Chapter 1

TeXelés

Elkezdtem véglegesen megoldani a nehéz laTeX problémát, hogy három különböző környezetben háromszor inkludálom a complexity.txt-t, és először csak feladatokat, aztán csak hinteket, aztán feladatokat és megoldásokat szeretnék látni, persze azonos számozással.

Egy compactForm nevű boolean beállításától függ, hogy ezt a háromszoros, bőbeszédű, végleges változatot fordítja le, vagy egy tömör, javítgatáshoz sokkal praktikusabb változatot.

Ismert hibák: A (*) hülyén eltávolodott a feladat sorszámától. Teljesen átnézendő, hogy mi és mi között mennyi vspace-nek kellene lennie, és most mennyi van ehelyett. Ha többet segítek, akkor azok azonos feladatszámot kapnak. (Ne segítsek többet.) Ha egy feladatnak több részfeladata van, akkor az (a) legyen egy sorban a feladat számával.

Az index-szel még egyáltalán semmit sem csináltam, pedig biztos ez is felveti majd a saját problémáit. A tételjegyzékről és a referenciákról még el sem döntöttem, hogy lesz-e, és hogyan.

Jelenleg lustaságból azok a fejezetek is megháromszorozódnak, amiknek nem kellene. Ki lehet ezt javítani új fájl bevezetése és a framecode megterhelése nélkül? Persze az új fájl bevezetése még jó is lehet: Ez lenne a self-documentation.

A nyelvosztályokra készített makróm csúnyán bánik a kötőjelekkel: A legcsúnyább a $\mathbf{co-NP}$, és pár másik co-, rájuk kell keresni. Bezzeg a CO-NP-vel semmi baj.

Az ... a tuti ellipszis?

Az $\overline{x_i}$ nem szép. Hogy csinálják ezt rendesen?

Rajtam kívül mindenki $\mathbf{\Sigma_2^p}$-val jelöli $\mathbf{\Sigma_2}$-t. És a p-t dőlt betűvel szokták

írni.

Chapter 2

Szándékok

Három részből áll a könyv. Feladatok-segítségek-megoldások. A megoldásos résznél megismételjük a feladat szövegét, a segítségesnél nem. Valami okosat ki kellene találni arra a helyzetre, amikor többször is akarok segíteni egy feladathoz. Lehet, hogy szélességi bejárás szerint sorolom fel a hinteket, és oda-vissza keresztreferenciákat teszek rájuk? Na ehhez komoly laTeX guru kellhet.

Kati szerint több segítés helyett mégiscsak legyen feladatra bontás. Az emelletti érvek teljesen agyonnyomják az én ösztönös averziómat.

Kati szerint legyen egy Jelölések fejezet, meg egy „felhasznált tételek" fejezet.

A Feladatok számozása a tételek szokásos számozásával azonos, fejezetszám.sorszám formában. A Definíciókat nem számozzuk. Lehetségesek továbbá a feladatok közti összekötő szövegek és kommentárok, (prochat és solchat parancs), de ezzel az eszközzel ritkán fogok élni.

A feladatok állítását tipikusan csak kijelentjük, „Bizonyítsuk be, hogy..." jellegű előtagokat nem használva. Eddig ettől az elvtől eltértem ott, ahol az előző mondatban egyszerhasználatos definíciót vezettem be, és így akartam megkülönböztetni a két kijelentő mondat szemantikáját. Például: DOUBLE-SAT a legalább két különböző kielégítéssel rendelkező Boole-formulák osztálya. Bizonyítsuk be, hogy DOUBLE-SAT **NP**-teljes.

A feladatok négy nehézségi szintbe vannak sorolva: (♡) : könnyű, definíció megértését ellenőrző. Az alap, ami nincs külön jelölve. (*) : nehéz. (**) : nagyon nehéz. Mindet át kell nézni, de a könnyű az, amit utólag vezettem be, és ezért még általában nincs kitéve.

Kati szerint az az állítás, amire egy közepes vagy gyengébb képességű

hallgató visszakérdezne, leírandó, hiszen itt nincs mód a visszakérdezésre.

A feladatok megoldásában nem hivatkozhatok a hintekben bevezetett fogalmakra és állításokra. A megoldások és hintek nem hivatkozhatnak más feladatoknak semmijére sem, beleértve a kitűzést is. Kati szerint hivatkozhassunk a feladatokra. A megoldásokban a korábbi feladatok kimondott állításait használhatnánk, meg például kijelenthetnénk, ha két feladat között valami viszonyulás van (alsó becslés-felső becslés).

Ha az állítás valakinek a tétele (Borogyin, Kannan), akkor azt illene odaírni.

A használt fogalmak közül azokra lesz definíció, amelyek nem szerepelnek a Lovászban. A jelölések ahol lehet, hozzá igazítandóak.

A könnyebb kérdések válaszait kicsit részletesebben, pedánsabban írjuk le. Ez egyrészt azért célszerű, mert gyakran nem pontosan ugyanannak a közönségnek lesznek feladva. Másrészt vannak olyan technikai bonyodalmak, amelyeknek a részletes feldolgozása egy nehéz feladat fókuszáról elvonná a figyelmet, de egy könnyűnél a nagyvonalú elhanyagolásuk triviálissá tenné a feladatot. (A Kolmogorov-bonyolultság fejezetben még így is szőnyeg alá söpörtük a prefix kódolás kérdését.)

A **LOGSPACE** versus **NL** jelölés következetlen, de az L gyakori használata indokolttá teszi. A gyakori „adjunk **LOGSPACE** algoritmust" mondatséma miatt sem lenne triviális javítani.

Bele kellene írni, hogy minden $\log n$ kettő alapú.

Rögzíteni kellene, hogy a Boole-függvények tárgyalásakor a 0-t a hamis, az 1-et az igaz szinonímájaként használom, akár mondat közben is szemrebbenés nélkül váltogatva a két jelölést.

Katiék könyvében a definíció vizuálisan kiemelkedik (egy kis krikszkraksszal.)

A definíciókban a definiált szó legyen dőlt betűs (vagy félkövér?).

Kéne bele párhuzamos bonyolultságelmélet.

Teljesen lerabolni az Arora-Barak könyvet: http://www.cs.princeton.edu/theory/complexity/ . (Brutális nagy munka lesz.)

Chapter 3

Bevezetés

(*Megírni.*)

Chapter 4

Döntési fák

1. A „körmentesség” gráftulajdonság zárkózott. (*Valahogy le kell szögezni, hogy ez és a következő iksz feladat mind irányítatlan gráfra vonatkozik. Lehet, hogy majd mindegyikbe beleírom, hogy „Az X irányítatlan gráftulajdonság zárkózott”, de valószínűleg nem ez a tuti megoldás.*)

Segítség: (*Le kell írni, hogy mi az az ördög-módszer? Kati szerint igen, legalábbis informálisan. Ha a textbookban nincsen leírva, akkor pontosabban.*) Egy megfelelő ördög-algoritmus a következő: Csak akkor válaszolunk nemmel egy él behúzottságát firtató kérdésre, ha az „igen” válasz után már minden szóbajöhető gráfban lenne kör. (Azaz igyekszünk igennel válaszolni, de kerüljük, hogy kör jöjjön létre az igennel megválaszolt élekből.)

Megoldás: (*Nem hivatkozhatunk a hintre. Másoljuk be, ha már stabilizálódott.*) Stratégiánk miatt a párbeszéd legvégén kapott gráf mindenképpen körmentes lesz. Legyen (u, v) az az él, amit utoljára, $\binom{n}{2}$-edikként (*Használjuk azt, hogy nem lett vége hamarabb?*) kérdezett meg a kérdező. Ördögstratégiánk sikere azt jelenti, hogy a gráf körmentessége függ attól, hogy ez az él be van húzva vagy sem. Tegyük fel indirekt módon, hogy a gráf az utolsó (u, v) él behúzásával is körmentes. Ekkor (u, v) elvágó él. Tekintsük az (u, v) elhagyásával szétváló két komponens között haladó tetszőleges (u, v)-tól különböző (w, z) csúcspárt. (*, ami nem-él. Miért nem lehet a két komponens egy-egy elemű?*) Könnyen látható, hogy (w, z) megkérdezésének pillanatában a behúzása nem befolyásolja az igennel megválaszolt élek gráfjának körmentességét. Ez azonban stratégiánk miatt ellentmondásban van azzal, hogy a (w, z) él behúzottságára nemmel válaszoltunk.

2. Az „összefüggőség” gráftulajdonság zárkózott.

Segítség: Az ördög-algoritmus legyen a következő: Csak akkor válaszolunk igennel, ha a „nem” válasz után már egyetlen gráf sem lenne összefüggő a szóbajöhetőek közül. (Azaz igyekszünk nemmel válaszolni, de vigyázunk arra, hogy a már behúzott és a még megkérdezetlen élek együtt összefüggő gráfot alkossanak.

Megoldás: *(Nem hivatkozhatunk a hintre. Másoljuk be, ha már stabilizálódott.)* Stratégiánk miatt a párbeszéd legvégén kapott gráf mindenképpen összefüggő lesz. Legyen (u, v) az az él, amit utoljára, $\binom{n}{2}$-edikként *(Használjuk azt, hogy nem lett vége hamarabb?)* kérdezett meg a kérdező. Ördög-stratégiánk sikere azt jelenti, hogy a gráf összefüggősége függ attól, hogy ez az él be van húzva, vagy sem. Tegyük fel indirekt módon, hogy a gráf az utolsó (u, v) él nélkül is összefüggő. Ekkor u és v között létezik P út. Válasszuk ki ennek az útnak az utoljára megkérdezett (w, z) élét. (u, v) és az út többi éle olyan utat alkot w és z között, amely (w, z) behúzásának pillanatában csupa behúzott vagy megkérdezetlen élből áll. Ebből könnyen következik, hogy (w, z) behúzása nem befolyásolja a már behúzott és a még megkérdezetlen élek együttes gráfjának összefüggőségét. Ez azonban stratégiánk miatt ellentmondásban van azzal, hogy a (w, z) él behúzottságára igennel válaszoltunk.

3. (**) A „van izolált pontja” gráftulajdonság zárkózott.

Megoldás: *(A Gács-Lovász 58-59. oldalán. Nagyon nehéz. (Kati szerint nem nagy ötlet a második feladatot nagyon nehézre venni.) Az az ötlete, hogy páratlan sok gráf jöjjön szóba minden pillanatban, és mindig azt a választ adjuk, ami ezt a tulajdonságot megőrzi.)*

4. A „feszítőfa” gráftulajdonság zárkózott.

5. A „kétszeresen összefüggő” gráftulajdonság zárkózott.

6. A „legalább k-élű erdő” gráftulajdonság zárkózott.

7. Skorpió-gráfnak hívjuk az alábbi felépítésű n csúcsú irányítatlan gráfokat: A gráfnak van egy egy fokú csúcsa (fullánk), amely szomszédos egy kettő fokú csúccsal (farok), amely szomszédos egy olyan csúccsal (test), amely a gráf további $n - 3$ csúcsával (lábak) is szomszédos. (Tehát a test a fullánk kivételével az összes csúccsal szomszédos.) A lábak által feszített részgráfról nem kötünk ki semmit. *(Ábra.)* Bizonyítsuk be, hogy egy gráfról $6n$ él lekérdezésével megállapítható, hogy skorpió-gráf-e. (n a csúcsok száma.)

Segítség: (*http://www.cse.nd.edu/courses/cse511/www/solution1.pdf és http://www.cse.iitd.ernet.in/ sbaswana/Puzzles/Algo/algo.html#Scorpio*)

8. Adott n darab különböző szám. Feladatunk kiválasztani a legkisebbet és a legnagyobbat egyszerre. Hány összehasonlításra van ehhez szükség?

(a) Megoldható $\lceil \frac{3n}{2} \rceil - 2$ összehasonlítással.

(b) (*) Ennyire szükség is van.

Megoldás: A felső becsléshez először osszuk párokba a számokat, és hasonlítsuk össze a párok két tagját. Ez $\lfloor n/2 \rfloor$ összehasonlítás. Ezek után már csak a „győztesek" (nagyobbak) között keressük a legnagyobbat, és a „vesztesek" között a legkisebbet. Mindkét részfeladat $\lceil n/2 \rceil - 1$ összehasonlítást igényel.

Az alsó becsléshez nevezzük nagyoknak azokat a számokat, amelyek az első összehasonlításuk alkalmával nagyobbnak lettek minősítve. A kicsik fogalmát analóg módon definiáljuk. Semmilyennek nevezzünk azokat a számokat, amelyek az algoritmus lefutásának adott pillanatában még egyszer sem lettek összehasonlítva másik számmal. Az alsó becsléshez a következő ördögmódszert alkalmazzuk: Ha egy nagyot és egy nem nagyot összehasonlítanunk, akkor jelöljük meg a nagyot nagyobbnak. Analóg módon járjunk el a kicsikkel is. Ha két semmilyet kell összehasonlítanunk, akkor természetesen tetszőlegesen dönthetünk. Ha két nagyot vagy két kicsit kell egymással összehasonlítanunk, akkor válaszoljunk tetszés szerint, de úgy, hogy ne kerüljünk ellentmondásba az addigi összehasonlításainkkal. (Más szóval az elvégzett összehasonlítások által meghatározott irányított gráfban ne jöjjön létre irányított kör. Ezt mindig megtehetjük, mert egy DAG-ba új élet behúzva az új él irányítható úgy, hogy a gráf DAG maradjon.)

Az algoritmus futásának végére minden szám vagy nagy vagy kicsi lesz, hiszen a semmilyenekről nem tudhatnánk, hogy nem ők-e a legnagyobbak. A nagyok halmazán az összehasonlítások gráfja összefüggő, mert ha lenne két komponense, akkor akkor azok tetszőlegesen lennének rendezhetőek egymáshoz képest. Hasonló állítás igaz a kicsikre is. Ez legalább $n - 2$ összehasonlítást jelent azonos kategóriájúak között. Ehhez jön a nagyok és kicsik között haladó összehasonlítások száma. Minden szám átesett legalább egy ilyenfajta összehasonlításon, hiszen a legelső összehasonlítása mindenképpen ilyen típusú volt. Tehát az ilyen összehasonlítások számára alsó becslés a két halmaz közül a nagyobbiknak a mérete, ami legalább $\lceil n/2 \rceil$.

9. Legfeljebb $2n - 2 - \lfloor \log n \rfloor$ kérdéssel eldönthető, hogy egy n résztvevős turnamentben van-e olyan játékos, aki az összes többit legyőzte (abszolút győztes).

(*Zolinak nem tetszik a turnament szó? Vagy definiálni akarja?*)

Megoldás: Először tartsunk egy $\lceil \log n \rceil$ fordulóból és $n - 1$ mérkőzésből álló standard kieséses bajnokságot. Ennek az egyértelműen létező győztese az egyetlen szóbajöhető jelölt abszolút győztesnek. Játsszon mindenkivel, akivel a kieséses bajnokság során még nem játszott.

10. (**) Az előző feladatban adott korlát optimális.

Segítség: Képzeljük el az algoritmus lefolyását úgy, hogy bajnokságot bonyolítunk le. Nevezzük valódinak azokat a mérkőzéseket, amelyek a mérkőzés megkezdéséig veretlenek között zajlottak. Vizsgáljuk a győztes által lejátszott valódi mérkőzések számát.

Segítség: Ördög-módszerünk legyen a következő: Egy veretlen és egy nem veretlen közötti mérkőzésen győzzön a veretlen. Két veretlen közti mérkőzésen győzzön az, akinek kevesebb valódi mérkőzése van.

Megoldás: (*Ide be kell egerezni a hinteket.*) Az ördög-módszerünk alkalmazása során mindig marad legyőzetlen játékos, tehát a párbeszéd végén kapott turnamentben van abszolút győztes. k szerinti teljes indukcióval bizonyítható, hogy mire egy játékos k valódi játszmát megnyert, addigra ő és az általa addig kiejtettek összesen legalább $2^k - 1$ valódi mérkőzést lejátszottak. A valódi mérkőzések összes száma $n - 1$. Az abszolút győztes tehát legfeljebb $\lfloor \log n \rfloor$ valódi játszmát játszott. Mindenkit megvert, tehát pontosan $n - 1$ (nem feltétlenül valódi) mérkőzést játszott. Ez az összes valódi mérkőzésekkel együtt már legalább $(n - 1) + (n - 1) - \lfloor \log n \rfloor$ játszma.

Chapter 5

NP-teljesség

Teljesség-visszavezetés. Ez többféle ismérv szerint is tovább finomítható: Egy kitüremkedő része a SAT-környéki feladatok. A Friedl-gyűjteményben erősen képviselt a fázisátmenet témája, tehát az olyan feladatpárok, ahol egy feladat **NP**-teljessége után egy hasonló feladat **P**-beliségét kell megmutatni.
1. (♡) Ha az L nyelv **NP**-teljes, akkor az $\mathrm{L}' = \{xx : x \in \mathrm{L}\}$ nyelv is az.

Megoldás: Ha L **NP**-beli, akkor L' is az: a xx szó tanúi legyenek az x szó L-beliségének tanúi. (Nem ilyen alakú szavakra ne létezzen tanú. *(Kati szerint ez pongyola. Világossá kell tenni, hogy most definiáltam egy nyelvet.)*) Ha L **NP**-nehéz, akkor L' is az: A visszavezetés L-ről L'-re egyszerűen a megkettőzés művelete.

2. Egy A halmaz nyesi a $\mathcal{H}$ halmazrendszert, ha annak minden H elemére $H \cap A$ nemüres és $H \cap \overline{A}$ sem üres. SET-SPLITTING-nek nevezzük azt a kérdést, hogy adott halmazrendszerhez létezik-e nyeső halmaz. SET-SPLITTING **NP**-teljes. *(Csúnya ez az overline.)*

Segítség: Vezessük vissza SET-SPLITTING-re a SAT feladatot. A literálokat vegyük fel az alaphalmaz elemeinek.

Megoldás: Vezessük vissza SET-SPLITTING-re a SAT feladatot. Az alaphalmaz álljon a literálokból, és egy kitüntetett 0 elemből. Minden i-re vegyük be $\mathcal{H}$-ba a kételemű $\{x_i, \overline{x_i}\}$ halmazt. Minden C klózra vegyük be $\mathcal{H}$-ba a $C \cup \{0\}$ halmazt. Könnyen látható, hogy a normálforma kielégítései kölcsönösen egyértelműen megfeleltethetők $\mathcal{H}$ nyeső halmazainak: Egy adott kielégítéshez tartozó nyeső halmaz legyen az igaz literálok halmaza, és fordítva, egy $A \cup \overline{A}, 0 \in \overline{A}$ nyeső partícióhoz válasszuk igaznak az A elemeit.

Definíció: SAT-k azon kielégíthető konjunktív normálformák nyelve, amelyekben minden változó legfeljebb k literálban fordul elő.

Definíció: k-SAT azon kielégíthető konjunktív normálformák nyelve, amelyekben minden klóz (clause) legfeljebb k-elemű.

3. SAT-3 **NP**-teljes.

Segítség: Egy háromnál többször előforduló változó előfordulásaihoz vegyünk fel új változókat, és alkalmas CNF (*valahol írni kellene, hogy így hívom a konjunktív normálformát, vagy pedig ki kellene írni*) segítségével biztosítsuk, hogy ezek egyenlőek legyenek egymással.

Megoldás: Egy konjunktív normálformát a kielégíthetősége megváltoztatása nélkül SAT-3 alakra hozunk. Ehhez minden egyes háromnál többször előforduló változóját kiküszöböljük, az alábbi módon: Tegyük fel, hogy az x változó (tagadásait is beleértve) k alkalommal szerepel. Vegyük fel az új $x_1, \ldots, x_k$ változókat, és rendre cseréljük le x előfordulásait ezekre. Ezek után azt kell garantálnunk, hogy az x_i változók minden kielégítésben egyenlőek, amit az alábbi klózok hozzáadásával elérhetünk: $x_1 \vee \overline{x_2}, \ldots, x_{k-1} \vee \overline{x_k}, x_k \vee \overline{x_1}$. Az új x_i változók mindegyike pontosan háromszor szerepel az új formulában.

4. (♡) SAT-2 **P**-beli.

Megoldás: Ha egy változó mindkét előfordulása állító, vagy mindkettő tagadó, akkor mindkét klózát kielégíthetjük a változó értékének alkalmas megválasztásával. Ha különböző előjelűek, és azonos klózban fordulnak elő, akkor a klóz kielégíthető. Ha az előfordulások különböző előjelűek, és különböző klózban fordulnak elő, akkor a két klózt összevonhatjuk, pontosabban mondva a $(C \vee x) \wedge (D \vee \overline{x})$ részformula helyettesíthető a $C \vee D$ klózzal úgy, hogy a normálforma kielégíthetősége nem változik. Ezt a redukciós lépést csak akkor nem tudjuk elvégezni, ha C és D üresek, azaz a $x \wedge \overline{x}$ részformulával találkoztunk. Ebben az esetben a normálforma nem kielégíthető.

5. 2-SAT **P**-beli.

Megoldás: Minden kételemű klóz felírható literálok közti implikáció formájában. Ezek az implikációk egy irányított gráfot hoznak létre a literálok csúcshalmazán. (A kontrapozíció két ekvivalens implikációt ad, mindkét megfelelő élet húzzuk be.)

Az algoritmus ismertetése előtt tárjuk fel ennek a gráfnak néhány tulajdonságát: A gráf tranzitív lezártja olyan implikációkat tartalmaz, amelyek logikailag következnek az eredetiekből. Nem okvetlenül körmentes gráfról lévén szó, a tranzitív lezártban előfordulhatnak oda-vissza irányított párhuzamos élek. (*Van ezeknek nevük?*) Egy ilyen kettő hosszú kör a két literál értékének azonosságát kényszeríti ki. Ha egy változó és a tagadása között kör halad, akkor a formula nem kielégíthető. Ennek az állításnak a megfordítása is igaz: Ha a tranzitív lezártban nincsen literál és tagadása között haladó kettő hosszú kör, akkor a formula kielégíthető. (*Ennek bizonyításához valami olyasféle állítás kell, mint a DAG-ok toprendezhetősége, csak most megengedünk klasztereket, ha nincs bennük változó és tagadása. A legegyszerűbb valószínűleg reprodukálni a toprendezhetőség bizonyítását:*) Tegyük fel, hogy a tranzitív lezártban egy változó két literálja között semelyik irányban nem megy él. Húzzuk be valamelyiket önkényesen, és végezzük el újra a tranzitív lezárást. Állítjuk, hogy ennek során nem keletkezett rossz hurok. (*Ezt még nem bizonyítottuk.*) A lépést ismételve végül minden literálpárok közti él be lesz húzva vagy az egyik, vagy másik irányban, ami kielégítését adja a formulának, hiszen a kiindulási implikáció-gráf továbbra is részgráfja a végül kapottnak. (*Kati szerint ne spóroljam meg azt a kis mondatot, hogy a turnamentből hogyan kapok kielégítést.*)

A fenti diszkusszió után maga a kielégíthetőséget eldöntő algoritmus már egyszerűen megadható: Határozzuk meg az implikációk gráfjának tranzitív lezártját, és vizsgáljuk meg, hogy van-e benne változó és a tagadása között haladó hurok.

(*Ha használhatnám az erősen összefüggő komponens, és az aszerinti redukált DAG fogalmát, akkor minden sokkal tisztább lenne.*)

Második megoldás, lehet, hogy kevésbé elegáns, de legalább már le van írva:

Keressünk a gráfban topologikus rendezést (*valamilyen lehivatkozott klasszikus algoritmussal*). Ha nincsen irányított kör a gráfban, akkor létezik topologikus rendezés, és ez alapján a formula kielégíthető: Az x_i változó értéke legyen igaz akkor és csakis akkor, ha a topologikus rendezésben az x_i literál előbb szerepel, mint az $\overline{x_i}$.

Ha létezik irányított kör, akkor a rajta szereplő literálok értéke szükségszerűen egyenlő. Ha a körön valamely literál és komplementere egyszerre szerepel, akkor a formula nem elégíthető ki. Ha nincs ilyen pár, akkor a kör (és a megfelelő komplementerekből álló szimmetrikus párja) összehúzható egyetlen új literállá. Ezt az eljárást ismételgetve a gráf csúcsainak száma csökken,

és előbb-utóbb találni fogunk egy kielégítést, vagy a kielégíthetetlenség bizonyítékát.

6. Az alábbi probléma **NP**-teljes: Természetes számok egy multihalmaza két részre osztható-e „igazságosan", azaz úgy, hogy az egyes részhalmazok összegzése egyenlő legyen? (*Azért illene megmondani, hogy miről vezetünk vissza. http://web.njit.edu/ marvin/cis341/hwsoln13.pdf*)

7. (♡) Adjunk polinomiális algoritmust egy diszjunktív normálforma kielégíthetőségének eldöntésére.

Segítség: Keressünk kielégíthető klózt.

8. (♡) Jelölje DOUBLE-SAT a legalább két különböző kielégítéssel rendelkező konjunktív normálformák nyelvét. Bizonyítsuk be, hogy DOUBLE-SAT **NP**-teljes.

Megoldás: Az **NP**-beliség nyilvánvaló. SAT-ot vezetjük vissza a feladatra. A visszavezetés egyszerűen álljon abból, hogy felveszünk egy új y változót, és ahhoz egy új $(y \vee \overline{y})$ klózt. Ennek a formulának pontosan akkor létezik legalább két kielégítése, ha az eredetinek létezik kielégítése.

9. Jelölje HALF-CLIQUE azon gráfok nyelvét, amelyeknek létezik legalább feleakkora klikkjük, mint amekkora a csúcsszámuk. Bizonyítsuk be, hogy HALF-CLIQUE **NP**-teljes.

Megoldás: Az **NP**-beliség nyilvánvaló. A klasszikus „Van-e G-nek legalább k méretű klikkje?" feladatot vezetjük vissza. A leképezés egy (G, k)) párról a következő legyen: Ha $k < n/2$ ($n = |V(G)|$), akkor a gráfot egészítsük ki $n - 2k$ csúccsal, amelyeket egymással és G összes eredeti csúcsával összekötünk. Ha $k \geq n/2$, akkor a gráfot egészítsük ki $2k - n$ izolált csúccsal. A két esetről külön-külön bizonyítjuk, hogy megfelelő leképezés. A $k < n/2$ esetben egy $2n - 2k$ csúcsú G' gráfot kapunk, amelynek az $n - k$ méretű klikkjei kölcsönösen egyértelműen megfeleltethetőek G k méretű klikkjeinek. Ha $k \geq n/2$, akkor egy $2k$ csúcsú gráfot kapunk, amelynek k méretű klikkjei kölcsönösen egyértelműen megfeleltethetőek G k méretű klikkjeinek.

10. (♡) A lineáris időben kiszámítható Karp-redukcióra nézve nem létezik **P**-teljes nyelv.

Megoldás: Tegyük fel, hogy létezik ilyen L nyelv. Az, hogy L **P**-ben van, azt jelenti, hogy $\mathbf{DTIME}(n^k)$-ban van valamilyen rögzített k-ra. Válasszunk egy tetszőleges L′ nyelvet $\mathbf{DTIME}(n^{k+1})$-ből. L **P**-teljessége azt jelenti, hogy L′-

re adható egy olyan eldöntési algoritmus, amely először alkalmaz egy lineáris idejű Karp-redukciót a bemenetre, amelynek eredménye legyen az x szó, aztán x L-beliségét eldönti $O(|x|^k)$ időben. A Karp-redukció linearitása miatt $|x| = O(n)$, tehát az algoritmus teljes futásideje $O(n^k)$, azaz L$'$ benne van $\mathbf{DTIME}(n^k)$-ban. Ezzel beláttuk, hogy $\mathbf{DTIME}(n^{k+1}) = \mathbf{DTIME}(n^k)$, ami ellentmond az Időhierarchia-tételnek. Az ellentmondás bizonyítja, hogy nem létezik a feltételnek megfelelő L nyelv.

11. (*) UNIQUE-SAT-nak nevezzük azon konjunktív normálformák nyelvét, amelyeknek egy és csakis egy kielégítésük van. Bizonyítsuk be, hogy ha UNIQUE-SAT eleme **NP**, akkor $\mathbf{NP} = \mathbf{co-NP}$.

Segítség: Vezessük vissza $\overline{\text{SAT}}$-ot UNIQUE-SAT-ra. Ha UNIQUE-SAT **NP**-beli és $\mathbf{co-NP}$-teljes, akkor $\mathbf{NP} = \mathbf{co-NP}$. *(döm: $\mathbf{co-NP}$-nehezet kéne inkább írni, mert a $\mathbf{co-NP}$-beliség szerintem nem igaz.)*

Megoldás: Vezessük vissza $\overline{\text{SAT}}$-ot UNIQUE-SAT-ra. $\overline{\text{SAT}}$ $\mathbf{co-NP}$-teljességéből már következik a bizonyítandó állítás. Legyen $F(x_1, \ldots, x_n)$ a konjunktív normálforma, amelynek a kielégíthetetlenségét kell eldöntenünk. A visszavezetés legyen a következő: Vezessünk be egy új x_0 változót, és konstuáljuk meg F-ből a következő F' formulát:

$$(x_1 \to F) \wedge ... \wedge (x_n \to F) \wedge (x_0 \to F).$$

Ha F nem kielégíthető, akkor F'-nek pontosan egyetlen kielégítése van, a csupa hamis értékadás. Ha F kielégíthető, akkor F'-t a csupa hamis értékadáson kívül még kielégíti az F bármelyik kielégítése, méghozzá az x_0 mindkét választása mellett. A visszavezetés alkalmasságát beláttuk, de a megadott F' formula nem konjunktív normálformájú. A megoldás befejezéséhez megmutatjuk, hogy F' ekvivalens módon konjunktív normálformára hozható. Álljon F a $C_1, \ldots, C_k$ klózokból. Ekkor

$$F' \equiv \bigwedge_{i=0}^{n} \bigwedge_{j=1}^{k} \overline{x_i} \vee C_j.$$

Ez a formula már konjunktív normálformában van, és hatékonyan előállítható az F bemenet alapján.

12. (Friedl 9.11.) Tegyük fel, hogy van egy polinomidejű eljárásunk, amely a kettes számrendszerben ábrázolt m, n bemenetekre megadja $m!$ értékét modulo n. Ezt felhasználva adjunk polinomidejű módszert természetes számok prímtényezőkre bontására.

13. (♡) (Friedl 9.13.) Pontos 3-fedésnek (exact 3-cover, X3C) hívjuk annak eldöntését, hogy egy háromelemű halmazokból álló halmazrendszernek létezik-e olyan részhalmazrendszere, amely partíció. Abból a feltételezésből kiindulva, hogy az X3C eldöntési feladat megoldására létezik polinomidejű eljárásunk, adjunk olyan eljárást, amely polinomidőben megoldja az X3C keresési feladatot, azaz adott halmazrendszerhez megkeresi a megfelelő partíciót, ha az létezik.

14. (Friedl 9.22.) A „Van-e G-ben legalább $|V(G)| - 2$ élhosszú egyszerű út?" feladat **NP**-teljes.

15. (Friedl 9.19.) Adjunk **PSPACE**-beli algoritmust, amely egy G páros gráfról eldönti, hogy teljes párosításainak száma pontosan egyenlő-e a bemenetként adott k természetes számmal.

16. (Friedl 9.24.) Adjunk polinomiális algoritmust, amely egy G páros gráfról eldönti, hogy teljes párosításainak száma pontosan egy-e.

17. (Friedl 9.20.) Adjunk **PSPACE**-algoritmust arra az eldöntési feladatra, amelynek bemenete egy (G, k) pár, és az eldöntendő kérdés, hogy van-e G-ben legalább k darab Hamilton-kör.

18. (♡) (Friedl 9.30., 9.31.) Az n csúcsú, $n+10$ élű gráfokon a HAMILTON és a 3-színezhetőség eldöntése polinomidőben lehetséges.

19. (Friedl 9.35.) Legyen $f : \{0,1\}^* \to \{0,1\}^*$ egy függvény, és tegyük fel, hogy a párokból álló $\{x\#f(x) : x \in \{0,1\}^*\}$ nyelv polinomidőben felismerhető. Adjunk ellenpéldát arra az állításra, hogy az f függvény polinomidőben kiszámítható.

20. (♡) (Friedl 9.37.) **NP**-teljes annak eldöntése, hogy egy n csúcsú gráfban létezik-e legalább $\sqrt{n}$ hosszú kör.

Megoldás: Az **NP**-beliség nyilvánvaló, maga a megfelelő hosszúságú kör a tanú. Az **NP**-teljesség bizonyításához a HAMILTON feladatot vezetjük vissza a feladatunkra. Legyen G az n csúcsú gráf, melynek Hamilton-tulajdonságát el kell döntenünk. Vegyünk hozzá $n^2 - n$ izolált csúcsot a gráfhoz. Ebben a G' gráfban pontosan akkor létezik $\sqrt{|V(G')|} = n$ csúcsú kör, ha G-ben létezik Hamilton-kör.

21. Ha $\mathbf{P} = \mathbf{NP}$, akkor tudunk polinomidőben prímfaktorizálni.

Segítség: Használjuk a kereséshez a „Létezik-e n-nek k-nál kisebb, de egynél nagyobb osztója?" eldöntési feladatot.

Megoldás: A „Létezik-e n-nek k-nál kisebb, de egynél nagyobb osztója?” eldöntési feladat nyilvánvalóan **NP**-ben van, tehát feltevésünk szerint **P**-ben is. Ezt szubrutinként használva felezéses iterációval meghatározhatjuk n legkisebb prímosztóját, majd azzal leoszthatunk. Ezt az eljárást ismételve sorban megkaphatjuk az összes prímosztót.

22. Ha $\mathbf{P} = \mathbf{NP}$, akkor 3-színezhető gráfokat tudunk polinomidőben 3-színezni.

Megoldás: Feltevésünk szerint gráfok 3-színezhetőségét el tudjuk dönteni polinomidőben. Haladjunk végig sorjában a gráf komplementerének élein. Ha az aktuálisan vizsgált élet a gráfhoz hozzávéve 3-színezhető marad a gráf, akkor vegyük be az élet a gráfba, és az iteráció további fázisaiban már ezzel a gráffal dolgozzunk, a korábbit elfelejtve. Az eljárás végén kapott gráfról könnyen látható, hogy Turán-gráf, legfeljebb három diszjunkt klikk komplementere. Színezése könnyen megtalálható, és az eredeti gráfnak is színezése egyben.

23. (Friedl 9.16.) Ha $\mathbf{P} = \mathbf{NP}$, akkor tudunk polinomidőben konjunktív normálformához kielégítést találni (ha a normálforma kielégíthető).

24. Ha $\mathbf{P} = \mathbf{NP}$, akkor polinomidőben meg tudjuk találni egy irányítatlan gráf valamelyik maximális méretű klikkjét.

Megoldás: A „Létezik-e G-nek k-nál nagyobb méretű klikkje?” eldöntési feladat **NP**-ben van, tehát feltevésünk szerint **P**-ben is. Ezt szubrutinként használva felezéses iterációval meghatározhatjuk a gráf klikkméretét. Ezután hagyjuk el sorjában a gráf éleit, de ügyelve arra, hogy ne hagyjunk el olyan élet, amelynek elhagyása csökkenti a klikkméretet. Könnyen látható, hogy az eljárás végén egy maximális méretű klikket kapunk.

Megjegyezzük, hogy a $\mathbf{P} = \mathbf{NP}$ feltétel sem elegendő ahhoz, hogy egy gráf összes maximális méretű klikkjét meghatározzuk polinomidőben. Egy párosítás komplementere példa olyan gráfra, amelynek exponenciális számú maximális klikkje van.

25. Ha $\mathbf{P} = \mathbf{NP}$, akkor polinomidőben meg tudjuk találni egy irányítatlan gráf valamelyik maximális hosszúságú körét.

Megoldás: A „Létezik-e G-nek k-nál hosszabb köre?” eldöntési feladat **NP**-ben van, tehát feltevésünk szerint **P**-ben is. Ezt szubrutinként használva felezéses iterációval meghatározhatjuk a gráf leghosszabb körének hosszát. Ezután hagyjuk el sorjában a gráf éleit, de ügyelve arra, hogy ne hagyjunk el

olyan élet, amelynek elhagyása csökkenti a leghosszabb kör hosszát. Könnyen látható, hogy az eljárás végén egy maximális hosszúságú kört kapunk.

Chapter 6

Klasszikus bonyolultsági osztályok

(Jobb fejezetcímet.)

1. (♡) Véges nyelv felismerhető konstans időben.

A következő két feladat a padding, azaz párnázás nevű módszer gyakorlására szolgál.

Definíció: Egy L nyelv f hosszúra párnázott változata, L$'$ azon szavakból áll, amelyeket L-beli x szavakból kapunk, olyan módon, hogy a végükhöz illesztünk egy $f(|x|) - |x|$ hosszú csupa-nulla szót, ahol f valamilyen (általában gyorsan növő, mindenesetre az identitásnál nagyobb) függvény.

Definíció: **EXP** azon nyelvek osztálya, amelyek kiszámíthatók $2^{poly(|x|)}$ futásidejű determinisztikus Turing-géppel. **NEXP** ennek nemdeterminisztikus megfelelője.

2. Ha $\mathbf{P} = \mathbf{NP}$, akkor $\mathbf{EXP} = \mathbf{NEXP}$.

Segítség: Ha L benne van $\mathbf{DTIME}(2^{n^c})$-ben, és a párnázó $f(n)$ függvényt is 2^{n^c}-nek választjuk, akkor a párnázott változat már **P**-ben is benne van.

3. $\mathbf{NP} \neq \mathbf{E}$. ($\mathbf{E} = \mathbf{DTIME}(2^{O(n)})$.

Segítség: Végezzünk párnázást.

4. $\mathbf{NP} \neq \mathbf{EXP}$. *(Ez később jött, befésülendő.)* *(döm: Ezt nem is tudtam)*

5. (*) $\mathbf{SPACE}(n) \neq \mathbf{P}$, azaz a lineáris tárral megoldható problémák nem ugyanazok, mint a polinomidőben megoldható problémák.

Segítség: A Tárhierarchia-tételből azonnal következik, hogy $\mathbf{SPACE}(n)$ valódi része **PSPACE**-nek. Használjuk fel ezt az eredményt.

Segítség: (*P zárt ennek és ennek a párnázásnak az inverzére (?),* $\mathbf{SPACE}(n)$ *pedig nem.*)

Megoldás: Tegyük fel, hogy a két osztály egybeesik. Egy **PSPACE**-beli L nyelvet párnázzunk fel annyira, hogy már $\mathbf{SPACE}(n)$-ben legyen. (*Írjuk ki konkrétan, hogy* $\mathbf{SPACE}(n^k)$*-ban van, és* n^k*-val párnázzuk.*) Indirekt feltevésünk szerint a párnázott nyelv **P**-ben van. Ekkor viszont a párnázatlan nyelv is **P**-ben van (*Miért is pontosan?*), azaz indirekt feltevésünket újra kihasználva $\mathbf{SPACE}(n)$-ben is. Tehát a feltevésből következik, hogy $\mathbf{PSPACE} = \mathbf{SPACE}(n)$, ami azonban ellentmond a Tárhierarchia-tételnek.

6. **TALLY**-nak nevezzük az egybetűs ábécé feletti nyelvek osztályát. Bizonyítsuk be, hogy létezik **TALLY** nyelv $\mathbf{E} \setminus \mathbf{P}$ -ben.

Megoldás: Az „egyes számrendszerre való áttérés” (*Definiálni.*) művelete kölcsönösen egyértelmű megfeleltetést képez **E** és $\mathbf{P} \cap \mathbf{TALLY}$ között. (*Miért is pontosan? Kati szerint olyannyira nem szabadna efölött a trivialitás fölött elsiklani, hogy akár feladható is lenne könnyű feladatként.*) Ugyanilyen megfeleltetés van $\mathbf{DTIME}(2^{O(2^n)})$ és $\mathbf{E} \cap \mathbf{TALLY}$ között. Az Időhierarchia-tétel szerint létezik nyelv $\mathbf{DTIME}(2^{O(2^n)}) \setminus \mathbf{E}$-ben. Ennek az egyes számrendszerű változata $\mathbf{TALLY} \cap (\mathbf{E} \setminus \mathbf{P})$-ben van.

7. $\mathbf{NE} \setminus \mathbf{E}$ pontosan akkor nemüres, ha $\mathbf{TALLY} \cap (\mathbf{NP} \setminus \mathbf{P})$ nemüres.

Megoldás: Az „egyes számrendszerre való áttérés” művelete kölcsönösen egyértelmű megfeleltetést képez **E** és $\mathbf{P} \cap \mathbf{TALLY}$ között. Analóg a viszonya **NE**-nek $\mathbf{NP} \cap \mathbf{TALLY}$-hez.

Chapter 7

Fejtörők

1. (**) Adott egy $n \times n$-es tábla. Minden mezőben egy lámpa és egy nyomógomb. Ha egy gombot megnyomunk, a sorában és oszlopában lévő összes lámpa állapotot vált (égett-elalszik, aludt-kigyullad), a saját lámpáját is beleértve. Bekapcsoláskor minden lámpa kikapcsolt állapotban van. Ezután valaki odalép a táblához, és össze-vissza nyomogatja a gombokat. Feladatunk, hogy helyreállítsuk az eredeti sötét állapotot. Hány gombnyomással tudjuk ezt mindig elérni?

(*döm: szerintem ugyanolyan nehéz ez és a következő feladat, legalábbis én egyikre sem látok egyszerűbb megoldást.*)

Segítség: A feladatot elképzelhetjük úgy, hogy minden gombhoz tartozik egy $GF(2)$ feletti $n \times n$ dimenziós vektor. Szükséges gombnyomások száma = log Lehetséges állapotok száma = Ahány dimenziós teret feszítenek a vektorok.

Megoldás: Ha n páros, akkor az i. sor és j. oszlop összes gombját megnyomva épp csak (i, j) lámpa gyullad ki, tehát a teljes teret feszítik a vektorok.
Ha n páratlan, akkor az i. és j. sor (vagy oszlop) összes gombját megnyomva semmi sem történik, tehát ezek a vektorok összefüggőek, azaz egyetlen sor (oszlop) már generálja mindet, tehát legfeljebb $n^2 - 2(n-1) = (n-1)^2 + 1$ dimenziós a tér. De ennyinek kell is lennie, hiszen a bal-felső $n-1 \times n-1$-es részmátrix gombjaihoz független vektorok tartoznak (hiszen $n-1$ páros) és ezektől független a jobb-alsó sarok vektora.
Tehát páros n-re n^2, páratlan n-re $(n-1)^2 + 1$ gombnyomás kell a kezdeti állapot visszaállításához.

2. (*) Könnyebb a fenti feladatot megoldani úgy, hogy ha a tábla működési szabályát kicsit megváltoztatjuk: Gombnyomásra a gomb sorában és os-

zlopában lévő összes lámpa állapotot vált, kivéve magát a gombhoz tartozó lámpát.

(*döm: hint ugyanaz, mint az előzőhöz*)

Megoldás: Itt az $(a,b),(a,d),(c,b),(c,d)$ gombokhoz tartozó vektorok összefüggőek. Tehát az első sor és első oszlop gombjaihoz tartozó vektorok már generálnak mindent, azaz legfeljebb $2n-1$ dimenziós lesz a tér. Az is világos, hogy legalább $2n-2$ dmienziós lesz, mert a bal-felső sarkot beállíthatjuk tetszőlegesre, majd bármely másik két lámpát, melyek egyike az első sorban, a másik az első oszlopban van, tudunk egyszerre kapcsolni egy gombnyomással úgy, hogy a többi lámpa nem változik.
Ha n páratlan, akkor bármely gomb páros sok lámpát kapcsol át az első sor és oszlop uniójából, tehát $2n-2$ dimenziós a tér.
Ha n páros, akkor bármely gomb páros sok lámpát kapcsol át az (első sor és oszlop uniója\ bal-felső sarok)-ból, tehát megintcsak $2n-2$ dimenziós a tér. Tehát $2n-2$ gombnyomás kell a kezdeti állapot visszaállításához.

3. Tekintsük a közismert 15-ös játék azon változatát, amely nem 4×4-es, hanem $2\times n$-es a táblán játszódik, n darab kék és $n-1$ darab piros kővel. Kezdetben a bal felső mező az üres, és az alsó sorban vannak a kék kövek. Aszimptotikusan hány lépésben tudjuk visszaállítani az eredeti állapotot, miután valaki összekeverte a köveket?

4. Rajzolás téglalapokkal: Adott egy nulla-egy mátrix. Adjunk hatékony algoritmust, amely minél kisebb számú olyan mátrix modulo 2 összegeként állítja elő az adott mátrixot, amely mindenhol nulla, csak egy összefüggő téglalap-blokkban egy. Elég, ha az algoritmusunk közelíti az optimális mátrixszámot.

Segítség: Minden 2×2-es összefüggő blokkra tekintsük elemei modulo 2 vett összegét. Így egy 'duális' mátrixot kapunk, amely értékeit az eredeti négyzetrács keresztezési pontjain veszi fel. Könnyen belátható, hogy az eredeti feladat ekvivalens azzal, hogy ezt a mátrixot állítsuk elő minél kisebb számú olyan mátrix összegeként, amely egy kétszer kettes, nem okvetlenül összefüggő részmátrix elemein egy, azon kívül nulla értékeket tartalmaz.

5. A Gonosz elfog három matematikust. Mindegyiknek a fejére húz egy sapkát. A sapka lehet piros vagy kék, és a Gonosz egyenként, pénzfeldobással dönti el, hogy melyikükön milyen sapka legyen. Nem látják a saját sapkájukat, de látják egymását. Mikor áttanulmányozták egymás sapkáját, vezényszóra egyszerre meg kell mondaniuk, hogy milyen színű a saját sap-

kájuk. Három választ adhatnak: piros, kék, passzolok. Ha bármelyikük tévedett (azaz nem passzolt, és nem találta el), akkor végez velük a Gonosz, mert buták voltak. Ha mindegyikük passzolt, akkor is végez velük, mert gyávák voltak.

Mielőtt megkapják a sapkákat, tarthatnak egy taktikai megbeszélést. Adjunk nekik módszert, hogy minél jobb esélyük legyen az életben maradásra. A módszer legyen eredményesebb a nyilvánvaló 50%-nál, amit például úgy érhetnek el, ha egyikük pirosat mond, a többi passzol.

Mi a legjobb életben maradási esély, ha n matematikus van? (Ez általános n-re megoldatlan probléma, oldjuk meg minél többféle n értékre.)

Segítség: A lehetséges sorsolásokat feleltessük meg az n-dimenziós kocka csúcsainak. Egy adott játékos helyzete ekkora a kocka valamely élének feleltethető meg. Egy tetszőleges megbeszélt stratégia tehát kódolható ennek az élhálózatnak egy irányításával. A stratégia sikerességének valószínűsége egyenlő *(félbehagyva)*.

6. Kémként beépültünk egy ellenséges szervezetbe. Ha üzenetet küldünk társainknak, könnyen kiderülhet, és elfoghatnak. Szerencsére az ellenséges szervezet minden nap közzétesz 64 bit információt (ezt elképzelhetjük egy 8×8-as fekete-fehérrel színezett saktáblának). Mielőtt a kimenő üzenetet publikálnák, átmegy a kezünkön, és nekünk módunkban áll egyetlen, tetszőleges bitet megváltoztatni rajta. Ennek segítségével üzenhetünk haza. Hány bit információt juttathatunk így ki naponta a szervezetből? (A protokollt természetesen még beépülésünk előtt egyeztettük társainkkal.)

7. A díszsortűz-feladat: Katonák díszsorfalat állnak. Mindegyik csak a két közvetlen szomszédjával tud kommunikálni, és csak egységnyi időközönként, szinkronizált módon. Az egyes katonák memóriája nagy, de véges konstans. Minden lépésben a memóriájuk állapota, és az előző lépésben hozzájuk bejövő jelzések alapján kimenő jelzéseket adhatnak le szomszédjaiknak. Feladatuk, hogy mindannyian egyidőben adjanak le lövést egylövetű fegyverükből. Ehhez a részletes szabályok a következőek: A nulla időpillanatban megkezdődik a protokoll végrehajtása. Ezután egy előre nem megjósolható időpillanatban a bal szélső kívülről parancsot kap a díszsortűzre. A protokoll akkor helyes, ha véges mennyiségű kommunikációs forduló után mindannyian egyszerre kerülnek a kitüntetett „tűz!" állapotba. Másrészt ezen pillanatot megelőzően egyikük sem kerülhet „tűz!" állapotba, a bal szélsőnek adott parancsot megelőzően sem.

Adjunk számukra protokollt, méghozzá olyat, ahol minél kevesebb lépés telik el a parancs kiadása és a sortűz között. A protokoll nem függhet a sorfal hosszától, és tetszőlegesen hosszú sorfalra is felkészültnek kell lennie. (Tehát olyan sorfalra is, amelynek hossza nem tárolható el a katonák rögzített véges méretű memóriájában.)

Segítség: Először oldjuk meg a feladatnak azt a változatát, ahol a cél az, csak egyetlen katona tüzeljen, méghozzá az, aki pontosan középen áll a sorfalban. (Az egyszerűség kedvéért feltehetjük, hogy páratlan sokan vannak.)

Megoldás: Az egyszerűbb tárgyalás végett a protokollt csak $2^n + 1$ hosszú sorfalakra adjuk meg, az általánosítás könnyű, bár munkaigényes.

Először megadunk egy segédprotokollt, amelyet szubrutinként fogunk használni a megoldáshoz. A segédprotokoll célja az, csak egyetlen katona tüzeljen, méghozzá az, aki pontosan középen áll a sorfalban. Itt is feltesszük, hogy a szélen álló által kapott külső parancs indítja a protokollt, feltehetjük, hogy a bal szélről. A bal szélen álló indítson el egy „piros" üzenetet jobbra, amelyet a megkapói majd körönként továbbküldenek. Ha a jobb szélsőhöz ér, az „pattintsa vissza". A piros üzenettel egyszerre indítson a bal szélső egy „kék" üzenetet is, amelyet a katonák csak minden harmadik körben küldenek tovább. (Ő maga is csak a harmadik körben indítsa.) Könnyen látható, hogy páratlan hosszú sorfal esetén a piros és a kék üzenet éppen a középső katonánál fog először találkozni. Ő ezt észlelve adja le a lövését, és függessze fel az üzenetek továbbküldését.

Az eredeti feladat megoldásához indítsuk el a segédprotokollt, a következőképpen módosítva: Amikor a középső katona megkapja a két üzenetet, akkor ne tüzeljen, hanem menjen át egy „kettévágó" állapotba, ami alatt azt értjük, hogy ő onnan kezdve a piros üzeneteket nem továbbküldi, hanem visszapattintja, mintha a sor szélén állna. Ezzel egyidőben indítsa el a segédprotokollt mindkét irányba egyszerre. (Két, őt is beleértve $2^{n-1}+1$ méretű részsorfal felé.) Kellő idő elteltével a sorfal negyedénél és háromnegyedénél állók (egyszerre) kerülnek kettévágó állapotba, és elindítják a saját két-két részprotokolljukat. Ez a rekurzív folyamat akkor ér véget, amikor az utolsó körben minden páros sorszámú katona kettévágóvá válik. (A páratlan sorszámúak már korábban azzá váltak.) A megoldás befejezéséhez csak azt a kiegészítést kell tennünk, hogy a kettévágó katonák minden körben értesítik szomszédaik kettévágó mivoltukról. Ha egy katona érzékeli, hogy ő és összes szomszédja kettévágó, akkor leadja a lövést, és felfüggeszti működését.

Chapter 8

Logaritmikus tár

1. (♡) Adjunk **LOGSPACE** algoritmust, ami eldönti, hogy egy szó palindróma-e.

2. (**) A fenti problémára nem létezik $o(\log n)$ tárkorlátos algoritmus.

Segítség: Alkalmazzuk azt az eredményt, hogy a szavak azonosságának eldöntése lineáris kommunikációs bonyolultságú feladat.

Megoldás: Tegyük fel indirekt módon, hogy létezik $o(\log n)$ tárkorlátos algoritmus a problémára. Azt az alapvető eredményt fogjuk alkalmazni, hogy a szavak azonosságának eldöntése lineáris kommunikációs bonyolultságú feladat. A bemenetet osszuk fel két azonos hosszúságú szakaszra, egyik felét Alízra, másik Bobra bízva. Feladatuk annak eldöntése, hogy Alíz bemenetének megfordítása megegyezik-e Bob bemenetével. Az ezt megvalósító kommunikációs protokollt a $o(\log n)$ tárkorlátos algoritmusunk futásából származtatjuk.

A Turing-gép a bemenet középső pontján való áthaladáskor legfeljebb $const^{o(\log n)}$-féle állapotban lehet. Az áthaladások száma nem lehet ennél nagyobb, mert akkor a Turing-gép végtelen ciklusba kerülne. A Turing-gép a középső ponton való áthaladáskor legfeljebb $o(\log n)$ bit információt szállít át a két kommunikáló fél „térfele" között. Tehát a futás alapján egy $const^{o(\log n)}o(\log n) = o(n)$ bonyolultságú kommunikációs protokollt kaptunk a szavak azonosságának eldöntésére, ami ellentmondás. (*Kati szerint leírandó, hogy végülis mi a protokoll.*)

3. Adott egy szintaktikusan helyes logikai formula, amiben csak a zárójelek, az $\wedge$, a $\vee$, a $\neg$, és a logikai konstansok szerepelnek. Értékeljük ki

LOGSPACE algoritmussal.

Megoldás: (*A megoldás nincs meg nekem, Borogyin tételének lemmája, és az az ötlete, hogy log-mélységű formulával nincs gond, és egy formula kiegyensúlyozható* **LOGSPACE** *algoritmussal.*)

4. (**) Az $\{a, b\}$ elemek által generált szabad csoport szóproblémájának nevezzük a következő feladatot: Adott egy szó az $\{a, b, a^{-1}, b^{-1}\}$ négyelemű ábécé felett. Kérdés, hogy ezt szabad csoport-beli szorzatként értelmezve értéke az egységelem-e. (Kombinatorikus megfogalmazásban: Ha az egymás mellett álló inverzeket egyszerűsítjük egymással, akkor ezt a lépést mohón ismételgetve az üres szóhoz jutunk-e.) Adjunk **LOGSPACE** algoritmus ennek a problémának a megoldására. (*Kati szerint ezt például szét kell vágni kisebb feladatokra.*)

Segítség:

$$A := \begin{pmatrix} 1 & 2 \\ 0 & 1 \end{pmatrix}$$

B legyen A transzponáltja. Használjuk fel azt az eredményt, mely szerint A és B a mátrixok szorzásával és invertálásával éppen a két elem által generált szabad csoportot generálják.

Segítség: Kis modulusú aritmetikában le tudjuk ellenőrizni egy hosszú mátrix-szorzatról logaritmikus memóriában, hogy identitás-e az eredménye. Használjuk a Kínai Maradéktételt, hogy nagy modulusra is következtethessünk.

Megoldás: A feladatnak egy lineáris algebrai átfogalmazását fogjuk megoldani.

$$A := \begin{pmatrix} 1 & 2 \\ 0 & 1 \end{pmatrix}$$

B legyen A transzponáltja. Felhasználjuk azt az eredményt, mely szerint A és B a mátrixok szorzásával és invertálásával éppen a két elem által generált szabad csoportot generálják. Feladatunk tehát n darab $\mathbb{Z}$ feletti 2×2-es mátrix szorzatáról eldönteni, hogy az egységmátrix-e. Ha $\mathbb{Z}$ helyett valamilyen rögzített véges modulus felett dolgoznánk, akkor a feladat triviális lenne, egyszerű balról jobbra haladó beszorzással. Sajnos $\mathbb{Z}$ felett a részeredmények elvben akár 3^n nagyságrendig növekedhetnek. (Jobban nem, hiszen minden egyes A-val vagy B-vel való szorzáskor a mátrix legnagyobb eleme

legfeljebb megháromszorozódik.) Ezzel a részeredmények bináris reprezentációja kimerítheti a logaritmikus tárat. A probléma áthidalásához a Kínai Maradéktételt hívjuk segítségül:

Végezzük el egyenként minden 2 és $2n$ közötti modulussal a beszorzást. (A $poly(n)$ nagyságú modulus feletti aritmetika elvégezhető **LOGSPACE**-ben.) Ha bármelyik modulus felett nem az identitásmátixot kapjuk eredményül, akkor befejezhetjük az eljárást, és kijelenthetjük, hogy a szorzat értéke nem az egységelem. Ha mindegyikre azt kapjuk, akkor a Kínai Maradéktétel szerint kijelenthetjük, hogy a modulusok legkisebb közös többszöröse szerinti modulusban is azt kapnánk. A 2 és m közötti számok legkisebb közös többszöröse egy ismert számelméleti tétel szerint kellően nagy m-re legalább 2^m. Ebbe a képletbe behelyettesítve azt kapjuk, hogy eljárásunk leellenőrizte a mátrixazonosságot egy 4^n-nél nagyobb modulus felett. De mint korábban láttuk, a $\mathbb{Z}$ feletti szorzáskor kapott részeredmények nem lehetnek 3^n-nél nagyobbak, ezért a 4^n nagyságrendű modulus felett végzett számításkor nem léphet fel „túlcsordulás", és ezért kijelenthetjük, hogy az azonosság $\mathbb{Z}$ felett is igaz, tehát a bemenet értéke az egységelem.

5. Adjunk **LOGSPACE** algoritmust, amely eldönti egy szóról a { (,) } kételemű ábécé felett, hogy helyesen zárójelezett-e.

Megoldás: Balról jobbra végighaladva a szón, minden pillanatban tartsuk nyilván, hogy mennyivel több nyitó-, mint záró zárójellel találkoztunk addig. Ha ez a szám bármikor nulla alá csökken, akkor helytelen a zárójelezés. Ha az utolsó betű utáni pozícióban nem nulla, akkor szintén. Minden más esetben a szó helyesen zárójelezett.

6. (*) Adjunk **LOGSPACE** algoritmust, amely eldönti egy szóról a { (,), [,] } négyelemű ábécé felett, hogy helyesen zárójelezett-e. (*Tipográfiailag csúnya.*)

Megoldás: Először ellenőrizzük le logaritmikus tárban mindkét fajta zárójelről, hogy a másik fajta zárójelektől elvonatkoztatva, önmagában helyesen zárójelezett-e. (Ehhez oldjuk meg az előző feladatot.) Ha igen, akkor minden (nyitó- vagy záró, gömbölyű vagy szögletes) zárójelnek egyértelműen létezik párja, és az logaritmikus memóriával meg is található. Ekkor a következő egyszerű struktúratételt mondhatjuk ki: A szó akkor és csak akkor helyesen zárójelezett, ha nem létezik két olyan, különböző típusú zárójel-pár, amelyek „egymásba gabalyodottak", azaz $(\ldots[\ldots)\ldots]$ mintájúak. Az ilyen tiltott konfigurációk megkereséséhez pedig egyszerűen végigmehetünk egy kettős

ciklusban minden lehetséges olyan páron, ami két különböző típusú nyitó zárójelből áll, és a párjaikat megkeresve megvizsgálhatjuk az „egymásba gabalyodottságot”. (*Egyrészt ez kicsit pongyola. Másrészt homályban hagyja, hogy a helyes zárójelezettség miféle definíciójából, és hogyan vezettük le a struktúratételt.*)

7. Labirintus a kockás papíron: Adott egy nulla-egy mátrix. Az egy értékű elemein gráfot definiálunk: Két csúcs össze van kötve, ha (vízszintesen vagy függőlegesen) szomszédosak a mátrixban. GRID-STCONN feladatnak hívjuk az (irányítatlan) s-t összefüggőség eldöntését az ilyen módon megadott gráfokban. GRID-STCONN eleme **LOGSPACE**. (*Kati szerint* STCONN-*ban máshogy kellene írni az st-t, mert összekeverhető a* STRONG-CONN-*nal.*)

Segítség: Egy csúcshoz keressük meg a csúcs komponensének egyértelműen létező „legészaknyugatibb” pontját.

8. STRONG-CONN **NL**-teljes.

Megoldás: STRONG-CONN **NL**-beliségét többféleképpen bizonyíthatjuk, mutatja például az alábbi visszavezetés STCONN-ra: Építsük meg (*Ugye valójában nem építjük meg. Kati szerint ezt precízen kellene leírni, olyannyira, hogy külön feladatokban kellene begyakoroltatni azt, hogy hogy zajlik egy* **LOGSPACE** *visszavezetés. Apropó fel lehetne adni a tranzitivitását.*) azt a G' gráfot, mely a $G(V, E)$ gráf $|V| + 1$ azonos, diszjunkt példányából áll, és az m. és $m + 1$. példány között előre irányított él vezet az $m + 1$ azonosítójú csúcsok között. Az utolsó példányba vezető él az 1 azonosítójú csúcsba vezessen.

A gráf akkor és csakis akkor erősen összefüggő, ha létezik olyan körséta a csúcsain, amelyen (*Félbehagyva.*)

Az **NL**-teljesség bizonyításához visszavezetjük az STCONN feladatot STRONG-CONN-ra. Adott tehát egy irányított G gráf egy kitüntett (s, t) csúcspárral. Egészítsük ki G-t a G' gráffá az alábbi módon: Húzzunk irányított élet t-ből az összes s-től és t-től különböző csúcsba. Továbbá húzzunk irányított élet az összes s-től és t-től különböző csúcsból s-be. Könnyen belátható, hogy G-ben akkor és csakis akkor létezik s-ből t-be menő út, ha G' erősen összefüggő.

9. 2-SAT **NL**-teljes.

Segítség: Egy G irányított gráf $s - t$-összefüggőségét fogjuk lefordítani

2-SAT formulára. A változók legyenek a csúcsok, és az $x \to y$ élhez vegyük fel az $\overline{x} \vee y$ klózt. (*És még kell valami t-s visszaél is.*)

10. (♡) BIPARTITE $\in$ **NL**.

Segítség: Használjuk az Immermann-Szelepcsényi tételt.

Megoldás: A nem-párosság feladatára könnyű **NL**-algoritmust adnunk: az algoritmus egyszerűen megtippel egy legfeljebb csúcsszám hosszú páratlan kört. Ezután a megoldás az Immermann-Szelepcsényi tétel azonnali következménye. (*És kábé ekvivalens vele, vagy van nagyágyútlan megoldás is?*)

11. Az alábbi probléma **NL**-teljes: Adott egy irányított gráf az s és t csúcsaival, és egy k szám. Igaz-e, hogy s és t távolsága pontosan k?

Segítség: Használjuk az Immermann-Szelepcsényi tételt.

Megoldás: Először bizonyítsuk az **NL**-beliséget. Az „Igaz-e, hogy s és t távolsága legfeljebb k?" feladat nyilvánvalóan **NL**-beli. Az Immermann-Szelepcsényi tétel szerint tehát a komplementer „Igaz-e, hogy s és t távolsága nagyobb, mint k?" feladat is **NL**-beli. A két **NL** algoritmus egymás után futtatásával a pontosan k távolság feladatát is meg tudjuk oldani **NL**-ben.

12. Melyik **LOGSPACE**-beli és melyik **NL**-teljes az alábbi problémák közül?

(a) s-t összefüggőség irányított gráfban, amelyben minden pont kifoka legfeljebb kettő.

(b) s-t összefüggőség szintezett DAG-ban. (*Mi az a szintezett? Meg van adva a bemenet*)

(c) s-t összefüggőség irányított gráfban, amelyben minden pont befoka legfeljebb egy.

Megjegyzés: (*Ne legyenek open-ended feladatok. Kati szerint akár lehetnek is, ő nem utálja őket.*)

13. (*) **LOGSPACE** akkor és csakis akkor zárt a Kleene-féle csillag műveletre, ha **LOGSPACE** = **NL**.

Segítség: A nehéz irányhoz adjunk meg olyan L **LOGSPACE** nyelvet, amelynek a Kleene-féle csillagára redukálható az $s-t$ összefüggő topologiku-

san rendezett irányított aciklikus gráfok nyelve. (Az L-beli szavakra való felbontás fogja kifejezni az utat s és t között.)

Segítség: Az $s - t$ összefüggő topologikusan rendezett irányított aciklikus gráfok nyelve **NL**-teljes, mivel a **LOGSPACE** Turing-gépről feltehető, hogy számolja, hogy indulása óta hány lépést tett meg.

Segítség: *(Mondani se kell, hogy ez még nincs kész.)* A redukció az alábbi sorokból fog állni: A DAG minden x csúcsára leírjuk x-et kétszer, majd felsoroljuk a szomszédait. x DELIMITER1 x D2 y_1 D3 y_2 D3 ... D3 y_k D4. Ezeket az izéket összefűzzük abban a topologikus sorrendben, amelyben a gráf csúcsai adva vannak. Az L nyelvben olyan mondatok vannak, amelyekre igaz, hogy az első szó szomszédai között szerepel az utolsó szó: x D2 y_1 D3 ... y_l ... D3 y_k D4, aztán teljes izéből jópár, aztán a végén D4 y_l D1.

14. $\mathbf{POLYLOGSPACE} = \bigcup_k \mathbf{SPACE}(\log^k n)$. Bizonyítsuk be, hogy nem létezik **POLYLOGSPACE**-teljes nyelv a **LOGSPACE**-redukcióra nézve. Hogyan viszonyulhat **POLYLOGSPACE** **P**-hez tartalmazás szempontjából? *(Ne legyenek ilyen argumentatív feladatok.)*

Segítség: Bizonyítsuk be, hogy adott k-ra $\mathbf{SPACE}(\log^k n)$ zárt a **LOGSPACE**-redukcióra nézve.

Definíció: Az alábbiakban definiáljuk a Branching Program nevű számítási modellt. Ez a Boole-hálózathoz hasonlóan fix változószámú Boole-függvények kiszámítására szolgál. A Branching Program egy irányított aciklikus gráf, egyetlen (START nevet viselő) forrással, és két nyelővel (az egyik a 0, a másik az 1 nevet viseli). A nem nyelő csúcsok kifoka mindig 2, és a két él közül az egyiken egy Boole-változó, a másikon ennek a változónak a tagadása áll. A változó értéke szerint a csúcsból továbblépünk a megfelelő élen. Adott bemenetre a START csúcsból elvégezve ezt a sétát, egy nyelőbe jutunk. Ennek értéke az adott bemeneten számított kimenet. A Branching Program mérete a csúcsok száma, mélysége a leghosszabb forrás-nyelő út hossza. *(Nincs ennek mégiscsak valami magyar neve? És jó az, hogy nagybetűvel írom? A google szerint Szegeden szemrebbenés nélkül elágazó programnak hívják. Hemaspaandra-Ogihara szerint egy Masek nevű arc még 1976-ban elnevezte döntési gráfnak, ami szerintem szép.)* *(Kati szerint érdemes egy külön megjegyzésben megalapozni, hogy a BP az olyan, mint egy döntési fa, csak nem okvetlenül fa.)*

Definíció: $B(f)$-fel jelöljük az f-et számító legkisebb Branching Program

méretét. $D(f)$-fel jelöljük az f-et számító legsekélyebb Branching Program mélységét.

Definíció: MAJORITY$(x_1, \ldots, x_n) = \lfloor \sum_i x_i/2 \rfloor$. *(Kicsit faramuci, hogy pont most vezetem be a MAJORITY-t.)*

15. Zárkózottnak nevezünk egy n-változós Boole-függvényt, ha $D(f) = n$. Bizonyítsuk be, hogy MAJORITY zárkózott. *(Elég faramuci, hogy most újra bevezetem a zárkózottságot.)*

Megjegyezzük, hogy egy Branching Program a mélység megváltoztatása nélkül döntési fává alakítható, tehát a zárkózottság definíciója ekvivalens a döntési fáknál már megismerttel.

Megoldás: A zárkózottság bizonyításához ördög-módszert alkalmazunk: Ördögként bármelyik addig még nem kérdezett bitet kérdezik tőlünk, felváltva 1-et és 0-t válaszolunk. Kezdjük az 1-gyel. Ez garantálja, hogy az utolsó kérdés előtt még nem megállapítható a függvény értéke.

16. **BP**-vel jelöljük azon nyelvek osztályát, amelyekre létezik poly méretű Branching Program-sorozat. Bizonyítsuk be, hogy **BP** = **LOGSPACE/poly**.

Megoldás: A **BP** $\subseteq$ **LOGSPACE/poly** tartalmazást bizonyító polinomiális méretű tanú-sorozat nem lesz más, mint az egyes Branching Programok leírása, a **LOGSPACE** algoritmus pedig az, amely egy adott bemeneten szimulál egy adott leírású Branching Programot. A szimuláció egyszerű, a START csúcsból élenként haladhatunk a nyelő felé, mindig csak az aktuális pozíciónkat rögzítve.

A fordított irányú tartalmazás bizonyításához rögzítsünk egy **LOGSPACE** algoritmust, egy bemeneti hosszat, és egy ahhoz tartozó tanút. Az ezzel ekvivalens Branching Program csúcsainak halmaza legyen a **LOGSPACE** algoritmus összes állapotainak halmaza, ahol az állapotba beleértjük a Turing-gép szalagpozícióját is. Ilyen állapotból összesen is csak polinomiális mennyiségű létezik. Minden állapotról megállapítható, hogy mely bemeneti szalagpozíció olvasását végzi, és hogy az olvasott értéktől függően milyen következő állapotba megy át. Az így felrajzolt irányított gráf Branching Programot alkot.

17. Nemdeterminisztikus Branching Programnak hívjuk a Branching Program azon általánosítását, amikor az élekre tetszőleges literálokat írhatunk. Az elfogadási feltétel itt az, hogy az igaz literálok által feszített részgráf-

ban van irányított út START-ból 1-be. (0-nak most nincs szerepe.) Az **NBP** nemuniform bonyolultsági osztályt **BP**-vel analóg módon definiáljuk. Bizonyítsuk be, hogy $\mathbf{NBP} = \mathbf{NL/poly}$.

Megoldás: *(Ezt akkor írom majd le, ha az előzővel, már nagyon elégedett vagyok, mert onnan kezdve nagyrészt copy-paste job.)*

18. (♡) Annak eldöntése, hogy két Branching Program ugyanazt a függvényt számítja-e ki, pontosan akkor oldható meg polinomidőben, ha $\mathbf{P} = \mathbf{NP}$.

Megoldás: Élesebb állítást bizonyítunk: Annak eldöntése, hogy két Branching Program **különböző** függvényt számít-e ki, **NP**-teljes.

Az **NP**-beliség tanúja a bemenet, amelyen a kimenetek különböznek. A teljesség belátásához vezessük vissza SAT-ot a feladatra. Az egy BP legyen az azonosan hamis kimenetet adó. A másik legyen egy olyan BP, amely az bemeneti konjunktív formulát értékeli ki. Ilyen BP-t könnyű építeni, és nyilvánvaló, hogy a két BP pontosan akkor számít különböző függvényt, ha a normálforma kielégíthető.

(Az alábbiaknak lehet, hogy jobb helye lenne máshol. És lehet, hogy szebb egyirányú gépekkel megfogalmazni őket.)

19. (*) Ha egy Turing-gép konstans mennyiségű tárat használ, akkor reguláris nyelvet ismer fel.

Megoldás: *(Az alábbiakban csak ész nélkül beegereztem Németh András rendkívül zavarosan leírt megoldását. Megfésülendő, köszönetnyilvánítandó.)* *(döm: Átírkáltam kicsit, még zavaros.)* Azt kell megmutatnunk, hogy ha A egy kétirányban mozgó véges automata, akkor egyirányúvá alakítható. Feltehetjük, hogy A csak akkor kerülhet végállapotba, amikor a „bemenet vége" szimbólumot ovassa. Az új automata konstrukciójának kulcsa egy $f : S \to S \cup \{\bot\}$ függvény kiszámítása (azaz az új automata ezt mindig számolja ki és jegyezze meg, ez megtehető, mivel ez konstans tárat igényel), ahol S az A automata állapottere, $\bot$ pedig egy S-ben nem szereplő szimbólum. Az f az jelenti, hogy az adott pillanatban ha A balra lép és s állapotba kerül, akkor amikor legközelebb visszajut a lépés előtti pozícióba a fej, akkor milyen állapotban lesz, ill. $f(s) = \bot$, ha valami baleset történik A-val mielőtt visszatérne (végtelen ciklus, elakadás, lefutás balra a szalagról, ...). Világos, hogy f értékét teljesen meghatározzák az automata mozgási szabályai és az aktuális pozíciót megelőző input szakasz. Sőt, az is látható, hogy valójában f meghatározásához elég az f előző pozícióbeli értéke. Tehát az új automata

úgy fog működni, hogy egy adott helyzetben az f segítségével „szimuláljuk" A működését, ill. annak azokat a pillanatait, amikor a fej éppen az aktuális pozícióban van, egészen addig, amíg egy jobbra lépés nem következik, amikor is jobbra lépünk mi is, áttérünk a megadott új állapotba, és meghatározzuk az új f-et. Miután végigolvastuk a szót, így megtudjuk, hogy az A automata milyen állapotban állt le.

20. Ha egy Turing-gép $o(\log\log n)$ tárat használ, akkor az is igaz, hogy csak konstans tárat használ.

Megoldás: *(Az alábbiakban csak ész nélkül beegereztem Németh András rendkívül zavarosan leírt megoldását. Megfésülendő, köszönetnyilvánítandó.) (döm: Átírkáltam kicsit, még zavaros.)*

Legyen T egy $o(\log\log n)$ tárat használó Turing-gép, jelölje $s(n)$ a pontosan n méretű inputokon használt maximális memória méretét, x pedig egy olyan n-hosszú inputot, amin a maximum felvevődik. Indirekt tegyük fel, hogy $s(n)$ nem korlátos, de $lim_{n\to\infty}\frac{s(n)}{\log\log n} = 0$. Az előző feladathoz hasonlóan csinálunk egy egyirányú véges automatát, ami ugyanazokat a szavakat fogadja el, ehhez tárolnunk kell az f-et, ami egy $cs(n)2^{s(n)}$ méretű S halmazból képez önmagára, ennyi féle belső állapot és társzalag tartalom lehet (Itt c a T belső állapotainak a száma, a társzalagon a fej $s(n)$ helyen állhat, a szalag tartalma $2^{s(n)}$ féle lehet.) A fő észrevétel az, hogy ha van két input pozíció, amire az egyirányú automatánk állapota (azaz f és T első rálépéskori belső állapota) ugyanaz, akkor a két pozíció közötti input darabot kidobva az automata lényegében ugyanúgy fog viselkedni. Tehát ha veszünk egy olyan n-t, amire $s(n)$ minden korábbi értéknél nagyobb, és az automatát elindítjuk az x inputtal, melyre ez felvétetik, majd találunk egy ismétlődő feljegyzést a szimuláció során, akkor az ellentmondás, hiszen a két ismétlődő feljegyzés közötti részt kivágva találtunk egy rövidebb inputot, amin a gép ugyanannyi memóriát használ, és ez ellentmond n választásának.

Elég tehát belátni, hogy ha n elég nagy, akkor mindenképpen lesz ismétlődő feljegyzés. Ehhez csak azt kell megfigyelni, hogy egy feljegyzés $(\log(cs(n)2^{s(n)}))^{cs(n)2^{s(n)}}$-féle, ami nem lehet $\geq n$ minden n-re, mert kétszer mindkét oldal logaritmusát és a kapott bal oldalt osztva a kapott jobb oldallal 0-hoz tartó kifejezést kapunk, speciálisan a bal oldal előbb-utóbb kisebb lesz.

(döm: RND véges automatás feladatok, egyelőre nem tudom mindnek a megoldását) **21.** Egy véletlen véges automatának van egy szalagja, melynek

minden pozíciójára 1/2 eséllyel 0, 1/2 eséllyel 1 van írva. Ezen is van egy külön olvasófeje. Megköveteljük, hogy minden szót $\geq 2/3$ eséllyel elfogadjon vagy $\geq 2/3$ eséllyel elutasítson.
(a) Ha mindkét fej egyirányú, akkor csak reguláris nyelvet ismerhet fel.
(b) Ha csak a bemenetet olvasó fej kétirányú, akkor felismerhető a $\{0^n 1^n\}$ nyelv.
(c) Ha csak a bemenetet olvasó fej kétirányú és a véletlen szalag csak $poly(n)$ hosszú, akkor csak reguláris nyelvet ismerhet fel.
(d) Ha mindkét fej kétirányú és a véletlen szalag csak $poly(n)$ hosszú, akkor felismerhető a $\{0^n 1^n\}$ nyelv.

Segítség: (b) Próbáljunk felváltva a bal és jobb oldalról beérni középre.
(d) Egy n^{100} hosszú sorozatban szinte mindig van pontosan k egymás utáni 1-es minden $1 \leq k \leq 2 \log n$-re. Ha $1 \leq a < b \leq n$, akkor van $k \leq \log n$, hogy $a \neq b \bmod k$.

Chapter 9

Polinomiális tár

Definíció: TQBF (True Quantified Boolean Formula) Probléma: Adott egy SAT formula az $x_1, \ldots, x_n$ változókkal. Kössük le a változókat $\forall$ illetve $\exists$ kvantorokkal. Ezáltal egy úgynevezett Teljesen Kvantifikált Boole Formulát kapunk. Egy ilyen formulának nincs szabad változója, tehát értéke egyszerűen vagy igaz vagy hamis. Feladatunk eldönteni, hogy igaz-e.

Példa: $\exists x_1 \forall x_2 : x_1 \wedge x_2$ QBF hamis.

A **PSPACE**-teljes nyelv fogalma teljesen analóg az **NP**-teljes nyelv fogalmával. Ismeretes, hogy TQBF **PSPACE**-teljes.

1. Lássuk be, hogy az alábbi probléma is **PSPACE**-teljes:

Általános Geográfia (GG) Probléma:

Adott egy irányított gráf, egyik csúcsán egy bábúval. Antal és Borbála felváltva lépnek a bábúval az aktuális helyéről egy olyanra, ahova vezet irányított él, de még nem járt ott a bábú. Az veszít, aki nem tud lépni. Kérdés: Kinek van nyerő stratégiája?

2. (**) Adjunk polinomiális időben kiszámolható nyerő stratégiát az Általános Geográfia irányítatlan változatára.

Segítség: A faktor-kritikus gráfok elmélete szükséges a megoldáshoz.

3. Ha minden **NP**-nehéz nyelv **PSPACE**-nehéz is, akkor **PSPACE** = **NP**.

4. (♡) Ha létezik **PH**-teljes nyelv, akkor összeomlik a polinomiális hierarchia.

Megoldás: Tegyük fel, hogy létezik **PH**-teljes nyelv, méghozzá a hierar-

chia k-dik szintjén lévő L nyelv. Ekkor a hierarchia magasabb szintjein álló nyelvek visszavezethetőek rá, és (*blabla kezdek teljesen elálmosodni*).

5. Adjunk **PSPACE** algoritmust arra a problémára, hogy két reguláris kifejezés ugyanazt a nyelvet fogadja-e el. (*Az nem feladat, hogy teljes is* **PSPACE***-ben?*)

6. Az alábbi probléma **PSPACE**-teljes: Adott egy M Turing-gép leírása, és egy x bemenet. Igaz-e, hogy az M gép elfogadja az x szót, miközben nem hagyja el az első $|x| + 1$ szalagpozíciót?

Segítség: Alkalmazzunk párnázást.

Chapter 10

Véletlen algoritmusok

1. Az L nyelv eleme **RP**-nek pontosan akkor, ha:
Létezik egy olyan $p(n)$ polinom, és egy olyan **P**-beli kétváltozós $M(x, y)$ reláció (olvasd: y tanúja x-nek), hogy ha x nem eleme L, akkor x-nek nincsen $p(|x|)$ hosszú tanúja, ha x eleme L, akkor viszont a $p(|x|)$ hosszú szavak legalább fele tanúja x-nek.

2. Bizonyítsuk be, hogy az **RP** definíciójában szereplő 1/2-et tetszőleges $0 < c < 1$ konstansnak is választhattuk volna.

3. Bizonyítsuk be, hogy a **BPP** definíciójában szereplő $(1/3, 2/3)$ helyett tetszőleges $(1/2 - c, 1/2 + c)$ is állhatna, ahol $0 < c < 1/2$.

4. (♡) $\mathbf{RP} \subseteq \mathbf{NP}$. $\mathbf{RP} \subseteq \mathbf{BPP}$. $\mathbf{co} - \mathbf{BPP} = \mathbf{BPP}$.

5. Az **RP**, **BPP**, **ZPP** nyelvosztályok mindegyike zárt a metszetre és unióra.

6. **ZPP** alábbi három definíciója ekvivalens:

(a) Az algoritmus 1/2-nél kisebb eséllyel passzolhat, és polinomiális futásidejű.

(b) Az algoritmus futásidejének várható értéke a bemenet hosszában polinomiális.

(c) $\mathbf{RP} \cap \mathbf{co} - \mathbf{RP}$.

7. Ha SAT eleme **BPP**, akkor SAT eleme **RP**. (Ebből azonnal következik, hogy ha **NP** része **BPP**, akkor **NP** = **RP**. (*Ennek prochat-nak kellene lennie, nem?*))

8. (♡) Véges test felett vagyunk (bizonyos bonyodalmak elkerülése végett). Adott három $n \times n$-es mátrix, A, B, C. Valaki azt állítja, hogy $AB = C$. Mi ellenőrizni akarjuk ezt az állítást, úgy, hogy csak $O(n^2)$ szorzást végzünk, de nem törekszünk teljes bizonyosságra: ha $AB \neq C$, akkor 0.99 valószínűséggel le akarjuk leplezni a csalást. Adjunk véletlen algoritmust a feladatra.

9. (♡) Adott egy n-edfokú polinom, és $n-1$ darab szám (nem feltétlen különbözőek). Ellenfelünk most azt állítja, hogy ennek a polinomnak multiplicitással éppen ezek a gyökei. Adjunk véletlent használó leleplező algoritmust, ami gyorsabb, mint a beszorzáson alapuló triviális determinisztikus eljárás.

10. Adott egy 2-SAT formula, amelyben minden klóz két különböző változót tartalmaz. MAX2SAT-nak nevezzük azt az optimalizálási feladatot, melynek célja a lehető legtöbb klózt igazzá tenni.

(a) Adjunk gyors *véletlen* algoritmust, amely nagy valószínűséggel olyan értékelést ad, amely a klózok legalább háromnegyedét igazzá teszi.

(b) Adjunk olyan determinisztikus polinomiális algoritmust, amely a klózok legalább háromnegyedét igazzá teszi.

11. Bizonyítsuk be, hogy a következő, RP-ACCEPTANCE-nak nevezett probléma algoritmikusan eldönthetetlen: Adott egy véletlen Turing-gép leírása. Igaz-e, hogy ez egy **RP**-típusú gép, azaz minden bemenetre vagy 0 valószínűséggel fogad el, vagy 1/2-nél nagyobb valószínűséggel?

12. $p(n)$ tetszőleges előre megadott polinom. **BPP** definíciója ekvivalens mindkét alábbi definícióval: Létezik a nyelvhez olyan véletlen Turing-gép, amelynek tévedési valószínűsége

(a) $2^{-p(n)}$ (erős **BPP** definíció)

(b) $1/2 - 2^{-p(n)}$ (gyenge **BPP** definíció)

13. Érveljünk amellett, hogy **SPACE**$(O(n^{\log n}))$ nem része **BPP**-nek. (*Ezt a lecture05.ps-ben láttam, nem ismerem a megoldást.*)

(*döm: szerintem ez nem nehéz, ha az alábbi megoldás jó*) **Segítség:** Átlós módszer. **Megoldás:** Felsoroljuk az összes véletlent használó Turing-gépet, mindet végtelen sokszor. Az n hosszú inputokon az n. Turing-gépet szimuláljuk addig, amíg $\leq n^{\log n}$ tárat használ. Ha ezt bármikor túllépné,

akkor leállunk. Mi nem tudunk véletlent használni, de sorban felírhatjuk az összes $n^{\log n}$ hosszú bitsorozatot, számoljuk, hogy melyikre mit outputolna a szimulált gép. (Ha több véletlen bitet használna, leállunk.) Végül pedig megnézzük, hogy mit outputolna többször és mi ennek az ellenkezőjét outputoljuk. Így biztos minden **BPP** géptől eltértünk.

14. (*Yao tétele, kijelentve, hogy felhasználhatják Neumann minimax tételét.*)

15. $\mathbf{PP} \subseteq \mathbf{PSPACE}$.

16. $\mathbf{NP} \subseteq \mathbf{PP}$.

17. Ha $\mathbf{NP} \neq \mathbf{BPP}$, akkor $\mathbf{EXP} \neq \mathbf{EXPSPACE}$.

Chapter 11

Bonyolultsági kérdések a nyelvek és automaták elméletében

1. A következő nyelvosztályok zártak a konkatenációra:

(a) **P**

(b) **NP**

(c) **ZPP**

(d) **LOGSPACE**

(e) **NL**

(f) **P/poly**

2. A következő nyelvosztályok zártak a Kleene-féle csillag műveletre:
(*Zoli: Kleene definiálandó. Kati: Na jó, de akkor már korábban.*)

(a) **P**

(b) **NP**

(c) **ZPP**

(d) **NL**

(e) **P/poly**

3. (**) Ha egy nyelv felismerhető egyszalagos, $o(n \log n)$ futásidejű Turing-géppel, akkor reguláris.

4. Annak eldöntése, hogy egy nemdeterminisztikus véges automata az üres nyelvet ismeri-e fel, **NL**-teljes.

5. (*) Ismeretes, hogy egy DFA-hoz polinomidőben meg tudjuk keresni a legkisebb állapotszámú ekvivalens DFA-t. Bizonyítsuk be, hogy az analóg NFA-minimalizálási feladat akkor és csak akkor oldható meg polinomidőben, ha $\mathbf{P} = \mathbf{NP}$.

Segítség: Építsünk a CNF-hez olyan NFA-t, amely éppen a nemkielégítő értékeléseket fogadja el.

6. (*) Adjunk példát **NL**-teljes környezetfüggetlen nyelvre. (*Sipser 8.17 van. Nem tudom a megoldást.*)

7. A CFG-EMPTY döntési feladat egy instanciája (*Kati nem szereti a szót, inkább körbeírná. Vagy esetleg lehet „bemenet”.*) egy környezetfüggetlen nyelvtan leírása. Az eldöntendő kérdés, hogy a generált nyelv üres-e. Bizonyítsuk be, hogy a feladat **P**-teljes.

8. A CFG-INFINITE döntési feladat egy instanciája egy környezetfüggetlen nyelvtan leírása. Az eldöntendő kérdés, hogy a generált nyelvnek végtelen sok eleme van-e. Bizonyítsuk be, hogy a feladat **P**-teljes.

9. Ha a CFG-INFINITE problémát megszorítjuk azokra a környezetfüggetlen nyelvtanokra, amelyekben nincsen üres szó a szabályok jobb oldalán, és nincsen használatlan nemterminális, akkor **NL**-teljes problémát kapunk. (*Pontosabban.*)

10. A CFG-MEMBERSHIP döntési feladat egy instanciája egy környezetfüggetlen nyelvtan leírása és egy szó. Az eldöntendő kérdés, hogy a generált nyelv tartalmazza-e az adott szót. Bizonyítsuk be, hogy a feladat **P**-teljes.

11. Adott két CFG az egyelemű ábécé felett. Feladat annak az eldöntése, hogy a két nyelv azonos-e. Bizonyítsuk be, hogy a feladat $\mathbf{\Sigma_2}$-teljes.

12. (*) Annak eldöntése, hogy egy nemdeterminisztikus véges automata a teljes $\Sigma^\star$ nyelvet ismeri-e fel, **PSPACE**-teljes. (*opg3.ps-ből van.*)

Chapter 12

Polinomiális hierarchia, orákulumok

1. $\mathbf{P}^{\mathbf{NP}} \supseteq \mathbf{co-NP}$.

2.

(a) $\mathbf{NP}^{\mathbf{NP}} \subseteq \mathbf{\Sigma_2}$.

(b) (*) $\mathbf{NP}^{\mathbf{NP}} = \mathbf{\Sigma_2}$.

3. Létezik olyan A orákulum, hogy $\mathbf{P}^{\mathrm{A}} = \mathbf{NP}^{\mathrm{A}}$.

4. Létezik olyan B orákulum, hogy $\mathbf{P}^{\mathrm{B}} \neq \mathbf{NP}^{\mathrm{B}}$.

5. Igaz-e, hogy „G-ben pontosan k méretű a legnagyobb klikk?" eleme $\mathbf{\Sigma_2}$?

6. (*) Bizonyítsuk be, hogy $\mathbf{\Sigma_4}$ nem része a $\mathbf{SIZE}(n^t)$ nemuniform bonyolultsági osztálynak, semmilyen fix t-re. (*SIZE definiálandó? Ha igen, hol?*)

Segítség: Az alternációt használjuk fel arra, hogy tippeljünk egy nagy bonyolultságú Boole-függvényt, és ellenőrizzünk is le.

7. (*) Az előző feladat és a Karp-Lipton tétel alkalmazásával lássuk be, hogy az előző feladat állítása $\mathbf{\Sigma_4}$ helyett $\mathbf{\Sigma_2}$-vel is igaz.

Segítség: Alkalmazzunk esetszétválasztást aszerint, hogy $\mathrm{SAT} \in \mathbf{SIZE}(n^t)$ (*befejezetlen*).

8. $\mathbf{NP}^{\mathbf{NP}\cap\mathbf{co}-\mathbf{NP}} = \mathbf{NP}$.

9. $\mathbf{BPP}^{\mathbf{BPP}} = \mathbf{BPP}$.

10. (*) $\mathbf{NP}^{\mathbf{BPP}} \subseteq \mathbf{BPP}^{\mathbf{NP}}$.

11. A MIN-DNF nyelv párokból áll. A pár első tagja egy ϕ DNF formula. A második egy k egész szám. A pár akkor van benne MIN-DNF-ben, ha létezik k-nál nem hosszabb ψ formula, amely ekvivalens ϕ-vel. Bizonyítsuk be, hogy MIN-DNF eleme $\mathbf{\Sigma_2}$.

12. Ha $\mathbf{\Sigma_k} = \mathbf{\Pi_k}$ akkor $\mathbf{PH} = \mathbf{\Sigma_k}$.

13. Ha $\mathbf{PH} = \mathbf{PSPACE}$, akkor a hierarchiának csak véges sok szintje van.

14. (*Ez lehet easy feladat is meg hint is az előzőhöz:*) Ha létezik **PH**-teljes probléma, akkor a hierarchiának csak véges sok szintje van.

15. $\mathbf{EXP}^{\mathbf{EXP}} = \mathbf{EEXP}$. ($\mathbf{EEXP} = \mathbf{DTIME}(2^{2^{n^c}})$.)

16. $\mathbf{BPP} \subseteq \mathbf{ZPP}^{\mathbf{NP}}$.

Segítség: $\mathbf{pBPP} = \mathbf{pRP}^{\mathbf{pRP}}$. (*opg5.ps*)

Chapter 13

Általános Boole-hálózatok

1. A méret polinomiális mértékű megnövelése árán elérhető, hogy az AND, OR, NOT hálózat összes NOT kapuja a legalsó szinten legyen.

2.

(a) Mely Boole-függvények számíthatóak ki Boole-hálózattal?

(b) Milyen $f(n)$-re igaz, hogy ekkora hálózat már minden n-változós Boole-függvényhez létezik?

3.

(a) Mely monoton Boole-függvények számíthatóak ki monoton Boole-hálózattal?

(b) Milyen $f(n)$-re igaz, hogy ekkora monoton hálózat már minden n-változós monoton Boole-függvényhez létezik?

4. Adjunk $O(n \log n)$ méretű Boole-hálózatot a MAJORITY függvényre.

5. (*) Adjunk $O(n)$ méretű Boole-hálózatot a MAJORITY függvényre.

Segítség: (*Igaz-e az, hogy a bináris fát alkotó ismételt összeadással $\Sigma_i O(i)n/2^i = O(n)$ méretben vagyunk? Ha igen, akkor nem is nehéz.*)

6. (**) Ha **NP** része **P/log**, akkor **P** = **NP**.

Segítség: Vegyük észre, hogy egy adott bemenetre egy adott **P/log** algoritmust le tudunk futtatni minden egyes logaritmikus hosszúságú tanáccsal,

polinomiális futásidőben maradva. De sajnos nincsen nyilvánvaló eljárás, amely eldönti, hogy a végigpróbálgatott tanácsok közül melyik a helyes.

Segítség: Azt kihasználva, hogy a SAT eldöntési feladat **P/log**-ban van, adjunk polinomidejű eljárást, amely megoldja a SAT keresési feladatot. Az eljárásnak ne legyen bemenete a logaritmikus hosszúságú tanács, csak használja ki annak létezését.

7. (**) Adjunk **PSPACE**-beli nyelvet, amelynek az általános Boole-hálózat bonyolultsága $O(n^3/\log n)$ és $O(n^3)$ közé esik.

Segítség: Először oldjuk meg a feladatot úgy, hogy elhagyjuk a **PSPACE**-beliség feltételét, aztán adjunk **PSPACE**-beli algoritmust a megtalált nyelvre.

Segítség: Az adott hálózati bonyolultságú nyelv megtalálásához először tekintsünk egy olyan nyelvet, amelynek a bonyolultsága $2^n/3n$ és 2^n közé esik. Ilyen nyelv Shannon tétele szerint létezik. Alkalmazzunk párnázást. (*Tényleg ilyen jó konstanst ad Shannon tétele úgy, ahogy azt mi tanuljuk?*)

Segítség: Ha egyáltalán létezik olyan nyelv, amelynek az általános Boole-hálózat bonyolultsága $c_2n^3/\log n$ és c_1n^3 közé esik, akkor ilyen nyelvet **PSPACE** (konkrétan **SPACE**(n^3)) algoritmussal is tudunk találni: Sorban haladjunk végig a c_1n^3 méretű hálózatokon. Álljunk meg, ha olyat találunk, amelynek egyetlen $c_2n^3/\log n$ méretű hálózattal sem egyezik meg az igazságtáblája. A megtalált hálózatot értékeljük ki a bemenetünkön.

Segítség: Élesebb megoldás az első hintre: Jelöljük az $n^3 - n - 1$ méretű hálózatokkal kiszámítható n-változós Boole-függvények halmazát F_n-nel. Ez a halmaz nem üres, és nem esik egybe az összes n-változós Boole-függvények halmazával. Léteznek tehát olyan $f \in F_n$, $g \notin F_n$ n-változós Boole-függvények, amelyek csak egyetlen a bemeneten különböznek. $g(x) = (x = a) \vee f(x)$ vagy $g(x) = (x = a) \vee f(x)$, tehát g is kiszámítható legfeljebb $(n^3-n-1)+n+1 = n^3$ méretű hálózattal.

(*Mindezt midterm-sol.ps-ból loptam, jelölésestül, úgyhogy majd kreditet kell adni Luca Trevisan-nak. Az élesebb megoldást feladattá lehetne különíteni.*)

8. Bizonyítsuk be, hogy a monoton Boole-hálózatok kiértékelési feladata **P**-teljes. (*Melyik redukcióra?*)

Megoldás: A nemtriviális irányhoz visszavezetjük a feladatra az általános Boole-hálózatok kiértékelési feladatát. A visszavezetés triviális, ha az általános Boole-hálózatainkat úgy definiáltuk, hogy csak a legalsó szintjén engedünk meg tagadó kapukat. Ekkor egyszerűen minden negált változóhoz felveszünk egy új változót, megkettőzve a változók számát, és monotonná téve a hálózatot. Ha az általános Boole-hálózatainknak magasabb szintjein is megengedünk tagadó kapukat, akkor ennek a triviális visszavezetésnek az alkalmazása előtt el kell végeznünk azt a polinom idejű átalakítást, amely a tagadó kapukat „lenyomja" a hálózat legalsó szintjére. (*Ez csak akkor elég részletes, ha ezt a lenyomást le tudom hivatkozni egy másik feladatba.*)

9. Bizonyítsuk be, hogy a stratified Boole-hálózatok kiértékelési feladata **NL**-teljes. Stratified-nak hívunk egy Boole-hálózatot, ha vagy csupa AND, vagy csupa OR kaput tartalmaz.

10. (♡) Egy nyelv ritka, ha n betűs szavainak száma $poly(n)$. A ritka nyelvek osztályát **SPARSE** jelöli. **P**-közelinek nevezünk egy L nyelvet, ha valamely $L' \in \mathbf{P}$ nyelvvel vett szimmetrikus differenciája ritka. $\mathbf{P-close}$ a **P**-közeli nyelvek osztálya. Bizonyítsuk be, hogy $\mathbf{P-close}$ valódi része $\mathbf{P/poly}$-nak.

Segítség: Része, hiszen a szimmetrikus differencia belekódolható a tanácsba. A tartalmazás valódiságának bizonyításához tekintsük természetes számok egy nemrekurzív H halmazát. Legyen $x \in$ L akkor és csakis akkor, ha $|x| \in H$. Ez az L nyelv $\mathbf{P/poly}$, de nem $\mathbf{P-close}$.

11. (*) Létezik olyan nyelv **PH**-ban, amely nincs benne $\mathbf{SIZE}(O(n))$ -ben.

Segítség: Ha SAT egy ilyen nyelv, akkor készen vagyunk, tehát a megoldáshoz feltehetjük, hogy SAT-ra létezik lineáris méretű hálózat-sorozat.

12. (**) (Kannan tétele) Bizonyítsuk be a fenti állítást **PH** helyett a $\mathbf{\Sigma_2 \cap \Pi_2}$ osztállyal.

Megoldás: (*Nem tudom, hogy hol a megoldás, de Kannan-é az eredmény, és írnak róla itt: http://citeseer.ist.psu.edu/594507.html*)

13. (♡) Ha $\mathbf{NP} \subseteq \mathbf{SIZE}(O(n))$, akkor $\mathbf{P} \neq \mathbf{NP}$. Használjuk fel Kannan tételét.

Chapter 14

Kis mélységű Boole-hálózatok

1.

(a) Adjunk alsó és felső becslést az n-változós PARITY függvényt kiszámító 2 mélységű Boole-hálózat méretére.

(b) Adjunk felső becslést az n-változós PARITY függvényt kiszámító 3 mélységű Boole-hálózat méretére.

2.

(a) Mutassunk lineáris méretű hálózatot, amelynek bemenete két n bit hosszú szám, kimenete a két szám ($n+1$ bit hosszú) összege. (Minden kettes számrendszerben történik.)

(b) Az $x \geq y$ (más néven COMPARE(x, y)) reláció $\mathbf{AC^0}$-ban van.

(c) Az összeadás $\mathbf{AC^0}$-ban van.

(d) (*) Mekkora lehet egy $\mathbf{AC^0}$ összeadó hálózat konstans mélysége?

3. (*Ugyanolyan nehéz az n darab n bites szám rendezése, mint az n darab egybitesé. Ugye ez igaz?*)

4. Tudjuk, hogy PARITY nem eleme $\mathbf{AC^0}$. Bizonyítsuk be, hogy

(a) MAJORITY sem.

(b) SORT sem.

(c) MULTIPLY(x, y) sem.

(d) DIVIDE$(x, y) = [x/y]$ sem.

5. Most azt tudjuk, hogy MULTIPLY nem eleme $\mathbf{AC^0}$. Bizonyítsuk be, hogy DIVIDE sem.

6. PARITY eleme $\mathbf{TC^0}$.

7. (**) MAJORITY eleme $\mathbf{NC^1}$.

Megoldás: *(Az alábbiakban csak ész nélkül beegereztem Németh András megoldását. Megfésülendő, köszönetnyilvánítandó.)*

A MAJORITY kiszámítására a triviális technikát lenne jó alkalmazni: kiszámítjuk valahogy egy hálózattal a bitek első felén az egyesek számát, ill. ezzel párhuzamosan a bitek második felén, ezeket ábrázoljuk bináris számként, majd összaadjuk, és a kapott bináris számot összahasonlítjuk $n/2$-vel. A bitek első felén az egyesek számát természetesen rekurzívan akarjuk számítani: a bitek első negyedén levő egyesek száma és a bitek második negyedén levő egyesek száma összegeként. És így tovább, egyre kisebb számokat kell összeadnunk.

Ez persze így bár helyes hálózatot eredményez, de mégsem oldja meg a feladatot, mert a hálózat túl mély lesz. Két n-bites szám összeadására ugyanis a triviális esetben cn mély hálózatot építenénk (általános iskolás összeadás hálósítása), így pl. $n = 2^k$ esetén összesen $1 + 2 + 3 + ... + k = O(k^2) = O(\log^2(n))$ mély hálóra lenne szükségünk. Valójában létezik módszer k bites számok $O(\log k)$ párhuzamos időben történő összeadására, de nekünk még ez sem lenne elég (ha jól sejtem ez $O(\log n \log\log n)$-et adna közvetlenül használva), de ennek a módszernek az egyik alapötletét fogjuk használni.

Ábrázoljuk minden lépésben az adott szakaszokon található egyesek számát kiegyensúlyozott (pl.) négyes számrendszerbeli számként, azaz $\sum_{i=0}^{k} 4^i a_i$ alakban, ahol $a_i \in \{-3, -2, -1, 0, 1, 2, 3\} =: J$. Ez az ábrázolás persze nem egyértelmű, de sebaj. Két ilyen számot konstans mély hálózattal össze lehet adni. Könnyen elenőrizhető ugyanis, hogy ha $x, y \in J$, akkor $x + y$ felírható $4a + b$ alakban, ahol $a \in -1, 0, 1$ és $b \in \{-2, -1, 0, 1, 2\}$. Ez a felírás nyilván elvégezhető egy konstans méretű E hálóval. Ha tehét két n jegyű számot akarok összeadni, akkor egyszerre alkalmazom E-t az összes azonos helyiértékű számjegypárra, majd az a-ba kerülő helyiértéket hozzáadom a következő helyiértéken számított b-hez, ami a b-re és a-ra tett megkötés miatt már nem képez maradékot. Ezt az összeadást alkalmazva

az első bekezdésben leírt módon egy $O(\log n)$ mélységű hálózattal ki tudom számítani az egyesek számát, amit egy kiegyensúlyozott, $O(\log n)$ jegyű négyes számrendszerbeli számként kapok meg. Ezt egy újabb $O(\log n)$ méretű hálózattal akár a triviális módon, a legkisebb helyiértéktől indulva standard alakra hozhatjuk. (Ehhez mindössze annyit kell észrevenni, hogy egy szám mod 4^k maradéka csak az utolsó k számjegytől függ kiegyensúlyozott alakban is.) Még ugyanilyen mélységet igényel az összehasonlítás, így sikerült igazolni, hogy $MAJORITY \in NC^1$ (Az, hogy az alkalmazott kapuk száma polinomiális n-ben, az triviális.)

Megjegyzés: Egy k jegyű kiegyensúlyozott szám valójában $O(\log k)$ párhuzamos időben is standard alakra hozható, így áll össze az $O(\log k)$ párhuzamos idejű összeadás.

8. SORT eleme monoton $\mathbf{AC^1}$.

9. (*) A reguláris nyelvek $\mathbf{NC^1}$-ben vannak.

10. n darab n-bites szám összeadása $\mathbf{TC^0}$-ban van.

11. MULTIPLY(x, y) eleme AC^1. *(Sőt, $\mathbf{TC^0}$-nak is eleme, ugye?)*

12. n darab n-bites szám rendezése $\mathbf{TC^0}$-ban van.

Segítség: Vegyük a páronkénti összehasonlításaikat.

13. Borogyin tétele: $\mathbf{NC^1} \subseteq \mathbf{LOGSPACE}$.

Segítség: *((Du-Ko 6.24.) Először végezzünk el egy* **LOGSPACE** *transzformációt, amely az adott bemeneti hálózatot formulává alakítja. Aztán értékeljük ki a formulát. Szétszedendő több feladatra.)*

14.

(a) Szimmetrikus Boole-függvény kiszámítható 3 mélységű, $2^{O(\sqrt{n}\log n)}$ méretű hálózattal.

(b) (*) Az állítás igaz $2^{O(\sqrt{n\log n})}$ méretkorlát mellett is.

(c) (**) Az állítás igaz $2^{O(\sqrt{n})}$ méretkorlát mellett is. *(Én sem tudom.)*

15. Adott egy alul-felül n csúcsú páros gráf, n^2 bittel leírva.

(a) Adjunk $2^{O(n)}$ méretű konjunktív normálformát, ami megmondja, hogy van-e a gráfnak teljes párosítása.

(b) Bizonyítsuk be, hogy nem létezik a feladatra $2^{O(n)}$ méretű diszjunktív normálforma.

16. Adaptálva Savitch tételének bizonyítását, bizonyítsuk be, hogy **NL** része $\mathbf{AC^1}$.

17. (*) Elhanyagolva a nagyszámú és nehézkes uniformitási részletet, bizonyítsuk be Russo tételét:

(a) $\mathbf{NC^i}$ azonos a logaritmikus tárat felhasználó, $O(log^i n)$-time alternáló Turing-gépekkel felismerhető nyelvekkel.

(b) $\mathbf{AC^i}$ azonos a logaritmikus tárat felhasználó, $O(log^i n)$-alternáló Turing-gépekkel felismerhető nyelvekkel. (*Ez igazából nemigen adható fel, túl sok fura részlete van. Csak azért írtam ide, mert fontos. barrington - circuits - 23.pdf -ben van benne.*)

Chapter 15

Kommunikációs bonyolultság

Ez a fejezet egyelőre lényegében csak Dömötör szövegének a beegerezése.

(@0715 Elnevezzük a két játékost Alice és Bobnak? Egyelőre nem, de lehetne definiálni nevüket. Jelöléseket bevezetni? Ez szerintem hasznos lenne, beírtam párat.)

Jelölések:
$D(f)$: A legrosszabb esetben szükséges determinisztikus kommunikáció hossza.
$N(f)$: A legrosszabb esetben szükséges nemdeterminisztikus kommunikáció hossza.
$R(f)$: A legrosszabb esetben szükséges random kommunikáció hossza, ami az esetek$\leq \frac{1}{3}$ részében téved.
$R^{pub}(f)$: A legrosszabb esetben szükséges random kommunikáció hossza, amikor a résztvevők közös random biteket kapnak és az esetek$\leq \frac{1}{3}$ részében téved.

1. Mindkét játékos megkapja egy-egy konvex n-szög csúcsainak koordinátáit. (Egy koordináta $O(\log n)$ hosszú.) Feladatuk eldönteni, hogy a két sokszögnek van-e közös belső pontja. $D(f) \leq O(\log^2 n)$.

Segítség: *(Yao-Ullmann-Yannakakis kell a legegyszerűbb megoldáshoz. De kijön anélkül is,* $\log n$ *lépésenként lehet felezni az egyik sokszög csúcsainak számát.)*

2. Adott egy G gráf. Egyik játékos kap egy klikket, a másik egy független csúcshalmazt. A kérdés, hogy van-e közös csúcsuk. $D(f) \leq O(\log^2 n)$.

3.

(a) Egyik játékos kap két n hosszú 0-1 sorozatot, míg a másik ezek közül az egyiket. A cél, hogy a másik is tudja, melyiket. $D(f) \leq \log n + O(1)$.

(b) (*) Most ugyanaz a feladat, csak k példányban, azaz az egyik játékosnak k-szor két inputja van, a másiknak pedig minden párból egy. $D(f) \leq \log n \log k \log\log k + O(k)$.

(c) Sőt, $D(f) \leq \log n \log k + O(k)$.

4. EQ feladat: Mindkét játékos kap egy számot 1-től 2^n-ig. $R^{pub}(\mathrm{EQ}) = O(1)$.

5. $R(\mathrm{EQ}) = O(\log n)$.

6. (**) Tudjuk, hogy az egyik játékosnak van n db valóban különböző inputja, azaz bármely kettőhöz van a másiknak legalább egy olyan inputja, amivel más az output. $R(f) \geq \log n - o(\log n)$.

7. (**) (*ezt szét akarom szedni több részre*) Jelölje $D_{--}(f)$ annak a kommunikációs feladatnak az optimális megoldásának a hosszát, amikor a két játékos nem beszélhet egymással, hanem csak egy köztük ülő harmadikkal. A kommunikációt itt is az összes küldött bitben mérjük. $\exists f : D_{--}(f) \leq \frac{3}{2}D(f) + O(1)$.

Chapter 16

Kolmogorov-bonyolultság

1. (♡) $|K_{Java}(x) - K_{C++}(x)| \leq c_{Java,C++}$. *(Kati szerint pongyola ez a jelölés.)*

Segítség: Java nyelven írhatunk C++ interpretert, és C++ nyelven Java interpretert.

2. (♡) $P_{|x|=n}(K(x) \geq |x| - c) \geq 1 - 1/2^c$.

3. (♡) Létezik olyan n szám, hogy az n-nél nagyobb prímek között nincsenek Kolmogorov-véletlenek.

Segítség: Létezik olyan algoritmus, amely a k bemenetre kiírja a k. prímszámot.

Megoldás: A Prímszámtétel szerint egy tetszőleges p prímszám legfeljebb a $\frac{9p}{8 \log p}$-edik a prímszámok sorában. Rögzítsünk egy olyan algoritmust, amely a k bemenetre kiírja a k. prímszámot. (Például Erasztotenész szitáját implementálva.) A k bemenetet a programba konstansként beledrótozva olyan algoritmust kapunk, amely $\log k + const$ méretű. Így egy legfeljebb $\log \frac{9p}{8 \log p} + const$ méretű algoritmust kaptunk tetszőleges rögzített p prím kiírására. Elegendően nagy p esetén ez a mennyiség kisebb lesz, mint $\log p - const$, amiből következik a bizonyítandó állítás.

4. (♡) Létezik olyan n szám, hogy a π n-nél hosszabb kettes számrendszerbeli közelítései között nincsenek Kolmogorov-véletlenek.

Segítség: Létezik olyan algoritmus, amely a k bemenetre kiírja π első k számjegyét.

Megoldás: Tekintsünk valamilyen π-hez konvergáló racionális végtelen soroza-

tot, amelynek becsülni tudjuk a konvergenciasebességét. (Ez lehet például a Leibniz-formula, lehet a $2\arctan(1)$ Taylor-sora, vagy a modern, úgynevezett Borwein iteráció.) Látható, hogy ez alapján építhetünk egy algoritmust, amely π egyre pontosabb racionális közelítéseit számolja, hibabecsléssel együtt. Ezt felhasználva könnyen építhetünk olyan algoritmust, amely a k bemenetre kiírja π első k számjegyét. A k bemenetet a programba konstansként beledrótozva olyan algoritmust kapunk, amely $\log k + const$ hosszú, és kiírja π első k számjegyét. Elegendően nagy k-ra ez kisebb lesz, mint $k - const$, tehát nagy k értékekre a kimenetek nem lehetnek Kolmogorov-véletlenek.

5. Nem létezik olyan algoritmus, amely minden x-hez megmondja $K(x)$-et.

Megoldás: Indirekt feltételezve $K(x)$-et meghatározó algoritmus létezését, azt szubrutinként használhatjuk olyan program írására, amely kiírja a lexikografikusan első olyan x szót, amelynek bonyolultsága egy adott k számnál nagyobb. Programunk bonyolultsága $const + \log k$-val becsülhető felülről, ahol a $\log k$ mennyiség a k szám programba való „beégetéséből" származik. Kellően nagy k számot választva ellentmondásra jutunk.

6. Nem létezik olyan algoritmus, amely a $K(x)$ olyan $L(x)$ közelítését számítja ki, amelyre minden x szóra $0.9K(x) \leq L(x) \leq 1.1K(x)$.

Megoldás: A megoldás lényegét tekintve azonos az előző feladat megoldásával, hiszen az $L(x)$ közelítést kiszámító szubrutin is alkalmazható arra, hogy megkeressünk egy olyan x szót, amelynek bonyolultsága biztosan nagyobb egy adott k számnál. (*Akkor itt most megsértettem azt az elvet, hogy nem hivatkoznak egymásra, de itt talán tényleg indokolt volt.*)

7. Bizonyítsuk be, hogy minden n hosszú x szóhoz létezik olyan y szó, amely x-ből csupán egyetlen bit megváltoztatásával áll elő, ugyanakkor $K(y|n) \leq n - \Omega(\log n)$. (*A $K(y|n)$ jelölést be kell vezetni.*)

Segítség: Használjunk Hamming-kódolást.

Megoldás: A Hamming-kódok elméletéből tudjuk, hogy az n-dimenziós kocka csúcsainak létezik egy olyan H részhalmaza, amelyre igaz, hogy 1. A kocka minden csúcsától legfeljebb egy távolságra található H-beli elem, 2. H mérete legfeljebb $(2^{n-\lfloor \log(n+1) \rfloor})$. H egy tetszőleges eleme n ismeretében specifikálható azzal, hogy a lexikografikus rendezés szerint hányadik elemről van szó H-ban. A feltételeket kielégítő $y \in H$ tehát $\lceil \log |H| \rceil = n - \Omega(\log n)$ bit információval írható le.

8. Létezik olyan c, hogy minden Kolmogorov-véletlen x szóra az egyesek számának és a nullák számának különbsége legfeljebb $c\sqrt{|x|}$.

Segítség: Ha k az egyesek száma a szóban, akkor kódolhatunk úgy, hogy megadjuk $|x|$-et, k-t, és azt, hogy az $\binom{|x|}{k}$ lehetőség közül melyik valósul meg. Alkalmas k-ra ez a kódolás tömörítés lesz.

Megoldás: Ha k az egyesek száma a szóban, akkor kódolhatunk úgy, hogy megadjuk $|x|$-et, k-t, és azt, hogy az $\binom{|x|}{k}$ lehetőség közül melyik valósul meg. Ennek a kódolásnak a hossza $O(\log|x|)+O(\log k)+O(\binom{|x|}{k}) \leq$ Alkalmas k-ra ez a kódolás tömörítés lesz. *(Számoljuk ki.)*

Definíció: Elegánsnak hívunk egy programot, ha ő a legrövidebb azok közül, akik az ő kimenetét adják. (Esetleg holtversenyben.)

9. (♡) Minden elegáns program Kolmogorov-véletlen.

Megoldás: Ha konstansnál több bitet tömöríthetünk egy programon, akkor a tömörített változatát kitömörítő majd lefuttató program rövidebb nála, miközben azonos kimenetet ad.

10. Minden axiómarendszerhez csak véges számú olyan p program létezik, amelynek eleganciája bizonyítható a rendszerben. (Mindeközben végtelen sok elegáns program létezik, tehát léteznek bizonyíthatatlan igaz állítások az axiómarendszerben.)

Megoldás: Tekintsünk egy olyan programot, amely az axiómarendszer összes lehetséges bizonyítását enumerálva bizonyítható állításokat bocsát ki. A program kimenetét vezessük át egy olyan szűrőprogramon, amely (tisztán formális ismérvek alapján) csak olyan állításokat enged át, amelyek valamely program eleganciáját jelentik ki. Legyen k később meghatározandó nagy természetes szám. Ha a szűrő végtelen sok állítást enged át magán, akkor előbb-utóbb kibocsát olyan állítást, amely egy k-nál hosszabb program eleganciáját jelenti ki. Az első ilyen állítás megtalálásakor futtassuk le a megtalált hosszú és elegáns programot, majd állítsuk le összes program-modulunkat. Mi a kiírt szó Kolmogorov-bonyolultsága? A szűrőn átengedett állítás szerint legalább k. Másrészt programunk minden összetevője (bizonyítás-enumeráló, szűrő, szimulátor) az axiómarendszer rögzítése mellett konstans hosszúságú, tehát a programunk bonyolultsága $const+\log k$-val becsülhető felülről, ahol a $\log k$ mennyiség a k szám programba való 'beégetéséből' származik. Kellően nagy k számot választva ellentmondásra jutunk, amelynek egyetlen lehetséges feloldása, hogy a szűrőprogram csak véges sok állítást enged át magán.

Chapter 17

Tartalékok

1. Egy n szám m modulus szerint vett t-edik hatványát ki tudjuk számolni polinomidőben. (Mindhárom bemenet kettes számrendszerben van megadva.)

Megoldás: Egy n szám m modulus szerint vett négyzetét természetesen ki tudjuk számolni polinomidőben. A négyzetre emelés műveletét ismételgetve a negyedik, nyolcadik, stb. hatványát is ki tudjuk számolni polinomidőben, amíg a négyzetre emelések összes száma polinomiális. t kettes számrendszerbeli felírása legyen $2^{i_1} + \ldots + 2^{i_k}$. Ez alapján $n^t = n^{2^{i_1}+\ldots+2^{i_k}} = \prod_l n^{2^{i_l}}$. Ennek a szorzatnak minden tényezője kiszámolható polinomidőben, és összesen polinomiális számúan vannak.

2. (♡) Egy n szám t-edik hatványát nem tudjuk kiszámolni polinomidőben az egészek között.

3. Egy $[n]$ feletti p permutáció t-edik hatványát ki tudjuk számolni polinomidőben. (t kettes számrendszerben van megadva.)

Megoldás: *(Annyira azonos a megoldás az előzővel, hogy restellem is leírni. n helyett p-t kell írni.)*

4. http://www.cl.cam.ac.uk/Teaching/2003/Complexity/exercise1.pdf HORN-SAT **P**-beli, és **P**-teljes a **LOGSPACE**-redukcióra nézve. Utóbbihoz kell a konkrét visszavezetés nemdeterminisztikus eldöntésről SAT-ra. Ez ügyesen csinálható úgy determinisztikus gépekre, hogy Horn-formulát adjon.

5. (*) *(Dömötöré. Mik ezek a példák? A szimmetrikus az, hogy xx alakú? Akkor hamis az állítás. Nem értem.)* A nem szimmetrikus szavak (pl *baba*

szimmetrikus, de *bababa* vagy *babab* nem) nyelve környezetfüggetlen.

(*döm: De igaz...*) **Segítség:** *uauvbv* nem szimmetrikus

Megoldás: A páros hosszú szavak előállításához az alábbi műveletek kellenek: $S \to AB, S \to BA, A \to xAy, B \to xBy, A \to a, B \to b$, ahol x és y bármely kisbetű lehet.

6. (*) (*Dömötöré.*) Egy szalagon adott n db szám, mindegyik 1-től n-ig. Valamelyik szám több, mint $\frac{n}{2}$-szer szerepel. Állapítsuk meg egy $O(\log n)$ tárhelyet használó egyirányú automatával, hogy melyik.

Megoldás: Minden lépésben el lesz tárolva egy szám és egy számláló. Kezgetben a szám bármi lehet, de a számláló 0. Miután olvastunk egy számot, ha a számláló 0 volt, akkor írjuk be ezt a számot és a számláló legyen 1. Ha a számláló nem 0, akkor ha a tárolt szám megegyezik az olvasott számmal, akkor növeljük a számlálót, egyébként pedig csökkentsük.

http://cs-www.bu.edu/faculty/homer/332/homework/hw6.html több értelmes kérdést is tartalmaz. http://cs-www.bu.edu/faculty/homer/332/homework/hw5.html már nagyrészt kivesézve.

7. Lássuk be, hogy a **P** és **EXP** osztályok zártak a Karp-redukcióra (más néven a polinomidejű many-one visszavezetésre). A bizonyítás nem megy át az **E** osztály esetére. Mi kellene ahhoz, hogy be tudjuk bizonyítani, hogy **E** nem zárt a Karp-redukcióra?

8. Az igen/nem/passzolós Turing-géppel felismerhető nyelvek osztálya része $\mathbf{NP} \cap \mathbf{co} - \mathbf{NP}$-nek: Define a strong nondeterministic Turing machine as one where each computation has three possible outcomes: accept, reject or maybe. If M is such a machine, we say that it accepts L, if for every $x \in L$, every computation path of M on x ends in either accept or maybe, with at least one accept and for $x \notin L$, every computation path of M on x ends in reject or maybe, with at least one reject. Show that if L is decided by a strong nondeterministic Turing machine running in polynomial time, then $L \in NP \cap co - NP$.

http://www.cs.unm.edu/ moret/computation/ (Ezt szerintem kimerítettem.)

http://www.cs.tau.ac.il/ safra/Complexity/exercises.htm (Ezt szerintem kimerítettem.)

Az angol Lovász: http://www.cs.elte.hu/ kiraly/complexity.pdf (Talán ezt is.)

9. Esetleg az a feladat final-sol.ps -ből, amikor egy hashP közelíthetőségéből következik, hogy **P** =**BPP**.

Még jó lenne valami a promise problémákról:
(http://www.wisdom.weizmann.ac.il/ oded/prpr.html)
A nemuniform fejezetben például elkelne egy feladat a BPP/log definíciójának komplikációiról:
(http://weblog.fortnow.com/2006/07/definitions-of-advice.html)
A véletlen algoritmusos fejezet nagyon sovány. Elkelne bele a egy-két feladat a Motwani-Raghavan 'Randomized Algorithms' könyvből. Mondjuk páronkénti függetlenségről, vagy akár entrópiáról jó lenne valami.
Sok remek NP-teljességi tétel bizonyítással: http://www.cs.tau.ac.il/ safra/Complexity/exercises.htm

10. A Formula-kiértékelés **LOGSPACE**, a Boole-hálózat-kiértékelés **P**-teljes: http://www.cs.huji.ac.il/ tamirr/complexity2001/ex/ex1.ps

11. Ha **PSPACE** $\subseteq$ **P/poly**, akkor **PSPACE** = $\mathbf{\Sigma_k}$ valamely k-ra. http://www.cs.huji.ac.il/ tamirr/complexity2001/ex/ex5.ps

12. Minden c konstanshoz létezik L ritka nyelv, és k konstans, hogy L $\in \mathbf{\Sigma_k}$, de L $\notin$ **SIZE**(n^c). Hint itt: http://www.cs.huji.ac.il/