a leggyakoribb betű.

A gyakoriságelemzés szemmel láthatóan hasznos a monoalfabetikus kriptorendszerek elleni támadás során. Ha például egy eltolásos rejtjelezővel rejtett német szövegben az „Y" betű lép fel leggyakrabban, ami a németben – akárcsak a legtöbb nyelvben – ritka, akkor ebből arra következtethetünk, hogy ez az „E" rejtett változata, az alkalmazott kulcs pedig az „U" ($k = 20$) (lásd a 3.3. ábrát). Az egyes betűk gyakorisága mellett vizsgálhatjuk a betűpárok (digrammok), a betűhármasok (trigrammok) stb. A támadásnak ez a módja működik a polialfabetikus kriptorendszerek esetében is, feltéve, hogy a periódus (vagyis a blokkhossz) ismert.

Az ismeretlen periódusú polialfabetikus kriptorendszerek ezzel szemben nagyobb biztonságot nyújtanak. A Vigenère-rejtjelező például hosszú ideig ellenállt minden feltörési kísérletnek. Csak 1863-ban, mintegy 300 évvel a feltalálása után, talált módszert Friedrich Wilhelm Kasiski német kriproanalitikus a Vigenère-rejtjelező feltörésére. Megmutatta, hogyan lehet meghatározni az alkalmazott periódust a kulcsszöveg szavainak ismétlődéséből akkor is, ha a periódus kezdetben ismeretlen volt. Végül a gyakoriságelemzés segítségével a

rejtjelezett szöveg visszafejthető. Singh írja, hogy a rendkívül sokoldalú Charles Babbage, akit sokan kora zsenijének tartanak, a Kasiski-féle módszert valószínűleg korábban,1854-ben felfedezte, bár nem hozta nyilvánosságra.

A kriptográfia történetében mérföldkőként kell megemlítenünk Claude Shannonnak (1916–2001). a modern kód- és információelmélet atyjának úttörő munkáját. Shannon bebizonyította, hogy vannak kriptorendszerek, amelyek egy bizonyos szigorú matematikai értelemben ***tökéletes titkosságot*** tesznek lehetővé. Pontosabban szólva egy kriptorendszer akkor tökéletes titkosságú, ha $|\mathcal{C}| = |\mathcal{K}|$, a kulcsok $\mathcal{K}$-ban egyenletes eloszlásúak, és minden $p \in \mathcal{P}$-re és minden $c \in \mathcal{C}$-re ***pontosan egy*** $k \in \mathcal{K}$ kulcs van, amelyre $E_k(p) = c$. Ez azt jelenti, hogy egy ilyen kriptorendszer a legtöbb gyakorlati célra nem használható, mert ahhoz, hogy tökéletes titkosságot garantálhassunk, egyrészt minden kulcsnak legalább olyan hosszúnak kell lennie, mint a rejtjelezendő üzenet, másrészt egyszeri használat után minden kulcsot el kell dobni. Gyakorlati célra megfelelő kriptorendszereket később fogunk ebben a fejezetben bemutatni.

3.1.3. Algebra, számelmélet és gráfelmélet

A később sorra kerülő algoritmusok és problémák közül néhánynak a megértéséhez segítségünkre lesz az algebra, és főként a csoport- és számelmélet köréből néhány alapvető fogalom és tétel. Vonatkozik ez mind a 3. fejezetbeli kriptorendszerekre és zéró-ismeretű protokollokra, mind pedig néhány, a 4. fejezetben vizsgált problémára. Az Olvasó megteheti, hogy ezt a pontot átugorja, és a szükséges fogalmaknak, eredményeknek csak akkor néz utána, ha később felbukkannak. A bizonyításoktól ebben a fejezetben többnyire eltekintünk.

3.2. definíció (csoport, gyűrű, test).

- $\mathfrak{G} = (S, \circ)$ ***csoport****, ha S nem üres halmaz, $\circ$ kétváltozós művelet S-en és teljesülnek a következő axiómák:*
  - *zártság:* $(\forall x \in S)\,(\forall y \in S)\,[x \circ y \in S]$;
  - *asszociativitás:* $(\forall x \in S)\,(\forall y \in S)\,(\forall z \in S)\,[(x \circ y) \circ z = x \circ (y \circ z)]$;
  - *semleges elem:* $(\exists e \in S)\,(\forall x \in S)\,[e \circ x = x \circ e = x]$;
  - *inverz elem:* $(\forall x \in S)\,(\exists x^{-1} \in S)\,[x \circ x^{-1} = x^{-1} \circ x = e]$.

  Az e elemet a $\mathfrak{G}$ csoport ***semleges elemének,*** *az x^{-1} elemet x* ***inverzének*** *nevezzük. $\mathfrak{G}$* ***félcsoport****, ha $\mathfrak{G}$-ben az asszociativitás és a zártság teljesül a $\circ$ műveletre, akkor is, ha $\mathfrak{G}$-nek nincs semleges eleme, vagy nem minden elemnek van inverze. Egy $\mathfrak{G} = (S, \circ)$ félcsoport, vagy csoport* ***kommutatív,*** *ha $x \circ y = y \circ x$ minden $x, y \in S$ esetén teljesül. Egy $\mathfrak{G}$ véges csoport elemeinek a száma a $\mathfrak{G}$* ***rendje*** *és ezt $|\mathfrak{G}|$ jelöli.*
- $\mathfrak{H} = (T, \circ)$*-t a $\mathfrak{G} = (S, \circ)$* ***csoport részcsoportjának*** *nevezzük ($\mathfrak{H} \leq \mathfrak{G}$ jelöli), ha $T \subseteq S$ és $\mathfrak{H}$ kielégíti a csoportaxiómákat.*
- *Egy $\mathfrak{R} = (S, +, \cdot)$ hármas* ***gyűrű****, ha $(S, +)$ Abel-csoport, $(S, \cdot)$ félcsoport és a disztributivitási szabály érvényben van:*

  $$(\forall x \in S)\,(\forall y \in S)\,(\forall z \in S)\,[(x \cdot (y + z) = (x \cdot y) + (x \cdot z)) \wedge ((x + y) \cdot z = (x \cdot z) + (y \cdot z))].$$

  Egy $\mathfrak{R} = (S, +, \cdot)$ gyűrű ***kommutatív****, ha az $(S, \cdot)$ félcsoport kommutatív. Az $(S, +)$ csoport semleges elemét (ha létezik) az $\mathfrak{R}$ gyűrű* ***nullelemének*** *(röviden* ***nullának****), az $(S, \cdot)$*

félcsoport semleges elemét az $\mathfrak{R}$ gyűrű ***egységelemének*** *(röviden* ***egy****) nevezzük.*

- *Legyen $\mathfrak{R} = (S, +, \cdot)$ egységelemes gyűrű. $\mathfrak{R}$ egy x elemét pontosan akkor nevezzük* ***invertálhatónak*** *(vagy $\mathfrak{R}$* ***egységének****), ha ez az elem az $(S, \cdot)$ félcsoportban invertálható. $\mathfrak{R}$ valamely x eleme* ***nullosztó,*** *ha nem nulla, és egy $\mathfrak{R}$-beli nem nulla y elemmel $x \cdot y = y \cdot x = 0$.*
- ***Test*** *egy kommutatív, egységelemes gyűrű, amelyben minden, a nullától különböző elem invertálható.*

3.3. példa. *Csoport, gyűrű, test.* Legyen $1 < k \in \mathbb{N}$. A $\mathbb{Z}_k = \{0, 1, \ldots, k-1\}$ halmaz az egész számok modulo k tekintett összeadásával véges csoport, amiben semleges elem a 0. $\mathbb{Z}_k$ a modulo k vett összeadással és szorzással kommutatív egységelemes gyűrű (lásd a 3. 3-1. feladatot). Abban az esetben, ha p prímszám, akkor $\mathbb{Z}_p$ a modulo k tekintett összeadásra és szorzásra test.

Legyen $\text{lnko}(n, m)$ az n és m számok legnagyobb közös osztója. Legyen $1 < k \in \mathbb{N}$ esetén $\mathbb{Z}_k^* = \{i \mid 1 \leq i \leq k \text{ és } \text{lnko}(i, k) = 1\}$. A számok modulo k vett szorzásával $\mathbb{Z}_k^*$ véges csoport az 1 semleges elemmel.

Ha a szövegösszefüggésből egyértelműen kiderül, hogy valamely $\mathfrak{G} = (S, \circ)$ csoportban a $\circ$ a művelet, akkor nem szükséges explicit módon megadni. A 3.3. példabeli $\mathbb{Z}_k^*$ csoport fontos szerepet játszik a 3.3. alfejezetben, ahol az RSA kriptorendszert mutatjuk be. Ennek a csoportnak a rendjét az ***Euler-féle φ függvény*** adja meg, vagyis $\varphi(k) = |\mathbb{Z}_k^*|$. φ alábbi tulajdonságai a definíció következményei:

- $\varphi(m \cdot n) = \varphi(m) \cdot \varphi(n)$ minden $m, n \in \mathbb{N}$ esetén, ha $\text{lnko}(m, n) = 1$ és
- $\varphi(p) = p - 1$ minden p prímszámra.

Ezeknek a tulajdonságoknak a bizonyítását az Olvasóra hagyjuk (lásd a 3.1-3. gyakorlatot). Főként a 3.3.1. pontban használjuk az alábbi állítást, amely ezekből a tulajdonságokból közvetlenül következik.

3.3. állítás. *Ha $n = p \cdot q$, ahol p és q különböző prímszámok, akkor $\varphi(n) = (p-1)(q-1)$.*

A Lagrange-tétel szerint egy véges, multiplikatív $\mathfrak{G}$ csoport minden a elemére és az e semleges elemre fennáll az $a^{|\mathfrak{G}|} = e$ összefüggés, ahol $|\mathfrak{G}|$ a csoport rendjét jelöli. Az alábbi, úgynevezett Euler-tétel ennek speciális esete (a $\mathbb{Z}_n^*$ csoportra). Az Euler-tétel speciális esete pedig, ha n prímszám, és a-nak nem osztója, „kis Fermat-tételként” lett ismert.

3.4. tétel (Euler). *Minden $a \in \mathbb{Z}_n^*$ esetén $a^{\varphi(n)} \equiv 1 \bmod n$.*

3.5. következmény (kis Fermat-tétel). *Ha p prímszám és $a \in \mathbb{Z}_p^*$, akkor $a^{p-1} \equiv 1 \bmod p$.*

A 4.4. alfejezetben gráfizomorfizmus problémára mutatunk algoritmusokat. Ez a probléma, amely a 3.5.2. pontban található zéró-ismeretű protokollnál fontos szerepet játszik, bizonyos csoportelméleti problémák speciális eseteként is felfogható. Különösen fontosak itt a ***permutációcsoportok.***

3.6. definíció (permutációcsoport).

- *Egy* ***permutáció*** *valamely halmaz bijektív leképezése önmagára. Egy $n \geq 1$ természetes szám esetén legyen $[n] = \{1, 2, \ldots, n\}$. Az $[n]$ összes permutációinak halmazát $\mathfrak{S}_n$ jelöli. Algoritmikus célokra a $\pi \in \mathfrak{S}_n$ permutációt n darab $[n] \times [n]$-beli $(i, \pi(i))$ rendezett párból álló listával reprezentáljuk.*
- *Ha $\mathfrak{S}_n$-en permutációk kompozíciójaként definiálunk egy műveletet, akkor $\mathfrak{S}_n$ csoport lesz. Ha π és τ $\mathfrak{S}_n$-beli permutációk, akkor $\pi\tau$* ***kompozíciójuk*** *az az $\mathfrak{S}_n$-beli permutáció, amelyet úgy kapunk, hogy először π-t, azután τ-t alkalmazzuk $[n]$ elemeire, tehát $(\pi\tau)(i) = \tau(\pi(i))$ minden $i \in [n]$ esetén. Az $\mathfrak{S}_n$ permutációcsoport semleges eleme az* ***identikus permutáció,*** *amelyet a következőképpen definiálunk:* $\mathrm{id}(i) = i$ *minden $i \in [n]$ esetén. $\mathfrak{S}_n$-nek azt a részcsoportját, amely csupán az* id *elemet tartalmazza,* **id** *jelöli.*
- *$\mathfrak{S}_n$-nek valamely $\mathfrak{T}$ részhalmaza esetén a* ***$\mathfrak{T}$ által generált $\langle\mathfrak{T}\rangle$ permutációcsoportot*** *$\mathfrak{S}_n$ legszűkebb olyan részcsoportjaként definiáljuk, amely $\mathfrak{T}$-t tartalmazza. Az $\mathfrak{S}_n$ valamely $\mathfrak{G}$ részcsoportját az őt generáló halmazzal reprezentáljuk, amelyet $\mathfrak{G}$* ***generátorrendszerének*** *is nevezzük. $\mathfrak{G}$-ben valamely $i \in [n]$* ***elem pályáját*** *$\mathfrak{G}(i) = \{\pi(i) \mid \pi \in \mathfrak{G}\}$ definiálja.*
- *$[n]$ valamely T részhalmaza esetén $\mathfrak{S}_n^T$ az $\mathfrak{S}_n$-nek az a részcsoportja, amelyik T minden elemét önmagára képezi. Ha $i \leq n$ és $\mathfrak{S}_n$ részcsoportja $\mathfrak{G}$, akkor* ***$[i]$ $\mathfrak{G}$-beli (pontonkénti) stabilizátorát*** *az alábbi módon definiáljuk:*

$$\mathfrak{G}^{(i)} = \{\pi \in \mathfrak{G} \mid \pi(j) = j \text{ minden } j \in [i] \text{ esetén}\} .$$

  A $\mathfrak{G}^{(n)} = \mathbf{id}$ és $\mathfrak{G}^{(0)} = \mathfrak{G}$ összefüggések fennállnak.
- *Legyenek $\mathfrak{G}$ és $\mathfrak{H}$ permutációcsoportok, és $\mathfrak{H} \leq \mathfrak{G}$. $\tau \in \mathfrak{G}$ esetén $\mathfrak{H}\tau = \{\pi\tau \mid \pi \in \mathfrak{H}\}$* ***$\mathfrak{H}$ jobb oldali mellékosztálya $\mathfrak{G}$-ben****. $\mathfrak{H}$ két jobb oldali mellékosztálya $\mathfrak{G}$-ben azonos, vagy diszjunkt. Emiatt a $\mathfrak{G}$ permutációcsoportot $\mathfrak{H}$ $\mathfrak{G}$-beli jobb oldali mellékosztályai osztályozzák:*

$$\mathfrak{G} = \mathfrak{H}\tau_1 \cup \mathfrak{H}\tau_2 \cup \cdots \cup \mathfrak{H}\tau_k. \tag{3.3}$$

  $\mathfrak{H}$ mindegyik $\mathfrak{G}$-beli jobb oldali mellékosztályának a számossága $|\mathfrak{H}|$. A $\{\tau_1, \tau_2, \ldots, \tau_k\}$ (3.3) halmazt ***$\mathfrak{H}$ $\mathfrak{G}$-beli jobb oldali reprezentánsrendszerének*** *nevezzük.*

A pontonkénti stabilizátor fogalma különösen fontos olyan algoritmusok tervezésénél, amelyek permutációcsoportokkal dolgoznak. Az a lényeges struktúra, amelyet ilyenkor használunk azt úgy nevezzük, hogy $\mathfrak{G}$ permutációcsoportbeli ***stabilizátor lánc:***

$$\mathbf{id} = \mathfrak{G}^{(n)} \leq \mathfrak{G}^{(n-1)} \leq \cdots \leq \mathfrak{G}^{(1)} \leq \mathfrak{G}^{(0)} = \mathfrak{G} .$$

Minden i-re $1 \leq i \leq n$ esetén legyen $\mathfrak{T}_i$ a $\mathfrak{G}^{(i)}$-nek $\mathfrak{G}^{(i-1)}$-ben teljes jobb oldali reprezentánsrendszere. Ekkor azt mondjuk, hogy $\mathfrak{T} = \bigcup_{i=1}^{n-1} \mathfrak{T}_i$ ***erős generátora $\mathfrak{G}$-nek,*** és $\mathfrak{G} = \langle\mathfrak{T}\rangle$. Ekkor minden $\pi \in \mathfrak{G}$ egyértelműen faktorizálható $\pi = \tau_1\tau_2\cdots\tau_n$ alakban, ahol $\tau_i \in \mathfrak{T}_i$. A permutációcsoportokra vonatkozó alábbi algoritmuselméleti eredmények hasznosak lesznek később, a 4.4. alfejezetben.

3.7. tétel. *Ha a $\mathfrak{G} \leq \mathfrak{S}_n$ permutációcsoport a generátorrendszerével adott, akkor fennáll a következő:*

1. *Minden $i \in [n]$ esetén i $\mathfrak{G}(i)$ pályája $\mathfrak{G}$-ben polinomiális idő alatt kiszámítható.*
2. *Az* **id** $= \mathfrak{G}^{(n)} \leq \mathfrak{G}^{(n-1)} \leq \cdots \leq \mathfrak{G}^{(1)} \leq \mathfrak{G}^{(0)} = \mathfrak{G}$ *stabilizátorlánc kiszámítható n-ben polinomidő alatt, ami azt jelenti, hogy meghatározzuk minden i, $1 \leq i \leq n$ esetén $\mathfrak{G}^{(i)}$ teljes jobb oldali reprezentánsrendszerét $\mathfrak{T}_i$-t $\mathfrak{G}^{(i-1)}$-ben és ezzel $\mathfrak{G}$ egy erős generátorrendszerét.*

Azokat a fogalmakat, amelyeket a 3.6. definícióban vezettünk be a permutációcsoporttal kapcsolatban, most a gráfelméletből vett konkrét példák alapján fogjuk megvilágítani. Elsősorban gráfok automorfizmuscsoportját és izomorfizmuscsoportját vizsgáljuk. Ehhez szükségünk lesz néhány gráfelméleti fogalomra.

3.8. definíció (gráfok izomorfizmusa és automorfizmusa). *Egy G gráf egy véges $V(G)$ csúcshalmazból, és egy véges $E(G)$ élhalmazból áll, melyben lévő élek bizonyos csúcsokat összekötnek egymással. Feltesszük, hogy nincsenek párhuzamos- és hurokélek. Ebben az alfejezetben kizárólag irányítatlan gráfokat vizsgálunk, ami azt jelenti, hogy az éleknek nincs irányításuk és rendezetlen csúcspárokként foghatjuk fel őket. Két gráf, G és H,* ***diszjunkt egyesítését,*** *$G \cup H$-t, amelyek $V(G)$ és $V(H)$ csúcshalmazai átnevezéssel diszjunkttá tehetők, a $V(G) \cup V(H)$ csúcshalmazzal és az $E(G) \cup E(H)$ élhalmazzal definiáljuk.*

Legyen G és H két gráf ugyanannyi csúccsal. ***G és H közötti izomorfizmus*** *G csúcshalmazáról H csúcshalmazára képező éltartó bijekció. Állapodjunk meg abban, hogy $V(G) = \{1, 2, \ldots, n\} = V(H)$. Ekkor tehát G és H pontosan akkor izomorf (röviden $G \cong H$), ha van olyan $\pi \in \mathfrak{S}_n$ permutáció, hogy minden $i, j \in V(G)$ csúcsra fennáll a következő:*

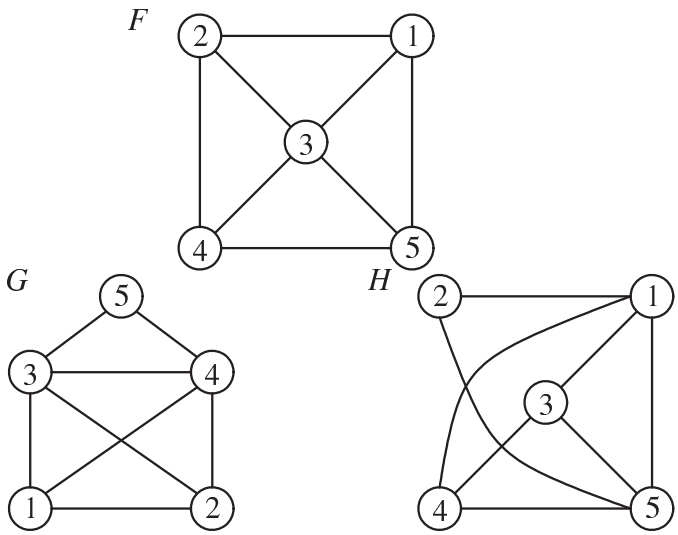
$$\{i, j\} \in E(G) \iff \{\pi(i), \pi(j)\} \in E(H) \,. \tag{3.4}$$

G egy automorfizmusa *G csúcshalmazának önmagára való éltartó bijekciója. Minden gráf esetén fennáll az* id *triviális automorfizmus. Jelöljük* $\mathrm{Iso}(G, H)$*-val a G és H közötti összes izomorfizmus halmazát,* $\mathrm{Aut}(G)$*-vel pedig G összes automorfizmusának halmazát. A* ***gráfizomorfizmus problémát*** *(röviden* `GI`*) és a* ***gráfautomorfizmus problémát*** *(röviden* `GA`*) a következő módon definiáljuk:*

$$\begin{aligned} \texttt{GI} &= \{(G, H) \mid G \text{ és } H \text{ izomorf gráfok}\} \,; \\ \texttt{GA} &= \{G \mid G \text{ tartalmaz nem triviális automorfizmust}\} \,. \end{aligned}$$

Algoritmikus célokra a gráfokat vagy csúcs- és él-listájukkal, vagy csúcsmátrixukkal reprezentáljuk, amelyben az (i, j) helyen 1 áll, ha $\{i, j\}$ egy él a gráfban, különben pedig 0. Valamely gráfnak ehhez az előállításához elegendő a $\Sigma = \{0, 1\}$ ábécével való kódolás. Ha gráfpárokat kívánunk szemléltetni, akkor olyan bijektív párosítási $(\cdot, \cdot)$ függvényt alkalmazunk, amely $\Sigma^* \times \Sigma^*$-ról Σ^*-ra képez, polinomiális időben kiszámítható, és van polinomiális időben kiszámítható inverze.

3.4. példa. (Gráfok izomorfizmusa és automorfizmusa.) A 3.5. ábrán látható G és H gráfok izomorfak. Egy $\pi : V(G) \to V(H)$ izomorfizmus, amelynél (3.4)-nek megfelelően adjuk meg a csúcsok szomszédosságát, megadható például a következő módon: $\pi = \left(\begin{smallmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 4 & 1 & 5 & 2 \end{smallmatrix}\right)$, illetve ciklikus módon írva $\pi = (1\,3)(2\,4\,5)$. G és H között van még három további izomorfizmus, és így $|\mathrm{Iso}(G, H)| = 4$, (lásd a 3.1-4. gyakorlatot). G és H azonban nem izomorfak F-fel. Ez azonnal látható, ha a ***fokszámok*** sorozatát nézzük (vagyis minden csúcs esetén a csúcsra illeszkedő élek számát). A G-hez, illetve H-hoz



3.5. ábra. A három gráf: G és H izomorfak, de F nem izomorf velük.

tartozó sorozat különbözik az F-hez tartozótól: G és H esetén ez a sorozat $(2,3,3,4,4)$, miközben F esetén $(3,3,3,3,4)$. G egy nem triviális $\varphi : V(G) \to V(G)$ automorfizmusa például $\varphi = \left(\begin{smallmatrix}1&2&3&4&5\\2&1&4&3&5\end{smallmatrix}\right)$, illetve $\varphi = (1\,2)(3\,4)(5)$ segítségével adható meg, egy másik pedig $\tau = \left(\begin{smallmatrix}1&2&3&4&5\\1&2&4&3&5\end{smallmatrix}\right)$ illetve $\tau = (1)(2)(3\,4)(5)$ segítségével. G-nek van még két további automorfizmusa és így $|\mathrm{Aut}(G)| = 4$ (lásd a 3.1-4. gyakorlatot).

Az $\mathrm{Iso}(G,H)$, $\mathrm{Aut}(F)$, $\mathrm{Aut}(G)$ és $\mathrm{Aut}(H)$ permutációcsoportok $\mathfrak{S}_5$ részcsoportjai. $\mathrm{Aut}(G)$-ben az $\mathrm{Aut}(G)^{(5)} \leq \mathrm{Aut}(G)^{(4)} \leq \cdots \leq \mathrm{Aut}(G)^{(1)} \leq \mathrm{Aut}(G)^{(0)}$ stabilizátorlánc az $\mathrm{Aut}(G)^{(5)} = \mathrm{Aut}(G)^{(4)} = \mathrm{Aut}(G)^{(3)} = \mathbf{id}$, $\mathrm{Aut}(G)^{(2)} = \mathrm{Aut}(G)^{(1)} = \langle\{\mathrm{id}, \tau\}\rangle$ és $\mathrm{Aut}(G)^{(0)} = \mathrm{Aut}(G)$ részcsoportokból áll. G automorfizmuscsoportjában, $\mathrm{Aut}(G)$-ben az 1 és 2 csúcsok pályája $\{1,2\}$, a 3 és 4 csúcsok pályája $\{3,4\}$, és az 5 pályája $\{5\}$.

Az $\mathrm{Iso}(G,H)$ és $\mathrm{Aut}(G)$ permutációcsoportok rendje pontosan akkor azonos, ha G és H izomorfak. Ha ugyanis G és H izomorfak, akkor $|\mathrm{Iso}(G,H)| = |\mathrm{Aut}(G)|$ következik abból, hogy $\mathrm{Aut}(G) = \mathrm{Iso}(G,G)$. Másrészt ha $G \ncong H$, akkor $\mathrm{Iso}(G,H)$ üres, ugyanakkor $\mathrm{Aut}(G)$-ben mindig benne van az id triviális automorfizmus. Ebből következik a 3.9. lemma (3.5) állítása, amire később a 4.4. alfejezetben szükségünk lesz. A (3.6) igazolására tegyük fel, hogy G és H összefüggőek; ellenkező esetben G illetve H helyett a $\overline{G}$ illetve $\overline{H}$ komplementer gráfokat tekintjük, (lásd a 3.1-5. gyakorlatban). $G \cup H$ egy automorfizmusa, amely G és H csúcsait felcseréli, egy $\mathrm{Iso}(G,H)$-beli és egy $\mathrm{Iso}(H,G)$-beli izomorfizmusból tevődik össze. Így $|\mathrm{Aut}(G \cup H)| = |\mathrm{Aut}(G)| \cdot |\mathrm{Aut}(H)| + |\mathrm{Iso}(G,H)|^2$, amiből (3.5) felhasználásával következik (3.6).

3.9. lemma. *Bármely két G és H gráf esetén fennállnak a következők:*

$$|\mathrm{Iso}(G,H)| = \begin{cases} |\mathrm{Aut}(G)| = |\mathrm{Aut}(H)|, & \text{ha } G \cong H \\ 0, & \text{ha } G \ncong H \ ; \end{cases} \tag{3.5}$$

$$|\mathrm{Aut}(G \cup H)| = \begin{cases} 2 \cdot |\mathrm{Aut}(G)| \cdot |\mathrm{Aut}(H)|, & \text{ha } G \cong H \\ |\mathrm{Aut}(G)| \cdot |\mathrm{Aut}(H)|, & \text{ha } G \ncong H \ . \end{cases} \tag{3.6}$$

Ha G és H izomorf gráfok, és $\tau \in \mathrm{Iso}(G,H)$, akkor $\mathrm{Iso}(G,H) = \mathrm{Aut}(G)\tau$. Ez azt jelenti, hogy $\mathrm{Iso}(G,H)$ $\mathrm{Aut}(G)$-nek jobb oldali mellékosztálya $\mathfrak{S}_n$-ben. Mivel két jobb oldali mellékosztály azonos, vagy diszjunkt, $\mathfrak{S}_n$ (3.3) szerint $\mathrm{Aut}(G)$ jobb oldali mellékosztályaira bomlik:

$$\mathfrak{S}_n = \mathrm{Aut}(G)\tau_1 \cup \mathrm{Aut}(G)\tau_2 \cup \cdots \cup \mathrm{Aut}(G)\tau_k \ , \tag{3.7}$$

ahol $|\mathrm{Aut}(G)\tau_i| = |\mathrm{Aut}(G)|$ minden i, $1 \le i \le k$ esetén. Ezek szerint az $\mathfrak{S}_n$ permutációiból álló $\{\tau_1, \tau_2, \ldots, \tau_k\}$ halmaz $\mathrm{Aut}(G)$-nek $\mathfrak{S}_n$-beli jobb oldali reprezentánsrendszerét alkotja. Ha $\pi(G)$ azt a $H \cong G$ gráfot jelöli, amelyet akkor kapunk, ha a $\pi \in \mathfrak{S}_n$ permutációt G csúcsaira alkalmazzuk, akkor $\{\tau_i(G) \mid 1 \le i \le k\} = \{H \mid H \cong G\}$. Mivel $\mathfrak{S}_n$-ben pontosan $n! = n(n-1)\cdots 2 \cdot 1$ permutáció van, (3.7)-ből következik:

$$|\{H \mid H \cong G\}| = k = \frac{|\mathfrak{S}_n|}{|\mathrm{Aut}(G)|} = \frac{n!}{|\mathrm{Aut}(G)|} \ .$$

Ezzel beláttuk az alábbi következményt.

3.10. következmény. *Ha a G gráfnak n csúcsa van, akkor pontosan $n!/|\mathrm{Aut}(G)|$ vele izomorf gráf van.*

A 3.4. példa 3.5. ábráján lévő G gráffal így pontosan $5!/4 = 30$ gráf izomorf. A következő lemmára később, a 4.4. alfejezetben lesz szükségünk.

3.11. lemma. *Legyenek G és H n csúcsú gráfok. Definiáljuk az alábbi halmazt:*

$$A(G,H) \quad = \quad \{(F,\varphi) \mid F \cong G \text{ és } \varphi \in \mathrm{Aut}(F)\} \cup \{(F,\varphi) \mid F \cong H \text{ és } \varphi \in \mathrm{Aut}(F)\}.$$

Ekkor fennáll a következő:

$$|A(G,H)| \quad = \quad \begin{cases} n!, & \text{ha } G \cong H \\ 2n!, & \text{ha } G \ncong H. \end{cases}$$

Bizonyítás. Ha F és G izomorfak, akkor $|\mathrm{Aut}(F)| = |\mathrm{Aut}(G)|$. A 3.10. következményből adódik, hogy

$$|\{(F,\varphi) \mid F \cong G \text{ és } \varphi \in \mathrm{Aut}(F)\}| = \frac{n!}{|\mathrm{Aut}(F)|} \cdot |\mathrm{Aut}(F)| = n!,$$

és hasonlóan megmutatható az is, hogy $|\{(F,\varphi) \mid F \cong H \text{ és } \varphi \in \mathrm{Aut}(F)\}| = n!$. Ha G és H izomorfak, akkor

$$\{(F,\varphi) \mid F \cong G \text{ és } \varphi \in \mathrm{Aut}(F)\} \quad = \quad \{(F,\varphi) \mid F \cong H \text{ és } \varphi \in \mathrm{Aut}(F)\} \ .$$

Ebből következik, hogy $|A(G,H)| = n!$. Ha pedig G és H nem izomorfak, akkor $\{(F,\varphi) \mid F \cong G \text{ és } \varphi \in \mathrm{Aut}(F)\}$ és $\{(F,\varphi) \mid F \cong H \text{ és } \varphi \in \mathrm{Aut}(F)\}$ diszjunkt halmazok. Eszerint $|A(G,H)| = 2n!$. ∎

Gyakorlatok

3.1-1. A 3.6. ábrán láthatunk két elfogott rejtjelezett szöveget, c_1-et és c_2-t. Tudjuk, hogy

c_1	W K L V V H Q W H Q F H L V H Q F U B S W H G E B F D H V D U V N H B
c_2	N U O S J Y A Z E E W R O S V H P X Y G N R J B P W N K S R L F Q E P

3.6. ábra. Ugyanannak a nyílt szövegnek két rejtett változata a 3.1-1. gyakorlatban.

mindkettő ugyanannak az m nyílt szövegnek a rejtett változata, és az egyik eltolásos rejtjelezővel, a másik pedig Vigenère-rejtjelezővel lett sifrírozva. Mindkét szöveget fejtsük vissza. *Útmutatás.* A visszafejtés után megállapítható, hogy az egyik rejtjelező módszernél egy igaz, a másiknál egy hamis állítás adódik. Van esetleg értelme gyakorisági elemzést alkalmazni?

3.1-2. Mutassuk meg, hogy $\mathbb{Z}$ a szokásos összeadással és szorzással nullosztómentes gyűrű. Vajon $\mathbb{Z}$ test? Mit mondhatunk az $(\mathbb{N}, +)$, $(\mathbb{N}, \cdot)$ és $(\mathbb{N}, +, \cdot)$ algebrai struktúrák tulajdonságairól?

3.1-3. Bizonyítsuk be az Euler-féle φ-függvény alábbi tulajdonságait:

a. $\varphi(m \cdot n) = \varphi(m) \cdot \varphi(n)$ minden $m, n \in \mathbb{N}$ esetén, ha $\text{lnko}(m, n) = 1$.

b. $\varphi(p) = p - 1$ minden p prímszám esetén.

Ezeknek a tulajdonságoknak a segítségével bizonyítsuk be a 3.3. állítást.

3.1-4. Adottak az F, G és H gráfok a 3.5. ábrán.

a. Határozzuk meg a G és H közötti összes izomorfizmust.

b. Határozzuk meg F, G és H összes automorfizmusát.

c. A G és H közötti melyik izomorfizmus esetén lesz $\text{Iso}(G, H)$ $\text{Aut}(G)$-nak jobb oldali mellékosztálya $\mathfrak{S}_5$-ben, vagyis melyik $\tau \in \text{Iso}(G, H)$ esetén lesz $\text{Iso}(G, H) = \text{Aut}(G)\tau$? Határozzuk meg $\text{Aut}(F)$, $\text{Aut}(G)$ és $\text{Aut}(H)$ jobb oldali reprezentánsrendszerét $\mathfrak{S}_5$-ben.

d. Határozzuk meg F minden csúcsának pályáját $\text{Aut}(F)$-ben és H minden csúcsának pályáját $\text{Aut}(H)$-ban.

e. Határozzuk meg az $\text{Aut}(F) \leq \mathfrak{S}_5$ és $\text{Aut}(H) \leq \mathfrak{S}_5$ részcsoportok stabilizátorláncát.

f. Hány 5 csúcsú gráf izomorf F-fel?

3.1-5. Valamely G gráf komplementer gráfját a következőképpen definiáljuk: a csúcshalmaza $V(\overline{G}) = V(G)$, az élhalmaza pedig $E(\overline{G}) = \{\{i, j\} \mid i, j \in V(\overline{G}) \text{ és } \{i, j\} \notin E(G)\}$. Mutassuk meg, hogy:

a. $\text{Aut}(G) = \text{Aut}(\overline{G})$.

b. $\text{Iso}(G, H) = \text{Iso}(\overline{G}, \overline{H})$.

c. $\overline{G}$ összefüggő, ha G nem összefüggő.

3.2. Kulcscsere Diffie és Hellman szerint

Ebben az alfejezetben, valamint a következőkben is szükségünk lesz a 3.1.3. pontban látott számelméleti alapfogalmakra. Elsősorban a 3.3. példabeli $\mathbb{Z}_k^*$ multiplikatív csoportra és az Euler-féle φ függvényre gondolunk; a maradékosztálygyűrűk aritmetikáját a fejezet végén fogjuk megvilágítani a 3-1. feladatban.

A 3.7. ábra a kulcscserére vonatkozó Diffie–Hellman-protokollt mutatja. Ez a protokoll az r alapú, p modulusú moduláris exponenciális függvényen alapszik, ahol p prímszám és r primitív gyök modulo p. $r \in \mathbb{Z}_n^*$ ***primitív gyök modulo*** n, ha $r^d \not\equiv 1 \bmod n$ minden d, $1 \leq d < \varphi(n)$ esetén. Egy r modulo n primitív gyök előállítja az egész $\mathbb{Z}_n^*$ csoportot, ami

Lépés	Aliz	Erik	Bob
1	Aliz és Bob megállapodnak egy nagy p prímszámban, és egy r modulo p primitív gyökben; p és r nyilvánosak		
2	véletlenszerűen választ egy nagy, titkos a számot, és kiszámítja a következőt: $\alpha = r^a \bmod p$		véletlenszerűen választ egy nagy, titkos b számot, és kiszámítja a következőt: $\beta = r^b \bmod p$
3		$\alpha \Rightarrow$ $\Leftarrow \beta$	
4	kiszámítja $k_A = \beta^a \bmod p$ értékét		kiszámítja $k_B = \alpha^b \bmod p$ értékét

3.7. ábra. Diffie és Hellman szerinti kulcscserére vonatkozó protokoll.

azt jelenti, hogy $\mathbb{Z}_n^* = \{r^i \mid 0 \le i < \varphi(n)\}$. Emlékeztetünk arra, hogy ha p prímszám, akkor a $\mathbb{Z}_p^*$ multiplikatív csoport rendje $\varphi(p) = p - 1$. $\mathbb{Z}_p^*$-nak pontosan $\varphi(p-1)$ primitív gyöke van (lásd a 3.2-1. gyakorlatot is).

3.5. példa. Nézzük $\mathbb{Z}_5^* = \{1,2,3,4\}$-et. Mivel $\mathbb{Z}_4^* = \{1,3\}$, így $\varphi(4) = 2$, tehát a modulo 5 primitív gyökök száma 2, modulo 5 primitív gyökök a 2 és a 3, így mind 2 mind 3 előállítja az egész $\mathbb{Z}_5^*$-ot, mivel

$$\begin{array}{llll} 2^0 = 1; & 2^1 = 2; & 2^2 = 4; & 2^3 \equiv 3 \bmod 5\ ; \\ 3^0 = 1; & 3^1 = 3; & 3^2 \equiv 4 \bmod 5; & 3^3 \equiv 2 \bmod 5\ . \end{array}$$

Nem minden n számnak van modulo n primitív gyöke; a 8 a legkisebb ilyen példa. Az elemi számelméletből tudjuk, hogy valamely n számra pontosan akkor van modulo n primitív gyök, ha n vagy $\{1,2,4\}$-nek eleme, vagy pedig $n = q^k$, illetve $n = 2q^k$ alakú, ahol q páratlan prímszám.

3.12. definíció (diszkrét logaritmus). *Legyen p prímszám és r egy modulo p primitív gyök. Az r **alapú,** p **modulusú** $\exp_{r,p}$ **moduláris exponenciális függvényt,** amely $\mathbb{Z}_{p-1}$-ről $\mathbb{Z}_p^*$-re képez, $\exp_{r,p}(a) = r^a \bmod p$-vel definiáljuk. Az inverz függvényét **diszkrét logaritmusnak** nevezzük, ez rögzített p és r esetén az $\exp_{r,p}(a)$ értéket a-ra képezi. Fennáll, hogy $a = \log_r \exp_{r,p}(a) \bmod p$.*

A 3.7. ábra protokollja működik, mert $k_A = \beta^a = r^{ba} = r^{ab} = \alpha^b = k_B$ miatt (modulo p aritmetikával) Aliz valójában ugyanazt a kulcsot számolja ki, mint Bob. Ezt a kulcsot könnyen kiszámíthatják, mivel az $\exp_{r,p}$ moduláris exponenciális függvény a MODULÁRIS-HATVÁNYOZÓ algoritmussal hatékonyan meghatározható. Ezzel szemben Erik arra irányuló törekvése, hogy a kulcsukat meghatározza, nehézségekbe ütközik, mert a diszkrét logaritmus igen nehéz problémának számít. Az $\exp_{r,p}$ moduláris exponenciális függvény így egy lehetséges ***egyirányú függvény,*** olyan függvény, amelyik bár könnyen kiszámítható, de csak nehezen invertálható. Sajnos máig nem tudjuk, hogy van-e egyáltalán egyirányú függvény. S ami még inkább kár: jóllehet nem tudjuk, hogy van-e ilyen, a kriptográfiában az egyirányú függvények kulcsszerepet játszanak, mert sok kriptorendszer biztonsága azon a feltevésen alapszik, hogy egyirányú függvények valóban vannak.

MODULÁRIS-HATVÁNYOZÓ(a, b, m)

1 ▷ m a modulus, $b < m$ az alap és a a kitevő
2 állítsuk elő a kitevőt kettes számrendszerben: $a = \sum_{i=0}^{k} a_i 2^i$, ahol $a_i \in \{0, 1\}$.
3 számoljuk ki egymás után a b^{2^i} értékeket, ahol $0 \le i \le k$, felhasználva, hogy $b^{2^{i+1}} = \left(b^{2^i}\right)^2$.
4 számoljuk ki modulo m aritmetikával $b^a = \prod_{\substack{i=0 \\ a_i=1}}^{k} b^{2^i}$ értékét.
5 ▷ ha beleszámoltuk a szorzatba b^{2^i}-t, és kiszámítottuk $b^{2^{i+1}}$-t, akkor b^{2^i}-t kitörölhetjük
6 **return** b^a

A $b^a = \prod_{\substack{i=0 \\ a_i=1}}^{k} b^{2^i}$ kiszámítása helyes, mert a modulo m aritmetikában fennáll a következő:

$$b^a = b^{\sum_{i=0}^{k} a_i 2^i} = \prod_{i=0}^{k} \left(b^{2^i}\right)^{a_i} = \prod_{\substack{i=0 \\ a_i=1}}^{k} b^{2^i} .$$

Miért lehet hatékonyan kiszámítani az $\exp_{r,p}(a) = r^a \bmod p$ hatványfüggvényt? Ha ezt a számítást naivan hajtjuk végre, sok szorzásra lehet szükségünk az a kitevő mindegyik értékére. A MODULÁRIS-HATVÁNYOZÓ algoritmussal azonban Aliznak nem $a - 1$ szorzást kell végeznie, mint a naiv módszernél, hanem csupán legfeljebb $2 \log a$ szorzásra van szüksége. A MODULÁRIS-HATVÁNYOZÓ algoritmus egy exponenciális tényezővel gyorsítja a moduláris hatványozást.

3.6. példa. (MODULÁRIS-HATVÁNYOZÓ a Diffie–Hellman-protokollban.) Aliz és Bob a $p = 5$ prímszámot és az $r = 3$-at, mint modulo 5 primitív gyököt választja. Aliz az $a = 17$ titkos számot választja. Mivel Aliz Bobnak az α számot akarja küldeni, ki szeretné számítani az $\alpha = 3^{17} = 129140163 \equiv 3 \bmod 5$ számot. A kitevő bináris alakban felírva $17 = 1 + 16 = 2^0 + 2^4$. Aliz lépésenként számítja ki az értéket:

$$3^{2^0} = 3;\ 3^{2^1} = 3^2 \equiv 4 \bmod 5;\ 3^{2^2} \equiv 4^2 \equiv 1 \bmod 5;\ 3^{2^3} \equiv 1^2 \equiv 1 \bmod 5;\ 3^{2^4} \equiv 1^2 \equiv 1 \bmod 5.$$

Ebből kiszámítja a $3^{17} \equiv 3^{2^0} \cdot 3^{2^4} \equiv 3 \cdot 1 \equiv 3 \bmod 5$ értéket. Figyeljük meg, hogy Aliz ahelyett, hogy 16-szor szorzott volna, csupán négyszer emelt négyzetre és egyszer szorzott $\alpha = 3 \bmod 5$ meghatározása érdekében. Ha Bob saját magának a $b = 23$ titkos kitevőt választja, akkor ugyanezzel az eljárással ki tudja számítani a kulcs őrá eső részét, tehát $\beta = 3^{23} = 94143178827 \equiv 2 \bmod 5$, végül Aliz és Bob a közös titkos kulcsukat a 3.7. ábrán látható Diffie–Hellman-protokollal meghatározzák (lásd a 3.2-2. gyakorlatot). A protokoll biztonsága ebben az esetben természetesen nincs garantálva, hiszen $p = 5$-tel nagyon kicsi prímszámot választottak; ez a játékpélda csak a könnyebb megértést szolgálja.

Ha Erik figyelmesen lehallgatja Aliz és Bob társalgását, amely a 3.7. ábra szerinti Diffie–Hellman-protokoll szerint zajlik, akkor ismeri a p, r, α és β értékeket, amelyekből szeretné meghatározni a $k_A = k_B$ titkos kulcsukat. Eriknek ezt a problémáját ***Diffie–Hellman-problémának*** nevezik. Ha Erik meg tudná határozni az $a = \log_r \alpha \bmod p$ és a $b = \log_r \beta \bmod p$ titkos számokat, akkor, akárcsak Aliz és Bob, ki tudná számolni a $k_A = \beta^a \bmod p = \alpha^b \bmod p = k_B$ kulcsot, és megoldaná a Diffie–Hellman-problémát. Ez a probléma tehát nem nehezebb, mint a diszkrét logaritmus problémája. A fordított kérdés, hogy vajon a Diffie–Hellman-probléma *legalább* olyan nehéz-e, mint a diszkrét logaritmusé, tehát hogy ugyanolyan nehezek-e, ez máig csupán nem bizonyított sejtés. Mint sok más protokoll, a Diffie–Hellman-protokoll biztonságossága mindezidáig nem bizonyított.

Mivel szerencsére eddig sem a diszkrét logaritmus, sem a Diffie–Hellman-probléma hatékonyan nem oldható meg, ez a közvetlen támadás nem jelent valóságos veszélyt. Másrészt azonban vannak más, nem közvetlen támadások, amelyeknél a kulcsot nem közvetlenül a Diffie–Hellman-protokollban átadott α, illetve β értékekből határozzák meg. Így például a Diffie–Hellman nem biztonságos a *középen állás támadással* szemben. A fent leírt támadás ***passzív,*** ez pedig ***aktív*** abban az értelemben, hogy a támadó Erik nem elégszik meg a passzív hallgatózással, hanem megpróbálja a protokollt aktívan a saját ízlésének megfelelően megváltoztatni. Úgyszólván beáll „középre", Aliz és Bob közé és elfogja Aliz Bobnak küldött $\alpha = r^a \bmod p$, és Bob Aliznak küldött $\beta = r^b \bmod p$ üzenetét. α és β helyett saját $\alpha_E = r^c \bmod p$ értékét továbbítja Bobnak, Aliznak pedig a szintén saját $\beta_E = r^d \bmod p$ értéket, miközben Erik a titkos c és d értékeket maga választotta. Ha most Aliz a $k_A = (\beta_E)^a \bmod p$ értéket kiszámítja, k_A valójában ezentúl nem a Bobbal – mint ahogy azt Aliz hiszi –, hanem az Erikkel való kommunikáció kulcsa, mert Erik ezt a kulcsot ugyanúgy értékeli ki, mint ők (modulo p aritmetikával): $k_E = \alpha^d = r^{ad} = r^{da} = (\beta_E)^a = k_A$ segítségével. Ugyanígy lefolytathat Erik egy észrevehetetlen kulcscserét Bobbal, és a jövőben vele kommunikálhat anélkül, hogy ezt Bob sejtené. A ***hitelességnek*** ezzel a problémájával később fogunk alaposabban foglalkozni a zéró-ismeretű protokollokról szóló 3.5. alfejezetben.

Gyakorlatok

3.2-1. *a.* Hány primitív gyök van $\mathbb{Z}_{13}^*$-ban, illetve $\mathbb{Z}_{14}^*$-ban?

b. Határozzuk meg $\mathbb{Z}_{13}^*$ valamint $\mathbb{Z}_{14}^*$ összes primitív gyökét, és bizonyítsuk be, hogy ezek valóban primitív gyökök.

c. Mutassuk meg az összes modulo 13, illetve modulo 14 primitív gyökről, hogy a teljes $\mathbb{Z}_{13}^*$-ot, illetve a teljes $\mathbb{Z}_{14}^*$-ot előállítják.

3.2-2. *a.* Határozzuk meg a $\beta = 3^{23} = 94143178827 \equiv 2 \bmod 5$ értéket (Bob 3.6. példabeli értékét) a MODULÁRIS-HATVÁNYOZÓ algoritmussal.

b. Határozzuk meg a 3.6. példában az α és β számokhoz Aliz és Bob közös titkos kulcsát a 3.7. ábrán lévő Diffie–Hellman-protokoll szerint.

3.3. RSA és faktorizálás

3.3.1. RSA

Az RSA-kriptorendszer, amelyet három feltalálójáról, Ron Rivest, Adi Shamir és Leonard Adleman-ról neveztek el, az első ismert ***nyilvános kulcsú kriptorendszer***. Manapság is igen népszerű, és sok kriptográfiai alkalmazásba beillesztik. A 3.8. ábrán összefoglaltuk az RSA protokoll egyes lépéseit, amelyeket végül részletesen is leírunk, valamint lásd a 3.7. példát.

Kulcsgenerálás. Bob választ véletlenszerűen két nagy prímszámot, p-t és q-t úgy, hogy $p \neq q$, és kiszámítja $n = pq$ szorzatukat. Azután választ egy $e \in \mathbb{N}$ kitevőt az alábbiak szerint

$$1 < e < \varphi(n) = (p-1)(q-1) \quad \text{és} \quad \text{lnko}(e, \varphi(n)) = 1\,, \tag{3.8}$$

majd meghatározza $e \bmod \varphi(n)$ inverzét, vagyis azt az egyértelmű d számot, amelyre:

$$1 < d < \varphi(n) \quad \text{és} \quad e \cdot d \equiv 1 \bmod \varphi(n). \tag{3.9}$$

Lépés	Aliz	Erik	Bob
1			Véletlenszerűen választ két nagy prímszámot, p-t és q-t és kiszámítja az $n = pq$ és $\varphi(n) = (p-1)(q-1)$ értékeket, nyilvános (n, e) kulcsát, és d titkos kulcsát, amelyek megfelelnek (3.8) és (3.9)-nek
2		$\Leftarrow (n, e)$	
3	$c = m^e \bmod n$ szerint rejtjelezi m-et		
4		$c \Rightarrow$	
5			visszafejti c-t $m = c^d \bmod n$ szerint

3.8. ábra. Az RSA-protokoll.

Az (n, e) pár Bob *nyilvános kulcsa, d* pedig Bob *titkos kulcsa.*

Rejtjelezés. Mint már a 3.1. alfejezetben is, az üzenetek egy Σ ábécé feletti szavak és blokkonként, állandó hosszúságú blokkhosszal $|\Sigma|$-adikus természetes számokként kódoljuk őket. Ezeket a számokat azután rejtjelezzük. Legyen $m < n$ az üzenet egyik rejtjelezett blokkjának megfelelő szám, amelyet Aliz szeretne Bobnak küldeni. Aliz ismeri Bob nyilvános (n, e) kulcsát, és m-et rejtjelezi $c = E_{(n,e)}(m)$ számként, ahol a rejtjelező függvény definíciója az alábbi:

$$E_{(n,e)}(m) = m^e \bmod n \ .$$

Visszafejtés. Legyen c, $0 \leq c < n$ az a szám, amely a rejtjelezett szöveg egyik blokkjának kódja, amelyet Bob megkap, Erik pedig lehallgatja. Bob desifrírozza c-t a d titkos kulcsa és az alábbi visszafejtő függvény segítségével:

$$D_d(c) = c^d \bmod n \ .$$

Azt, hogy az RSA-módszer valóban kriptorendszer a 3.1. definíció értelmében, a 3.13. tétel állítja. Ennek a tételnek a bizonyítását az Olvasóra hagyjuk (lásd a 3.3-1. gyakorlatot).

3.13. tétel. *Legyenek (n, e) a nyilvános, d a titkos kulcsok az RSA-protokollban. Ekkor minden m, $0 \leq m < n$ üzenet esetén $m = (m^e)^d \bmod n$. Így az RSA (nyilvános kulcsú) kriptorendszer.*

Az RSA-eljárás hatékonysága érdekében megint a MODULÁRIS-HATVÁNYOZÓ algoritmust használjuk a gyors-hatványozásra. Hogyan válasszuk meg a 3.8. ábrán bemutatott RSA protokollban a p és q prímszámokat? Először is elég nagyoknak kell lenniük, egyébként Erik az n számot Bob nyilvános (n, e) kulcsában faktorizálná, és n p és q prímfaktorait meghatározná, így a kiterjesztett euklideszi algoritmussal könnyen meghatározhatná Bob d titkos kulcsát, amely e mod $\varphi(n)$ egyértelmű inverze, ahol $\varphi(n) = (p-1)(q-1)$. A p és q prímszámoknak titkosaknak kell maradniuk, s ezért elegendően nagyoknak kell lenniük. A gyakorlatban p-t és q-t mindig legalább 80 decimális jegyűnek kell választani. Elő kell tehát állítani ilyen nagyságú véletlen számokat, és tesztelni kell egy ismert véletlen prímteszttel, hogy vajon valóban prímek-e. Mivel a prímszámok elmélete szerint körülbelül $N/\ln N$ N-nél kisebb prímszám van, nagy a valószínűsége annak, hogy nem túl sok próbálkozás után prímszámra akadjunk.

Elméletileg *determinisztikusan* polinomiális időben eldönthető, hogy p és q prímek-e. Agrawal, Kayal és Saxena a közelmúltban azt a meglepő eredményt publikálták, hogy a

PRIMES = {bin(n) | n prím} prímszámprobléma a P bonyolultsági osztályba tartozik. Ez az áttörés megoldotta a bonyolultságelmélet régóta nyitott problémáját, mert a gráfizomorfizmus probléma mellett a prímszámprobléma tűnt olyannak, amely se nem P-beli, se nem NP-teljes.[3] Bár az algoritmus futási idejének eredeti – $O(n^{12})$ – korlátját azóta $O(n^6)$-ra javították, az algoritmus a gyakorlati alkalmazások számára nem elég gyors.

A legkedveltebb véletlen prímteszt Rabin következő algoritmusa, amely Miller determinisztikus algoritmusának elvére épül.

MILLER–RABIN(n)

végezzük el az $n-1 = 2^k m$ átalakítást, ahol m és n páratlanok
válasszunk véletlenszerűen egy $z \in \{1, 2, \ldots, n-1\}$ számot egyenletes eloszlással
számítsuk ki az $x = z^m \bmod n$ értéket
if $x \equiv 1 \bmod n$
  then return „n prímszám”
  else for $j \leftarrow 0$ **to** $k-1$
    do if $x \equiv -1 \bmod n$
      then return „n prímszám”
      else $x \leftarrow x^2 \bmod n$
        return „n nem prímszám”

A MILLER–RABIN teszt úgynevezett ***Monte-Carlo-algoritmus,*** „nem” válasza megbízható, „igen” válasza azonban egy bizonyos hibavalószínűséggel értendő. A Miller–Rabin-teszthez hasonló Solovay és Strassen prímtesztje. Mindkettő $O(n^3)$ időben dolgozik. A Solovay–Strassen-teszt azonban kevésbé népszerű, mert a gyakorlatban nem olyan hatékony, mint a Miller–Rabin-teszt és kevésbé pontos.

Azon problémák osztályának, amelyek a Monte-Carlo-algoritmussal mindig megbízható „igen” válasszal oldódnak meg, külön nevük van: RP, a ***R***andomized ***P***olynomial Time kifejezésből származó betűszó. A komplementer osztály, coRP = $\{L \mid \overline{L} \in \text{RP}\}$, mindazokat a problémákat tartalmazza, amelyekre a Monte-Carlo-algoritmus mindig megbízható „nem” választ ad. RP-t formálisan a nemdeterminisztikus polinomiális idejű Turing-gépekkel definiáljuk (röviden NPTM; lásd a 4.1. alfejezetet és főként a 4.1., 4.2. és 4.3. definíciókat), amelyeknek a működése véletlen folyamatként fogható fel: minden nemdeterminisztikus elágazásnál a gép úgymond pénzfeldobással működik, és a következő két elágazás mindegyike 1/2 valószínűséggel következik be. Az NPTM-ben megvalósuló utak száma szerint minden bemeneti adathoz tartozik egy bizonyos megvalósulási valószínűség, ahol hibák is felléphetnek. RP definíciója megkívánja, hogy az elfogadott bemeneti értékek esetén a hibavalószínűség soha nem lehet nagyobb 1/2-nél, miközben az elutasított bemenetek esetén egyáltalán nem léphet fel hiba.

3.14. definíció (véletlenített polinomiális idő). *Az* RP *osztály pontosan azokat az A problémákat tartalmazza, amelyekhez van olyan NPTM M, hogy M mindegyik $x \in A$-t legalább* 1/2 *valószínűséggel fogadja el, $x \notin A$-t pedig* 0 *valószínűséggel fogadja el.*

3.15. tétel. *A* PRIMES *a* coRP *ben van.*

[3] A P és NP bonyolultsági osztályokat a 4.1. alfejezetben definiáljuk, az NP-teljességet pedig a 4.2. alfejezetben.

Üzenet	R S	A I	S T	H E	K E	Y T	O P	U B	L I	C K	E Y	C R	Y P	T O	G R	A P	H Y
m_b	460	8	487	186	264	643	379	521	294	62	128	69	639	508	173	15	206
c_b	697	387	229	340	165	223	586	5	189	600	325	262	100	689	354	665	673

3.9. ábra. Példa RSA-val történő rejtjelezésre.

A fenti tétel azt mondja, hogy a Miller–Rabin-teszt a prímszámproblémára Monte-Carlo-algoritmus. A bizonyítást itt csak vázoljuk. Megmutatjuk, hogy a Miller–Rabin-teszt `PRIMES`-t egyoldalú hibavalószínűséggel fogadja el: ha a (binárisan előállított) n bemenő érték prímszám, akkor az algoritmus nem válaszolhatja tévesen azt, hogy n nem prímszám. Hogy ellentmondásra jussunk, tegyük fel, hogy n prím, de a Miller–Rabin-teszt azzal a kimenettel áll le, hogy „n nem prímszám”. Következésképpen $z^m \not\equiv 1 \bmod n$ áll fenn. Mivel x-et a **for** ciklus minden lefutásakor négyzetre emeljük, modulo n a z^m, z^{2m}, …, $z^{2^{k-1}m}$ egymás utáni értékeket vizsgáljuk. Ezen értékek egyikénél sem válaszolja azt az algoritmus, hogy n prím. Ebből következik, hogy $z^{2^j m} \not\equiv -1 \bmod n$ minden j, $0 \le j \le k-1$ esetén. Mivel $n-1 = 2^k m$, a kis-Fermat-tételből $z^{2^k m} \equiv 1 \bmod n$ következik (lásd a 3.5. következményt). Ezért $z^{2^{k-1}m}$ 1 négyzetgyöke modulo n. Mivel n prím, 1-nek modulo n csak két négyzetgyöke van, nevezetesen a $\pm 1 \bmod n$, (lásd a 3.3-2. gyakorlatot). $z^{2^{k-1}m} \not\equiv -1 \bmod n$ miatt tehát $z^{2^{k-1}m} \equiv 1 \bmod n$ teljesül. De ekkor megint $z^{2^{k-2}m}$ 1-nek négyzetgyöke modulo n. Mint fent, ebből megint $z^{2^{k-2}m} \equiv 1 \bmod n$ következik. Ezt a érvet ismételten alkalmazva végül azt kapjuk, hogy $z^m \equiv 1 \bmod n$, ami ellentmondás. Következésképpen a Miller–Rabin-teszt minden prímszám esetén helyes választ ad. Ha n nem prímszám, akkor megmutatható, hogy a Miller–Rabin-teszt valószínűsége nem lépi túl az $1/4$ küszöböt. Ismételt független tesztfuttatásokkal azt kapjuk, – természetesen a futási idő kárára, ami mindamellett $\log n$-ben polinomiális marad – hogy a hibavalószínűség tetszőlegesen közel kerül nullához.

3.7. példa. *RSA.* Bob a $p = 67$ és $q = 11$ prímszámokat választja. Így $n = 67 \cdot 11 = 737$ és $\varphi(n) = (p-1)(q-1) = 66 \cdot 10 = 660$. Ha most Bob a $\varphi(n) = 660$-hoz a lehető legkisebb kitevőt választja, ami $e = 7$, akkor az ő nyilvános kulcsa az $(n, e) = (737, 7)$ pár. A kiterjesztett euklideszi algoritmus Bobnak a $d = 283$ titkos kulcsot szolgáltatja, és fennáll $e \cdot d = 7 \cdot 283 = 1981 \equiv 1 \bmod 660$ (lásd a 3.3-3. gyakorlatot). Amint a 3.1. alfejezetben is tettük, a $\Sigma = \{\mathrm{A}, \mathrm{B}, \ldots, \mathrm{Z}\}$ ábécét a $\mathbb{Z}_{26} = \{0, 1, \ldots, 25\}$ halmazzal azonosítjuk. Az üzenetek a Σ fölötti szavak, és azonos hosszúságú blokkonként 26-os számrendszerben előállított számokként kódoljuk. Példánkban a blokkhossz $\ell = \lfloor \log_{26} n \rfloor = \lfloor \log_{26} 737 \rfloor = 2$.

Egy $b = b_1 b_2 \cdots b_\ell$ blokkot, amelynek a hossza ℓ, és $b_i \in \mathbb{Z}_{26}$, az $m_b = \sum_{i=1}^{\ell} b_i \cdot 26^{\ell-i}$ szám ábrázolja. Az $\ell = \lfloor \log_{26} n \rfloor$ blokkhossz definíció következtében:

$$0 \le m_b \le 25 \cdot \sum_{i=1}^{\ell} 26^{\ell-i} = 26^\ell - 1 < n.$$

Az RSA rejtjelező függvénnyel a b blokkot, illetve a megfelelő m_b számot $c_b = (m_b)^e \bmod n$ szerint sifrírozzuk. A b blokknak megfelelő rejtett szöveg ekkor $c_b = c_0 c_1 \ldots c_\ell$, ahol $c_i \in \mathbb{Z}_{26}$. Az RSA tehát az ℓ hosszúságú blokkokat injektíven képezi le $\ell + 1$ hosszúságú blokkokra. A 3.9. ábrán láthatjuk, hogy egy 34 hosszú üzenetet 17 darab 2 hosszú blokkba tördeltünk és az egyes blokkokat számokként rejtjeleztük. Például az első blokkot, amely „RS”, számmá transzformáltuk a következőképpen: „R”-nek a 17 és „S”-nek a 18 felel meg, és $17 \cdot 26^1 + 18 \cdot 26^0 = 442 + 18 = 460$.

A kapott c_b számot megint előállíthatjuk 26-os számrendszerben és $\ell + 1$ lehet a hossz: $c_b = \sum_{i=0}^{\ell} c_i \cdot 26^{\ell-i}$, ahol $c_i \in \mathbb{Z}_{26}$ (lásd a 3.3-3. gyakorlatot is). Így az első blokk $697 = 676 + 21 =$

Lépés	Aliz	Erik	Bob
1	A következő értékeket választja: $n = pq$, (n, e) nyilvános kulcs és d titkos kulcs, ahogyan Bob tette a 3.8. ábrán lévő RSA-protokollban		
2		$(n, e) \Rightarrow$	
3	Aláírja az m üzenetet: $\text{sig}_A(m) = m^d \bmod n$		
4		$(m, \text{sig}_A(m)) \Rightarrow$	
5			Ellenőrzi Aliz aláírását $m \equiv (\text{sig}_A(m))^e \bmod n$ segítségével

3.10. ábra. Digitális aláírás RSA-val.

$1 \cdot 26^2 + 0 \cdot 26^1 + 21 \cdot 26^0$, a rejtjelezett szövegben „BAV" lett belőle.

A visszafejtés is blokkonként történik. Ahhoz, hogy az első blokkot a $d = 283$ titkos kulccsal visszafejtsük, kiszámoljuk a következőt: $697^{283} \bmod 737$, ismét gyors-hatványozással, a MODULÁRIS-HATVÁNYOZÓ algoritmus segítségével. Azért, hogy a számok ne legyenek nagyon nagyok, ajánlatos minden szorzásnál modulo $n = 737$ redukálni a kapott értékeket. A kitevő bináris kifejtése $283 = 2^0 + 2^1 + 2^3 + 2^4 + 2^8$, és – amint kívántuk – adódik a következő:

$$697^{283} \equiv 697^{2^0} \cdot 697^{2^1} \cdot 697^{2^3} \cdot 697^{2^4} \cdot 697^{2^8} \equiv 697 \cdot 126 \cdot 9 \cdot 81 \cdot 15 \equiv 460 \bmod 737.$$

3.3.2. Digitális aláírás RSA segítségével

A 3.8. ábrán szereplő RSA nyilvános kulcsú kriptorendszer módosítható úgy, hogy protokollként digitális aláírásra alkalmas legyen. Ilyet mutattunk be a 3.10. ábrában. Meggyőződhetünk arról, hogy a protokoll működik (lásd a 3.3-4. gyakorlatot). Ez a protokoll érzékeny az olyan támadásra, amelynél a támadó maga meg tudja választani a rejtjelezendő nyílt szöveget (***választott nyílt szövegű támadás***.

3.3.3. Az RSA biztonsága és lehetséges támadások az RSA ellen

Már említettük, hogy az RSA biztonsága döntő mértékben attól függ, hogy nagy számok nem könnyen faktorizálhatók. Mivel elszánt keresés ellenére sem sikerült ezidáig hatékony faktorizáló algoritmust találni, azt gyanítjuk, hogy ilyen algoritmus nincs, vagyis a faktorizálási probléma nehéz. Erre a sejtésre azonban még nem született bizonyítás. Önmagában ennek a sejtésnek a bizonyításából nem következne, hogy az RSA-rendszer biztonságos. Tudjuk, hogy az RSA feltörése legfeljebb olyan nehéz, mint a faktorizálási probléma, azt azonban nem tudjuk, hogy ugyanolyan nehéz-e. Elképzelhető az is, hogy az RSA könnyen feltörhető n faktorizálása nélkül is.

Az RSA-rendszer elleni lehetséges támadások listája helyett, amely különben is hiányos lenne, utalunk a további hivatkozásokat tartalmazó idevágó irodalomra, valamint a 3-4. feladatra. Mindegyik eddig ismert RSA elleni támadásra van megfelelő ellenintézkedés, amelyek meghiúsítják, vagy hatástalanítják a támadást, tehát a támadás sikerességének a valószínűsége elhanyagolhatóan kicsivé tehető. Főként a p és q prímszámokat, az n mo-

Lépés	Aliz	Erik	Bob
1	véletlenszerűen választ két nagy számot, x-et és y-t, x-et titokban tartja, és kiszámolja az $x\sigma y$ értékét		
2		$(y, x\sigma y) \Rightarrow$	
3			véletlenszerűen választ egy z nagy számot, z-t titokban tartja, és kiszámolja az $y\sigma z$ értéket
4		$\Leftarrow y\sigma z$	
5	kiszámolja a $k_A = x\sigma(y\sigma z)$ értéket		kiszámolja a $k_B = (x\sigma y)\sigma z$ értéket

3.11. ábra. A kulcscserére vonatkozó, σ-n alapuló Rivest–Sherman-protokoll.

dulust, az e kitevőt és a d titkos kulcsot kell megfelelő gondossággal megválasztani.

Mivel az RSA elleni ***faktorizálási támadás*** különösen központi szerepet játszik, bemutatunk egy egyszerű ilyen támadást, amely John Pollard $(p-1)$-módszerén alapszik. A $(p-1)$-módszer p prímfaktorú összetett n szám esetén akkor működik, ha $p-1$ prímfaktorai kicsik. Ekkor ugyanis meg lehet határozni $p-1$ egyik ν többszörösét anélkül, hogy p-t ismernénk, és alkalmazhatjuk a kis Fermat-tételt (lásd a 3.5. következményt), amely szerint $a^\nu \equiv 1 \bmod p$ minden a egész szám esetén fennáll, amely p-hez relatív prím. Következésképpen p egy osztója $a^\nu - 1$-nek. Ha n nem osztója $a^\nu - 1$-nek, akkor $\mathrm{lnko}(a^\nu - 1, n)$ valódi osztója n-nek, s így n-et faktorizáltuk.

Hogyan lehet meghatározni $p-1$ egy ν többszörösét? Pollard $(p-1)$-eljárása ν számára azt a számot alkalmazza jelöltként, amely egy választott S korlát alatti összes prímszámhatványok szorzataként áll elő:

$$\nu = \prod_{q \text{ prím}, q^k \le S < q^{k+1}} q^k \,.$$

Ha $p-1$ mindegyik prímhatvány osztója kisebb S-nél, akkor ν többszöröse $p-1$-nek. Az algoritmus kiszámítja $\mathrm{lnko}(a^\nu - 1, n)$ értékét egy alkalmas a alappal. Ha eközben nem sikerül n egyetlen valódi osztóját sem meghatározni, akkor az algoritmus újra indul egy új $S' > S$ értékkel.

Más faktorizálási módszer többek között a ***négyzetes szita***. Ez a következő egyszerű ötleten alapul. Az n szám faktorizálásához egy bizonyos szitával olyan a és b számokat keresünk, amelyekre:

$$a^2 \equiv b^2 \bmod n \quad \text{és} \quad a \not\equiv \pm b \bmod n \,. \tag{3.10}$$

Következésképpen n osztja az $a^2 - b^2 = (a-b)(a+b)$ számot, de nem osztja sem $a-b$-t, sem $a+b$-t. Így $\mathrm{lnko}(a-b, n)$ egy nem triviális tényezője n-nek. A négyzetes szita mellett vannak más szitamódszerek is, amelyek ettől abban a módszerben különböznek, ahogyan a (3.10)-et kielégítő a és b számokat meghatározzák. Egy példa erre az „általános számtest-szita”.

Gyakorlatok

3.3-1.★ Bizonyítsuk be a 3.13. tételt. *Útmutatás.* A kis Fermat-tétel 3.5. következményére támaszkodva mutassuk meg, hogy $(m^e)^d \equiv m \bmod p$ és $(m^e)^d \equiv m \bmod q$. Mivel p és q prímszámok, $p \neq q$ és $n = pq$, a kínai maradéktételből következik az $(m^e)^d \equiv m \bmod n$ állítás.

3.3-2.⋆ A 3.15. tétel bizonyításának vázlatában felhasználtuk azt, hogy egy n prímszám esetén 1-nek csak két négyzetgyöke van modulo n, nevezetesen a ± 1 mod n. Bizonyítsuk be ezt. *Útmutatás.* Használjuk fel azt, hogy r pontosan akkor négyzetgyöke 1-nek modulo n, ha n az $(r-1)(r+1)$ számot osztja.

3.3-3. *a.* Mutassuk meg, hogy a 3.7. példában szereplő $\varphi(n) = 660$ és $e = 7$ értékekre a kiterjesztett euklideszi algoritmus segítségével valóban a $d = 283$ titkos kulcs adódik, amely 7 mod 660 inverze.

b. Kódoljuk a 3.7. példában a 3.9. ábrabeli nyílt szöveget a $\Sigma = \{A, B, \ldots, Z\}$ ábécé betűivel mind a 17 blokk esetén.

c. Fejtsük vissza a 3.9. ábrán látható rejtett szövegnek mind a 17 blokkját, és mutassuk meg, hogy az eredeti nyílt szöveget kapjuk vissza.

3.3-4. Mutassuk meg, hogy a 3.10. ábrában szereplő digitális aláírásra vonatkozó RSA protokoll működik.

3.4. Rivest, Rabi és Sherman protokolljai

Rivest, Rabi és Sherman javasoltak kulcscserére és digitális aláírásra protokollokat. A 3.11. ábra kulcscsere protokollja Rivest-től és Sherman tól ered. Könnyen módosítható úgy, hogy digitális aláírásra alkalmas legyen (lásd 3.4-1. gyakorlatot).

Rivest és Sherman protokollja egy ***totális, erősen neminvertálható, asszociatív egyirányú függvényen*** alapszik. Ezen a következőt értjük. Egy totális (tehát mindenhol definiált) σ függvény, amelyik $\mathbb{N} \times \mathbb{N}$-ből $\mathbb{N}$-be képez, pontosan akkor ***asszociatív***, ha $(x\sigma y)\sigma z = x\sigma(y\sigma z)$ minden $x, y, z \in \mathbb{N}$ esetén teljesül, ahol a $\sigma(x, y)$ jelölés helyett az $x\sigma y$ jelölést alkalmaztuk. Ebből a tulajdonságból következik, hogy a protokoll működik, mert $k_A = x\sigma(y\sigma z) = (x\sigma y)\sigma z = k_B$ miatt Aliz és Bob valójában ugyanazt a kulcsot számítják ki.

Az erős neminvertálhatóságot itt nem kívánjuk formálisan definiálni. Tájékoztatásul annyit mondunk, hogy σ-t ***erősen neminvertálhatónak*** nevezzük, ha σ nem csupán egyirányú függvény, hanem akkor sem invertálható hatékonyan, ha a függvényérték mellett ehhez a függvényértékhez tartozó két argumentum egyikét is ismerjük. Ez a tulajdonság megakadályozza azt, hogy Erik y és $x\sigma y$, illetve $y\sigma z$ ismeretében a titkos x, illetve z számokat ki tudja számolni, amelynek a segítségével a $k_A = k_B$ kulcsot könnyen meg tudná határozni.

Gyakorlatok

3.4-1. Módosítsuk Rivest és Sherman 3.11. ábrán látható, kulcscserére vonatkozó protokollját úgy, hogy digitális aláírásra alkalmas protokollt kapjunk.

3.4-2. Melyik közvetlen támadás lenne a 3.11. ábrán szereplő Rivest–Sherman-protokoll ellen hatásos, és hogyan akadályozhatnánk meg? *Útmutatás.* Érveljünk a σ asszociatív egyirányú függvény „erős neminvertálhatóságának” fogalmával, amelyen a protokoll alapszik.

3.4-3. *a.* Adjunk formális definíciót az „erős neminvertálhatóság” fogalmára.

b. Adjunk formális definíciót az „asszociativitás” fogalmára ***parciális,*** $\mathbb{N} \times \mathbb{N}$-ből $\mathbb{N}$-be képező függvények számára. Egy ilyen definíciónak nem kell mindenhol érvényesnek lennie. Mi okoz gondot a következő definíció kísérletnél: „Egy (esetleg parciális) $\sigma : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ függvényt ***asszociatívnak*** nevezzük, ha $x\sigma(y\sigma z) = (x\sigma y)\sigma z$ minden olyan $x, y, z \in \mathbb{N}$ esetén érvényben van, amikor az (x, y), (y, z), $(x, y\sigma z)$ és $(x\sigma y, z)$ párok mindnyájan σ

értelmezési tartományában vannak."

Útmutatás. Ezeknek a fogalmaknak kielégítő definícióját találjuk Hemaspaandra munkájában.

3.5. Interaktív bizonyítási rendszerek és zéró ismeret

3.5.1. Interaktív bizonyítási rendszerek, Artúr–Merlin-játékok és zéró-ismeretű protokollok

Az a probléma, amelyet a Diffie–Hellman-protokoll elleni *középen állás támadásnál* a 3.2. alfejezetben említettünk, nyilvánvalóan abból adódik, hogy Bob a protokoll lejátszása előtt nem győződött meg beszélgető partnere valódi kilétéről. Bob azt hiszi, hogy Alizzal hajtja végre a protokollt, a valóságban azonban Erikkel. Másképpen megfogalmazva Aliznak az a feladata, hogy Bobot kétséget kizáróan meggyőzze személyazonosságáról. A kriptográfiának ezt a feladatát ***személyazonosításnak*** nevezzük. A digitális aláírással szemben, amely az elektronikus úton közvetített ***dokumentum,*** mint például az e-mail valódiságát hitelesíti, most azoknak a ***személyeknek*** a hitelesítéséről van szó, akik a protokollban a résztvevő felek. Ennek a fogalomnak további értelmezése is van: „személy", vagy „résztvevő" nem csak egy élő személy lehet, hanem például egy számítógép, amelyik egy másik számítógéppel automatikusan lefolytat egy protokollt.

Önmaga hitelességének bizonyítására Aliz felhasználhatna egy csupán általa ismert titkos adatot, mondjuk a PIN kódját (**P**ersonal **I**dentifaction **N**umber), vagy más információt, amit rajta kívül más nem ismer. Itt azonban van egy bökkenő. Aliz kénytelen titkát Bobnak elárulni személyazonossága valódiságának bizonyítására. Ekkor azonban az nem lesz többé már titok. Bob egy harmadik féllel lefolytatott protokoll során közölhetné mondjuk Kristóffal, hogy ő Aliz, mert már ismeri Aliz titkát. Az a kérdés tehát, hogyan bizonyíthatná az ember egy titok ismeretét anélkül, hogy magát a titkot elárulná. Pontosan erről van szó a zéró-ismeretű protokollokban. Ezek speciális interaktív bizonyítási rendszerek, amelyeket Goldwasser, Micali és Rackoff vezettek be. Tőlük függetlenül fejlesztette ki Babai és Moran az ezzel lényegében ekvivalens Artúr–Merlin-játékok tervét, amelyet most informálisan leírunk.

A hatalmas Merlin varázsló, akit egy M NP gép reprezentál, és a bizalmatlan Artúr király, akit egy A randomizált polinomidejű gép reprezentál, közösen meg akarnak oldani egy L problémát, tehát el akarják dönteni, hogy egy x bemenet L-hez tartozik-e, vagy sem. Mindegyik bemenő adatért játszanak, miközben felváltva lépnek. Merlin szándéka mindig az, hogy Artúrt meggyőzze arról, hogy a közös x bemenő szavuk L-hez tartozik, függetlenül attól, hogy ez valóban így van-e. Merlin egy lépése az „$x \in L$" (állítólagos) bizonyítására vonatkozó adat. Ezt megkapja egy $M(x, y)$ szimulációval, ahol x a bemenetet és y az eddigi lépéseket kódolja. Az y szó tehát leírja az M eddigi nemdeterminisztikus választásait, illetve A eddigi véletlen választásait.

Artúr király azonban bizalmatlan. A hatalmas varázsló állítólagos bizonyítását természetesen nem tudja közvetlenül ellenőrizni, ehhez nincs meg a számítási ereje. Kétségbe tudja azonban vonni Merlin bizonyítását, és képes egy ügyes kihívással válaszolni, amelyben a Merlin szolgáltatta bizonyítás ***véletlenszerűen választott*** részleteihez ellenőrizhető tanúsítványt követel. Hogy Artúrt kielégítse, Merlinnek lenyűgöző valószínűséggel meg kell őt győznie bizonyításának helyességéről. Artúr egyik lépése tehát abból áll, hogy $A(x, y)$

számításokat szimuláljon, ahol x megint a bemenetet, y pedig a játék eddigi lefutását írják le.

Az Artúr–Merlin-játék gondolata kifejezhető váltakozva egzisztenciális és valószínűségi kvantorokkal, ahol az előbbiek Merlin NP-számítását, az utóbbiak pedig Artúr randomizált polinomiális számításait formalizálják.[4] Ilyen módon bonyolultsági osztályok hierarchiáját definiálhatjuk, az úgynevezett Artúr–Merlin-hierarchiát. Itt az MA osztály definíciójára szorítkozunk, ez egy olyan két lépéses Artúr–Merlin-játéknak felel meg, amelynél Merlin lép először.

3.16. definíció (MA az Artúr–Merlin-hierarchiában). *Az* MA *osztály pontosan azokat az L problémákat tartalmazza, amelyekre van egy M NPTM és egy A randomizált polinomidejű Turing-gép, hogy minden x bemenetre fennáll:*

- *Ha $x \in L$, akkor van egy olyan y út $M(x)$-ben, hogy $A(x,y) \geq 3/4$ valószínűséggel elfogad, (vagyis Arthur Merlinnek az „$x \in L$”-re vonatkozó y bizonyítását nem tudja megcáfolni, és Merlin nyer).*
- *Ha $x \notin L$, akkor $M(x)$ mindegyik y útját $A(x,y) \geq 3/4$ valószínűséggel elutasítja (vagyis Artúrt Merlin „$x \in L$”-re vonatkozó hamis bizonyítása nem téveszti meg, és nyer).*

Ennek megfelelően lehet definiálni az AM, MAM, AMA, ... osztályokat (lásd a 3.5-1. gyakorlatot).

A 3/4 valószínűségi küszöb a 3.16. definícióban, amely szerint Artúr elfogad vagy elutasít, önkényesen lett megválasztva, és eleinte nem tűnik elég nagynak. Valójában lehet növelni a siker valószínűségét, és 1-hez tetszőlegesen közel hozni. Másként megfogalmazva, a definícióban alkalmazhatunk $1/2 + \varepsilon$ valószínűséget egy tetszőleges rögzített $\varepsilon > 0$ konstanssal, s így még mindig ugyanazt az osztályt kapjuk. Továbbá ismert az, hogy a lépések egy konstans száma esetén ez a hierarchia beleolvad az AM osztályba („stabilizálódik”): $\mathrm{NP} \subseteq \mathrm{MA} \subseteq \mathrm{AM} = \mathrm{AMA} = \mathrm{MAM} = \cdots$. Az a kérdés azonban még nyitott, hogy az $\mathrm{NP} \subseteq \mathrm{MA} \subseteq \mathrm{AM}$ tartalmazás valódi-e.

A fent említett ***interaktív bizonyítási rendszerek*** az Artúr–Merlin-játékok változatai. Jelentéktelen különbség a terminológiában az, hogy Merlin az „igazoló”, Artúr pedig az „ellenőrző”, és a kommunikáció nem játékként zajlik, hanem protokoll formájában. Első pillantásra a két modell között az a jelentős különbség, hogy Artúr véletlen bitje nyilvánosan – és elsősorban Merlin számára – ismert, ezzel ellentétben az ellenőrző véletlen bitje az interaktív bizonyítási rendszerekben titkos. Goldwasser és Sipser megmutatták azonban, hogy valójában lényegtelen az, hogy a véletlen bit titkos, vagy nyilvános. Tehát az Artúr–Merlin-játékok és az interaktív bizonyítási rendszerek egymással ekvivalensek.

Ha a játékban konstans számú lépés helyett polinomiálisan sokat engedünk meg – és többet a polinomiális időkorlátozás miatt nem lehet –, akkor az IP osztályt kapjuk. Definíció szerint IP tartalmazza az egész NP-t, és főként a gráfizomorfizmus-problémát. Később látni fogjuk, hogy IP tartalmaz $\mathrm{coNP} = \{\overline{L} \mid L \in \mathrm{NP}\}$-beli problémákat is, amelyekről feltesszük, hogy nincsenek NP-ben. A 4.21. tétel bizonyítása elsősorban azt mutatja meg, hogy a gráfizomorfizmus-probléma komplementere AM-ben és így IP-ben van. Shamir egyik híres eredménye azt mondja, hogy IP még PSPACE-szel is megegyezik, azon problémák osztályával, amelyek a polinomiális térben eldönthetők.

[4] Ez hasonlít a polinomiálisidő-hierarchia lépéseinek $\exists$ és $\forall$ kvantorokkal váltakozva történő jellemzéséhez, (lásd a 4.4. alfejezetet és főként a 4.16. tétel 3. részét).

Térjünk azonban vissza a hitelesség előbb vázolt problémájához, és a zéró-ismeretű protokollok fogalmához. Nézzük az alapgondolatot. Tegyük fel, hogy Artúr és Merlin egyik játékukat játsszák. Az interaktív bizonyítási rendszerek terminológiája szerint ebben az IP-protokollban Merlin nehéz bizonyítást küld Artúrnak. Hogy honnan varázsolta ezt a bizonyítást, az Merlin titka, és mivel csak ő ismeri, ezzel hitelesíteni tudja magát Artúr előtt.

Amit egyikük sem tud: a gonosz varázsló Marvin varázsital segítségével Merlin hasonmásává változott, és Artúr előtt Merlinként akarja kiadni magát. Merlin titkát azonban nem ismeri. Hasonlóképpen nincs meg neki Merlin hatalmas varázslóereje sem, mágiája nem nagyobb, mint egy közönséges randomizált polinomidejű Turing-gép számítási ereje. Ugyanolyan csekély, mint amennyire Artúr képes Marvinnak Merlin-bizonyítást találni. Ennek ellenére megkísérli, hogy a Merlin és Artúr közötti kommunikációt szimulálja. Egy IP-protokoll pontosan akkor ***zéró-ismeretű,*** ha az az információ, amely Marvin és Artúr között kicserélésre kerül, nem különbözik a Merlin és Artúr közötti kommunikációtól. Ekkor Marvin, aki Merlin titkos bizonyítását nem ismeri, természetesen nem tud róla a szimuláló protokollba semmiféle információt sem áramoltatni. Bár abban a helyzetben van, hogy az eredeti protokollt pontosan lemásolta, úgy, hogy egy független megfigyelő semmiféle különbséget nem tudna felfedezni, a protokoll semmilyen információt nem tud közölni: Ahol nincs, ott ne keress!

3.17. definíció (zéró-ismeretű protokoll). *$L \in$ IP esetén legyen M NPTM és A pedig randomizált polinomidejű Turing-gép úgy, hogy (M, A) interaktív bizonyítási rendszer L számára. Az (M, A) IP-protokoll pontosan akkor* ***zéró-ismeretű protokoll L számára,*** *ha van egy olyan $\widehat{M}$ randomizált polinomidejű Turing-gép, hogy $(\widehat{M}, A)$ az (M, A) eredeti protokollt szimulálja, és minden $x \in L$ esetén az (M, A)-beli, illetve az $(\widehat{M}, A)$-beli kommunikációt reprezentáló $(m_1, m_2, \ldots, m_k)$ és $(\widehat{m}_1, \widehat{m}_2, \ldots, \widehat{m}_k)$ sorozatoknál azonos a pénzfeldobás eloszlása.*

A fent definiált fogalmat az irodalomban úgy nevezik, hogy „***tökéletes zéró ismeret becsületes igazolóval***” (honest-verifier perfect zero-knowledge). Ez a következőket jelenti: (a) feltesszük, hogy az ellenőrző Artúr ***becsületes*** (aminek nem feltétlenül kell teljesülnie a kriptográfiai alkalmazásokban), és (b) megkívánjuk, hogy a szimuláló protokollban közölt információ ***tökéletesen*** megegyezzék az eredeti protokollban közölt információval. Az első feltétel némileg idealista, a második alkalmasint túl szigorú. Ezért vizsgálják a zéró-ismeretű protokollok más változatait (lásd a fejezet végén levő megjegyzéseket).

3.5.2. Zéró-ismeretű protokoll gráfizomorfizmusra

Most egy konkrét példát vizsgálunk. Már említettük, hogy GI NP-ben van, a komplementer probléma pedig $\overline{\text{GI}}$, AM-ben (lásd a 4.21. tétel bizonyítását). Így mindkét probléma IP-ben van. Most adunk egy zéró-ismeretű protokollt GI-re.

Bár manapság nincs hatékony eljárás GI-re, Merlin meg tudja oldani ezt a problémát, mert GI NP-ben van. Jóllehet nem szükséges, hogy ezt tegye. Megteheti, hogy egyszerűen választ egy nagy G_0 gráfot, amelynek n csúcsa van, valamint véletlenszerűen egy $\pi \in \mathfrak{S}_n$ permutációt, és előállítja a $G_1 = \pi(G_0)$ gráfot. A (G_0, G_1) párt nyilvánosságra hozza, a G_0 és G_1 közötti π izomorfizmust saját információjaként titokban tartja. A 3.12. ábra mutatja a Merlin és Artúr közötti IP-protokollt.

Természetesen Merlin nem küldheti el Artúrnak egyszerűen π-t, mert akkor elárulná a

Lépés	Merlin		Artúr
1	választ egy ρ véletlen permutációt $V(G_0)$-ra és egy $a \in \{0,1\}$ bitet, kiszámolja $H = \rho(G_a)$-t		
2		$H \Rightarrow$	
3			választ egy $b \in \{0,1\}$ véletlen bitet, majd G_b és H között követel egy izomorfizmust
4		$\Leftarrow b$	
5	kiszámolja a σ izomorfizmust $\sigma(G_b) = H$-val: ha $b = a$, akkor $\sigma = \rho$; ha $0 = b \neq a = 1$, akkor $\sigma = \pi\rho$; ha $1 = b \neq a = 0$, akkor $\sigma = \pi^{-1}\rho$.		
6		$\sigma \Rightarrow$	
7			ellenőrzi, hogy $\sigma(G_b) = H$, és akkor fogadja el, ha ez teljesül

3.12. ábra. Goldreich, Micali és Wigderson GI-re vonatkozó zéró-ismeretű protokollja.

titkát. Annak bizonyítására, hogy az adott G_0 és G_1 gráfok valóban izomorfak, Merlin választ véletlenszerűen egyenletes eloszlással egy ρ izomorfizmust és egy a bitet, és előállítja a $H = \rho(G_a)$ gráfot. Ezután elküldi Artúrnak H-t. Ő egy kihívással válaszol: küld Merlinnek egy véletlenszerűen egyenletes eloszlással választott b bitet, és követel tőle egy σ izomorfizmust G_b és H között. Ezt pontosan akkor fogadja el Artúr, ha Merlin σ-jára valóban teljesül $\sigma(G_b) = H$.

A protokoll működik, mert Merlin ismeri saját titkos π izomorfizmusát és véletlenszerűen választott ρ permutációját. Nem okoz problémát Merlinnek, hogy kiszámolja a σ izomorfizmust G_b és H között és Artúr előtt hitelesítse magát. A π titkot azonban nem árulja el. Mivel G_0 és G_1 izomorfak, Artúr 1 valószínűséggel elfogad. Itt egyáltalán nem kell vizsgálni két nem izomorf gráf esetét, mivel Merlin a protokollnak megfelelően izomorf G_0 és G_1 gráfokat választ (lásd a 4.21. tétel bizonyítását is).

Tegyük fel, hogy Marvin szeretné magát Artúr előtt Merlinként kiadni. Ismeri a G_0 és G_1 gráfokat, de nem ismeri a titkos π izomorfizmust. Mégis szeretné színlelni, hogy ismeri π-t. Ha az Artúr által választott b bit véletlenül megegyezik az a bittel, amelyet Marvin ***előre*** határozott meg, akkor ő nyer. Ha azonban $b \neq a$, akkor a $\sigma = \pi\rho$ vagy $\sigma = \pi^{-1}\rho$ kiszámítása igényli π ismeretét. Mivel GI egy randomizált polinomidejű Turing-gép számára túl nehéz, és hatékonyan nem számítható ki, Marvin elég nagy G_0 és G_1 gráfok esetén nem tudja meghatározni a π izomorfizmust. Nem ismerve a π-t, csak találgathat. Az ő esélye arra, hogy véletlenül elcsípjen egy b bitet, amelyre $b = a$, legfeljebb $1/2$. Természetesen Marvin mindig találgathat, és így az ő sikervalószínűsége pontosan $1/2$. Ha Artúr azt kívánja, hogy r független körben végezzék el a protokollt, akkor a csalás valószínűsége a 2^{-r} értékre szorítható le. Ez már $r = 20$ esetén is elenyészően csekély: Marvin sikervalószínűsége kisebb, mint egy a millióhoz.

Meg kell még mutatnunk, hogy a 3.12. ábrán szereplő protokoll valóban zéró-ismeretű protokoll. A 3.13. ábrán látható egy szimuláló protokoll Marvinnal, aki Merlin π titkát nem ismeri, de azt színleli, hogy ismeri. Az az információ, amelyik a protokoll egy lefutása alatt kicserélésre kerül, a (H, b, σ) hármasként adható meg. Ha Marvin véletlenszerűen választ egy a bitet, és $a = b$, akkor egyszerűen elküldi $\sigma = \rho$-t és nyer: Artúr, vagy bárki más

Lépés	Marvin		Artúr
1	választ egy ρ véletlen permutációt $V(G_0)$-ra, és egy $a \in \{0,1\}$ bitet, kiszámolja $H = \rho(G_a)$-t		
2		$H \Rightarrow$	
3			választ egy $b \in \{0,1\}$ véletlen bitet, és követel egy izomorfizmust G_b és H között
4		$\Leftarrow b$	
5	ha $b \neq a$, akkor $\widehat{M}$ kitöröl minden ebben a körben eddig közvetített információt és ismétel; ha $b = a$, akkor $\widehat{M}$ elküldi $\sigma = \rho$-t		
6		$\sigma \Rightarrow$	
7			$b = a$ esetén $\sigma(G_b) = H$, ezért Artúr elfogadja Marvin hamis személyazonosságát

3.13. ábra. A GI-re vonatkozó zéró-ismeretű protokoll szimulációja π ismerete nélkül.

független megfigyelő nem tud semmilyen szabálytalanságot felfedezni. Másrészt, ha $a \neq b$, Marvin csalása lelepleződik. Ez azonban nem okoz gondot az álnok varázslónak: egyszerűen kitörli a szimuláló protokollból ezt a kört és újra próbálkozik. Így (H, b, σ) formájú hármasok sorozatát tudja előállítani, amelyek a Merlin és Artúr közötti eredeti protokoll hármasaiból álló megfelelő sorozataitól megkülönböztethetetlenek. Így Goldreich, Micali és Wigderson GI protokollja zéró-ismeretű protokoll.

Gyakorlatok

3.5-1. Definiáljuk az Artúr–Merlin-hierarchia AM, MAM, AMA, ... osztályait a 3.16. definíció MA osztályához hasonlóan.

3.5-2. Melyik osztályt kapjuk egy olyan Artúr–Merlin-játékban, amelyik csak egy lépésből áll, amelyet Merlin, illetve amelyet Artúr végez?

3.5-3. Folytassuk le a 3.12. ábrán látható zéró-ismeretű protokollt a $G_0 = G$ és a $G_1 = H$ gráfok, valamint a G és H közötti $\pi = \left(\begin{smallmatrix}1&2&3&4&5\\3&4&1&5&2\end{smallmatrix}\right)$ izomorfizmus esetén, ahol G és H a 3.1.3. pontban a 3.4. példabeli gráfok. Játsszuk le ezt az Artúr–Merlin-játékot egy magunk választotta ρ izomorfizmussal $a \in \{0,1\}$ és $b \in \{0,1\}$ mindegyik változatára. Ismételjük meg ezt a játékot egy számunkra ismeretlen ρ izomorfizmussal, amelyet valaki más választott.

3.5-4. Módosítsuk a 3.12. ábra protokollját úgy, hogy Marvin csalásvalószínűsége a 2^{-10} érték alá kerüljön. Adjuk meg ennek a valószínűségnek egy formálisan helyes elemzését.

Feladatok

3-1. Aritmetika a $\mathbb{Z}_k$ maradékosztálygyűrűben

Legyenek $k \in \mathbb{N}$ és $x, y, z \in \mathbb{Z}$. Azt mondjuk, hogy *x kongruens y-nal modulo k* (röviden: $x \equiv y \bmod k$), ha k osztja az $y - x$ különbséget.

Az $x + k\mathbb{Z} = \{y \in \mathbb{Z} \mid y \equiv x \bmod k\}$ halmazt *$x \bmod k$ vett maradékosztályának* nevezzük. Például 3 mod 7 maradékosztálya a $3 + 7\mathbb{Z} = \{3, 3 \pm 7, 3 \pm 2 \cdot 7, \ldots\} = \{3, 10, -4, 17, -11, \ldots\}$

halmaz. ***Egy*** x mod k ***maradékosztály reprezentánsa*** legyen mindig a legkisebb természetes szám $x + k\mathbb{Z}$-ben; például 3 reprezentálja a 3 mod 7 maradékosztályt. Az összes mod k vett maradékosztály reprezentánsa $\mathbb{Z}_k = \{0, 1, \ldots, k-1\}$. $\mathbb{Z}_k$-n definiálunk ***összeadást*** az $(x + k\mathbb{Z}) + (y + k\mathbb{Z}) = (x + y) + k\mathbb{Z}$ és ***szorzást*** az $(x + k\mathbb{Z}) \cdot (y + k\mathbb{Z}) = (x \cdot y) + k\mathbb{Z}$ szabályok segítségével. A modulo 7 vett aritmetikában például $(3+7\mathbb{Z})+(6+7\mathbb{Z}) = (3+6)+7\mathbb{Z} = 2+7\mathbb{Z}$ és $(3 + 7\mathbb{Z}) \cdot (4 + 7\mathbb{Z}) = (3 \cdot 4) + 7\mathbb{Z} = 5 + 7\mathbb{Z}$.

Bizonyítsuk be, hogy modulo k aritmetikát alkalmazva:

a. $(\mathbb{Z}_k, +, \cdot)$ kommutatív egységelemes gyűrű;

b. A 3.3. példában definiált $\mathbb{Z}_k^*$ halmaz multiplikatív csoport;

c. $(\mathbb{Z}_p, +, \cdot)$ minden p prímszám esetén test. Milyen struktúra $(\mathbb{Z}_p^* \cup \{0\}, +, \cdot)$, ha p prímszám?

d. Bizonyítsuk be, hogy egy csoport semleges eleme, valamint minden csoportelem inverze egyértelműen meghatározott.

e. Mutassuk meg, hogy egy kommutatív egységelemes $\Re$ gyűrű invertálható elemei csoportot alkotnak, az $\Re$ úgynevezett ***egységcsoportját***. Mi a $\mathbb{Z}_k$ gyűrű egységcsoportja?

f. Határozzuk meg a $\mathbb{Z}_k$ maradékosztálygyűrű nullosztóit. Mutassuk meg, hogy $\mathbb{Z}_k$-ban nincs nullosztó, ha k prímszám.

3-2. Faizomorfizmus

Speciális gráfosztályok esetén, például a fák osztályában, a `GI` probléma hatékonyan megoldható. Egy (irányítatlan) ***fa*** összefüggő, körmentes gráf, ahol egy ***kör*** egymásután következő élekből áll, úgy, hogy a kiindulási csúcsba visszatérünk, és minden csúcs legfeljebb egyszer szerepel benne. Egy fa ***levelei*** az 1-fokú csúcsok. Tervezzünk hatékony algoritmust a ***faizomorfizmusra***, amely P-ben van. Ezt a problémát így definiáljuk:

$$\texttt{TI} \quad = \quad \{(G, H) \mid G \text{ és } H \text{ izomorf fák}\} .$$

Útmutatás. Jelöljük meg fokozatosan az adott fák csúcsait megfelelő számsorozatokkal, és mindegyik lépésben hasonlítsuk össze a kapott jelölő sorozatokat. Kezdjük a levelekkel és lépésről lépésre haladjunk a fa belseje felé, amíg az összes csúcsot meg nem jelöltük.

3-3. Determináns számítása

Írjuk meg egy olyan algoritmus pszeudokódját, amelyik egy mátrix determinánsát hatékonyan kiszámítja. Ültessük át az algoritmust egy tetszőleges programnyelvre. Ki lehet-e hatékonyan számítani egy mátrix inverzét?

3-4. Alacsony kitevőjű támadás

a. A 3.8. ábrán látható RSA-rendszerben az $e = 3$ kitevő hatékonysági okok miatt népszerűségnek örvend. Ez azonban veszélyes lehet. Tegyük fel, hogy Aliz, Bob és Kristóf ugyanazt az m üzenetet ugyanazzal a nyilvános $e = 3$ kitevővel, de esetleg különböző n_A, n_B és n_C modulusokkal rejtjelezik. Erik elcsípi a három keletkező rejtett szöveget: $c_i = m^3 \bmod n_i$, ahol $i \in \{A, B, C\}$. Ezután Erik az m üzenetet könnyen vissza tudja fejteni. Hogyan?

Útmutatás. Erik ismeri a kínai maradéktételt. Egy javasolt érték a kitevő számára $e =$

$2^{16}+1$, amelynek bináris előállításában csak két egyes van, s így a MODULÁRIS-HATVÁNYOZÓ algoritmus nagyon gyorsan feldolgozza.

b. A fent leírt támadásokat ki lehet terjeszteni k darab olyan rejtett szövegre, amelyek egymással kapcsolatban vannak. Legyenek ismertek valamilyen a_i és b_i $1 \leq i \leq k$ értékek, és tegyük fel, hogy k darab $c_i = (a_i m + b_i)^e \bmod n_i$ üzenetet küldtek el és mindegyiket el is fogták, ahol $k > e(e+1)/2$ és $\min(n_i) > 2^{e^2}$. Hogyan tudja egy támadó kideríteni, hogy mi az eredeti m üzenet?

Útmutatás. Az úgynevezett rácsredukciós technikával (lásd például Micciancio és Goldwasser cikkét).

c. Hogyan lehet ezt a támadást elhárítani?

Megjegyzések a fejezethez

Diffie és Hellman nyilvános kulcsú kriptográfiáról szóló munkája [14]. Az angol, francia, német és finn nyelvű szövegek betűinek gyakoriságáról [57]-ban olvashatunk. Charles Babbageról Singh ír a könyvében [66]. Claude Shannon korszakalkotó munkája a tökéletes titkosságú kriptorendszerekről [65]. Ron Rivest, Adi Shamir és Leonard Adleman [52] munkájukban írták le az első ismert nyilvános kulcsú kriptorendszert. Agrawal, Kayal és Saxena[1] munkájukban publikálták azt az eredményt, hogy a PRIMES = {bin(n) | n prím} prímszámprobléma a P bonyolultsági osztályba tartozik.

John Pollard $(p-1)$-módszeréről [48]-ben olvashatunk. Rabin és Miller *véletlen prímtesztje* a [44, 50] cikkekben, Solovay és Strassen prímtesztje pedig a [67] cikkben található meg. ***Választott szövegű támadásról*** olvashatunk a [54] dolgozatban. Az RSA-rendszer elleni lehetséges támadásokról a további utalásokat tartalmazó [6]-ben lehet olvasni. A négyzetes szita leírását megtalálhatjuk például [68]-ben, az „általános számtest-szita” leírását pedig az [39] cikkben.

Stinsonnál [68] olvashatunk arról, hogy ha p és q prímszámok, $p \neq q$ és $n = pq$, a kínai maradéktételből hogyan következik az $(m^e)^d \equiv m \mod n$ állítás. A 3.4-3. gyakorlat megoldásához kielégítő definíciót találunk a [27, 28] munkákban. A zéró-ismeretű protokollokat Goldwasser, Micali és Rackoff [22] vezették be. Az Artúr–Merlin-játékok tervét Babai és Moran [3, 4] fejlesztették ki.

Goldwasser és Sipser [23] munkájában olvashatunk arról, hogy az interaktív bizonyítási rendszerek esetén lényegtelen az, hogy a véletlen bit titkos, vagy nyilvános. A [64]cikkben található meg Shamir híres eredménye arról, hogy IP PSPACE-szel megegyezik. A GI problémára adott zéró-ismeretű protokoll Goldreich, Micali és Wigderson munkájáig [20] nyúlik vissza.

A faizomorfizmusra vonatkozó 3-2. feladathoz lásd [34]-t is. A 3-4. feladatban említett rácsredukciós technikát lásd például [43]-ben. Az itt említett támadás Johan Håstad [25] munkájára nyúlik vissza, és Don Coppersmith [11] javította.

Singh [66] könyve szép bepillantást nyújt a kriptológia fejlődésének történetébe, az ókori gyökerektől kezdve a modern rejtjelező eljárásokig. Például ott olvashatjuk, hogy a brit Állami Kommunikációs Központ (Government Communications Headquarters; GCHQ) egyik különleges egysége, a Kommunikációs-elektronikai Biztonsági Csoport (Communications Electronics Security Group; CESG) azt állítja, hogy Ellis, Cocks és Williamson nevű

munkatársai mind a 3.8. ábrán látható RSA-rendszert, mind a 3.7. ábrabeli Diffie–Hellman-protokollt hamarabb feltalálták, mint Rivest, Shamir és Adleman, illetve hamarabb, mint Diffie és Hellman, érdekes módon fordított sorrendben. Az RSA-rendszer jószerével minden, a kriptográfiáról szóló könyvben le van írva. Az RSA elleni támadásokról, mint amilyen a 3.3. alfejezetben szerepel, átfogó listát találunk például a [6, 54] áttekintő cikkekben.

Prímszámtesztek, mint amilyen a Miller–Rabin-algoritmus, valamint faktorizáló algoritmusok szintén sok könyvben megtalálhatók, például [19, 57, 68] -ben.

Az erősen neminvertálható, asszociatív egyirányú függvények fogalma, amelyen az 3.11. ábrán látható titkos kulcscserére vonatkozó protokoll alapszik, Rivest-től és Shermantól származik. Ennek a protokollnak a módosítása, amely digitális aláírásra alkalmas, Rabinak és Sherman-nak köszönhető, akik [49] munkájukban azt is bebizonyították, hogy kommutatív, asszociatív egyirányú függvény pontosan akkor létezik, ha $P \neq NP$. Kétségtelen, hogy az általuk konstruált egyirányú függvények sem nem totálisak, sem erősen nem invertálhatók, még $P \neq NP$ fennállása esetén sem. Hemaspaandra és Rothe [28] megmutatták, hogy totális, erősen nem invertálható, kommutatív, asszociatív egyirányú függvény pontosan akkor adódik, ha $P \neq NP$ teljesül (lásd a [5, 27] cikkeket is).

Az interaktív bizonyítási rendszerek és zéró-ismeretű protokollok területére vonatkozólag a legjobb és legátfogóbb forrást Goldwasser, Micali és Rackoff mutatták be [22] munkájukban, ez Goldreich könyvében [19] a 4. fejezet. Hasonlóképpen szép kimutatást találhatunk például Köbler és társai [35] és Papadimitriou [46] könyvében, valamint a [18, 21, 54] áttekintő cikkekben. Az Artúr–Merlin-játékokat főként Babai és Moran [3, 4], valamint Zachos és Heller [74] vizsgálták.

[19] részletesen foglalkozik a zéró-ismeretű protokollok olyan változataival, amelyek technikai részletekben térnek el a 3.17. definíciótól. Némi ízelítő található róluk például a [18, 21, 54] dolgozatokban.

A kriptográfiával foglalkozik magyar nyelvű könyvében Ködmön József [36], Buttyán Levente és Vajda István [8], valamint Simon Singh [66].

4. Bonyolultságelmélet

A 3. fejezetben kriptográfia módszerek és protokollok szempontjából fontos algoritmusokkal ismerkedtünk meg, mint amilyen a square-and-multiply algoritmus. Az algoritmusok fejlesztői akkor boldogok, ha sikerül egy probléma hatékony megoldását előállítaniuk. Ennek során viszont sok fontos probléma sajnos megmakacsolja magát, és konok módon ellenáll minden kísérletnek, melynek célja hatékony megoldó algoritmus kifejlesztése lenne. Ilyen problémák például a Boole-kifejezések kielégíthetőség problémája, a párosítás és a gráfizomorfizmus problémái, melyeket a fejezetben részletesen tárgyalunk.

A bonyolultságelmélet egyik legfontosabb feladata ilyen és hasonló problémák *kiszámíthatósági bonyolultság* szerinti osztályozása. Miközben az algoritmusfejlesztő akkor elégedett, ha egy, a problémáját megoldó konkrét algoritmus fejlesztése során – konkrét futási idő mellett – meg tud őrizni egy, a lehetőségekhez mérten legjobb *felső bonyolultsági korlátot*, a bonyolultságelméleti szakember egy lehető legjobb *alsó korlát* meghatározásán fáradozik. Az algoritmusok elmélete és a bonyolultságelmélet ilyen szempontból kiegészítik egymást. Ha a felső és alsó korlát egybeesik, akkor a problémát osztályoztuk.

Annak bizonyítéka, hogy egy probléma nem oldható meg hatékonyan, gyakran „negatív” hatású és egyáltalán nem kívánatos. Van azonban a bizonyításnak egy pozitív aspektusa is: a kriptográfia tárgykörében (lásd a 3. fejezetet) épp a rossz hatékonyság hiányában vagyunk érdekeltek. Annak bizonyítása, hogy egyes problémák csak rossz hatékonysággal oldhatók meg, mint például a faktorizálás probléma vagy a diszkrét logaritmus, titkosított üzenetek átvitelekor a biztonság növekedését jelentik.

A 4.1. alfejezetben lefektetjük a bonyolultságelmélet alapjait, különös tekintettel a P és NP bonyolultsági osztályok definícióira. Nagyon fontos kérdés, hogy ez a két osztály különböző-e vagy sem. Ez évtizedek óta a bonyolultságelmélet és az egész elméleti informatika központi kérdése. A mai napig sem a $P \neq NP$ sejtést, sem pedig P és NP egyenlőségét nem sikerült bizonyítani. A 4.2. alfejezet rövid bevezetőt ad az NP-teljesség elméletébe, amely ezzel a kérdéskörrel különösen intenzíven foglalkozik.

A leghíresebb NP-teljes problémák egyike a SAT, az ítéletlogika kielégíthetőség problémája: *kielégíthető-e* egy adott Boole-formula igazságértékeket behelyettesítve változóiba, vagyis igazzá teheti-e valamely behelyettesítés? A SAT probléma NP-teljessége miatt nagyon valószínűtlen, hogy SAT hatékony (és determinisztikus) megoldó algoritmussal rendelkezzen. A 4.3. alfejezetben bemutatunk SAT-hoz egy determinisztikus és egy valószínűségi algoritmust, melyek mindketten exponenciális időben működnek. Futási idejük a *gyakorlatban fontos* bemenetméretek mellett elviselhető szinten tartható még úgy is, hogy ezek az

algoritmusok *aszimptotikusan rossz hatékonyságúak*, tehát nagyon nagy bemenetekre csillagászati méretű ráfordítást igényelnek.

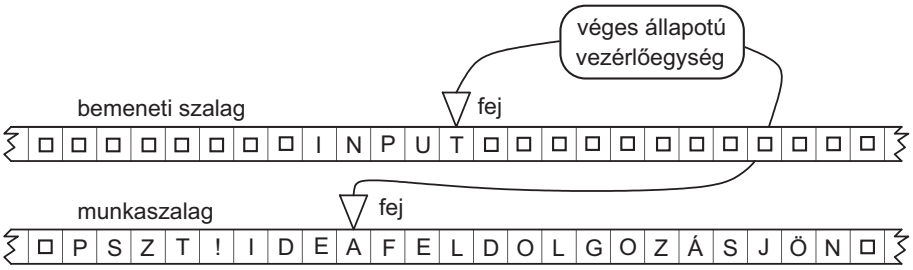
A 4.4. alfejezetben újra elővesszük a GI gráfizomorfizmus problémát, amit a 3.1.3. alfejezet 3.8. definíciójában ismertettünk, és a zéró ismeretű protokolloknál játszott szerepet. Ez a probléma azon NP-beli természetes problémák egyike, melyek valószínűleg (az elfogadhatónak tűnő P $\neq$ NP feltevés mellett) nem oldhatók meg hatékonyan, és nem is NP-teljesek. Ilyen szempontból GI különleges helyzetet élvez az NP-beli problémák halmazában. Az erre utaló jelek az úgynevezett *alsóság*-elméletből (angolul „lowness theory") erednek, amit a 4.4. alfejezetben ismertetünk. Többek között megmutatjuk, hogy GI az NP-n belüli alsó-hierarchiában helyezkedik el, ami nagyban valószínűsíti, hogy GI nem NP-teljes. Ezen kívül megmutatjuk, hogy GI az SPP bonyolultsági osztályban fekszik, és ezáltal *alsóbb* („low") az ismert valószínűségi bonyolultságosztályoknál. A formális megfogalmazást kerülve egy halmaz *alsóbb* egy C bonyolultsági osztálynál, ha mint „orákulum", mint jós semmiféle hasznos információt nem szolgáltat a C-számítások számára. A fent nevezett következmény bizonyításához, annak belátásához, hogy GI alsóbb bizonyos bonyolultsági osztályoknál, nagyon hasznosnak mutatkoznak egyes csoportelméleti algoritmusok.

4.1. Alapok

A bevezetésben említettük, hogy a bonyolultságelmélet többek között alsó korlátok meghatározásával foglalkozik. Ennek nehézsége abban rejlik, hogy nem elég a vizsgált problémát megoldó *egyetlen* konkrét algoritmus futási idejét elemezni, hanem meg kell mutatnunk, hogy a problémát megoldó *összes* elképzelhető algoritmus futási ideje *legalább akkora*, mint a felmutatott alsó korlát. Ezek közé az algoritmusok közé olyanok is tartozhatnak, melyeket talán még ki sem találtak, ebből következően első lépésként az algoritmus fogalmát formális és matematikailag pontos módon kell megragadnunk, máskülönben nem beszélhetünk az elképzelhető algoritmusok összességéről.

Az 1930-as évek óta már sok különböző formális algoritmus-modellre tettek javaslatot. Ezek a modellek ekvivalensek egymással olyan értelemben, miszerint bármelyik modell átalakítható egy tetszőlegesen választott másik ilyen modellé. Kissé pongyolán fogalmazva ez a transzformáció felfogható mint egyfajta – programnyelvek közti – fordítás. Az eddigi ismert modellek ekvivalenciája alapján az úgynevezett *Church-tézis* felteszi, hogy minden egyes algoritmusmodell pontosan a - természeténél fogva elég bizonytalanul körülhatárolható - „intuitív kiszámítások" fogalmát írja le. A bonyolultságelméletben használt szokásos algoritmusmodell az 1936-ban Alan Turing (1912–1954) által bevezetett Turing-gép. A Turing-gép a számítógépek egy nagyon egyszerű absztrakt modellje, melyet a következőkben szintaxisán és szemantikáján keresztül definiálunk. Eközben egyúttal bevezetünk két különböző kiszámíthatóságelméleti vezérgondolatot: a determinisztikusságot és a nemdeterminisztikusságot. Célszerűen először az általánosabb nem-determinisztikus Turing-gépeket írjuk le, amiből aztán speciális esetként közvetlenül adódnak a determinisztikus Turing-gépek.

Mindenek előtt a Turing-gép néhány technikai részletét és működését ismertetjük. Egy Turing-gép rendelkezik k darab mindkét irányban végtelen munkaszalaggal, melyek mezőkre vannak felosztva. A mezőkben betűk (jelek) állhatnak. Azt a tényt, hogy egy mező üres, egy speciális jellel, a $\square$-szimbólummal jelöljük. A tényleges számítás a munkasza-



4.1. ábra. Turing-gép.

lagokon történik. Egy számítás kezdetekor a bemeneti szó egy meghatározott szalagon, a bemeneti szalagon található, aminek minden bemeneti szóhoz nem használt további mezője a □ jelet tartalmazza. A számítás befejeztével az eredmény egy másik meghatározott szalagon, a kimeneti szalagon jelenik meg. Minden szalaghoz egy-egy író-olvasó fej férhet hozzá, amely a gép minden lépésében felülírhatja az aktuálisan olvasott betűt, ezután elmozdulhat egy mezőt jobbra vagy balra, illetve állva maradhat az aktuális mezőn. Eközben a gép aktuális állapota megváltozhat, melyet a belső memóriájában (***véges állapotú vezérlőegység***) feljegyez.[1] A 4.1. ábrán egy két szalagos Turing-gép látható.

4.1. definíció (a Turing-gép szintaxisa). *Egy k* ***szalagos nemdeterminisztikus Turing-gép*** *(röviden k-szalagos NTM) egy $M = (\Sigma, \Gamma, Z, \delta, z_0, \square, F)$ hetes, ahol Σ a* ***bemeneti ábécé****, Γ a munka ábécé (szalagjelek) melyre $\Sigma \subseteq \Gamma$, Z véges* ***állapothalmaz,*** *ahol $Z \cap \Gamma = \emptyset$, $\delta : Z \times \Gamma^k \to \mathfrak{P}(Z \times \Gamma^k \times \{L, R, N\}^k)$ az* átmenetfüggvény*, $z_0 \in Z$ a* ***kezdőállapot****, $\square \in \Gamma - \Sigma$ az* ***üres jel*** *és $F \subseteq Z$ a* ***végállapotok halmaza****. $\mathfrak{P}(S)$ egy S halmaz* ***hatványhalmazát*** *jelöli, tehát S összes részhalmazának halmazát.*

Minden $z, z' \in Z$, $x \in \{L, R, N\}$ és $a, b \in \Gamma$ mellett $(z', b, x) \in \delta(z, a)$ helyett a rövidített $(z, a) \mapsto (z', b, x)$ jelölést is használhatjuk. Ha az aktuális állapot z, a fej pedig egy a feliratú mezőn áll, akkor a fenti Turing-gép parancs a következő lépések végrehajtását jelenti:

- *a felülírását b-vel,*
- *egy új z' állapot felvételét és*
- *az $x \in \{L, R, N\}$-nak megfelelő fejmozgást, azaz egy mezőnyi lépést* balra *(L*, az angol „left” után), vagy egy lépést *jobbra* (R, azaz „right”), vagy a fej helyben marad az aktuális mezőn (N mint „neutral”, *semleges*).

A ***k szalagos determinisztikus Turing-gép*** *(röviden a k-szalagos DTM) speciális eseténél akkor jutunk, ha a δ átmenetfüggvény $Z \times \Gamma^k$-ról $Z \times \Gamma^k \times \{L, R, N\}^k$-ra képez.*

A $k = 1$ esetben az 1-szalagos Turing-gépekhez jutunk, melyeket NTM-ként, illetve DTM-ként rövidítünk. Minden k-szalagos NTM illetve k-szalagos DTM szimulálható megfelelő egy szalagos Turing-géppel, amelynél a számítási idő legfeljebb négyzete az eredetinek. Ésszerű lehet több szalag használata, amennyiben a hatékonyság szerepét nem hanya-

[1] Lehetséges további megkötések bevezetése is, például lehet a bemeneti szalag csak olvasható, illetve a kimeneti szalag csak írható. Hasonlóképp a technikai részletek kidolgozásakor számos további megoldás lehetséges, megkövetelhetjük, hogy meghatározott fejek csak meghatározott irányba mozdulhassanak el, vagy a szalagok csak egyik irányban legyenek végtelenek és így tovább.

golhatjuk el.

A Turing-gépek felfoghatók mint olyan elfogadó automaták, melyek nyelveket (szóhalmazokat) fogadnak el. Emellett a Turing-gépek használhatók függvények kiszámításához is.

4.2. definíció (a Turing-gép szemantikája). *Legyen $M = (\Sigma, \Gamma, Z, \delta, z_0, \square, F)$ egy NTM. **M egy konfigurációja** egy $k \in \Gamma^* Z \Gamma^*$ szó. Ha $k = \alpha z \beta$, akkor $\alpha\beta$ jelentése az aktuális szalagfelirat (az aktuális szó a fej által már meglátogatott szalagrészről), a fej β első szimbólumán áll, z pedig az M aktuális állapota.*

M konfigurációinak $\mathfrak{K}_M = \Gamma^* Z \Gamma^*$ halmazára definiálunk egy $\vdash_M$ bináris relációt, amely a $k \in \mathfrak{K}_M$ konfigurációból $k' \in \mathfrak{K}_M$ konfigurációba vezető átmenetet írja le a δ átmenetfüggvény segítségével. Legyen minden α és β Γ^*-beli szóra, ahol $\alpha = a_1 a_2 \cdots a_m$ és $\beta = b_1 b_2 \cdots b_n$, $m \geq 0$, $n \geq 1$ és minden $z \in Z$-re:

$$\alpha z \beta \vdash_M \begin{cases} a_1 a_2 \cdots a_m z' c b_2 \cdots b_n\,, & \text{ha } (z, b_1) \mapsto (z', c, N),\ m \geq 0 \text{ és } n \geq 1\,, \\ a_1 a_2 \cdots a_m c z' b_2 \cdots b_n\,, & \text{ha } (z, b_1) \mapsto (z', c, R),\ m \geq 0 \text{ és } n \geq 2, \\ a_1 a_2 \cdots a_{m-1} z' a_m c b_2 \cdots b_n, & \text{ha } (z, b_1) \mapsto (z', c, L),\ m \geq 1 \text{ és } n \geq 1. \end{cases}$$

Megvizsgálandó még két speciális eset:

1. *Ha $n = 1$ és $(z, b_1) \mapsto (z', c, R)$ (tehát M jobbra mozdul és egy $\square$-szimbólumot talál), akkor $a_1 a_2 \cdots a_m z b_1 \vdash_M a_1 a_2 \cdots a_m c z' \square$.*
2. *Ha $m = 0$ és $(z, b_1) \mapsto (z', c, L)$ (tehát M balra mozdul és egy $\square$-szimbólumot talál), akkor $z b_1 b_2 \cdots b_n \vdash_M z' \square c b_2 \cdots b_n$.*

*Az **M kezdőkonfigurációja x bemenet mellett** mindig $z_0 x$, az **M termináló konfigurációi x bemenet mellett** pedig $\alpha z \beta$ alakúak, ahol $z \in F$ és $\alpha, \beta \in \Gamma^*$.*

Legyen $\vdash_M$ reflexív, tranzitív lezártja $\vdash_M^*$. Eszerint $k, k' \in \mathfrak{K}_M$-re $k \vdash_M^* k'$ akkor és csak akkor, ha létezik egy véges $k_0, k_1, \ldots, k_t$ $\mathfrak{K}_M$-beli konfiguráció-sorozat, melyre

$$k = k_0 \vdash_M k_1 \vdash_M \cdots \vdash_M k_t = k',$$

*ahol lehetséges, hogy $k = k_0 = k_t = k'$. Ha az M kezdőkonfigurációja x bemenet mellett $k_0 = z_0 x$, akkor ezt a sorozatot **$M(x)$ véges kiszámításának** nevezzük, ha a k'-beli állapot végállapot, és azt mondjuk, hogy **M megáll az x bemenetre**. Az **M által elfogadott nyelvet** a következőképp definiáljuk:*

$$L(M) = \{x \in \Sigma^* \mid z_0 x \vdash_M^* \alpha z \beta,\ z \in F \text{ és } \alpha, \beta \in \Gamma^*\}.$$

Az M termináló állapotainak F halmazát feloszthatjuk az F_a elfogadó végállapotok és a F_r elutasító végállapotok halmazaira, ahol $F = F_a \cup F_r$ és $F_a \cap F_r = \emptyset$. Ebben az esetben $L(M) = \{x \in \Sigma^* \mid z_0 x \vdash_M^* \alpha z \beta,\ z \in F_a \text{ és } \alpha, \beta \in \Gamma^*\}$ az M által elfogadott nyelv.

*Az **M kiszámít egy $f : \Sigma^* \to \Delta^*$ szófüggvényt**, amennyiben minden $x \in \Sigma^*$-ra és minden $y \in \Delta^*$-ra*

1. *$x \in D_f \iff M$ az x bemenet mellett véges sok lépésben megáll,*
2. *minden $x \in D_f$-re: $f(x) = y \iff z_0 x \vdash_M^* z y$ egy megfelelő $z \in F$-re,*

*ahol D_f az **f értelmezési tartományát** jelöli. Azokat a szófüggvényeket, melyeket egy Turing-gép kiszámít, **kiszámítható szófüggvényeknek** nevezzük. Egy $f : \mathbb{N}^k \to \mathbb{N}$ függvény kiszámítható, ha egy $g : \{0, 1, \#\}^* \to \{0, 1\}^*$,*

$$g(bin(x_1)\#bin(x_2)\#\cdots\#bin(x_k)) = bin(f(x_1, x_2, \ldots, x_k))$$

$(z_0, a) \mapsto (z_1, \$, R)$	$(z_2, \$) \mapsto (z_2, \$, R)$	$(z_5, c) \mapsto (z_5, c, L)$
$(z_1, a) \mapsto (z_1, a, R)$	$(z_3, c) \mapsto (z_3, c, R)$	$(z_5, \$) \mapsto (z_5, \$, L)$
$(z_1, b) \mapsto (z_2, \$, R)$	$(z_3, \square) \mapsto (z_4, \square, L)$	$(z_5, b) \mapsto (z_5, b, L)$
$(z_1\$) \mapsto (z_1, \$, R)$	$(z_4, \$) \mapsto (z_4, \$, L)$	$(z_5, a) \mapsto (z_5, a, L)$
$(z_2, b) \mapsto (z_2, b, R)$	$(z_4, \square) \mapsto (z_6, \square, R)$	$(z_5, \square) \mapsto (z_0, \square, R)$
$(z_2, c) \mapsto (z_3, \$, R)$	$(z_4, c) \mapsto (z_5, c, L)$	$(z_0, \$) \mapsto (z_0, \$, R)$

4.2. ábra. Az M Turing-gép $L = \{a^n b^n c^n \mid n \geq 1\}$ nyelvhez tartozó δ parancsai.

Z	Jelentés	Teendő
z_0	kezdőállapot	új ciklust kezdeni
z_1	felismertük a-t	megkeresni a következő b-t
z_2	együttesen felismertük a, b-t	megkeresni a következő c-t
z_3	a, b, c ki van írva	megkeresni a jobb oldali szélt
z_4	jobb oldali szélre értünk	visszalépni és ellenőrizni, hogy minden a, b, c ki van-e írva
z_5	a teszt sikertelen	visszalépni a bal szélre és új ciklust kezdeni
z_6	a teszt sikerült	elfogadás

4.3. ábra. Az M állapotainak értelmezése.

által definiált g szófüggvények segítségével kiszámítható. A $\mathit{bin}(n)$ jelölés az $n \in \mathbb{N}$ kettes számrendszerbeli ábrázolását jelenti felvezető nullák nélkül, mint például $\mathit{bin}(17) = 10001$.

Mivel egy NTM minden konfigurációja több rákövetkező konfigurációval rendelkezhet, ezért természetesen adódik egy *kiszámítási fa*, melynek gyökere a kezdőkonfiguráció, levelei pedig a termináló konfigurációk. A fák speciális gráfok (lásd a 3.8. definíciót a 3.1.3. pontban, valamint a 3-2. feladatot), tehát csúcsokból és élekből állnak. Az M kiszámítási fájának csúcsai megfelelnek az M x bemenet melletti konfigurációi. Két k és k' konfiguráció között pontosan akkor vezet egy irányított él k-ból k'-be, ha $k \vdash_M k'$. Egy út az M kiszámítási fájában egy $k_0 \vdash_M k_1 \vdash_M \cdots \vdash_M k_t \vdash_M \cdots$ konfiguráció-sorozat, tehát $M(x)$ egy számításának felel meg. Egy NTM kiszámítási fája tartalmazhat végtelen utakat. Egy DTM esetében a kezdőkonfiguráció kivételével minden konfigurációt egyértelműen (*determinisztikusan*) meghatározza a megelőző konfigurációja, ezért a kiszámítási fa elfajul egy lineáris lánccá. A lánc a kezdőkonfigurációval indul, és – amennyiben a gép megáll az adott bemenetre – a termináló konfigurációval végződik, ellenkező esetben pedig a lánc végtelen.

4.1. példa. *Turing-gép.* Legyen egy L nyelv a következő: $L = \{a^n b^n c^n \mid n \geq 1\}$. Egy, az L nyelvet elfogadó Turing-gép definíciója az

$$M = (\{a, b, c\}, \{a, b, c, \$, \square\}, \{z_0, z_1, \ldots, z_6\}, \delta, z_0, \square, \{z_6\}),$$

ahol a δ átmenetfüggvénynek megfelelő Turing-gép parancsok listáját a 4.2. ábra tartalmazza. A 4.3. ábra egyenként jellemzi az M állapotait azok jelentésével, valamint a hozzájuk tartozó elvégzendő lépésekkel.

A bonyolultságelmélet szakemberei alapos emberek, szívesen teszik rendbe, foglalják

rendszerbe a fontos problémák óriási, változatos halmazát. Ezen cél érdekében osztályozzák és katalogizálják a problémákat, majd bonyolultsági osztályokba sorolják őket. Minden egyes ilyen osztályba olyan problémák kerülnek, melyek a megoldásukhoz egy meghatározott bonyolultsági mérték szerint nagyjából azonos ráfordításokat igényelnek. A legelterjedtebb bonyolultsági mértékek az *időmérték* (egy algoritmus által a megoldáshoz szükséges lépések száma) és a *tármérték* (a számítógépen szükséges tárhely). Mi most az időmérték tárgyalására szorítkozunk.

Az algoritmus által a megoldáshoz szükséges „idő” fogalmán az elvégzett lépések számát értjük, mint a bemenet méretének függvényét. Formális modellünk, a Turing-gép esetében egy lépés a Turing-gép δ átmenetfüggvényének egyszeri alkalmazását jelenti, tehát a számítás egy konfigurációjának átmenetét a következőbe. Esetünkben a hagyományos *legrosszabb eset* bonyolultsági modell vizsgálatára szorítkozunk. Ennek megfelelően egy Turing-gép időfüggvényéhez a lehetséges tetszőleges n hosszú bemenetek közül azokat a döntő hatású bemeneteket vizsgáljuk, melyek mellett a gép a legtöbb lépést igényli – tehát a legrosszabb esetet feltételezzük. Ezzel ellentétben *átlagos eset* bonyolultság esetén egy algoritmus várható futási idejét mint egy (tetszőlegesen választott méretű bemenethez adott) valószínűségi eloszlásnak megfelelő átlagot kezeljük.

A következőkben definiáljuk a determinisztikus és nemdeterminisztikus időbonyolultsági osztályokat.

4.3. definíció (determinisztikus és nemdeterminisztikus időbonyolultság).

- *Legyen M egy DTM, melyre $L(M) \subseteq \Sigma^*$, valamint legyen $x \in \Sigma^*$ egy bemenet. Definiáljuk az $M(x)$ **időfüggvényét**, amely Σ^*-ból $\mathbb{N}$-be képez a következőképp:*

$$Time_M(x) = \begin{cases} m, & \text{ha } M(x) \text{ pontosan } m+1 \text{ konfigurációval rendelkezik} \\ \text{nem definiált} & \text{egyébként} \end{cases},$$

Definiáljuk a $\mathrm{time}_M : \mathbb{N} \to \mathbb{N}$ *függvényt:*

$$time_M(n) = \begin{cases} \max_{x:|x|=n} Time_M(x), & \text{ha } Time_M(x) \text{ definiált minden} \\ & |x| = n \text{ méretű } x\text{-re} \\ \text{nem definiált} & \text{egyébként} \end{cases},$$

- *Legyen M egy NTM, melyre $L(M) \subseteq \Sigma^*$, valamint legyen $x \in \Sigma^*$ egy bemenet. Definiáljuk az $M(x)$ **időfüggvényét**, amely Σ^*-ból $\mathbb{N}$-be képez a következőképp:*

$$\mathrm{TIME}Mx = \begin{cases} \min\{Time_M(x,\alpha) \mid M \text{ az } \alpha \text{ úton elfogadja } x\text{-et}\}, & \text{ha } x \in L(M), \\ \text{nem definiált}, & \text{egyébként}. \end{cases}$$

Definiáljuk az $\mathrm{NTime}_M : \mathbb{N} \to \mathbb{N}$ *függvényt:*

$$\mathrm{TIME}Mn = \begin{cases} \max_{x:|x|=n} NTime_M(x) & \text{ha } NTime_M(x) \text{ értelmezve van minden olyan} \\ & x\text{–re}, \text{melyre}, |x| = n \\ \text{nem definiált}, & \text{egyébként}. \end{cases},$$

- *Legyen t egy $\mathbb{N}$-ből $\mathbb{N}$-be képező kiszámítható függvény. Definiáljuk a **determinisztikus***

$t(n)$	$n = 10$	$n = 20$	$n = 30$	$n = 40$	$n = 50$	$n = 60$
n	.00001 mp	.00002 mp	.00003 mp	.00004 mp	.00005 mp	.00006 mp
n^2	.0001 mp	.0004 mp	.0009 mp	.0016 mp	.0025 mp	.0036 mp
n^3	.001 mp	.008 mp	.027 mp	.064 mp	.125 mp	.256 mp
n^5	.1 mp	3.2 mp	24.3 mp	1.7 perc	5.2 perc	13.0 perc
2^n	.001 mp	1.0 mp	17.9 perc	12.7 nap	35.7 év	366 évszd
3^n	.059 mp	58 perc	6.5 év	3855 évszd	$2 \cdot 10^8$ évszd	$1.3 \cdot 10^{13}$ évszd

4.4. ábra. Néhány polinomiális és exponenciális időfüggvény összevetése.

és a ***nemdeterminisztikus időbonyolultsági osztályokat*** t ***időfüggvény mellett*** *a következőképp:*

$$\mathrm{DTIME}t = \left\{ A \;\middle|\; \begin{array}{l} A = L(M) \text{ egy } M \text{ DTM-el és} \\ \text{minden } n \in \mathbb{N}\text{-re } time_M(n) \leq t(n) \end{array} \right\},$$

$$\mathrm{NTIME}t = \left\{ A \;\middle|\; \begin{array}{l} A = L(M) \text{ egy } M \text{ NTM-el és} \\ \text{minden } n \in \mathbb{N}\text{-re } ntime_M(n) \leq t(n) \end{array} \right\}.$$

- *Legyen* $\mathbb{P}\mathrm{ol}$ *az összes polinom halmaza. Definiáljuk a P és NP bonyolultsági osztályokat a következőképp:*

$$\mathrm{P} = \bigcup_{t \in \mathbb{P}\mathrm{ol}} \mathrm{DTIME}t \qquad \text{és} \qquad \mathrm{NP} = \bigcup_{t \in \mathbb{P}\mathrm{ol}} \mathrm{NTIME}t\,.$$

DPTM-*mel illetve* NPTM-*mel jelöljük a polinomiális idejű DTM-eket illetve NTM-eket.*

Miért is olyan fontosak a P és az NP polinom idejű osztályok? Nyilvánvaló, hogy az exponenciális futási idejű algoritmusok általában bajosan nevezhetők hatékonynak. A 4.4. ábrán összehasonlítjuk néhány polinomiális és exponenciális időfüggvény növekedési ütemét, gyakorlati szempontból fontos bemeneti méretekre. Ehhez egy olyan számítógépből indulunk ki, amely másodpercenként egymillió művelet elvégzésére képes. Látható, hogy míg a polinomiális idejű algoritmusok a bemenet $n = 60$ méretéig elfogadható idő alatt előállítják az eredményt, addig például egy $t(n) = 3^n$ futási idejű algoritmus a bemenet viszonylag szerényebb, $n = 30$ méreténél 6 évnél is tovább dolgozna, az $n = 40$ méretű problémákhoz már majdnem 400 évezredre, és körülbelül $n = 50$-től pedig valóban csillagászati mértékű időre lenne szüksége.

Az utóbbi évtizedekben a számítógéptudományok és a hardver-technológia meggyőző fejlődését figyelhettük meg. Ha azonban fel is tesszük, hogy a gyorsabb és gyorsabb processzorok fejlesztési üteme megmarad az eddigi szinten, a 4.5. ábra szerint ez sem segít lényegesen az exponenciális futási idejű algoritmusok teljes végrehajtási idejének csökkentésében. Mi történne, ha egy olyan számítógépet használnánk, amely 100-szor vagy akár 1000-szer gyorsabb napjaink leggyorsabb számítógépeinél? Egy $t_i(n)$, $1 \leq i \leq 6$ függvényhez jelölje N_i azt a maximális feladatméretet, amelyet egy $t_i(n)$ időfüggvényű algoritmus egy órán belül meg tud oldani. A 4.5. ábra alapján látható, hogy még ezerszeres sebességnövekedés is csak körülbelül tízzel növeli N_5 értékét $t_5(n) = 2^n$ esetén. Ezzel szemben

$t_i(n)$	Mai számítógép	100-szor gyorsabb	1000-szer gyorsabb
$t_1(n) = n$	N_1	$100 \cdot N_1$	$1000 \cdot N_1$
$t_2(n) = n^2$	N_2	$10 \cdot N_2$	$31.6 \cdot N_2$
$t_3(n) = n^3$	N_3	$4.64 \cdot N_3$	$10 \cdot N_3$
$t_4(n) = n^5$	N_4	$2.5 \cdot N_4$	$3.98 \cdot N_4$
$t_5(n) = 2^n$	N_5	$N_5 + 6.64$	$N_5 + 9.97$
$t_6(n) = 3^n$	N_6	$N_6 + 4.19$	$N_6 + 6.29$

4.5. ábra. Mi lenne, ha a számítógépek gyorsabbak lennének?

egy n^5 időfüggvényű algoritmus ugyanekkora sebességnövekedést feltételezve hozzávetőleg négyszer nagyobb problémákat tud megoldani egy óra alatt.

A következő dogma azt a széles körben elterjedt nézetet fogalmazza meg, miszerint a polinomiális idejű algoritmusok hatékonynak tekinthetők, míg a csak exponenciális alsó korláttal rendelkező algoritmusok kimondottan rosszak, hatékonynak nem nevezhetők.

4.4. dogma. *A polinomiális idő a hatékonyság intuitív fogalmának felel meg. Az exponenciális idő a rossz hatékonyság intuitív fogalmának felel meg.*

Természetesen egy dogma mindig is csak egy dogma marad, ennélfogva 4.4.-et is kritikusan kell szemlélnünk. Egy $n^{10^{77}}$ lépésben dolgozó algoritmus például ugyan formálisan nézve egy $\mathbb{N}$ feletti, polinom lépésszámú algoritmus, de a polinom fokszáma éppen olyan nagy, mint az egész látható univerzumban lévő összes atom mostanság becsült darabszáma. Emiatt egy ilyen algoritmus egyáltalán nem hatékony, használata még a legkisebb problémaméret mellett is gyakorlatilag értelmetlen. Másrészt viszont egy $2^{0.00001 \cdot n}$ exponenciális időkorlát éppenséggel elfogadható lehet a gyakorlatban fontos problémaméretek mellett. Természetesen, ha valamikor is jelentkezik az exponenciális növekedés, a $0.00001 \cdot n$ kitevő esetében ez először csak nagyon nagy n mellett esedékes. Ez a két szélsőséges eset a valóságban mindenesetre szinte soha nem fordul elő. A természetes P-beli problémák lenyűgöző többsége megoldható olyan algoritmussal, aminek futási ideje alacsony fokú polinom, mint $O(n^2)$ vagy $O(n^3)$. Negyed-, ötöd- vagy ennél magasabb fokú polinomok csak nagyon ritkán lépnek fel.

A P osztály a 4.4. dogma szerint pontosan a hatékonyan megoldható problémákat tartalmazza. Az NP osztály sok olyan, gyakorlatban fontos problémát tartalmaz, melyekre a mai napig nem találtak hatékony megoldó algoritmust, mint például a kielégíthetőség és a gráfizomorfizmus problémái. Ezeket a 4. fejezetben részletesen vizsgáljuk. A kérdés, hogy a P és NP osztályok azonosak-e, szintén máig megoldatlan. Ez a híres P–NP probléma, amit tekinthetünk az elméleti informatika legfontosabb máig nyitott kérdésének. A kérdés különösen nagy szerepet játszik a kriptográfia területén, mivel a legtöbb ma használatos titkosítási rendszer azon a feltevésen alapszik, hogy bizonyos problémák nehezen megoldhatók. Ide tartoznak a faktorizálási probléma és a diszkrét logaritmus problémája, melyeket a 3. fejezetben vizsgálunk részletesebben. Amennyiben sikerülne a P = NP egyenlőséget bizonyítani, minden ilyen titkosítási rendszer elvesztené biztonságos tulajdonságát, és ezáltal haszontalanná válna.

A P–NP kérdés különösen nagy szerepet játszott az NP-teljesség elméletének létrejötté-

ben. Ez az elmélet módszereket biztosít a legnehezebb NP-beli problémák alsó korlátainak bizonyításához. Egy ilyen bizonyításnál egyetlen egy nehéz NP-beli problémából kell kiindulni. Számos további NP-probléma NP-nehézsége ezután egy visszavezetés segítségével következik, ami során egy problémát egy másikká alakítunk át. Érdekes módon – mivel ezek visszavezetések – léteznek olyan hatékony algoritmusok, melyekkel lehetővé válik a nehéz problémák NP-nehézségének bizonyítása. Azokat a problémákat, melyek NP-beliek és NP-nehezek, NP-teljes problémáknak nevezzük. Ezek nem tartozhatnak P-be, ezért nem is lehetnek hatékonyan megoldhatók, kivéve, ha esetleg P = NP fennáll. Az NP-teljesség elméletét a 4.2. alfejezetben mutatjuk be.

Gyakorlatok

4.1-1. Be lehet-e bizonyítani a Church-tézist? A választ indokoljuk.

4.1-2. Tekintsük a 4.1. példában szereplő M Turing-gépet.

a. Adjuk meg M konfigurációinak sorozatát $x = a^3b^3c^2$, illetve $y = a^3b^3c^3$ bemenet mellett.

b. Bizonyítsuk be M helyességét, azaz igazoljuk az $L(M) = \{a^n b^n c^n \mid n \geq 1\}$ egyenlőséget.

c. Adjunk becslést az M futási idejére.

4.1-3. Mutassuk meg, hogy a 3.8. definícióban szereplő `GI` és `GA` problémák NP-beliek.

4.2. NP-teljesség

Az NP-teljesség elmélete módszereket biztosít NP-beli problémák alsó korlátainak igazolásához. Egy NP-probléma NP-*ben teljes*, ha az osztály legnehezebb problémái közé tartozik. Ennél fogva egy X probléma NP-nehézségének belátásához NP összes problémáját össze kell hasonlítani X-el, és meg kell mutatni, hogy X legalább olyan nehéz, mint az aktuálisan vizsgált másik probléma. Két probléma bonyolultsága polinomiális idejű visszavezetések segítségével mérhető össze egymással. A sok különbözőképp definiálható visszavezethetőségi típus közül számunkra most az úgynevezett *sok-egy-visszavezethetőség* lényeges, amit $\leq_{\mathrm{m}}^{\mathrm{p}}$-vel fogunk jelölni. Mivel ebben az alfejezetben nem foglalkozunk más visszavezethetőségi osztállyal, ezért egyszerűen „visszavezethetőség”-ről beszélünk. A 4.4. alfejezetben általánosabb visszavezethetőségeket is megismerhetünk, az úgynevezett *Turing-visszavezethetőséget* és az (*erős*) *nemdeterminisztikus Turing-visszavezethetőséget.*

4.5. definíció (visszavezethetőség, NP-teljesség). *Egy A halmaz **visszavezethető** egy B halmazra (formálisan $A \leq_{\mathrm{m}}^{\mathrm{p}} B$, ha létezik egy polinomiális időben kiszámítható r függvény, melyre minden $x \in \Sigma^*$ esetén $x \in A \iff r(x) \in B$. Egy B halmaz **$\leq_{\mathrm{m}}^{\mathrm{p}}$-nehéz NP-ben,** ha minden $A \in$ NP halmazra $A \leq_{\mathrm{m}}^{\mathrm{p}} B$. Egy B halmaz **$\leq_{\mathrm{m}}^{\mathrm{p}}$-teljes NP-ben** (vagy röviden **NP-teljes**), ha B $\leq_{\mathrm{m}}^{\mathrm{p}}$-nehéz NP-ben és $B \in$ NP.*

Egy adott X probléma NP-nehézségének bizonyításhoz látszólag végtelen sok hatékony algoritmus szükséges, majd minden egyes ilyen NP-beli problémát hatékonyan vissza kell vezetni X-re. Egy alapvető kutatási eredmény szerint azonban nem szükséges *végtelen sok* ilyen visszavezetés elvégzése, elég egyetlen NP-teljes V probléma visszavezetése X-re. Mivel a $\leq_{\mathrm{m}}^{\mathrm{p}}$-visszavezethetőség tranzitív (lásd a 4.2-2. gyakorlatot) valamint V NP-nehéz, ezért az $A \leq_{\mathrm{m}}^{\mathrm{p}} V \leq_{\mathrm{m}}^{\mathrm{p}} X$ visszavezetéssel következik X NP-nehézsége. Itt A befutja az NP-beli

nyelveket.

1971-ben Stephen Cook megtalálta az ilyen NP-teljes problémák egyikét, az ítéletlogikai kifejezések kielégíthetőségének problémáját, amit röviden SAT-tal jelölünk. Sok NP-teljességi bizonyításhoz elég, ha a kielégíthetőség probléma egy úgynevezett 3-SAT megszorításából indulunk ki, ahol az adott Boole-formulák konjunktív normálformában állnak rendelkezésre, és minden klóz pontosan három literált tartalmaz. A 3-SAT is NP-teljes. A diszjunktív normálformájú Boole-kifejezések kielégíthetőségének eldöntése hatékonyan megoldható.

4.6. definíció (kielégíthetőség probléma). *A* HAMIS *és az* IGAZ *Boole-konstansokat rendre* 0*-val és* 1*-gyel jelöljük. Legyenek* $x_1, x_2, \ldots, x_m$ *Boole-változók, tehát* $x_i \in \{0,1\}$ *minden* i*-re. A változókat és azok negáltját* ***literáloknak*** *nevezzük. Egy* φ ***Boole-formula kielégíthető,*** *ha létezik* φ *változóinak olyan behelyettesítése, ami a formulát igazzá teszi. Egy* φ *Boole-formula* ***konjunktív normálformájú*** *(röviden* ***KNF****,) ha* $\varphi(x_1, x_2, \ldots, x_m) = \bigwedge_{i=1}^{n} \left(\bigvee_{j=1}^{k_i} \ell_{i,j} \right)$ *formájú, ahol az* $\ell_{i,j}$ *literálok* $\{x_1, x_2, \ldots, x_m\}$*-ből valók. A literálok* $\bigvee_{j=1}^{k_i} \ell_{i,j}$ *diszjunkcióit* φ ***klózainak*** *hívjuk. Egy* φ *Boole-formula* ***k-KNF,*** *ha* φ *KNF és* φ *minden klóza pontosan* k *literált tartalmaz. Definiáljuk a következő két problémahalmazt:*

$$\begin{aligned} \text{SAT} &= \{\varphi \mid \varphi \textit{ kielégíthető KNF Boole-formula}\}\,, \\ \texttt{3-SAT} &= \{\varphi \mid \varphi \textit{ kielégíthető 3-KNF Boole-formula}\}\,. \end{aligned}$$

4.2. példa. *Boole-kifejezések.* A következő két formula kielégíthető Boole-kifejezés (lásd még a 4.2-1. gyakorlatot):

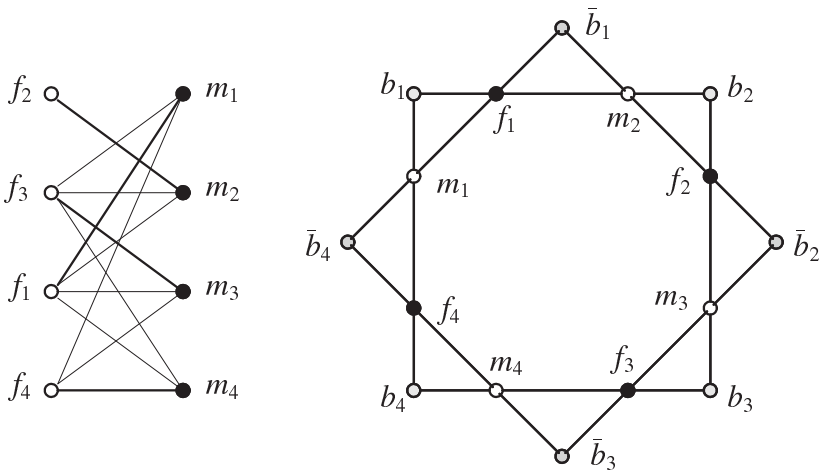
$$\begin{aligned} \varphi(w,x,y,z) &= (x \vee y \vee \neg z) \wedge (x \vee \neg y \vee \neg z) \wedge (w \vee \neg y \vee z) \wedge (\neg w \vee \neg x \vee z)\,; \\ \psi(w,x,y,z) &= (\neg w \vee x \vee \neg y \vee z) \wedge (x \vee y \vee \neg z) \wedge (\neg w \vee y \vee z) \wedge (w \vee \neg x \vee \neg z)\,. \end{aligned}$$

Itt φ egy 3-KNF formula, ezért φ 3-SAT-beli. Ezzel szemben ψ nem 3-KNF, mivel az első klóz négy literált tartalmaz, ezért ψ ugyan SAT-beli, de nem 3-SAT-beli.

A 4.7. tétel Cook korábban említett eredménye, amelyben az elsők között adott meg SAT-tal egy NP-teljes problémát. A bizonyítás alapötlete szerint kódoljuk egy tetszőleges M NPTM működését egy $\varphi_{M,x}$ Boole-formulával úgy, hogy $\varphi_{M,x}$ pontosan azon x-ekre legyen kielégíthető az x bemenettel, melyeket az M elfogad. A kielégíthetőség problémából kiinduló visszavezetésekhez sokszor célszerű, ha az adott kifejezések szigorú 3-KNF formában állnak rendelkezésre. Ez megoldható, mivel SAT visszavezethető 3-SAT-ra, ezért 3-SAT szintén NP-teljes (lásd az *Új algoritmusok* megfelelő fejezetét).

4.7. tétel (Cook). *A SAT és* 3-SAT *problémák NP-teljesek.*

Napjainkra több ezer NP-teljes problémát találtak már. Mi most a nagy választékból szemléltetési célokra a *háromdimenziós párosítás problémát* választjuk ki, és egy 3-SAT-ból való visszavezetéssel megmutatjuk NP-teljességét. A párosítási feladatoknál egymáshoz illő párokat vagy hármasokat alakítunk ki. Egy *kétdimenziós* (vagy *bipartite*) *párosítás* egymáshoz illő párok halmaza, egy *háromdimenziós* (vagy *tripartite*) *párosítás*, más néven *hármasítás* egymáshoz illő hármasok halmaza. Kétdimenziós párosítások jól szemléltethetők (irányítatlan) gráfokkal, lásd bővebben a 3.8. definíciót.



4.6. ábra. *Balra*: A házasítás probléma megoldása. *Jobbra*: Az R reláció igazságérték-komponensei.

4.8. definíció (kétdimenziós párosítás probléma). *Egy $2n$ csúccsal rendelkező G gráf **páros**, ha a csúcshalmaza két n méretű, diszjunkt V_1 és V_2 halmazra osztható, ahol mindkettő **független halmaz**, azaz sem V_1, sem pedig V_2 csúcsai nincsenek élekkel összekötve. Csak V_1 és V_2-beli csúcsok között fordulhat elő él. G **tökéletes kétdimenziós párosítása** egy olyan n csúcsú $M \subseteq E(G)$ halmaz, melynél minden két különböző $\{v,w\}$ és $\{x,y\}$ M-beli élre $(v,x \in V_1,\ w,y \in W_2)$ $v \neq x$ és $w \neq y$. A **kétdimenziós párosítás probléma** kérdése az, hogy egy adott páros gráfhoz létezik-e kétdimenziós párosítás.*

4.3. példa. (Kétdimenziós párosítás probléma.) Képzeljünk el n darab házasodni szándékozó hölgyet és n darab házasodni szándékozó férfit, akik egy páros gráf csúcsait alkotják. A $V(G)$ csúcshalmaz tehát két részre bomlik, $V_{\text{vőlegény}} = \{f_1, f_2, \ldots, f_n\}$ és $V_{\text{menyasszony}} = \{m_1, m_2, \ldots, m_n\}$, valamint $V(G) = V_{\text{menyasszony}} \cup V_{\text{vőlegény}}$ és $V_{\text{menyasszony}} \cap V_{\text{vőlegény}} = \emptyset$. A $V_{\text{vőlegény}}$-beli és a $V_{\text{menyasszony}}$-beli csúcsokat élek köthetik össze, de nincs él $V_{\text{vőlegény}}$-beli csúcsok vagy $V_{\text{menyasszony}}$-beli csúcsok között. Egy kétdimenziós párosítás akkor áll elő, ha rendezhető n darab esküvő az n menyasszony és n vőlegény között úgy, hogy (Garey és Johnson művéből idézve) „a poligámia elkerülendő, valamint mindenki elfogadható feleséget illetve férjet kapjon”. Emiatt az értelmezés miatt a kétdimenziós párosítás problémát *házasítás problémának* is nevezzük. A 4.6. ábrán (bal oldalon) látható a házasítás probléma egy megoldása, ahol a vastagon nyomtatott élek a négy új házaspárt jelölik. Ismert, hogy a házasítás probléma hatékonyan megoldható. A valós élet gyakran igazolja az állítást: *Házasodni könnyű!*

A következőkben általánosítjuk a páros gráfokat és a kétdimenziós párosításokat három dimenzióra.

4.9. definíció (háromdimenziós párosítás probléma). *Legyen U, V és W három, páronként diszjunkt, n méretű halmaz. Legyen $R \subseteq U \times V \times W$ egy hármas reláció, tehát (u,v,w) hármasok halmaza, ahol $u \in U$, $v \in V$ és $w \in W$. R **egy háromdimenziós párosítása** egy n méretű $M \subseteq R$ halmaz, ahol minden két különböző (u,v,w) és $(\hat{u},\hat{v},\hat{w})$ hármasra $u \neq \hat{u}$, $v \neq \hat{v}$ és $w \neq \hat{w}$. Ez azt jelenti, hogy egy háromdimenziós párosítás semelyik két eleme nem illeszked-*

het egymáshoz valamely koordinátájában. Definiáljuk tehát a ***háromdimenziós párosítás problémát*** *a következőképp:*

$$\texttt{3-DM} = \left\{ (R,U,V,W) \;\middle|\; \begin{array}{l} U,\ V \text{ és } W \text{ páronként diszjunkt, nem üres, azonos méretű} \\ \text{halmaz és } R \subseteq U \times V \times W \text{ egy hármas reláció, amely} \\ \text{egy } |U| \text{ méretű háromdimenziós párosítást tartalmaz} \end{array} \right\}.$$

4.4. példa. *Háromdimenziós párosítás probléma.* Kilenc hónap telt el. Egy reggel az n darab boldog házastárspárunk útra kel a kórházba. Néhány órával később n csecsemő születik, akik hangos sírás mellett rögtön jelentősen megnövelik szüleik életének bonyolultságát például azáltal, hogy elcserélik a névtáblájukat, amin rajta áll, hogy melyik szülőpárhoz tartoznak. Ez nagy zűrzavarhoz vezet a szülőszobában. Még tovább rontva a helyzeten az újdonsült apák mindegyike – talán az izgalmas pillanatoktól összezavarva és a többi nő szépségétől elcsábítva – kijelenti, hogy sosem látta azt a hölgyet, aki makacsul kitart amellett, hogy az ő gyermekét hozta épp világra. Ehelyett hűtlen módon kijelenti, hogy egy *másik* hölgy a házastársa, aki az első mellett közvetlenül balra fekszik. A káosz teljes! A szülőszoba főnővére nehéz problémával szembesül: melyik szülőpároshoz melyik csecsemő tartozik? Másképp fogalmazva n darab boldog, harmonikus és páronként diszjunkt családot kell újra összeállítania, tehát egy háromdimenziós párosítás feladatot kell megoldania n édesapa, n édesanya és n csecsemő között. Nem csoda, hogy – szemben a kétdimenziós párosítás feladat hatékony megoldhatóságával – a 3-DM probléma NP-teljes. Végül a főnővérnek $3n$ darab vérvételt és kifinomult DNS teszteket kell elvégeznie, ám ennek tárgyalása már meghaladná könyvünk terjedelmét. Ismét átfogalmazva a 3-DM NP-teljességét a valós élet tapasztalataira: *Ha jönnek a gyermekek, nagyon nehéz feladat boldog és harmonikus, diszjunkt családokban élni!*

4.10. tétel. *A* 3-DM *probléma* NP*-teljes.*

Bizonyítás. A 4.2-4. gyakorlat szerint könnyen belátható, hogy 3-DM NP-beli. A 3-DM NP-nehézségének bizonyítása mögötti intuíciót legkönnyebb akkor megérteni, ha először megnézzük, hogyan jár el a szülőszobai főnővér a háromdimenziós párosítás probléma megoldásakor. Először is mindenkit ellát a teremben egy névtáblával, olyannal, ami megbízhatóan nem fog ismét eltűnni. Feltesszük, hogy az édesanyák az $m_1, m_2, \ldots, m_n$, az édesapák az $f_1, f_2, \ldots, f_n$, a csecsemők pedig a $b_1, b_2, \ldots, b_n$ elnevezéseket kapják. Ezután a főnővér egy újabb adag $\{\bar{b}_1, \bar{b}_2, \ldots, \bar{b}_n\}$ csecsemőt állít elő, ahol minden egyes $\bar{b}_i$ a b_i-vel megegyező klón[2], azaz ugyanúgy néz ki, és a DNS-e b_i-vel azonos öröklődési információkat tartalmaz. Ezt elvégezve két kört alakít ki a $4n$ emberből. A $2n$ darab szülő egy belső kört formáz, amiben apák és anyák váltakoznak. A külső körben az n csecsemő és n klónja helyezkedik el, ugyancsak váltakozva. A két kör szomszédos személyei kapcsolatban állnak egymással úgy, ahogy a 4.6. ábra (jobb oldalon) $n = 4$ esetében mutatja: minden apa két anyával és két csecsemővel van összekötve.

A következő gondolatmenetben az i indexek modulo n értendők, a példában $n = 4$. Vizsgáljunk minden i-t, melyre[3] i modulo $n = 4$. Az f_i-edik apa az m_{i-1}-edik anyával tartja magát házasnak, és vele közös gyermekének pedig az $(i-1)$ csecsemőt. Ezzel szemben az m_i anya ahhoz ragaszkodik, hogy *ő* az f_i apa felesége, és a közös gyermekük pedig az i-edik csecsemő. Mindkét ellentmondó állítást egy-egy háromszög szemlélteti a 4.6. ábrán

[2] A csecsemőklónozás technikai kérdései és az ezzel kapcsolatban felmerülő etikai kérdések szintén túlmutatnak ezen könyv határain, ennélfogva szép csendben mellőzzük őket.

[3] A modulo n aritmetikáját a 3. fejezet végén, a 3-1. példával kapcsolatban ismertettük.

(jobb oldalon), melyek csúcsai f_i, m_{i-1} és $\bar{b}_{i-1}$, illetve m_i, f_i és b_i. A $2n$ darab háromszög mindegyike egy-egy potenciális családot képez. A főnővérnek azt kell megállapítania, hogy mely háromszögek reprezentálják az *eredeti* n darab családot, és melyek nem. Az egyetlen lehetőség n darab diszjunkt család létrehozására az, ha vagy minden háromszögbe egy b_i csecsemőt, vagy pedig minden háromszögbe egy $\bar{b}_i$ klóncsecsemőt választunk. Azáltal, hogy a $3n$ vérvételt és a fentebb említett DNS-teszteket elvégezte, a főnővér meghozhatja a helyes döntést és minden apához kijelölheti igazi feleségét és igazi gyermekét. A főnővér tehát így állítja elő az n darab eredeti családot. A fennmaradó n darab csecsemőt (és ez a főnővér módszerének szomorú oldala – és a csecsemőklónozásé úgy általában) adoptálják, vagy árvaházakban helyezik el.

A bonyolultságelmélettel foglalkozók nem nagyon értenek a DNS-tesztekhez vagy a klónozáshoz, szerencsére azonban a kielégíthetőség-problémát jól ismerik. A 3-DM probléma NP-nehézségének belátásához definiáljuk 3-SAT egy visszavezetését 3-DM-re. Legyen adott φ Boole-formula 3-KNF-ben, vagyis $\varphi(x_1, x_2, \ldots, x_\ell) = C_1 \wedge C_2 \wedge \cdots \wedge C_n$, ahol φ minden C_j klóza pontosan három literált tartalmaz. Előállítandó polinomiális időben egy (R, U, V, W) négyes 3-DM-ből, ahol $R \subseteq U \times V \times W$ egy hármas reláció a páronként diszjunkt, nem üres, azonos méretű U, V és W halmazokkal úgy, hogy

$$\varphi \text{ kielégíthető} \iff R \text{ egy } |U| \text{ méretű, } M \text{ háromdimenziós párosítást tartalmaz .} \qquad (4.1)$$

R különböző jellegű hármasokból áll, melyek mögött mindig más és más szándék rejtőzik. Azonos jellegű hármasokat elem-csoportokká, komponensekké fogunk össze. Az első ilyen komponens azokból az R-beli hármasokból áll, melyek φ formula változóinak olyan meghatározott hozzárendelését biztosítják, ami konzisztens φ összes klózára nézve. Ennek megfelelően ha ugyanaz a változó előfordul különböző klózokban, akkor minden egyes ilyen előforduláshoz ugyanazt az igazságértéket kell hozzárendelni, ennélfogva ezeket az alkotóelemeket R-ből *igazságérték-komponensnek* nevezzük.

Minden φ-beli x_i változóhoz készítünk pontosan $2n$ darab U-beli $b_1^i, b_2^i, \ldots, b_n^i$ és $\bar{b}_1^i, \bar{b}_2^i, \ldots, \bar{b}_n^i$ elemet, ahol n a φ klózainak száma. A b_j^i reprezentálja x_i, a $\bar{b}_j^i$ pedig $\neg x_i$ előfordulását φ j-edik C_j klózában. Mivel az összes literál nem fordul elő minden klózban, ezért néhány b_j^i vagy $\bar{b}_j^i$ nem feleltethető meg valamely φ-beli literál előfordulásának. Ezeken kívül minden φ-beli x_i változóhoz létrehozunk további n darab V-beli $m_1^i, m_2^i, \ldots, m_n^i$ és n darab W-beli $f_1^i, f_2^i, \ldots, f_n^i$ elemet, melyek a 4.6. ábrán (jobb oldalon) lévő belső kört alkotják, ha $n = 4$ és az ábrán nem szerepelnek a fenti indexek. Kapcsoljuk most össze az m_j^i, f_j^i és b_j^i, továbbá az f_j^i, m_{j-1}^i és $\bar{b}_{j-1}^i$ elemeket egymással, ahogy a 4.6. jobb oldali ábráján vázoltuk. Az így létrehozott komponensek háromszögei megfelelnek az R hármasainak. A belső körből m_j^i és f_j^i csak azokban a komponensekben fordulnak elő, melyek az x_i változónak felelnek meg, miközben a külső körből b_j^i és $\bar{b}_j^i$ más komponensekben is előfordulhatnak. Formálisan az X igazságérték-komponensek $X = \bigcup_{i=1}^{\ell} X_i$ alakúak, ahol $X_i = F_i \cup T_i$ minden φ-beli x_i-re, a következő két halmaz által definiálva:

$$\begin{aligned} F_i &= \{(b_j^i, m_j^i, f_j^i) \mid 1 \le j \le n\} ; \\ T_i &= \{(\bar{b}_j^i, m_j^i, f_{j+1}^i) \mid 1 \le j < n\} \cup \{(\bar{b}_n^i, m_n^i, f_1^i)\} . \end{aligned}$$

Mivel a belső kör m_j^i és f_j^i elemei csak X_i komponenseiként szerepelhetnek, ezért R minden M párosítása pontosan n darab hármast tartalmaz, vagy minden ilyen hármast F_i-ből,

vagy minden hármast T_i-ből. Az F_i és T_i közötti választás biztosítja az x_i változók megfeleltetését az *igaz* vagy a *hamis* értékeknek. Mivel X_i tartalmazza φ összes x_i előfordulását, ezért ez a választás az egész formulára nézve konzisztens, következésképp R minden egyes M párosítása a φ formula egy hozzárendelését határozza meg úgy, hogy a hozzárendelés minden x_i változója akkor és csak akkor lehet igaz, ha $M \cap X_i = T_i$.

Most hozzáillesztjük R-hez az $Y = \bigcup_{j=1}^{n} Y_j$ hármasok halmazát úgy, hogy minden Y_j a φ megfelelő C_j klózát ellenőrizze. Ennek megfelelően az Y komponenst R *kielégíthetőségi-komponensének* nevezzük. Minden C_j klózhoz készítünk két $v_j \in V$ és $w_j \in W$ elemet, melyek csak Y_j-ben fordulnak elő. Ezen kívül Y_j három további elemet tartalmaz az $\bigcup_{i=1}^{\ell} \left(\{b_j^i\} \cup \{\bar{b}_j^i\}\right)$ halmazból, melyek megfelelnek C_j három literáljának, és R más komponenseiben is előfordulhatnak. Formálisan Y_j-t φ minden C_j klózára a következő halmazzal definiáljuk:

$$Y_j \quad = \quad \{(b_j^i, v_j, w_j) \mid x_i \text{ előfordul } C_j\text{-ben}\} \cup \{(\bar{b}_j^i, v_j, w_j) \mid \neg x_i \text{ előfordul } C_j\text{-ben}\} \; .$$

Mivel a v_j és w_j $(1 \le j \le n)$ elemek egyike sem szerepelhet az R halmaz Y_j-től különböző bármelyik másik hármasában, ezért R minden M háromdimenziós párosítása pontosan egy Y_j-beli hármast tartalmaz, vagy (b_j^i, v_j, w_j)-t, vagy pedig $(\bar{b}_j^i, v_j, w_j)$-t. Továbbá M pontosan akkor tartalmaz egy Y_j-beli hármast – vagy b_j^i-t (ha x_i előfordul C_j-ben), vagy pedig $\bar{b}_j^i$-t (ha $\neg x_i$ fordul elő C_j-ben) – ha ez az elem nem fordul elő az $M \cap X_i$ hármasai között. Ez pedig pontosan akkor következik be, ha az M igazságérték-komponensével meghatározott hozzárendelés kielégíti a C_j klózt.

Mostanra U pontosan $2n\ell$ elemet tartalmaz, azonban V és W mindössze $n\ell + n$ elemű. Bővítsük további $n(\ell - 1)$ elemmel V-t és W-t, amivel a három halmaz azonos méretű lesz. Először is hozzáadjuk a $v_{n+1}, v_{n+2}, \ldots, v_{n\ell}$ elemeket V-hez, valamint a $w_{n+1}, w_{n+2}, \ldots, w_{n\ell}$ elemeket W-hez. Ezeken kívül kiegészítjük R-et hármasok következő halmazával:

$$\begin{aligned} Z \quad = \quad & \{(b_j^i, v_k, w_k) \mid 1 \le i \le \ell \text{ és } 1 \le j \le n \text{ és } n+1 \le k \le n\ell\} \cup \\ & \{(\bar{b}_j^i, v_k, w_k) \mid 1 \le i \le \ell \text{ és } 1 \le j \le n \text{ és } n+1 \le k \le n\ell\} \; . \end{aligned}$$

A vicc pedig az, hogy ha létezik $R - Z$ egy háromdimenziós párosítása, és az összes, R igazságérték- és kielégíthetőségi-komponenseivel biztosított feltétel teljesül, akkor ez a háromdimenziós párosítás U-ból pontosan $n(\ell - 1)$ elemet szabadon hagy, amit aztán egy egyértelműen meghatározott Z-beli (v_k, w_k) párral „párosíthatunk”. Az $R - Z$ háromdimenziós párosításainak ilyen bővítése R egy háromdimenziós párosítását adja. Formálisan az U, V és W halmazokat a következőképp definiáljuk:

$$\begin{aligned} U \quad &= \quad \{b_j^i \mid 1 \le i \le \ell \text{ és } 1 \le j \le n\} \cup \{\bar{b}_j^i \mid 1 \le i \le \ell \text{ és } 1 \le j \le n\} \; ; \\ V \quad &= \quad \{m_j^i \mid 1 \le i \le \ell \text{ és } 1 \le j \le n\} \cup \{v_k \mid 1 \le k \le n\ell\} \; ; \\ W \quad &= \quad \{f_j^i \mid 1 \le i \le \ell \text{ és } 1 \le j \le n\} \cup \{w_k \mid 1 \le k \le n\ell\} \; . \end{aligned}$$

Az $R \subseteq U \times V \times W$ relációt a $R = X \cup Y \cup Z$ egyenlőség határozza meg. Mivel R pontosan $2n\ell + 3n + 2n^2\ell(\ell - 1)$ hármast tartalmaz, tehát polinomiális számút a φ formula méretére nézve, és R szerkezete könnyen meghatározható φ szerkezetéből, ezért a visszavezetés polinomiális időben kiszámítható. A (4.1) állítás következik az R konstrukciója közbeni észrevételekből.

(4.1) formális bizonyítását a 4.2-5. gyakorlat keretében az Olvasóra bízzuk. ■

Gyakorlatok

4.2-1. Adjunk meg egy-egy kielégítő behelyettesítést a 4.2. példában szereplő φ és ψ Boole-formulákhoz.

4.2-2. Mutassuk meg az $\leq_{\mathrm{m}}^{\mathrm{p}}$-visszavezethetőség tranzitivitását: $(A \leq_{\mathrm{m}}^{\mathrm{p}} B \wedge B \leq_{\mathrm{m}}^{\mathrm{p}} C) \Longrightarrow A \leq_{\mathrm{m}}^{\mathrm{p}} C$.

4.2-3. Adjunk meg egy SAT $\leq_{\mathrm{m}}^{\mathrm{p}}$ 3-SAT visszavezetést. Alakítsuk át hozzá egy adott Boole-formula minden olyan klózát KNF klózzá, amely csak egy, csak kettő vagy háromnál több literált tartalmaz. Eközben a formula kielégíthetősége ne változzon.

4.2-4. Mutassuk meg, hogy a SAT és 3-DM problémák NP-beliek.

4.2-5. Lássuk be a 4.10. bizonyításban szereplő (4.1) egyenlőséget.

4.3. Az ítéletlogika kielégíthetőség-problémája

4.3.1. 3-SAT determinisztikus időbonyolultsága

A SAT kielégíthetőség probléma és annak 3-SAT megszorítása a 4.7. tétel alapján NP-teljes. Ha SAT esetleg P-beli lenne, akkor az általánosan elfogadott sejtéssel szemben P = NP rögtön következne, ezért nagyon valószínűtlennek tűnik, hogy léteznének SAT-hoz vagy 3-SAT-hoz hatékony determinisztikus algoritmusok. Akkor viszont mekkora a legjobb algoritmusok futási ideje 3-SAT-ra? Mivel a formula pontos szerkezete nyilvánvalóan befolyásolhatja a futási időt, ezért mi ebben a pontban 3-SAT problémára koncentrálunk, ahol minden klóz pontosan három literálból áll. Az itt bemutatott eredmények közvetlenül átvihetők k-SAT-ra, SAT megszorítására pontosan k literállal klózonként. 3-SAT „naiv” determinisztikus algoritmusa a következőképp működik: egy adott n-változós φ Boole-formulánál egymás után az összes lehetséges behelyettesítést kipróbáljuk, kiértékelve a formulát a mindenkori értékekkel. Ha φ valamely behelyettesítésre igaz, az algoritmus elfogad, ellenkező esetben mind a 2^n behelyettesítést eredmény nélkül kipróbálja, és az algoritmus elutasít. Ez a módszer nyilvánvalóan $O(2^n)$ idővel dolgozik. Megoldható-e a probléma gyorsabban?

Igen, megoldható gyorsabban is. De mielőtt megmutatnánk *hogyan*, a következő kérdést szeretnénk feltenni: *miért?* Mi haszna annak, ha 3-SAT felső időkorlátját $O(2^n)$ alá szorítjuk, valahová $O(c^n)$-re, ahol a c konstansra $1 < c < 2$ teljesül, ami még mindig egy exponenciális időkorlát? A haszon ott jelentkezik, hogy elérhető annak az n_0 küszöbértéknek a hátrébb tolása, ahonnan kezdve az exponenciális idő jellemző lesz, azaz ahol az algoritmus abszolút futási ideje $n \geq n_0$ méretű bemenetek mellett elviselhetetlenül nagyra nő. A 3-SAT „naiv” algoritmusának $O(2^n)$ korlátját hozzávetőleg annyira le lehet szorítani, hogy egy $O(c^n)$ algoritmussal kétszeres méretű bemenetek feldolgozása válik lehetővé azonos idő alatt, ami nagy gyakorlati jelentőséggel bír. Pontosan ez az eset áll fenn $c = \sqrt{2} \approx 1.4142$ mellett, ahol az algoritmus $O(\sqrt{2}^{2n}) = O(2^n)$ időben dolgozik (lásd még a 159. oldalon a 4.9. táblázatot).

A következőkben bemutatunk egy algoritmust 3-SAT-hoz, amely a *visszalépésen* alapul. Ez az algoritmusfejlesztési technika azon problémák megoldására alkalmas, ahol a megoldás n darab alkotóelemből tevődik össze, melyeknél több választási lehetőség adódik.

Például 3-SAT egy megoldása egy kielégítő behelyettesítés, ami n darab igazságértékéből áll, továbbá minden ilyen igazságértékhez két választási lehetőség adódik: IGAZ vagy HAMIS, illetve 1 vagy 0. Az ötlet ezután abból áll, hogy az üres megoldásból (részleges behelyettesítés, ahol a változók nincsenek kitöltve) kiindulva lépésről lépésre, a VISSZALÉPÉS eljárás rekurzív hívásával a probléma mind nagyobb részleges megoldását hozzuk létre, amíg meg nem találjuk a teljes megoldást, már amennyiben az létezik. A keletkező rekurziós fában[4] a gyökeret megjelöljük az üres megoldással, a probléma teljes megoldásai pedig a levelek szintjére kerülnek. Ha az algoritmus végrehajtása során megállapítjuk, hogy a rekurziós fa aktuális ága „halott", vagyis az addig konstruált részmegoldás semmi esetben sem egészíthető ki a probléma teljes megoldásává, akkor az eddig elért csúcs alatti részfát nyugodtan levághatjuk. A meghívandó eljárást visszaléptethetjük, próbálkozhatunk az előzőleg konstruált részmegoldás újabb folytatásával. Ennek a visszalépésnek köszönheti nevét a *visszalépés* algoritmikai elve. A rekurziós fa „halott" részeinek levágásával időt tudunk megtakarítani.

VISSZALÉPÉS-SAT(φ, β)

```
1  if β-ban φ minden változója ki van töltve
2    then return φ(β)
3    elseif β hamissá teszi φ egy klózát                ▷ Halott ág.
4          then return 0
5    elseif VISSZALÉPÉS-SAT(φ, β0)
6          then return 1
7    else return VISSZALÉPÉS-SAT(φ, β1)
```

A VISSZALÉPÉS-SAT algoritmus egy φ Boole-formula bemenet és φ valamely változóinak β részleges hozzárendelése mellett egy Boole-értéket ad vissza: 1-et, ha a β parciális megoldás egy minden változót tartalmazó φ-t kielégítő hozzárendeléssé bővíthető, és 0-át egyébként. A parciális hozzárendeléseket ebben az esetben a $\{0, 1\}$ ábécé fölötti, n-nél nem hosszabb szavaknak tekintjük. Az algoritmus kezdő hívása a VISSZALÉPÉS-SAT(φ, λ), ahol λ az üres hozzárendelés. Látható, hogy ha az előzetesen konstruált β parciális hozzárendelés φ valamely klózát hamissá teszi, akkor az már soha nem lesz kiegészítve egy φ-t kielégítő hozzárendeléssé, a rekurziós fa aktuális csúcs alatti részfáját levágjuk (lásd még a 4.3-1. gyakorlatot).

A VISSZALÉPÉS-SAT futási idejének felső becsléséhez az adott φ formulából egy tetszőlegesen választott C_j klózt vizsgálunk. A φ minden β kitöltendő hozzárendelésének – és ebből most elsősorban a három C_j-ben előforduló változónak – értékül igazságértékeket kell kapnia. A 0 vagy 1 sorozatok $2^3 = 8$ lehetőségéből az algoritmus biztosan kiválasztja azt az egyet, amely C_j-t hamissá teszi. A szóban forgó csúcs VISSZALÉPÉS-SAT(φ, β) rekurziós fájában tehát egy „halott" részfához vezet, amit nyugodtan levághatunk. A φ formula szerkezetéből adódóan további „halott" részfák állnak elő, melyeket nem kell figyelembe venni, ebből adódik a VISSZALÉPÉS-SAT egy felső korlátjára legrosszabb esetben $O((2^3 - 1)^{n/3}) = O(\sqrt[3]{7}^n) \approx O(1.9129^n)$, ami a „naiv" 3-SAT algoritmus $O(2^n)$ korlátját még

[4] A félreértések elkerülése végett hangsúlyozni kell, hogy a rekurziós fa más, mint egy NTM kiszámítási fája. A VISSZALÉPÉS-SAT algoritmus teljesen determinisztikusan működik, egy rekurziós fa mélységi bejárásának megfelelően. Egy ilyen fa belső csúcsai az algoritmus rekurzív hívásait reprezentálják, a gyökere az első hívást, a leveleknél pedig további hívások nélkül terminál az algoritmus. A rekurziós fa egy $\hat{k}$ csúcsa pontosan akkor gyereke egy k csúcsnak, ha a $\hat{k}$ hívás az algoritmus k után kiváltott számításán belül következik be, továbbá nincs olyan $k^\star$ hívás, ami k-n belül van és $k^\star$-on belül van.

mindig javítja valamelyest.

A 3-SAT determinisztikus időbonyolultsága még lejjebb szorítható, például Monien és Speckenmeyer oszd-meg-és-uralkodj algoritmusa $O(1.618^n)$ felső korláttal rendelkezik. A felső korlát világrekordját determinisztikus 3-SAT-algoritmusra jelenleg egy $O(1.481^n)$ korlátos, lokális keresésen alapuló megoldás tartja, melyet Dantsin és társai készítettek. Vannak más, nemdeterminisztikus megközelítések is. Ezek közül egyet mutatunk be, egy Schöning munkáiból merítő, úgynevezett „véletlen séta” algoritmust.

4.3.2. 3-SAT valószínűségi időbonyolultsága

Egy *véletlen séta* egy (véletlen) bejárás adott struktúrán, például egy euklideszi térben, egy végtelen rácson vagy egy gráfon. Esetünkben a gráfok bejárása érdekes, azon belül is olyan gráfok bejárása, melyek egy meghatározott sztochasztikus automatát ábrázolnak. Egy sztochasztikus automata egy speciális *véges automata.*

Egy véges automata szemléltethető állapotgráfjának segítségével. A véges automata állapotait csúcsok, az állapotok közti átmeneteket irányított, egy Σ ábécé szimbólumaival címkézett élek jelölik. Egy csúcs a kitüntetett *kezdőállapot* szerepet kapja, ennél kezdődik az automata működése, és akkor fejeződik be, ha a teljes bemenet feldolgozásra került. Az automata minden lépésben pontosan egy bemeneti szimbólumot olvas. Az állapotgráf bizonyos csúcsait *végállapotként* jelöljük meg, és amennyiben egy végállapotot érünk el a feldolgozás végén, akkor az automata elfogadja a bemenetet.

Véges automatákkal szavakat ismerhetünk fel. Egy Σ^*-beli $w = w_1 w_2 \cdots w_n$ szót az automata pontosan akkor fogad el, ha a kezdőállapotból kiindulva w minden egyes w_i szimbólumát a megfelelő sorrendben elolvassa, az állapotátmeneteket mindig a w_i-vel jelölt él követésének megfelelően hajtja végre, és végül elér egy végállapotot. Egy véges automata által elfogadott nyelv pontosan az ilyen módon elfogadott szavakból áll.

Egy $\mathcal{S}$ *sztochasztikus automata* éleit utólagosan számokkal is felcímkézzük. Az $\mathcal{S}$ állapotgráfjában egy u-ból v-be vezető élhez rendelt $p_{u,v}$ $(0 \leq p_{u,v} \leq 1)$ érték annak valószínűségét adja meg, hogy $\mathcal{S}$ az u állapotból v állapotba lép. Egy sztochasztikus automata (véletlen) állapotátmeneteinek folyamatát a sztochasztika tárgykörében *Markov-láncoknak*, a végállapotokat pedig *abszorpciós* állapotoknak nevezhetjük. Egy $\mathcal{S}$ sztochasztikus automata esetében így természetesen egy w szó elfogadása (és ezzel az $\mathcal{S}$ által elfogadott nyelv definiálása) csak adott valószínűséggel következik be, mely valószínűséget a w feldolgozása során bejárt élek címkéiből ismerhetjük.

Esetünkben azonban nem a nyelvfelismerés érdekes a sztochasztikus automata feladatai közül. Az automatát egy bejárásra szeretnénk alkalmazni, amit a Vélelenített-SAT valószínűségi algoritmus valósít meg. A Vélelenített-SAT algoritmus megpróbál az n-változós φ Boole-formulához egy kielégítő behelyettesítést találni, amennyiben létezik ilyen.

Vélelenített-SAT valamely φ bemenet mellett először is készít egy véletlenszerű β kezdeti hozzárendelést, ahol β minden egyes bitje függetlenül választott egyenletes eloszlás mellett, tehát β minden bitje 0 vagy 1 értéket vesz fel 1/2–1/2 valószínűséggel. A behelyettesítéseket ismét mint $\{0,1\}$ feletti n-hosszú szavakat kezeljük. Tegyük fel, hogy φ kielégíthető, és legyen $\tilde{\beta}$ φ tetszőlegesen választott kielégítő behelyettesítése φ-be. Legyen X az a valószínűségi változó, ami a β *és* $\tilde{\beta}$ *Hamming-távolságát* fejezi ki, vagyis azon bitek számát, ahol β és $\tilde{\beta}$ nem egyeznek meg. X nyilvánvalóan a $j \in \{0, 1, \ldots, n\}$ értékeket veheti fel, valamint binomiális eloszlású n és 1/2 paraméterekkel. Ennek megfelelően $X = j$ valószínűsége

$\binom{n}{j}2^{-n}$.

VÉLELENÍTETT-SAT(φ)

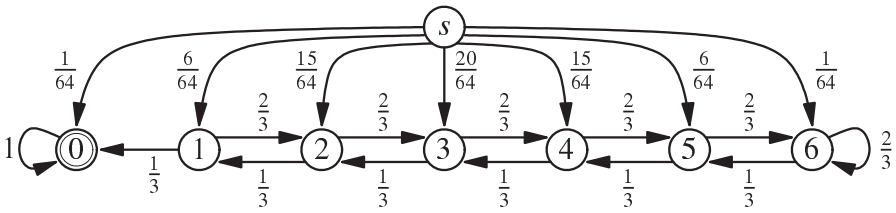
```
1  for i ← 1 to ⌈(4/3)^n⌉                          ▷ n a φ változóinak száma.
2      do válasszunk egy β ∈ {0,1}^n behelyettesítést, egyenletes eloszlás szerint
3      for j ← 1 to n
4          do if φ(β) = 1
5                  then return β                    ▷ β kielégíti φ-t.
6                  else válasszunk egy C = (x ∨ y ∨ z) klózt, ahol C(β) = 0
7                       válasszunk véletlenszerűen egy ℓ ∈ {x, y, z} literált,
                                egyenletes eloszlás szerint
8                       állapítsuk meg β ℓ-edik lefoglalt β_ℓ ∈ {0,1} bitjét
9                       változtassuk meg β-ban β_ℓ -t (1 − β_ℓ)-re
10 return „φ kielégíthetetlen”
```

A VÉLELENÍTETT-SAT algoritmus ezután ellenőrzi, hogy a kezdetben választott β hozzárendelés kielégíti-e a φ formulát, és amennyiben igen, akkor elfogadja a bemenetet. Egyébként pedig ha β nem teszi igazzá φ-t, akkor léteznie kell egy φ-beli klóznak, amit β nem elégít ki. A VÉLELENÍTETT-SAT tetszőlegesen kiválaszt egy ilyen klózt, a kiválasztott klózban egyenletes eloszlás mellett véletlenszerűen kijelöl egy literált, és megcseréli az ehhez a literálhoz rendelt bitet az aktuális β behelyettesítésben. Ezt n-szer megismétli, majd amennyiben az aktuális behelyettesítés még mindig nem elégíti ki a φ formulát, akkor a VÉLELENÍTETT-SAT újraindul egy új kezdeti hozzárendeléssel és megismétli a teljes fent leírt próbát t-szer, ahol $t = \lceil (4/3)^n \rceil$.

A 4.7. ábrán egy $\mathcal{S}$ sztochasztikus automata szerepel, melynek éleit nem szimbólumokkal, hanem mindössze átmenet-valószínűségekkel címkéztük. A VÉLELENÍTETT-SAT algoritmus φ bemenet melletti működését a következőkben mint $\mathcal{S}$ egy bejárását mutatjuk be. Az s kezdőállapotból kiindulva – amit később már sosem érintünk – VÉLELENÍTETT-SAT(φ) az n és $1/2$ paraméterű binomiális eloszlásnak megfelelően lép tovább egy $j \in \{0, 1, \ldots, n\}$ állapotba. Ezt mutatja a kép felső része egy $n = 6$ változós φ Boole-formula esetében. Egy ilyen j állapot azt jelenti, hogy a véletlenszerűen választott β kezdeti behelyettesítés és a $\tilde{\beta}$ kielégítő behelyettesítés állandó közötti Hamming-távolság j. Amíg $j \neq 0$, addig VÉLELENÍTETT-SAT(φ) a belső **for** ciklusban a kielégítő megoldást keresve mindig megváltoztatja az aktuális β behelyettesítés egy β_ℓ bitjét $(1 - \beta_\ell)$-re. Az $\mathcal{S}$ bejárásban ez megfelel egy lépésnek a $j - 1$ állapotba balra vagy a $j + 1$ állapotba jobbra, ahol csak n-nél kisebb vagy n-nel egyenlő állapot érhető el.

Az állandóra választott $\tilde{\beta}$ behelyettesítés kielégíti φ-t, tehát φ minden klózából legalább egy literált igazzá tesz. Ha minden klózban *pontosan* egyet kijelölünk ezek közül a $\tilde{\beta}$ által kielégített literálok közül, úgy pontosan akkor lépünk egy lépést balra, ha ezt az ℓ literált VÉLELENÍTETT-SAT kiválasztja. Egy balra lépés valószínűsége ($j > 0$-ból $j - 1$-be) nyilvánvalóan mindig $1/3$, a jobbra lépés (j-ből $j + 1$-be) valószínűsége pedig mindig $2/3$.

Bármikor is érjük el a $j = 0$ állapotot, ott β és $\tilde{\beta}$ Hamming-távolsága 0 lesz. Ennek megfelelően β kielégíti a φ formulát, és VÉLELENÍTETT-SAT(φ) visszaadja β aktuális értékét, majd megáll elfogadó állapotban. Egy $j \neq 0$ állapotban természetesen belefuthatunk egy $\tilde{\beta}$-tól különböző kielégítő behelyettesítésbe is, de mivel ez a lehetőség az elfogadás valószínűségét



4.7. ábra. Egy VÉLELENÍTETT-SAT bejárás sztochasztikus automatájának állapotgráfja.

csak növelheti, ezért az elfogadás valószínűségének becslésénél most figyelmen kívül hagyjuk. Ha a $j = 0$ állapotot nem érjük el β behelyettesítés n-szeri bitcseréjével, akkor a kezdő behelyettesítést olyan rosszul választottuk, hogy VÉLELENÍTETT-SAT(φ) egyszerűen eldobja, és egy új kezdeti értékkel próbál ismét szerencsét.

Mivel annak valószínűsége, hogy a (számunkra kedvező) 0 végállapottól jobbra távolodjunk, nagyobb, mint annak valószínűsége, hogy 0 irányába balra haladjunk, azt gondolhatjuk, hogy VÉLELENÍTETT-SAT nem túl nagy valószínűséggel jár sikerrel. Annak esélyét sem szabad azonban alábecsülni, hogy s-ből rögtön a nulladik lépés után a 0 közelébe kerülünk. Minél közelebbi a 0-hoz a kezdő pozíciónk, annál nagyobb annak valószínűsége, hogy a rákövetkező véletlen jobbra- vagy balra-lépések lefutásával 0-ba ütközünk.

VÉLELENÍTETT-SAT eredményességének valószínűségére és az algoritmus futási idejére vonatkozó elemzést itt most nem mutatjuk be teljes részletességében, csak vázlatszerűen. Az egyszerűség kedvéért tegyük fel, hogy n a 3 egész értékű többszöröse. Legyen p_i annak valószínűsége, hogy VÉLELENÍTETT-SAT n lépésen belül eléri a 0 állapotot, azzal a feltétellel, hogy VÉLELENÍTETT-SAT a bejárás nulladik lépésében (β kezdeti véletlen hozzárendelés megválasztásakor) az $i \leq n/3$ állapotba kerül. Amennyiben például kezdetben az $n/3$ állapotba kerülünk, akkor legfeljebb $n/3$ lépés megengedett a „rossz” irányba jobbra, ami kiegyenlíthető a „helyes” irányba, balra tett lépésekkel. Egyéb esetekben a 0 állapot nem érhető el n lépésben. Általában az i állapotból kiindulva legfeljebb $(n - i)/2$ lépést tehetünk jobbra, amiből a következő becslés adódik p_i-re:

$$p_i = \binom{n}{\frac{n-i}{2}} \left(\frac{2}{3}\right)^{(n-i)/2} \left(\frac{1}{3}\right)^{n-(n-i)/2} . \tag{4.2}$$

Legyen továbbá q_i annak valószínűsége, hogy VÉLELENÍTETT-SAT a bejárás nulladik lépésében az $i \leq n/3$ állapotba jut. Ekkor természetesen fennáll

$$q_i \quad = \quad \binom{n}{i} \cdot 2^{-n} . \tag{4.3}$$

Végül pedig legyen p az eredményesség valószínűsége, amikor VÉLELENÍTETT-SAT a külső *for* ciklus egy körében eléri a 0 állapotot. Ez lehetséges $j > n/3$ állapotokból is. Ennélfogva fennáll

$$p \quad \geq \quad \sum_{i=0}^{n/3} p_i \cdot q_i .$$

Ez az összeg közelíthető az entrópiafüggvény segítségével, továbbá az összeg tagjaiban

(4.2) és (4.3) binomiális együtthatói közelíthetők a Stirling-formulával, így végül $\Omega((3/4)^n)$ adódik p alsó korlátjának.

A hibás futás elkerülése érdekében a Vélelenített-SAT algoritmus összesen t független kísérletet hajt végre, melyek mindegyike új kezdőértékkel indul, és legalább a fent megadott hozzávetőleges $(3/4)^n$ valószínűséggel sikerrel jár. Mivel a kísérletek függetlensége miatt ezek a valószínűségi értékek szorzódnak, ezért Vélelenített-SAT eredményes futásának valószínűsége – tehát annak valószínűsége, hogy Vélelenített-SAT megadja φ egy kielégítő behelyettesítését, amennyiben az létezik – nagyon közel van 1-hez. Ha pedig φ kielégíthetetlen, Vélelenített-SAT sosem követ el hibát, tehát ezekben az esetekben a kimenet mindig „φ kielégíthetetlen” lesz.

Az algoritmus futási ideje megegyezik az eredményes futás $p \approx (3/4)^n$ valószínűségének reciprokával, mivel egy hiba valószínűsége (amikor a t próba során egyszer sem találunk kielégítő behelyettesítést, pedig φ kielégíthető) $(1-p)^t \leq e^{-t\cdot p}$-vel becsülhető. Amennyiben tehát nem szeretnénk egy előre adott ε hibavalószínűségi értéket átlépni, elegendő t értékét úgy megválasztani, hogy $e^{-t\cdot p} \leq \varepsilon$, illetve $t \geq \ln(1/\varepsilon)/p$ fennálljon. A konstans tényezőktől eltekintve ez $t = \lceil (4/3)^n \rceil$ választásával elérhető, az algoritmus futási ideje tehát $O((4/3)^n)$.

Gyakorlatok

4.3-1. Indítsuk el a Visszalépés-SAT algoritmust a $\varphi = (\neg x \vee y \vee \neg z) \wedge (x \vee \neg y \vee z) \wedge (\neg u \vee y \vee z) \wedge (u \vee \neg y \vee z)$ Boole-formulával, és konstruáljuk meg lépésről lépésre φ egy kielégítő behelyettesítését. Rajzoljuk fel az így kialakuló rekurziós fát, és határozzuk meg a fa azon részeit, melyeket levágunk, mert nem vezethetnek megoldáshoz.

4.4. Gráfizomorfizmus és alsóság

A következő alfejezetben szükségünk lesz a 3.1.3. alfejezetben leírt csoport- és gráfelméleti alapokra. Különösen fontos felidézni a 3.6. definíció permutációcsoport fogalmát, a `GI` gráfizomorfizmus problémát, valamint a 3.8. definíció `GA` gráfautomorfizmus problémáját (lásd még a 3.4. példát is).

4.4.1. Visszavezethetőségek és bonyolultsági hierarchiák

A 4.2. pontban megismerkedtünk a hatékonyan megoldható házasítás problémával, valamint az NP-teljes SAT, `3-SAT` és 3-`DM` problémákkal. Könnyen meggondolhatjuk, hogy P = NP pontosan akkor teljesül, ha *minden* NP-beli probléma – beleértve az NP-teljes problémákat is – P-ben helyezkedik el. Amennyiben P ≠ NP, akkor P-ben nem lehetnek NP-teljes problémák. Megoldható-e vajon hatékonyan minden NP-beli probléma a hihetőnek tűnő P ≠ NP feltevés mellett, tehát P-beliek-e, vagy pedig NP-teljesek? Tudunk-e vajon olyan NP-beli problémákat adni P ≠ NP feltevés mellett, melyek nem oldhatók meg hatékonyan, de nem is NP-teljesek? A választ a következő tétel adja meg.

4.11. tétel (Ladner). *Ha* P ≠ NP, *akkor léteznek olyan problémák NP-ben, melyek se nem P-beliek, se nem NP-teljesek.*

A Ladner által megadott problémák valamelyest „mesterségesek” olyan szempontból, hogy közvetlenül a 4.11. tétel bizonyításához készültek. Vannak azonban olyan termész-

tes problémák is NP-ben, melyek megfelelő jelöltek arra, hogy se P-beliek, se pedig NP-teljesek ne legyenek. Ezek közül egy GI, a gráfizomorfizmus probléma, és ezt szeretnénk most bizonyítani. Ehhez definiálunk NP-n belül két, Schöning által bevezetett bonyolultsági osztályhierarchiát, az úgynevezett *alsó-*, valamint a *felső-hierarchiát.* Ahhoz, hogy ezt a két hierarchiát definiálni tudjuk, be kell vezetnünk az NP-re épülő *polinomiális idő-hierarchia* fogalmát, ennek definíciójához pedig szükségünk van egy, a 4.5. definícióban bevezetett sok-egy visszavezethetőségnél általánosabb visszavezethetőségre, nevezetesen a *Turing-visszavezethetőség*, a $\leq_T^p$ fogalmára. Definiáljuk továbbá a *nemdeterminisztikus* és az *erősen nemdeterminisztikus Turing-visszavezethetőség* – $\leq_T^{NP}$ és $\leq_{sT}^{NP}$ – fogalmakat, melyek a polinomiális idő-hierarchia esetében fontosak. Ezek a visszavezethetőségek az *orákulumos Turing-gép* fogalmára építkeznek. Most tehát definiáljuk a nevezett fogalmakat.

4.12. definíció (orákulumos Turing-gép). *Az* ***orákulum-halmaz*** *(röviden* ***orákulum****) szavak egy halmaza. Egy B orákulummal rendelkező* orákulumos M Turing-gép *egy olyan Turing-gép, ami rendelkezik egy speciális* ***kérdés-szalaggal****, állapotainak halmaza pedig tartalmaz egy speciális $z_?$* ***kérdés-állapotot****, valamint z_{yes} és z_{no}* ***válasz-állapotokat****. Amíg M nincs a $z_?$ állapotban, addig pontosan úgy viselkedik, mint egy szokványos Turing-gép. Ha azonban működése során $z_?$ kérdés-állapotba kerül, akkor megszakítja futását, és kérdést intéz az orákulumhoz az éppen a kérdés-szalagon álló q szóról. Az orákulumot egyfajta „fekete dobozként" képzelhetjük el: B orákulum egy lépésben eldönti, hogy q szó B-beli-e, függetlenül attól, hogy ennek eldöntése milyen nehézségű feladat. Ha $q \in B$, akkor M a következő lépésben z_{yes} válasz-állapotba kerül és folytatja működését. Ellenkező esetben (ha $q \notin B$) M z_{no} állapotban folytatja működését. Ekkor azt mondjuk, hogy az M Turing-géppel* ***x bemenet kiszámítása relatív B orákulumra nézve****, a kiszámítást pedig $M^B(x)$-el jelölünk. Legyen $L(M^B)$ az M^B által elfogadott nyelv. Egy C bonyolultsági osztályt* ***relativizálhatónak*** *nevezünk, ha reprezentálható orákulumos Turing-gépekkel (üres orákulum-halmaz mellett) az előzőeknek megfelelően. Definiáljuk egy C relativizálható bonyolultság-osztály és egy B orákulum mellett a B-re relatív C osztályt a következőképp:*

$$C^B = \{L(M^B) \mid M \text{ egy } C\text{-t reprezentáló orákulumos Turing-gép}\} .$$

Ha $\mathcal{B}$ halmazok egy osztálya, akkor legyen $C^{\mathcal{B}} = \bigcup_{B\in\mathcal{B}} C^B$.

NPOTM (illetve DPOTM) jelöli a *nemdeterminisztikus* (illetve a *determinisztikus*) *polinomiális idejű orákulumos Turing-gépeket.* Definiálhatók például a következő osztályok:

$$\mathrm{NP}^{\mathrm{NP}} = \bigcup_{B\in\mathrm{NP}} \mathrm{NP}^B = \{L(M^B) \mid M \text{ egy NPOTM, } B \text{ pedig NP-beli}\} ;$$
$$\mathrm{P}^{\mathrm{NP}} = \bigcup_{B\in\mathrm{NP}} \mathrm{P}^B = \{L(M^B) \mid M \text{ egy DPOTM, } B \text{ pedig NP-beli}\} .$$

Amennyiben az orákulum üres halmaz ($\emptyset$), akkor a nem relativizálható $\mathrm{NP} = \mathrm{NP}^\emptyset$, illetve $\mathrm{P} = \mathrm{P}^\emptyset$ osztályokhoz jutunk. NPOTM helyett ekkor NPTM-ről beszélhetünk, DPOTM helyett pedig DPTM-ről. Különösen jól alkalmazhatók az orákulumos Turing-gépek keresési feladatokhoz, mint amilyen a prefixkeresés is, ahogy a következő példa mutatja. Az itt alkalmazott Pre-Iso NP-orákulum szolgáltatja azt az információt, hogyan lehet az üres szóból kiindulva bitről bitre haladva az NP-beli GI probléma legkisebb megoldását előállítani, amennyiben létezik ilyen megoldás.

4.5. példa. *Prefixkeresés a legkisebb izomorfizmus után orákulumos Turing-géppel.* A GI gráfizomorfizmus problémát az 3.1.3. alfejezet 3.8. definíciójában fogalmaztuk meg. Legyen G és H két adott gráf $n \geq 1$ darab csúccsal. A G és H közti izomorfizmusokat „$(G,H) \in \mathtt{GI}$" *megoldásának* nevezzük. Az $\mathrm{Iso}(G,H)$ izomorfizmus-csoport tartalmazza a „$(G,H) \in \mathtt{GI}$" összes megoldását, valamint $\mathrm{Iso}(G,H) \neq \emptyset \iff (G,H) \in \mathtt{GI}$. Ha $(G,H) \in \mathtt{GI}$, akkor egy lexikografikusan legkisebb megoldás előállítása a célunk, ellenkező esetben pedig az üres λ szó kimenet segítségével tudatnunk kell, hogy „$(G,H) \notin \mathtt{GI}$". Tehát, ki szeretnénk számítani a következőképp definiált f függvényt:

$$f(G,H) \quad = \quad \begin{cases} \min\{\pi \,\big|\, \pi \in \mathrm{Iso}(G,H)\}, & \text{ha } (G,H) \in \mathtt{GI}\ , \\ \lambda, & \text{ha } (G,H) \notin \mathtt{GI}\ , \end{cases}$$

ahol a lexikografikus rendezésre vonatkozó minimumot $\mathfrak{S}_n$-ből képezzük. A $\mathfrak{S}_n$ definíciója a következő: kezeljük a $\pi \in \mathfrak{S}_n$ egy permutációját mint n hosszú, $[n] = \{1,2,\ldots,n\}$ ábécé fölötti $\pi(1)\pi(2)\cdots\pi(n)$ szót. Pontosan akkor írható $\pi < \sigma$ $(\pi,\sigma \in \mathfrak{S}_n)$, ha létezik egy olyan $j \in [n]$, melyre minden $i < j$ esetén $\pi(i) = \sigma(i)$, valamint $\pi(j) < \sigma(j)$. Amennyiben egy $\sigma \in \mathfrak{S}_n$ permutációból kitörlünk néhány $(i,\sigma(i))$ párt, akkor *parciális permutációt* kapunk, amit szintén kezelhetünk $[n]$ fölötti szóként. A $\sigma \in \mathfrak{S}_n$ egy $k \leq n$ *hosszú prefixe* σ egy olyan parciális permutációja, amely minden $(i,\sigma(i))$ párt tartalmaz $i \leq k$ mellett, viszont egyetlen $(i,\sigma(i))$ párt sem $i > k$ esetén. A $k = 0$ esetben σ prefixe a λ üres szó, valamint $k = n$ esetben a teljes permutáció. Ha π a $\sigma \in \mathfrak{S}_n$ egy $k < n$ hosszú prefixe és $w = i_1 i_2 \cdots i_{|w|}$ egy $[n]$ fölötti $|w| \leq n-k$ hosszú szó, akkor jelölje πw azt a parciális permutációt, ami π-t a $(k+1,i_1),(k+2,i_2),\ldots,(k+|w|,i_{|w|})$ párokra bővíti. Ha $1 \leq j \leq |w|$ esetén $\sigma(k+j) = i_j$, akkor πw is σ egy prefixe. Definiáljuk G és H gráfok mellett az $\mathrm{Iso}(G,H)$-beli izomorfizmusok prefixeinek halmazát:

$$\text{Pre-Iso} \quad = \quad \{(G,H,\pi) \,\big|\, (\exists w \in \{1,2,\ldots,n\}^*)\ [w = i_1 i_2 \cdots i_{n-|\pi|} \text{ és } \pi w \in \mathrm{Iso}(G,H)]\}\ .$$

Figyeljük meg, hogy $n \geq 1$-re a λ üres szót egy permutáció sem képzi le $\mathfrak{S}_n$-be, valamint hogy $\mathrm{Iso}(G,H) = \emptyset$ pontosan akkor áll fenn, ha $(G,H,\lambda) \notin$ Pre-Iso, ami viszont akkor és csak akkor igaz, ha $(G,H) \notin \mathtt{GI}$.

N-Pre-Iso(G,H)

```
1  if (G,H,λ) ∉ Pre-Iso
2     then return 0
3     else π ← λ
4          j ← 0
5          while j < n                          ▷ G és H mindig n csúcsú.
6                do i ← 1
7                   while (G,H,πi) ∉ Pre-Iso
8                         do i ← i + 1
9                         π ← πi
10                        j ← j + 1
11         return π
```

Az N DPOTM prefixkereséssel, a Pre-Iso orákulum segítségével számítja ki az f függvényt (lásd még a 4.4-2. gyakorlatot). Jelölje FP a polinomiális időben kiszámítható függvények osztályát. Ekkor $f \in \mathrm{FP}^{\text{Pre-Iso}}$. Mivel Pre-Iso egy NP-beli halmaz (lásd a 4.4-2. gyakorlatot), ezért $f \in \mathrm{FP}^{\mathrm{NP}}$ következik.

A 4.5. példa alapján elláthatók orákulummal a függvényeket kiszámító Turing-gépek is, valamint relativizálhatók függvény osztályokra, mint például az FP. Másrészt alkalmazhatók orákulumként halmazok helyett függvények is. A 4.12. definícióval összevetve az

$f : \Sigma^* \to \Sigma^*$ függvényorákulum egy q kérdésre egy lépésben nem az „igen” vagy „nem” válaszokat adja, hanem $|f(q)|$ lépésben visszaadja az $f(q)$ függvényértékeket. A következő tétel kimondja, hogy konstruálható teljes izomorfizmus két izomorf gráf közötti parciális izomorfizmusból egy f függvényorákulum segítségével.

4.13. tétel. *Legyenek G és H izomorf gráfok. Legyen f egy függvényorákulum, melyre $f(G,H) = (x,y)$, ahol $x \in V(G)$, valamint fennáll $y \in V(H)$, és $\sigma(x) = y$ egy $\sigma \in \mathrm{Iso}(G,H)$ izomorfizmussal. Ekkor létezik egy M DPOTM, amely az f orákulummal kiszámol egy $\varphi \in \mathrm{Iso}(G,H)$ izomorfizmust.*

A 4.13. tétel szerint tehát az NP-beli `GI` probléma egy teljes megoldásának konstrukciója visszavezethető a `GI` egy parciális megoldására (érdemes lehet ezt összevetni a Visszalépés-SAT algoritmussal, ami bitről bitre bővíti a `3-SAT` kifejezések parciális megoldását, amíg azok teljesek nem lesznek).

Képzeljük el, hogy Merlin, a 3.5. alfejezet 3.12. ábráján szereplő `GI` probléma zéró ismeretű protokolljában egy $\sigma \in \mathrm{Iso}(G,H)$ izomorfizmust küld Artúrnak, ahogy az kívánta, sajnos azonban az átvitelnél némely bitek elvesznek vagy használhatatlanná válnak. Artúr tehát csak egy parciális π izomorfizmust kap meg σ-ból. Rekonstruálhat azonban Merlin segítségével a 4.13. tételnek köszönhetően egy $\varphi \in \mathrm{Iso}(G,H)$ teljes izomorfizmust még akkor is, ha π csak egyetlen csúcspárból áll. Figyeljük meg, hogy φ-nek nem kell a Merlin által eredetileg elküldött σ izomorfizmusnak lennie.

A 4.6. példa a bizonyítás alapötletét szemlélteti konkrét gráfokkal. A formális bizonyítástól most eltekintünk. Egy lényeges tulajdonság, amit itt kihasználunk `GI` úgynevezett *ön-visszavezethetősége*. Anélkül, hogy belemennénk a technikai részletekbe, ezt a fontos fogalmat a következőképp írhatjuk le: egy A halmaz pontosan akkor *ön-visszavezethető*, ha létezik egy M DPOTM, ami egy A orákulum segítségével elfogadja magát az A halmazt. M az x bemeneti szóról egyszerűen megkérdezhetné az A orákulumot, amikor is x A-beliségének eldöntése természetesen triviális lenne, ezért az ön-visszavezethetőségnél megtiltjuk a bemenetre magára vonatkozó kérdéseket. Ehelyett az M csak olyan kérdéseket tehet fel az A orákulumnak, melyek *kisebbek* a bemenetnél, ahol a kisebb jelzőt a hagyományos lexikografikus rendezésből általánosított értelemben használjuk. Ezt a definíciót a 4.13. tételnek megfelelően szintén átültethetjük halmazok helyett függvényekre.

4.6. példa. (Teljes izomorfizmus előállítása parciális izomorfizmusból.) A **??**. ábra két izomorf G és H gráfjára $\mathrm{Iso}(G,H) = \{\sigma, \varphi\}$, ahol $\sigma = \left(\begin{smallmatrix}1&2&3&4&5\\1&5&4&3&2\end{smallmatrix}\right)$ és $\varphi = \left(\begin{smallmatrix}1&2&3&4&5\\5&1&4&3&2\end{smallmatrix}\right)$. A következőkben leírunk és körüljárunk egy kudarchoz vezető naiv megközelítést. Tegyük fel, hogy a 4.13. tétel algoritmusa az f orákulumhoz intézett kérdésekkel meghatároz egy (x,y) csúcspárt, ahol $\sigma(x) = y$ vagy $\varphi(x) = y$. Így felismerve (x,y)-t az algoritmus törli x-et G-ből, illetve y-t H-ból, és hasonló módon fokozatosan addig halad, amíg a gráfok üresek nem lesznek. Ezzel biztosítottuk, hogy az algoritmus legfeljebb $n = 5$ lépésben terminál jon, és remélhetőleg a tárolt csúcspárok sorrendje a keresett $\mathrm{Iso}(G,H)$-beli izomorfizmust adja, tehát vagy σ-t, vagy pedig φ-t. Ez azonban nem feltétlenül következik be. Tegyük fel például, hogy az első (G,H) párra vonatkozó kérdésre az f orákulum az $(5,2)$ csúcspárt választja. A 4.13. tétel algoritmusa ekkor egyszerűen törli G-ből az 5, valamint H-ból a 2 csúcsokat (valamint az 5-ből, illetve a 2-ből kiinduló éleket). Ekkor a **??**. ábra $\widehat{G}$ és $\widehat{H}$ gráfjaihoz jutunk. Az $\mathrm{Iso}(\widehat{G},\widehat{H})$ azonban hat izomorfizmust tartalmaz, amiből csak kettő *összeegyeztethető* a $(5,2)$ párral (lásd a 4.4-3. gyakorlatot). Ez tehát azt jelenti, hogy $\mathrm{Iso}(\widehat{G},\widehat{H})$ hat izomorfizmusából csak kettő parciális izomorfizmusa σ-nak és φ-nek. Ezután az algoritmus következő lépésében tárolhatja például az új $(4,5)$ párt, ami sem σ-hoz, sem pedig φ-hez nem tartozik.

Annak érdekében, hogy kizárhassuk ezeket az eseteket, a 4.13. tétel f orákulumos M *DPOTM*-ja másképp jár el. Nem csak egyszerűen törli azokat a csúcspárokat, melyeket az orákulum segítségével meghatározott, hanem elegendően nagy méretű klikkek segítségével kijelöli a törölt csúcsok szomszédos csúcsait. Egy k méretű ***klikk*** olyan gráf, aminek k darab csúcsa páronként egy-egy éllel mindenhol össze van kötve. Példánkban tehát az első $(5,2)$ csúcspár törlése után a G-beli 4 és H-beli 3 csúcsokat egy 5 méretű klikkel jelöljük meg. Ebből adódik a (G_1, H_1) új pár, lásd a **??**. ábrán. Figyeljük meg, hogy most minden $\pi \in \mathrm{Iso}(G_1, H_1)$ izomorfizmus összeegyeztethető a $(5,2)$ csúcspárral az eredeti $\mathrm{Iso}(G,H)$-beli σ-ból és φ-ből.

Ha M fokozatosan e szerint az eljárás szerint halad, akkor az orákulum válaszok egy meghatározott sorozatára például az **??**. ábrán szereplő (G_2, H_2), (G_3, H_3) és (G_4, H_4) gráfpárok állhatnak elő. Ekkor a (G_4, H_4)-nél egyértelműen meghatároztuk az utolsó $(4,3)$ csúcspárt, M pedig ebben az esetben előállította a $\varphi = \left(\begin{smallmatrix}1&2&3&4&5\\5&1&4&3&2\end{smallmatrix}\right)$ teljes izomorfizmust $\mathrm{Iso}(G,H)$-ból.

Most pedig különböző visszavezethetőségeket definiálunk az orákulumos Turing-gép fogalmából kiindulva. Minden itt vizsgált visszavezethetőség hatékonyan – tehát polinomiális időben – kiszámítható.

4.14. definíció (Turing-visszavezethetőség). *Legyen $\Sigma = \{0,1\}$ egy bináris ábécé, legyenek A és B ezen Σ feletti szavak halmazai, és legyen C egy bonyolultsági osztály. A C-beli halmazok komplementereinek osztályát definiáljuk mint $\mathrm{co}C = \{\overline{L} \mid L \in C\}$. Definiáljuk a következő visszavezethetőségeket:*

- *Turing-visszavezethetőség: $A \leq_{\mathrm{T}}^{\mathrm{p}} B \iff A = L(M^B)$ egy M DPOTM-ra.*
- *Nemdeterminisztikus Turing-visszavezethetőség: $A \leq_{\mathrm{T}}^{\mathrm{NP}} B \iff A = L(M^B)$ egy M NPOTM-ra.*
- *Erős nemdeterminisztikus Turing-visszavezethetőség: $A \leq_{\mathrm{sT}}^{\mathrm{NP}} B \longleftrightarrow A \in \mathrm{NP}^B \cap \mathrm{coNP}^B$.*
- *Ha $\leq_r$ egy fent definiált visszavezethetőség, akkor egy B halmazt pontosan akkor nevezünk **$\leq_r$-nehéznek C-ben**, ha $A \leq_r B$ teljesül minden $A \in C$ halmazra. Egy B halmaz pontosan akkor **$\leq_r$-teljes C-ben**, ha B $\leq_r$-nehéz C-re és $B \in C$.*
- $\mathrm{P}^C = \{A \mid (\exists B \in C)\,[A \leq_{\mathrm{T}}^{\mathrm{p}} B]\}$ *halmaz a C* lezárása $\leq_{\mathrm{T}}^{\mathrm{p}}$-visszavezethetőség mellett.
- $\mathrm{NP}^C = \{A \mid (\exists B \in C)\,[A \leq_{\mathrm{T}}^{\mathrm{NP}} B]\}$ *halmaz a C* lezárása $\leq_{\mathrm{T}}^{\mathrm{NP}}$-visszavezethetőség mellett.

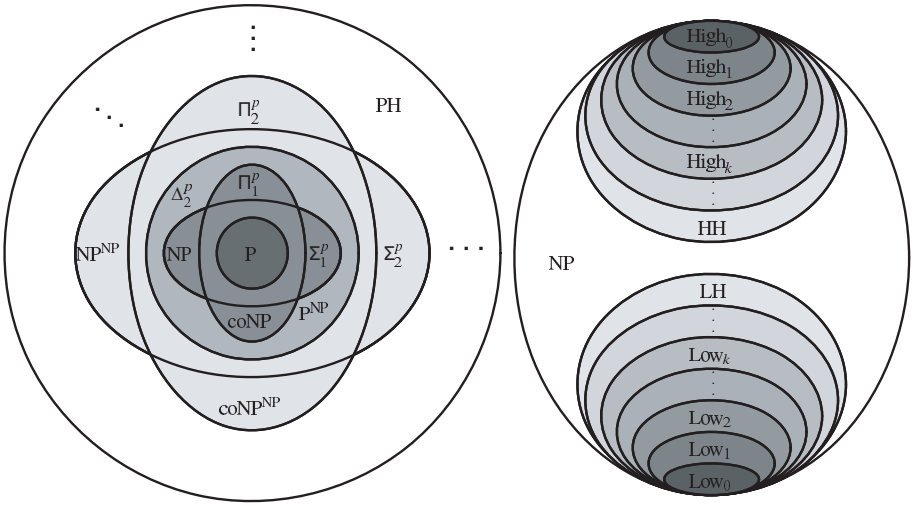
A 4.14. definíció $\leq_{\mathrm{T}}^{\mathrm{p}}$- és $\leq_{\mathrm{T}}^{\mathrm{NP}}$-visszavezethetőségeinek segítségével bevezetjük a polinomiális idő-hierarchiát, valamint az NP-ben definiált alsó- és felső-hierarchiákat.

4.15. definíció (polinomiális idő-hierarchia). *A* $\mathrm{PH} = \bigcup_{k\geq 0} \Sigma_k^p$ polinomiális idő-hierarchiát *a következőképp definiáljuk:* $\Delta_0^p = \Sigma_0^p = \Pi_0^p = \mathrm{P}$, $\Delta_{i+1}^p = \mathrm{P}^{\Sigma_i^p}$, $\Sigma_{i+1}^p = \mathrm{NP}^{\Sigma_i^p}$ *és* $\Pi_{i+1}^p = \mathrm{co}\Sigma_{i+1}^p$ *minden $i \geq 0$-ra.*

Ebből speciális esetként $\Delta_1^p = \mathrm{P}^{\Sigma_0^p} = \mathrm{P}^{\mathrm{P}} = \mathrm{P}$ és $\Sigma_1^p = \mathrm{NP}^{\Sigma_0^p} = \mathrm{NP}^{\mathrm{P}} = \mathrm{NP}$ és $\Pi_1^p = \mathrm{co}\Sigma_1^p = \mathrm{coNP}$ adódik. A következő bizonyítás nélküli tétel megadja ezen hierarchiák néhány tulajdonságát (lásd még a 4-2. feladatnál).

4.16. tétel (Meyer és Stockmeyer). *Minden egyes $i \geq 1$-re:*

1. $\Sigma_{i-1}^p \cup \Pi_{i-1}^p \subseteq \Delta_i^p \subseteq \Sigma_i^p \cap \Pi_i^p$.
2. Σ_i^p, Π_i^p, Δ_i^p *és* PH *mind $\leq_{\mathrm{m}}^{\mathrm{p}}$-lezártak, Δ_i^p továbbá a $\leq_{\mathrm{T}}^{\mathrm{p}}$-visszavezetésekre nézve is lezárt.*



4.8. ábra. A polinomiális idő-, valamint az alsó- és felső-hierarchiák.

3. Σ_i^p *pontosan azt az* A *halmazt tartalmazza, amelyre léteznek* $B \in \mathrm{P}$ *halmaz és egy* p *polinom úgy, hogy minden* $x \in \Sigma^*$*-ra:* $x \in A \iff (\exists^p w_1)\,(\forall^p w_2) \cdots (\mathfrak{Q}^p w_i)\,[(x, w_1, w_2, \ldots, w_i) \in B]$*, ahol a* $\exists^p$ *és* $\forall^p$ *kvantorok polinomiális hosszkorlátúak és* $\mathfrak{Q}^p = \exists^p$ *ha* i *páratlan, vagy* $\mathfrak{Q}^p = \forall^p$ *ha* i *páros.*
4. *Ha* $\Sigma_{i-1}^p = \Sigma_i^p$*, akkor* PH *összeomlik* $\Sigma_{i-1}^p = \Pi_{i-1}^p = \Delta_i^p = \Sigma_i^p = \Pi_i^p = \cdots =$ PH*-ba.*
5. *Ha* $\Sigma_i^p = \Pi_i^p$*, akkor* PH *összeomlik* $\Sigma_i^p = \Pi_i^p = \Delta_{i+1}^p = \Sigma_{i+1}^p = \Pi_{i+1}^p = \cdots =$ PH*-ba.*
6. Σ_i^p*-ben,* Π_i^p*-ben és* Δ_i^p*-ben léteznek* $\leq_{\mathrm{m}}^{\mathrm{p}}$*-teljes problémák. Amennyiben* PH*-ban létezik* $\leq_{\mathrm{m}}^{\mathrm{p}}$*-teljes probléma, akkor* PH *egy véges szintjébe omlik össze, vagyis egy adott* k*-val* $\mathrm{PH} = \Sigma_k^p = \Pi_k^p$.

4.17. definíció (az NP-beli alsó- és felső-hierarchia). *Legyen* $k \geq 0$ *mellett az*

- *az* NP*-beli* $\mathrm{LH} = \bigcup_{k\geq 0} \mathrm{Low}_k$ ***alsó-hierarchia*** k*-adik szintje* $\mathrm{Low}_k = \{L \in \mathrm{NP} \mid \Sigma_k^{p,L} \subseteq \Sigma_k^p\}$,
- *az* NP*-beli* $\mathrm{HH} = \bigcup_{k\geq 0} \mathrm{High}_k$ ***felső-hierarchia*** k*-adik szintje* $\mathrm{High}_k = \{H \in \mathrm{NP} \mid \Sigma_{k+1}^p \subseteq \Sigma_k^{p,H}\}$.

Egy L halmaz tehát akkor és csak akkor Low_k-beli, ha egy Σ_k^p kiszámításhoz orákulumként használhatatlan, vagyis minden információt, amit L kínálhat, egy Σ_k^p-gép is kiszámíthat, önmagában, orákulum nélkül. Másrészről egy H halmaz High_k-ból olyan gazdag hasznos információkban, hogy egy Σ_k^p-gép kiszámítási erejét egy NP-halmazban maximális értékre növeli. Egy High_k-beli H orákulum segítségével tehát egy Σ_k^p-gép minden Σ_{k+1}^p kiszámítást szimulálhat. Egy Σ_k^p-gép számára tehát H pontosan olyan sokat nyújt, mint egy NP-teljes halmaz. A fent definiált hierarchiák egymásba ágyazott szerkezetét a 4.8. ábra szemlélteti, ahol a világosabb tónusú bonyolultsági osztályok tartalmazzák a sötétebb árnyalatú osztályokat. Egyik ilyen $\mathcal{C} \subseteq \mathcal{D}$ tartalmazásról sem ismert, hogy *valódi*-e, tehát vajon $\mathcal{C} \neq \mathcal{D}$

fennáll-e. A kérdés, hogy $\Sigma_k^p \neq \Sigma_{k+1}^p$ igaz-e, $k = 0$ esetében pontosan a P-versus-NP kérdés. Most (ismét bizonyítás nélkül) felsoroljuk a hierarchiák néhány fontosabb tulajdonságát (lásd még a 4-2. feladatot). Ladner tétele (4.11.) közvetlenül következik a 4.18. tétel utolsó állításának $n = 0$ speciális esetéből.

4.18. tétel (Schöning).

1. $\mathrm{Low}_0 = \mathrm{P}$, $\mathrm{Low}_1 = \mathrm{NP} \cap \mathrm{coNP}$ *és* $\mathrm{NP} \cap \mathrm{coAM} \subseteq \mathrm{Low}_2$.
2. $\mathrm{High}_0 = \{H \mid H \leq_{\mathrm{T}}^{\mathrm{p}}$*-teljes* NP*-ben*$\}$ *és* $\mathrm{High}_1 = \{H \mid H \leq_{\mathrm{sT}}^{\mathrm{NP}}$*-teljes* NP*-ben*$\}$.
3. $\mathrm{Low}_0 \subseteq \mathrm{Low}_1 \subseteq \cdots \subseteq \mathrm{Low}_k \subseteq \cdots \subseteq \mathrm{LH} \subseteq \mathrm{NP}$.
4. $\mathrm{High}_0 \subseteq \mathrm{High}_1 \subseteq \cdots \subseteq \mathrm{High}_k \subseteq \cdots \subseteq \mathrm{HH} \subseteq \mathrm{NP}$.
5. *Minden* $n \geq 0$*-ra* $\mathrm{Low}_n \cap \mathrm{High}_n$ *pontosan akkor nem üres, ha* $\Sigma_n^p = \Sigma_{n+1}^p = \cdots = \mathrm{PH}$.
6. *Minden* $n \geq 0$*-ra* NP*-ben akkor és csak akkor léteznek sem* Low_n*-beli, sem pedig* High_n*-beli halmazok, ha* $\Sigma_n^p \neq \Sigma_{n+1}^p$. NP*-ben pontosan akkor léteznek sem* LH*-beli, sem pedig* HH*-beli halmazok, ha* PH *valódi végtelen halmaz, tehát nem omlik össze egy véges szintjére.*

4.4.2. A gráfizomorfizmus alsó-hierarchiabeli

Hozzákezdve az említett eredmény bizonyításához belátjuk, hogy `GI` Low_2-beli, ami egy erős fegyvertény `GI` NP-teljessége ellen. Ha `GI` NP-teljes lenne, akkor $\mathrm{High}_0 \subseteq \mathrm{High}_2$-beli is lenne, mivel a 4.18. tétel alapján High_0 pontosan NP $\leq_{\mathrm{T}}^{\mathrm{p}}$-teljes részét tartalmazza, így tehát az NP $\leq_{\mathrm{m}}^{\mathrm{p}}$-teljes részét is. Szintén a 4.18. tételből következik viszont, hogy $\mathrm{Low}_2 \cap \mathrm{High}_2$ akkor és csak akkor nem üres, ha PH összeomlik Σ_2^p-be, ami viszont nagyon valószínűtlennek tűnik.

A tétel bizonyításához szükségünk van még az úgynevezett hasítás lemmára, lásd 4.20. lemma. A hasítás egy dinamikus adatrendező eljárás. Minden adategységhez egy kulcs tartozik, ami azt egyértelműen azonosítja. A kulcsok U halmaza meglehetősen nagy, a következőkben mint *univerzum* tekintjük, mivel a ténylegesen felhasznált kulcsok $V \subseteq U$ halmaza jóval kisebb lehet. A célunk az, hogy U elemeit egy *hasító függvény* segítségével bejegyezzük egy $T = \{0, 1, \ldots, k-1\}$ *hasító táblázatba*, ahol több U-beli kulcshoz is tartozhat ugyanaz a T-beli cím. Lehetőség szerint különböző V-beli kulcsoknak különböző T-beli címet kell kapniuk, tehát elkerülendők a ténylegesen használt kulcsok *ütközései*. Másképp fogalmazva a hasító függvény lehetőség szerint legyen injektív V-re.

A sok különböző ismert hasító eljárás változat közül számunkra leginkább az *univerzális hasítás* az érdekes. Ennél az alapötlet abból áll, hogy egy alkalmasan választott hasító függvény családból *véletlenszerűen* választunk egy hasító függvényt. Ez a hasítás módszer univerzális olyan értelemben, hogy már nem függ egy meghatározott V halmaztól, hanem *minden* megfelelően kicsi V halmazon nagy valószínűséggel elkerüli az ütközéseket. A valószínűség itt a hasító függvény véletlenszerű választására vonatkozik. A következőkben a kulcsokat mint $\Sigma = \{0, 1\}$ ábécé fölött kódolt szavakat kezeljük, és Σ^n-el jelöljük a Σ^*-beli, n hosszú szavak halmazát.

4.19. definíció (hasítás). *Legyen* $\Sigma = \{0, 1\}$*, valamint legyenek* m *és* t *természetes számok,* $t > m$*. Egy* $h : \Sigma^t \to \Sigma^m$ hasító függvény *lineáris leképezés, ami egy* $(t \times m)$ *Boole-mátrixszal*

adott, mégpedig $B_h = (b_{i,j})_{i,j}$*-vel, ahol* $b_{i,j} \in \{0,1\}$*. Minden* $x \in \Sigma^t$ *és* $1 \leq j \leq m$ *mellett az* $y = h(x) \in \Sigma^m$ *j-edik bitje mint* $y_j = (b_{1,j} \wedge x_1) \oplus (b_{2,j} \wedge x_2) \oplus \cdots \oplus (b_{t,j} \wedge x_t)$ *adódik, ahol* $\oplus$ *a logikai paritásműveletet jelenti, tehát* $a_1 \oplus a_2 \oplus \cdots \oplus a_n = 1 \iff |\{i \mid a_i = 1\}| \equiv 1 \bmod 2$.

Legyen $\mathcal{H}_{t,m} = \{h : \Sigma^t \to \Sigma^m \mid B_h$ *egy* $(t \times m)$ *Boole-mátrix*$\}$ *hasító függvények egy családja, t és m paraméterekkel. A* $\mathcal{H}_{t,m}$*-re egyenletes eloszlást tételezünk fel:* $\mathcal{H}_{t,m}$*-ből egy h hasító függvényt emelünk ki, amelynél a* B_h*-beli* $b_{i,j}$*-ket függetlenül és egyenletes eloszlás mellett választjuk.*

Legyen $V \subseteq \Sigma^t$*. A* $\mathcal{H}_{t,m}$ *egy* $\widehat{\mathcal{H}}$ *részcsaládjára akkor lép fel* ütközés *V-n, ha*

$$(\exists v \in V)\,(\forall h \in \widehat{\mathcal{H}})\,(\exists x \in V)\,[v \neq x \wedge h(v) = h(x)]\ .$$

Egyéb esetekben $\widehat{\mathcal{H}}$ *a V-n* ütközésmentes.

Egy V fölötti ütközés azt jelenti tehát, hogy megsérül egy tetszőlegesen választott $\widehat{\mathcal{H}}$ családbeli hasító függvény injektivitása V-n. A következő lemma kimondja, hogy minden kellően kis V halmazon $\mathcal{H}_{t,m}$ egy véletlenszerűen választott részcsaládja ütközésmentes, ha pedig V túl nagy, az ütközés elkerülhetetlen. A 4.20. lemma bizonyításától eltekintünk.

4.20. lemma (hasítás lemma). *Legyenek* $t, m \in \mathbb{N}$ *paraméterek,* $V \subseteq \Sigma^t$ *és* $\widehat{\mathcal{H}} = (h_1, h_2, \ldots, h_{m+1})$ *egy egyenletes eloszlás mellett véletlenül választott hasító függvény család* $\mathcal{H}_{t,m}$*-ből. Legyen*

$$K(V) = \{\widehat{\mathcal{H}} \mid (\exists v \in V)\,(\forall h \in \widehat{\mathcal{H}})\,(\exists x \in V)\,[v \neq x \wedge h(v) = h(x)]\}$$

$\widehat{\mathcal{H}}$*-ban egy ütközés fellépésének eseménye V-n. Ekkor*

1. *ha* $|V| \leq 2^{m-1}$*, akkor* $K(V)$ *legfeljebb* $1/4$ *valószínűséggel következik be,*
2. *ha* $|V| > (m+1)2^m$*, akkor* $K(V)$ 1 *valószínűséggel következik be.*

A 3.5. alfejezetben definiáltuk az Artúr-Merlin hierarchiát és említettük, hogy ez a hierarchia a második szintjére omlik össze. Számunkra most a coAM osztály érdekes, lásd a 3.16. definíciót.

4.21. tétel (Schöning). *A* `GI` *probléma* Low_2*-beli.*

Bizonyítás. A 4.18. tétel alapján minden coAM-beli NP-halmaz Low_2-beli. `GI` Low_2-beliségének belátásához elégséges tehát azt megmutatni, hogy `GI` coAM-beli. Legyenek G és H n csúccsal rendelkező gráfok. A hasítás lemmát szeretnénk alkalmazni. Ígéretesnek tűnik, hogy a 3.11. lemmában definiált

$$A(G,H) \quad = \quad \{(F,\varphi) \mid F \cong G \text{ és } \varphi \in \mathrm{Aut}(F)\} \cup \{(F,\varphi) \mid F \cong H \text{ és } \varphi \in \mathrm{Aut}(F)\}$$

halmaz a 4.20. lemma V-jének szerepét vegye fel. A 3.11 lemma szerint $|A(G,H)| = n!$ amennyiben $G \cong H$ és $|A(G,H)| = 2n!$, ha $G \ncong H$. Ahhoz, hogy a `GI`-t megoldó coAM-gép polinomiális időben dolgozzon, a hasítás lemma t és m paraméterei n-ed fokú polinomok kell legyenek. A lemma alkalmazásához az $m = m(n)$ polinomot úgy kellene választanunk, hogy

$$n! \leq 2^{m-1} < (m+1)2^m < 2n! \tag{4.4}$$

fennálljon, mivel ekkor lesz a $V = A(G,H)$ halmaz elég nagy ahhoz, hogy elegendően nagy

valószínűséggel meg tudjunk különböztetni két izomorf G és H gráfot két nem izomorf gráftól. Sajnos nem létezik olyan m polinom, hogy a (4.4) egyenlőtlenség fennálljon, ehelyett választunk egy másik V halmazt, amivel a felső és alsó korlátok közötti rést elég naggyá tudjuk tenni.

Legyen $V = A(G,H)^n = \underbrace{A(G,H) \times A(G,H) \times \cdots \times A(G,H)}_{n\text{-szer}}$. Ezzel (4.4) a következőképp alakul:

$$(n!)^n \leq 2^{m-1} < (m+1)2^m < (2n!)^n \ , \tag{4.5}$$

ahol az új egyenlőtlenség kielégíthető az $m = m(n) = 1 + \lceil n \log n! \rceil$ választással.

Készítsünk egy M coAM-gépet GI-hez a következőképp. Az n csúcsú G és H gráfok bemenete mellett M mindenekelőtt kiszámítja az m paramétert. A $V = A(G,H)^n$ halmaz n darab (F,φ) formájú párost tartalmaz, ahol F egy n csúcsú gráf és $\varphi \in \mathrm{Aut}(F)$. Tekintsük a V elemeit mint $t(n)$ hosszú, $\Sigma = \{0,1\}$ ábécé fölötti szavakat, ahol t egy alkalmasan választott polinom. Ekkor M egyenletes eloszlás mellett véletlenszerűen választ egy $\widehat{\mathcal{H}} = (h_1, h_2, \ldots, h_{m+1})$ hasító függvény családot $\mathcal{H}_{t,m}$-ből, ami megfelel Artúr lépésének. Minden $h_i \in \widehat{\mathcal{H}}$ hasító függvényt egy $(t \times m)$ Boole-mátrixszal reprezentálunk, ezért az $m+1$ darab, $\widehat{\mathcal{H}}$-beli h_i hasító függvény ábrázolható mint $p(n)$ hosszú $z_{\widehat{\mathcal{H}}} \in \Sigma^*$ szó, ahol p egy alkalmas polinom. Módosítva a hasítás lemma $K(V)$ ütközés predikátumát

$$B = \{(G,H,z_{\widehat{\mathcal{H}}}) \mid (\exists v \in V)\,(\forall i : 1 \leq i \leq m+1)\,(\exists x \in V)\,[v \neq x \wedge h_i(v) = h_i(x)]\} \ .$$

A B-ben elforduló $\forall$ kvantor csak polinomiálisan sok i-t határoz meg, ezért determinisztikus módon polinomiális időben kiértékelhető, valamint B mindkét $\exists$ kvantora összefogható *egy* darab polinomiális hosszra korlátozott $\exists$ kvantorrá. A 4.16. tétel alapján tehát B egy Σ_1^p = NP-beli halmaz. Legyen N egy NPTM B-hez. A $z_{\widehat{\mathcal{H}}}$ készülő szóhoz – amit $m+1$ függetlenül és egyenletes eloszlás mellett választott hasító függvény reprezentál – M szimulálja az $N(G,H,z_{\widehat{\mathcal{H}}})$ kiszámítását, ami megfelel Merlin lépésének. Az M pontosan akkor fogadja el a (G,H) bemenetét, ha $N(G,H,z_{\widehat{\mathcal{H}}})$ elfogad.

Becsüljük meg (a $z_{\widehat{\mathcal{H}}}$-ban kódolt hasító függvény véletlenszerű választása mellett) annak valószínűségét, hogy M elfogadja (G,H) bemenetét. Ha G és H izomorfak, akkor a 3.11 lemma alapján $|A(G,H)| = n!$. A (4.5) egyenlőtlenségből $|V| = (n!)^n \leq 2^{m-1}$ következik. A 4.20. lemma alapján annak valószínűsége, hogy $(G,H,z_{\widehat{\mathcal{H}}})$ B-beli, következésképp $M(G,H)$ elfogad, legfeljebb 1/4. Ha ezzel szemben G és H nem izomorfak, akkor a 3.11. lemma szerint $|A(G,H)| = 2n!$, a (4.5) egyenlőtlenségből pedig $|V| = (2n!)^n > (m+1)2^m$ adódik. A 4.20. lemma alapján ekkor annak valószínűsége, hogy $(G,H,z_{\widehat{\mathcal{H}}})$ B-beli, tehát $M(G,H)$ elfogad, közel 1. Ebből következik tehát, hogy GI coAM-beli. ■

4.4.3. A gráfizomorfizmus SPP-beli

A 3.3.1. alfejezet 3.14. definíciójában bevezettük az RP valószínűségi osztályt. A következő alfejezetben két további, itt definiált valószínűségi osztály játszik szerepet: PP és SPP, melyek a *Probabilistic Polynomialtime* és a *Stoic Probabilistic Polynomialtime* angol kifejezések rövidítései.

4.22. definíció (PP és SPP). *A* PP ***osztály*** *azokat az A problémákat foglalja magában, melyekhez létezik egy M NPTM úgy, hogy minden x bemenetre: ha* $x \in A$*, akkor* $M(x)$ $1/2$*-nél nem kisebb valószínűség mellett elfogad, és ha* $x \notin A$*, akkor* $M(x)$ *valamely* $1/2$*-nél kisebb valószínűséggel fogad el.*

Jelölje $\mathrm{acc}_M(x)$ *az* $M(x)$ *elfogadó útjainak számát egy M NPTM-nél x bemenet mellett, valamint* $\mathrm{rej}_M(x)$ *az* $M(x)$ *visszautasító útjainak számát. Legyen továbbá* $\mathrm{gap}_M(x) = \mathrm{acc}_M(x) - \mathrm{rej}_M(x)$.

Az SPP *osztály azokból az A problémákból áll, melyekre létezik olyan M NPTM, hogy minden x-re* $(x \in A \Longrightarrow \mathrm{gap}_M(x) = 1)$ *és* $(x \notin A \Longrightarrow \mathrm{gap}_M(x) = 0)$ *teljesül.*

Egy SPP-gép „sztoikus” tehát abban az értelemben, hogy a „rése” – tehát az elfogadó és a visszautasító utak különbsége – mindig csak két értéket vesz fel az exponenciális számú lehetséges érték közül. A PP-vel ellentétben az SPP így egy úgynevezett *ígéret osztály* („promise class”), mivel egy M SPP-gép azt „ígéri”, hogy minden x-re $\mathrm{gap}_M(x) \in \{0, 1\}$ teljesül (lásd még a 4.4-4. gyakorlatot).

Az alsóság fogalmát tetszőleges relativizálható C bonyolultsági osztályra definiálhatjuk: egy A halmaz akkor és csak akkor C*-alsó*, ha $C^A = C$ teljesül. Speciálisan minden k-ra a 4.17. definíció alsóság-hierarchiájának Low_k szintje pontosan azokból az NP-halmazokból áll, melyek Σ_k^p-alsó tulajdonságúak. Minden, a fent definiált SPP osztályba tartozó halmaz PP-alsó. A következő tételben ezek mellett bizonyítás nélkül összefoglalunk néhány hasonló tulajdonságot.

4.23. tétel.

1. *Az* SPP *osztály* PP*-alsó, tehát* $\mathrm{PP}^{\mathrm{SPP}} = \mathrm{PP}$.
2. *Az* SPP *önmagával alsó (angolul „self-low”), tehát* $\mathrm{SPP}^{\mathrm{SPP}} = \mathrm{SPP}$.
3. *Legyen* $A \in \mathrm{NP}$ *egy N NPTM-el egyeztetve és* $L \in \mathrm{SPP}^A$ *egy M NPOTM-mal egyeztetve úgy, hogy az* $\mathrm{M\text{-}A}(x)$ *minden x bemenetre csak olyan q kérdést tesz fel, melyre* $\mathrm{acc}_N(q) \leq 1$ *fennáll. Ekkor L* SPP*-beli.*
4. *Legyen* $A \in \mathrm{NP}$ *egy N NPTM-mel egyeztetve és* $f \in \mathrm{FP}^A$ *egy M DPOTM-ma egyeztetve úgy, hogy* $\mathrm{M\text{-}A}(x)$ *minden x bemenet mellett csak olyan q kérdést tesz fel, melyre* $\mathrm{acc}_N(q) \leq 1$. *Ekkor f* $\mathrm{FP}^{\mathrm{SPP}}$*-beli.*

A következő tétel kimondja, hogy egy jobb oldali co-halmaz lexikografikusan legkisebb permutációja hatékonyan kiszámítható. A $\mathfrak{S}_n$ lexikografikus rendezését a 4.5. példában definiáljuk.

4.24. tétel. *Legyen* $\mathfrak{G} \leq \mathfrak{S}_n$ *egy permutációcsoport* $\mathfrak{G} = \langle G \rangle$ *mellett és* $\pi \in \mathfrak{S}_n$ *egy permutáció. Ekkor létezik olyan algoritmus, amely a* (G, π) *bemenetre polinomiális időben meghatározza a* $\mathfrak{S}_n$*-beli* $\mathfrak{G}$ *jobb oldali co-halmazát,* $\mathfrak{G}\pi$*-t.*

Bizonyítás. A LERC algoritmus a lexikografikusan legkisebb permutáció kiszámítását végzi a $\mathfrak{S}_n$-beli $\mathfrak{G}$ jobb oldali co-halmazában $\mathfrak{G}\pi$-ben, ahol a $\mathfrak{G}$ permutációcsoport egy G generátorfüggvényen keresztül adott, lásd a 3.1.3. pont 3.6. definícióját.

A 3.7. tétel segítségével polinomiális időben számítható az $\mathbf{id} = \mathfrak{G}^{(n)} \leq \mathfrak{G}^{(n-1)} \leq \cdots \leq \mathfrak{G}^{(1)} \leq \mathfrak{G}^{(0)} = \mathfrak{G}$, $\mathfrak{G}$ stabilizátor lánc. Pontosabban fogalmazva minden egyes i-re meghatározzuk $\mathfrak{G}^{(i)}$ jobb oldali T_i reprezentáns rendszerét, ahol $1 \leq i \leq n$, és ebből képezzük $\mathfrak{G}$ egy

erősebb $S = \bigcup_{i=1}^{n-1} T_i$ generátorát. Mivel $\varphi_0 = \pi$ és $\mathfrak{G}^{(n-1)} = \mathfrak{G}^{(n)} = \mathbf{id}$, ezért a bizonyításhoz elég a LERC algoritmus helyességét megmutatni, tehát azt, hogy $0 \leq i \leq n-1$ esetén minden i-re a $\mathfrak{G}^{(i+1)}\varphi_{i+1}$-beli $\mathfrak{G}^{(i)}\varphi_i$ legkisebb permutációját kapjuk.

LERC(G, π)

```
1  számítsuk ki a 𝔊-beli stabilizátorok 𝔊^(n) ≤ 𝔊^(n−1) ≤ ··· ≤ 𝔊^(1) ≤ 𝔊^(0) láncát
2      φ_0 ← π
3      for i ← 0 to n − 1
4          do x ← i + 1
5              számítsuk ki a 𝔊^(i)(x) pályán az y elemet, melyre φ_i(y) minimális
6              határozzunk meg egy τ_i permutációt 𝔊^(i)-ben τ_i(x) = y mellett
7              φ_{i+1} ← τ_i φ_i
8      return φ_n
```

Ebből indukcióval következik, hogy $\mathfrak{G}^{(n)}\varphi_n = \{\varphi_n\}$ a $\mathfrak{G}\pi = \mathfrak{G}^{(0)}\varphi_0$ lexikografikusan legkisebb permutációja. Következésképp tehát a LERC algoritmus valóban $\mathfrak{G}\pi$ lexikografikusan legkisebb φ_n permutációját adja vissza.

A fenti állítás bizonyításához jelöljük egy $\mathfrak{H} \leq \mathfrak{S}_n$ permutációcsoportbeli $x \in [n]$ elemek pályáját $\mathfrak{H}(x)$-el. Legyen τ_i az a $\mathfrak{G}^{(i)}$-beli permutáció, amely a $i+1$-et a $\mathfrak{G}^{(i)}(i+1)$ pályában lévő y-ra képzi le, és amelyre $\varphi_i(y) = x$ a $\{\varphi_i(z) \mid z \in \mathfrak{G}^{(i)}(i+1)\}$ halmaz minimális eleme. A 3.7. tétel alapján a $\mathfrak{G}^{(i)}(i+1)$ pálya polinomiális időben kiszámítható, és mivel $\mathfrak{G}^{(i)}(i+1)$ legfeljebb $n-i$ elemet tartalmaz, ezért y hatékonyan meghatározható. Az algoritmus definíciója alapján fennáll $\varphi_{i+1} = \tau_i\varphi_i$. Mivel $\mathfrak{G}^{(i)}$ minden permutációja minden $[i]$-beli elemet önmagára képez le és mivel $\tau_i \in \mathfrak{G}^{(i)}$, ezért minden j-re, amelyre $1 \leq j \leq i$, minden $\tau \in \mathfrak{G}^{(i)}$-re és minden $\sigma \in \mathfrak{G}^{(i+1)}$-re:

$$(\sigma\varphi_{i+1})(j) = \varphi_{i+1}(j) = (\tau_i\varphi_i)(j) = \varphi_i(j) = (\tau\varphi_i)(j).$$

Ebből speciálisan következik, hogy minden $\mathfrak{G}^{(i)}\varphi_i$-beli μ lexikografikusan legkisebb permutációra fennáll, hogy minden $\mathfrak{G}^{(i+1)}\varphi_{i+1}$-beli permutációnak egyeznie kell μ-vel az első i elemre.

Ezen kívül minden $\sigma \in \mathfrak{G}^{(i+1)}$-re és a fent definiált $x = \varphi_i(y)$ elemre fennáll

$$(\sigma\varphi_{i+1})(i+1) = \varphi_{i+1}(i+1) = (\tau_i\varphi_i)(i+1) = x.$$

Természetesen $\mathfrak{G}^{(i+1)}\varphi_{i+1} = \{\varphi \in \mathfrak{G}^{(i)}\varphi_i \mid \varphi(i+1) = x\}$. Az állítás ezután abból a tényből következik, hogy a $\mathfrak{G}^{(i)}\varphi_i$-beli lexikografikusan legkisebb μ permutációra fennáll $\mu(i+1) = x$.

Ezzel megmutattuk, hogy a LERC algoritmus hatékonyan és helyesen működik. ■

A 4.24. tétel könnyen kibővíthető a 4.25. következménnyé (lásd a 4-3.. feladatot). Ehhez kapcsolódóan bebizonyítjuk az alfejezet fő tételét, a 4.26. tételt.

4.25. következmény. *Legyen $\mathfrak{G} \leq \mathfrak{S}_n$ egy permutációcsoport, ahol $\mathfrak{G} = \langle G \rangle$, valamint legyen π és ψ két permutáció $\mathfrak{S}_n$-ben. Ekkor létezik olyan algoritmus, amely (G, π, ψ) bemenet mellett polinomiális időben meghatározza $\psi\mathfrak{G}\pi$ lexikografikusan legkisebb permutációját.*

4.26. tétel (Arvind és Kurur). *A* GI *probléma* SPP*-beli.*

Bizonyítás. Az AUTO probléma a következőképp definiált: számítsuk ki egy adott G gráfhoz az $\mathrm{Aut}(G)$ automorfizmus-csoportot (a fogalmak értelmezéséhez lásd a 3.6 definíciót és az azt követő bekezdést, valamint a 3.8 definíciót). Mathon eredményei alapján alapján az AUTO és a GI problémák Turing-ekvivalensek, tehát $\mathtt{AUTO} \in \mathrm{FP}^{\mathtt{GI}}$ és $\mathtt{GI} \in \mathrm{P}^{\mathtt{AUTO}}$, ezért elég AUTO $\mathrm{FP}^{\mathrm{SPP}}$-beliségét megmutatni. Ezután SPP 4.23. tételbeli önmagával alsó („self-low") tulajdonságából következik, hogy $\mathtt{GI} \, \mathrm{P}^{\mathtt{AUTO}} \subseteq \mathrm{SPP}^{\mathrm{SPP}} \subseteq$ SPP-beli, amivel a tételt be is láttuk.

Célunk tehát, hogy találjunk AUTO-hoz egy $\mathrm{FP}^{\mathrm{SPP}}$-algoritmust. Ennek az algoritmusnak egy adott G gráf esetén egy $S = \bigcup_{i=1}^{n-1} T_i$ erős generátort kell kiszámítania $\mathrm{Aut}(G)$-hez, ahol

$$\mathbf{id} = \mathrm{Aut}(G)^{(n)} \leq \mathrm{Aut}(G)^{(n-1)} \leq \cdots \leq \mathrm{Aut}(G)^{(1)} \leq \mathrm{Aut}(G)^{(0)} = \mathrm{Aut}(G)$$

az $\mathrm{Aut}(G)$ stabilizátorainak lánca, T_i $(1 \leq i \leq n)$ pedig $\mathrm{Aut}(G)^{(i)}$ egy teljes jobb oldali reprezentáns rendszere $\mathrm{Aut}(G)^{(i-1)}$-ben. A triviális $\mathrm{Aut}(G)^{(n)} = \mathbf{id}$ esettel kezdve lépésről lépésre építünk fel egy erős generátort $\mathrm{Aut}(G)^{(i)}$-hez, csökkenő i mellett, végül tehát kapunk egy erős generátort $\mathrm{Aut}(G)^{(0)} = \mathrm{Aut}(G)$-hez. Tegyük fel tehát, hogy már találtunk egy $S_i = \bigcup_{j=i}^{n-1} T_j$ erős generátort $\mathrm{Aut}(G)^{(0)} = \mathrm{Aut}(G)$-hez. Most leírjuk, hogyan határozza meg az $\mathrm{FP}^{\mathrm{SPP}}$-algoritmus $\mathrm{Aut}(G)^{(i)}$ egy teljes jobb oldali reprezentáns rendszerét $\mathrm{Aut}(G)^{(i-1)}$-ben, T_{i-1}-et. Ehhez definiáljuk az

$$A = \left\{ (G, S, i, j, \pi) \;\middle|\; \begin{array}{l} S \subseteq \mathrm{Aut}(G) \text{ és } \langle S \rangle \text{ egy pontonkénti stabilizátora } [i]\text{-nek} \\ \mathrm{Aut}(G)\text{-ben, } \pi \text{ egy pontonként } [i-1] \text{ által stabilizált} \\ \text{parciális permutáció és } \pi(i) = j, \text{ valamint létezik egy} \\ \tau \in \mathrm{Aut}(G)^{(i-1)}, \text{ ahol } \tau(i) = j \text{ és } \mathrm{LERC}(S, \tau) \text{ a } \pi \text{ bővítése} \end{array} \right\}$$

halmazt. A 4.24. tétel alapján a $\langle S \rangle \tau$ jobb oldali co-halmazhoz tartozó $\mathrm{LERC}(S, \tau)$ lexikografikusan legkisebb permutáció polinomiális időben kiszámítható a LERC algoritmus segítségével. A π parciális permutáció a (G, S, i, j, π) bemenet része, mivel az A halmazt mint orákulumot szeretnénk használni a lexikografikusan legkisebb $\tau \in \mathrm{Aut}(G)^{(i-1)}$, $\tau(i) = j$ melletti permutáció prefixkereséses meghatározásakor (vessük ezt össze a 4.5. példa N-Pre-Iso algoritmusával).

Az N algoritmus egy NPTM A orákulumhoz, tehát A NP-beli. Eldöntendő, hogy ha $\tau(i) = j$, akkor fennáll-e $\langle S \rangle \tau$ jobb oldali co-halmaz minden σ permutációjára $\sigma(i) = j$.

$N(G, S, i, j, \pi)$

1 ellenőrizzük, hogy $S \subseteq \mathrm{Aut}(G)^{(i)}$
2 találjunk ki nemdeterminisztikusan egy $\tau \in \mathfrak{S}_n$ permutációt ▷ G n csúcsú.
3 **if** $\tau \in \mathrm{Aut}(G)^{(i-1)}$ és $\tau(i) = j$ és τ a π bővítése és $\tau = \mathrm{LERC}(S, \tau)$
4 **then** elfogadás és megállás
5 **else** elutasítás és megállás

Most megmutatjuk, hogy N elfogadó útjainak száma (G, S, i, j, π) bemenet mellett vagy 0 vagy 1, amennyiben $\langle S \rangle = \mathrm{Aut}(G)^{(i)}$, valamint általában fennáll

$$\mathrm{acc}_N(G, S, i, j, \pi) \in \left\{ 0, |\mathrm{Aut}(G)^{(i)}| / |\langle S \rangle| \right\} .$$

Először egy pszeudokódot adunk meg.

M-A(G)

```
1  T_i ← {} minden i-re, ahol 0 ≤ i ≤ n − 2                    ▷ G n csúcsú.
2  ▷ T_i az Aut(G)^(i+1) egy teljes jobb oldali reprezentáns rendszere lesz  Aut(G)^(i)-ben.
3  S_i ← ∅ minden i-re, ahol 0 ≤ i ≤ n − 2
4  S_{n−1} ← {id}              ▷ S_i egy erős generátor lesz Aut(G)^(i)-hez.
5  for i ← n − 1 downto 1
6      do   ▷ Az i-edik iteráció kezdetén S_i-t már megtaláltuk, most S_{i−1}-et számítjuk ki.
7         legyen π : [i − 1] → [n] parciális permutáció π(a) = a-val minden a ∈ [i − 1]-re
8         ▷ i = 1 esetén π a definiálatlan parciális permutáció.
9         for j ← i + 1 to n
10            do π̂ ← πj                  ▷ Tehát π̂ bővíti π-t az (i, j) párral π̂(i) = j mellett.
11               if (G, S_i, i, j, π̂) ∈ A
12                  then ▷ A prefixkeresés előállítja a legkisebb Aut(G)^(i−1)-beli,
                            i-ről j-re képező permutációt.
13                       for k ← i + 1 to n
14                          do keressük meg az ℓ elemet, π̂ leképezésein kívül,
                               (G, S_i, i, j, π̂ℓ) ∈ A mellett
15                             π̂ ← π̂ℓ
16                       ▷ π̂ most egy teljes 𝔖_n-beli permutáció.
17                       T_{i−1} ← T_{i−1} ∪ π̂
18        ▷ T_{i−1} most az Aut(G)^(i) egy teljes jobb oldali reprezentáns
            rendszere Aut(G)^(i−1)-ben.
19        S_{i−1} ← S_i ∪ T_{i−1}
20 return S_0                                  ▷ S_0 egy erős generátor Aut(G) = Aut(G)^(0)-hoz.
```

Tegyük fel, hogy (G, S, i, j, π) A-beli. Ha $\tau(i) = j$ egy $\tau \in \mathrm{Aut}(G)^{(i-1)}$ mellett és $j > i$, akkor az $\langle S \rangle \tau$ jobb oldali co-halmaz pontosan azokból az $\mathrm{Aut}(G)^{(i-1)}$-beli permutációkból áll, melyek i-t j-re képezik le. Következésképp $N(G, S, i, j, \pi)$ egyetlen elfogadó útja megfelel az egyértelműen meghatározott lexikografikusan legkisebb $\tau = \mathrm{LERC}(S, \tau)$ permutációnak. Ha viszont $\langle S \rangle$ az $\mathrm{Aut}(G)^{(i)}$ egy valódi alcsoportja, akkor $\mathrm{Aut}(G)^{(i)}\tau$ ábrázolható az $\langle S \rangle$ $k = |\mathrm{Aut}(G)^{(i)}|/|\langle S \rangle|$ darab diszjunkt jobb oldali co-halmazának egyesítéseként. Általában tehát $N(G, S, i, j, \pi)$ akkor rendelkezik k elfogadó úttal, ha (G, S, i, j, π) A-beli, egyébként pedig nincs elfogadó útja.

Az M-A egy FP^A-algoritmus `AUTO` megoldásához. Itt az M DPOTM csak olyan $q = (G, S_i, i, j, \pi)$ kérdéseket tesz fel az A orákulumnak, melyekre $\langle S_i \rangle = \mathrm{Aut}(G)^{(i)}$, következésképp minden valóban feltett q kérdésre fennáll $\mathrm{acc}_N(q) \leq 1$. A 4.23. tétel 4. része alapján ebből $\texttt{AUTO} \in \mathrm{FP}^{\mathrm{SPP}}$ következik.

Az állítást – miszerint M-A(G) S_0 kimenete egy erős generátor $\mathrm{Aut}(G) = \mathrm{Aut}(G)^{(0)}$-hoz – n szerinti teljes indukcióval mutatjuk meg. Az indukció kezdete az $n-1$, $S_{n-1} = \{\mathrm{id}\}$ pedig természetesen $\mathrm{Aut}(G)^{(n-1)} = \mathbf{id}$-t állítja elő. Az indukciós lépéshez feltesszük, hogy az i-edik iteráció kezdetekor már rendelkezésre áll egy S_i erős generátor $\mathrm{Aut}(G)^{(i)}$-hez. Megmutatjuk, hogy az i-edik iteráció végére az $S_{i-1} = S_i \cup T_{i-1}$ halmaz egy erős generátor $\mathrm{Aut}(G)^{(i-1)}$-hez. Minden j-hez, ahol $i + 1 \leq j \leq n$ a „$(G, S_i, i, j, \hat{\pi}) \in A$?” kérdéssel megvizsgáljuk,

hogy $\mathrm{Aut}(G)^{(i-1)}$-ben létezik-e olyan permutáció, amely i-t j-re képezi le. A rákövetkező perfixkeresés az A orákulumhoz intézett megfelelő kérdésekkel előállítja a lexikografikusan legkisebb $\hat{\pi}$ permutációt $\mathrm{Aut}(G)^{(i-1)}$-ben $\hat{\pi}(i) = j$ mellett. Mint fent megállapítottuk, ekkor A-hoz csak $\mathrm{acc}_N(q) \leq 1$-nek megfelelő q kérdéseket intézünk, mivel S_i egy erős generátor $\mathrm{Aut}(G)^{(i)}$-hez, tehát $\langle S_i \rangle = \mathrm{Aut}(G)^{(i)}$. A permutáció előállítása után az i-edik iteráció végén T_{i-1} $\mathrm{Aut}(G)^{(i)}$ egy teljes jobb oldali átlója $\mathrm{Aut}(G)^{(i-1)}$-ben, következésképp $S_{i-1} = S_i \cup T_{i-1}$ egy erős generátor $\mathrm{Aut}(G)^{(i-1)}$-hez. Végül n iteráció után megtalálunk egy S_0 erős generátort $\mathrm{Aut}(G) = \mathrm{Aut}(G)^{(0)}$-hoz. ■

A 4.23. tétel első két állításából közvetlenül adódik a 4.27. következmény.

4.27. következmény. *$\mathtt{GI}$ alsó* SPP*-re és* PP*-re, tehát* $\mathrm{SPP}^{\mathtt{GI}} = \mathrm{SPP}$ *és* $\mathrm{PP}^{\mathtt{GI}} = \mathrm{PP}$.

Gyakorlatok

4.4-1. A 4.14. definíció alapján fennáll $A \leq_{\mathrm{T}}^{\mathrm{p}} B \iff A \in \mathrm{P}^B$. Mutassuk meg, hogy $A \leq_{\mathrm{T}}^{\mathrm{p}} B \iff \mathrm{P}^A \subseteq \mathrm{P}^B$.

4.4-2. Mutassuk meg, hogy a 4.5. példában definiált Pre-Iso halmaz NP-beli, valamint hogy az N-Pre-Iso algoritmus N gépe egy DPOTM, tehát polinomiális időben dolgozik.

4.4-3. Határozzuk meg a 4.6. példában definiált $\widehat{G}$ és $\widehat{H}$ gráfokhoz az $\mathrm{Iso}(\widehat{G}, \widehat{H})$ izomorfizmuscsoportot.

4.4-4. A következő osztályok közül melyek „ígéret”-osztályok: NP és coNP, RP és coRP, AM és coAM, MA és coMA? Tartozhatnak-e az „ígéret”-osztályokhoz teljes problémák? A válaszokat indokoljuk.

Feladatok

4-1. Erős NPOTM

Egy *erős NPOTM* egy három végállapot típussal rendelkező NPOTM, vagyis a végállapotok F halmaza F_a, F_r és $F_?$ halmazokra bontható. Ekkor teljesül, hogy amennyiben $x \in A$, akkor $M^B(x)$ rendelkezik egy olyan úttal, ami egy F_a-beli állapotban végződik, és nem rendelkezik olyan úttal, amely F_r-beli állapotban végződik. Ha azonban $x \notin A$, akkor $M^B(x)$ rendelkezik egy olyan úttal, ami egy F_r-beli állapottal végződik, és nem rendelkezik olyan úttal, ami F_a-beli állapottal végződik. Az $M^B(x)$ mindkét esetben végződhet bizonyos utakon $F_?$-beli állapotban, ami megfelel a „nem tudom” válasznak. Az erős NPOTM tehát olyan gép, ami sosem hazudik. Bizonyítsuk a következő két állítást:

- $A \leq_{\mathrm{sT}}^{\mathrm{NP}} B \iff$ létezik M erős NPOTM, mellyel $A = L(M^B)$.
- $A \leq_{\mathrm{sT}}^{\mathrm{NP}} B \iff \mathrm{NP}^A \subseteq \mathrm{NP}^B$.

Útmutatás. Általánosítsuk a 4.4-1. gyakorlatot.

4-2. Bizonyítás

Bizonyítsuk a 4.16. és 4.18. tételek állításait.

4-3. Bizonyítás módosítása

Módosítsuk a 4.24. tétel bizonyítását úgy, hogy abból adódjon a 4.25. következmény.

Algoritmus	Típus	3-SAT	4-SAT	5-SAT	6-SAT
vISSZALÉPÉS	det.	$O(1.913^n)$	$O(1.968^n)$	$O(1.987^n)$	$O(1.995^n)$
Monien és Speckenmeyer [45]	det.	$O(1.618^n)$	$O(1.839^n)$	$O(1.928^n)$	$O(1.966^n)$
Dantsin et al. [13]	det.	$\boldsymbol{O(1.481^n)}$	$\boldsymbol{O(1.6^n)}$	$\boldsymbol{O(1.667^n)}$	$\boldsymbol{O(1.75^n)}$
Paturi et al. [47]	val.	$O(1.362^n)$	$O(1.476^n)$	$\boldsymbol{O(1.569^n)}$	$\boldsymbol{O(1.637^n)}$
Schöning [60]	val.	$O(1.334^n)$	$O(1.5^n)$	$O(1.6^n)$	$O(1.667^n)$
Iwama és Tamaki [31]	val.	$\boldsymbol{O(1.324^n)}$	$\boldsymbol{O(1.474^n)}$	—	—

4.9. ábra. Néhány – a kielégíthetőség problémát megoldó – algoritmus futási ideje.

Megjegyzések a fejezethez

A 3. és 4. fejezetek egyes részei a [56] könyvön alapulnak. Amit itt csak erősen tömörített formában tudtunk bemutatni, ott megtalálható átfogó és minden technikai részletre kiterjedő változatban, terjedelmes példákkal, nagyobb számokkal, szebb és több ábrával, pontosabb magyarázatokkal és kidolgozottabb bizonyításokkal. Ilyen módon megtalálhatók [56]-ben olyan bizonyítások, melyekről itt most lemondtunk, mint a 4.16., 4.18. és 4.23. tételek, valamint a 4.20. lemma bizonyításai. Ehhez képest a 3. és ezen 4. fejezet előnye, hogy rövid, feszes és tömör, mindazonáltal érthető és pontos.

Bonyolultságelmélethez nagyobb háttéranyag található például a [29, 46, 72, 73] könyvekben. A Turing-gépet Alan Turing vezette be úttörő munkájában [70]. A 4.4. és a 4.5. táblázatot Garey és Johnson [16] készítették. A klasszikus [16] az NP-teljesség elméletéhez még ma is értékes forrásnak számít. A 4.7. tételnél említett több ezer NP-teljes probléma egy több száz darabos gyűjteménye szintén megtalálható Garey és Johnson [16] könyvében. A 4.10. tétel bizonyítása megtalálható más könyvekben is, például [16] és [46]. A 3-SAT probléma determinisztikus időbonyolultságánál említett oszd-meg-és-uralkodj algoritmust Monien [45], a lokális keresésen alapuló megoldást pedig Dantsin és társai [13] publikálták. A véletlen séta algoritmus Schöning [60, 62] munkáin alapszik. A $\leq_T^p$-visszavezethetőséget Cook [10], a $\leq_m^p$-visszavezethetőséget Karp [32] vezették be. Ladner, Lynch és Selman [38] cikke átfogó és mélyreható mű a bonyolultság-korlátozott visszavezethetőségek tanulmányozásához. A 4.4-1. gyakorlat, valamint a 4-1. feladat alapjait Selman [63] cikke jelenti.

Az eddigi legjobb $O(1.481^n)$ felső korlátot Dantsin és társai [13] érték el k-SAT determinisztikus időbonyolultságára $k \geq 3$ esetén. Schöning itt bemutatott valószínűségi algoritmusa a „korlátozott lokális keresés ismétléssel” [60] ötleten alapszik. A k-SAT problémához $k \geq 4$ mellett Paturi és társai [47] algoritmusa kissé még jobb. A jelenlegi legjobb valószínűségi algoritmus 3-SAT-hoz és 4-SAT-hoz Iwama és Tamaki [31] munkáján alapul. Az ő algoritmusuk ügyesen ötvözi Paturi et al. [47] és Schöning [60] algoritmusait, és megközelítőleg $O(1.324^n)$ futási idővel rendelkezik. A k-SAT ezen algoritmusa $k \geq 5$ mellett nem jobb mint Paturi et al. [47] algoritmusa.

A 4.9. ábra áttekintést nyújt a fejezetben tárgyalt és néhány további kielégíthetőség problémát megoldó algoritmusról. A jelenlegi legjobb algoritmusok félkövér szedéssel szerepelnek.

A gráfizomorfizmus problémával Köbler, Schöning és Torán [35] könyve foglalko-

zik átfogó jelleggel, elsősorban bonyolultságelméleti szempontok szerint. Hoffman [30] csoportelméleti algoritmusokat vizsgált GI-hoz. A 4.11. tételt Ladner [37] publikálta, az alacsony- és magas bonyolultsági osztályhierarchiákat Schöning [58] vezette be. A 4.13. tételnél használt lexikografikus rendezés megtalálható Rothe [56] cikkében. Gál et al. [17] mutatták meg, hogy teljes izomorfizmus konstruálható $O(\lg n)$ méretű parciális izomorfizmusból két, egyenként n csúcsú izomorf gráfhoz. Ezt az eredményt a 4.13. tétel optimálisra javítja fel, kimondja, hogy ehhez már egy 1 méretű parciális izomorfizmus is elegendő. Ezen tétel, valamint a 4.6. példa Größe, Rothe és Wechsung [24] munkájára alapszik. A polinomiális idő-hierarchiát Meyer és Stockmeyer [42, 69] vezették be, akik bizonyították többek között a 4.16. tétel állításait. Az alsó- és felső-hierarchiát Schöning vezette be [58], valamint bizonyította a 4.18. tétel állításait [58] és megmutatta, hogy GI Low_2-beli [59]. Köbler et al. [33, 34] közöltek először eredményeket GI alsóságára vonatkozóan valószínűségi osztályok, mint amilyen PP ellenében. Eredményeiket Arvind és Kurur [2] javították, akik bebizonyították, hogy GI még SPP-beli is. Az univerzális hasítás módszerét Carter és Wegman [9] írták le 1979-ben. A 4.20. lemma Carter és Wegman [9] munkáján alapszik. Az SPP a Valiant [71] által bevezetett UP osztály általánosítása. Az SPP és más „ígéret” osztályok alapos tanulmányozásával számos munka foglalkozik, például [2, 7, 15, 26, 33, 34, 51, 55], a 4.23 tételben felsorolt tulajdonságok is megtalálhatók ezen művekben [15, 34, 35]. A 4.26 tétel bizonyításához szükséges Turing-ekvivalenciát AUTO és GI problémák közt Mathlon [41] mutatta meg (lásd még [35]).

Magyar nyelvű, bonyolultságelmélettel foglalkozó szakkönyvek Cormen, Leiserson, Rivest és Stein [12], Lovász [40], Papadimitriou [46], valamint Rónyai Lajos, Ivanyos Gábor és Szabó Réka [53] műve.

A szerző köszönetet mond Uwe Schöning-nek hasznos észrevételeiért ezen fejezet egy korábbi változatához. Leginkább Uwe Schöning előadásfóliáin alapul a 4.3. alfejezet VÉLETLENÍTETT-SAT algoritmusának valószínűségi elemzése, amely az eredeti érvelés egy egyszerűsítése. Ennek részletesebb elemzése olvasható könyvében [61]. Ezen kívül hálás köszönet illeti Dietrich Stoyant, Sigurd Assingt és Holger Spakowskit a 3. és 4. fejezetek korábbi változatainak korrektúra-olvasásáért. Támogatta a szerzőt a Német Kutatási Társaság (Deutsche Forschungsgemeinschaft – DFG) a RO 1202/9-1 azonosítószám alatt.

Irodalomjegyzék

[1] M. Agrawal, N. Kayal, N. M. Saxena. PRIMES is in P. Available at http://www.cse.iitk.ac.in/users/manindra/primality.ps, 2002. 123

[2] V. Arvind, P. Kurur. Graph isomorphism is in SPP. In *Proceedings of the 43rd IEEE Symposium on Foundations of Computer Science*, 743–750. o. IEEE Computer Society Press, 2002. 160

[3] L. Babai. Trading group theory for randomness. In *Proceedings of the 17th ACM Symposium on Theory of Computing*, 421–429. o. ACM Press, 1985. 123, 124

[4] L. Babai, S. Moran. Arthur-Merlin games: A randomized proof system, and a hierarchy of complexity classes. *Journal of Computer and Systems Sciences*, 36(2):254–276, 1988. 123, 124

[5] A. Beygelzimer, L. Hemaspaandra, C. Homan, J. Rothe. One-way functions in worst-case cryptography: Algebraic and security properties are on the house. *SIGACT News*, 30(4):25–40, 1999. 124

[6] D. Boneh. Twenty years of attacks on the RSA cryptosystem. *Notices of the AMS*, 46(2):203–213, 1999. 123, 124

[7] B. Borchert, L. Hemaspaandra, J. Rothe. Restrictive acceptance suffices for equivalence problems. *London Mathematical Society Journal of Computation and Mathematics*, 86:86–95, 2000. 160

[8] L. Buttyán, I. Vajda. *Kriptográfia és alkalmazásai (Cryptography and its Applications)*. Typotex, 2004. 124

[9] J. L. Carter, M. N. Wegman. Universal classes of hash functions. *Journal of Computer and System Sciences*, 18(2):143–154, 1979. 160

[10] S. Cook. The complexity of theorem proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, 151–158. o. ACM Press, 1971. 159

[11] D. Coppersmith. Small solutions to polynomial equations, and low exponent RSA vulnerabilities. *Journal of Cryptology*, 10(4):233–260, 1997. 123

[12] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein. *Introduction to Algorithms*. The MIT Press/McGraw-Hill, 2004 (Második kiadás ötödik, javított utánnyomása. Magyarul: *Új algoritmusok*. Scolar Kiadó, 2003). 160

[13] E. Dantsin, A. Goerdt, R. Kannan, J. Kleinberg, C. Papadimitriou, P. Raghavan, U. Schöning. A deterministic $(2-2/(k+1))^n$ algorithm for k-sat based on local search. *Theoretical Computer Science*, 289(1):69–83, 2002. 159

[14] W. Diffie, M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976. 123

[15] S. Fenner, L. Fortnow, S. Kurtz. Gap-definable counting classes. *Journal of Computer and System Sciences*, 48(1):116–148, 1994. 160

[16] M. R. Garey, D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979. 159

[17] A. Gál, S. Halevi, R. Lipton, E. Petrank. Computing from partial solutions. In *Proceeding of the 14th Annual IEEE Conference on Computational Complexity*, 34–45. o. IEEE Computer Society Press, 1999. 160

[18] O. Goldreich. Randomness, interactive proofs, and zero-knowledge – a survey. In R. Herken (szerkesztő), *The Universal Turing Machine: A Half-Century Survey*, 377–405. o. Oxford University Press, 1988. 124

[19] O. Goldreich. *Foundations of Cryptography*. Cambridge University Press, 2001. 124

[20] O. Goldreich, S. Micali, A. Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM*, 38(3):691–729, 1991. 123

[21] S. Goldwasser. Interactive proof systems. In J. Hartmanis (szerkesztő), *Computational Complexity Theory*, *AMS Short Course Lecture Notes: Introductory Survey Lectures. Proceedings of Symposia in Applied Mathematics* 38. kötete, 108–128. o. American Mathematical Society, 1989. 124

[22] S. Goldwasser, S. Micali, C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989. 123, 124

[23] S. Goldwasser, M. Sipser. Private coins versus public coins in interactive proof systems. In S. Micali (szerkesztő), *Randomness and Computation*, *Advances in Computing Research* 5. kötete, 73–90. o. JAI Press, 1989. A preliminary version appeared in *Proc. 18th Ann. ACM Symp. on Theory of Computing*, 1986, pp. 59–68. 123

[24] A. Große, J. Rothe, G. Wechsung. Computing complete graph isomorphisms and hamiltonian cycles from partial ones. *Theory of Computing Systems*, 35(1):81–93, 2002. 160

[25] J. Håstad. Solving simultaneous modular equations of low degree. *SIAM Journal on Computing*, 17(2):336–341, 1988. Special issue on cryptography. 123

[26] J. Hartmanis, L. Hemachandra. Complexity classes without machines: On complete languages for UP. *Theoretical Computer Science*, 58(1–3):129–142, 1988. 160

[27] L. Hemaspaaandra, K. Pasanen, J. Rothe. If P ≠ NP then some strongly noninvertible functions are invertible. In *Proceedings of the 13th International Symposium on Fundamentals of Computation Theory*, *Lecture Notes in Computer Science* 2138. kötete, 162–171. o. Springer-Verlag, 2001. 123, 124

[28] L. Hemaspaandra, J. Rothe. Creating strong, total, commutative, associative one-way functions from any one-way function in complexity theory. *Journal of Computer and Systems Sciences*, 58(3):648–659, 1999. 123, 124

[29] L. A. Hemaspaandra, M. Ogihara. *The Complexity Theory Companion*. EATCS Texts in Theoretical Computer Science. Springer-Verlag, 2002. 159

[30] C. Hoffman (szerkesztő). *Group-Theoretic Algorithms and Graph Isomorphism*, *Lecture Notes in Computer Science* 136. kötete. Springer-Verlag, 1982. 160

[31] K. Iwama, S. Tamaki. Improved upper bounds for 3-SAT. Technical Report TR03-053, Electronic Colloquium on Computational Complexity, 2003. 159

[32] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller, J. W. Thatcher (szerkesztők), *Complexity of Computer Computations*, 85–103. o. Plenum Press, 1972. 159

[33] J. Köbler, U. Schöning, S. Toda, J. Torán. Turing machines with few accepting computations and low sets for PP. *Journal of Computer and System Sciences*, 44(2):272–286, 1992. 160

[34] J. Köbler, U. Schöning, J. Torán. Graph isomorphism is low for PP. *Computational Complexity*, 2:301–330, 1992. 123, 160

[35] J. Köbler, U. Schöning, J. Torán. *The Graph Isomorphism Problem: Its Structural Complexity*. Birkhäuser, 1993. 124, 159, 160

[36] J. Ködmön. *Kriptográfia: Az informatikai biztonság alapjai, a Pgp kriptorendszer használata (Cryptography: Bases of the Security of Computer Systems. Use of Cryptosystem Pgp)*. Computerbooks, 2002. 124

[37] R. Ladner. On the structure of polynomial time reducibility. *Journal of the ACM*, 22(1):155–171, 1975. 160

[38] R. Ladner, N. Lynch, A. Selman. A comparison of polynomial time reducibilities. *Theoretical Computer Science*, 1(2):103–124, 1975. 159

[39] A. Lenstra, H. Lenstra. *The Development of the Number Field Sieve*, *Lecture Notes in Mathematics* 1554. kötete. Springer-Verlag, 1993. 123

[40] L. Lovász. *Algoritmusok bonyolultsága (Complexity of Algorithms)*. ELTE TTK, 1990. 160

[41] R. Mathon. A note on the graph isomorphism counting problem. *Information Processing Letters*, 8(3):131–132, 1979. 160

[42] A. Meyer, L. Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential space. In *Proceedings of the 13th IEEE Symposium on Switching and Automata Theory*, 125–129. o., 1972. 160

[43] D. Micciancio, S. Goldwasser. *Complexity of Lattice Problems: A Cryptographic Perspective*, *The Kluwer International Series in Engineering and Computer Science* 671. kötete. Kluwer Academic Publishers, 2002. 123

[44] G. L. Miller. Riemann's hypothesis and tests for primality. *Journal of Computer and Systems Sciences*, 13(3):300–317, 1976. 123

[45] B. Monien, E. Speckenmeyer. Solving satisfiability in less than 2^n steps. *Discrete Applied Mathematics*, 10:287–295, 1985. 159

[46] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994 (Magyarul: *Kiszámítási bonyolultság*, Novadat, 1999). 124, 159, 160

[47] R. Paturi, P. Pudlák, M. Saks, F. Zane. An improved exponential-time algorithm for k-SAT. In *Proceedings of the 39th IEEE Symposium on Foundations of Computer Science*, 628–637. o. IEEE Computer Society Press, 1998. 159

[48] J. M. Pollard. Theorems on factorization and primality testing. *Proceedings of the Cambridge Philosophical Society*, 76:521–528, 1974. 123

[49] M. Rabi, A. Sherman. An observation on associative one-way functions in complexity theory. *Information Processing Letters*, 64(5):239–244, 1997. 124

[50] M. O. Rabin. Probabilistic algorithms for testing primality. *Journal of Number Theory*, 12(1):128–138, 1980. 123

[51] R. Rao, J. Rothe, O. Watanabe. Upward separation for FewP and related classes. *Information Processing Letters*, 52(4):175–180, 1994 (Corrigendum appears in the same journal,74(1–2):89, 2000). 160

[52] R. L. Rivest, A. Shamir, L. M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978. Lásd még az U.S. Patent 4 405 829 számú szabadalmat. 123

[53] L. Rónyai, G. Ivanyos, R. Szabó. *Algoritmusok* (*Algorithms*). Typotex, 1999. 160

[54] J. Rothe. Some facets of complexity theory and cryptography: A five-lecture tutorial. *ACM Computing Surveys*, 34(4):504–549, 2002. 123, 124

[55] J. Rothe. A promise class at least as hard as the polynomial hierarchy. *Journal of Computing and Information*, 1(1):92–107, 1995. 160

[56] J. Rothe. *Complexity Theory and Cryptology. An Introduction to Cryptocomplexity*. EATCS Texts in Theoretical Computer Science. Springer-Verlag, 2005. 159, 160

[57] A. Salomaa. *Public-Key Cryptography*, *EATCS Monographs on Theoretical Computer Science* 23. kötete. Springer-Verlag, 2. kiadás, 1996. 123, 124

[58] U. Schöning. A low and a high hierarchy within NP. *Journal of Computer and System Sciences*, 27:14–28, 1983. 160

[59] U. Schöning. Graph isomorphism is in the low hierarchy. *Journal of Computer and System Sciences*, 37:312–323, 1987. 160

[60] U. Schöning. A probabilistic algorithm for k-SAT based on limited local search and restart. In *Proceedings of the 40th IEEE Symposium on Foundations of Computer Science*, 410–414. o. IEEE Computer Society Press, 1999. 159

[61] U. Schöning. *Algorithmik*. Spektrum Akademischer Verlag, 2001. 160

[62] U. Schöning. *Ideen der Informatik*. Oldenbourg Verlag, 2002. 159

[63] A. Selman. Polynomial time enumeration reducibility. *SIAM Journal on Computing*, 7(4):440–457, 1978. 159

[64] A. Shamir. IP = PSPACE. *Journal of the ACM*, 39(4):869–877, 1992. 123

[65] C. Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, 28(4):657–715, 1949. 123

[66] S. Singh. *The Code Book. The Secret History of Codes and Code Breaking*. Fourth Estate, 1999 (Magyarul: *Kódkönyv. A rejtjelezés és rejtjelfejtés története*. Park Könyvkiadó, 2002). 123, 124

[67] R. Solovay, V. Strassen. A fast Monte Carlo test for primality. *SIAM Journal on Computing*, 6:84–85, 1977. Erratum appears in the same journal, 7(1):118, 1978. 123

[68] D. Stinson. *Cryptography: Theory and Practice*. CRC Press, 2002 (2. kiadás). 123, 124

[69] L. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3(1):1–22, 1977. 160

[70] A. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society, ser. 2*, 2:230–265, 1936 (Correction, *ibid*, vol. 43, pp. 544–546, 1937). 159

[71] L. G. Valiant. The relative complexity of checking and evaluating. *Information Processing Letters*, 5(1):20–23, 1976. 160

[72] K. Wagner, G. Wechsung. *Computational Complexity*. D. Reidel Publishing Company, 1986 (and Kluwer Academic Publishers, 2001). 159

[73] G. Wechsung. *Vorlesungen zur Komplexitätstheorie*, 32. kötet. B. G. Teubner Verlagsgesellschaft, 2000. 159

[74] S. Zachos, H. Heller. A decisive characterization of BPP. *Information and Control*, 69:125–135, 1986. 124

Tárgymutató

Névmutató

Tartalomjegyzék