Mi a Kolmogorov-bonyolultsága ennek a példatárnak?

2020. december 21.

Bevezető

A példatár az ELTE matematikus és alkalmazott matematikus szak BSc harmadéves Számítástudomány és MSc elsőéves Bonyolultságelmélet tárgyak gyakorlataihoz használt feladatokat és megoldásaik vázlatát tartalmazza. A forrásfile-ok az EGRES csoport példatárán alapulnak. Köszönet ...

Budapest, 2020. május 35.

Tartalomjegyzék

Jelölések

- * : nehezebb példa,** : nehéz példa.
- Nyelv: L egy nyelv a Σ ábécé felett, ha elemei karaktersorozatok, ahol a karakterek Σ-beliek. Többnyire $\Sigma = \{a, b\}$ vagy $\Sigma = \{0, 1\}$ lesz, ha mást nem mondunk.

I. rész

Feladatok

1. fejezet

Véges Automaták

1.1. DFA

Definíció: Véges automata: $M = (Q, \Sigma, \delta, q_0, F)$ véges automatának Q az állapotainak halmaza (véges sok), Σ ábécé felett van értelmezve, δ az átmeneti függvénye (azaz adott betűt olvasva egy állapotban milyen új állapotba kerülünk), q_0 a kezdőállapota, F a végállapotainak halmaza. Azon szavak halmazát, melyeket elolvasva az automata valamelyik végállapotába kerül, $L(M)$-lel jelöljük. Azt is mondjuk, hogy M felismeri az $L(M)$ nyelvet és elfogadja az $L(M)$-beli szavakat. Részletesebben lásd: http://en.wikipedia.org/wiki/Deterministic_finite-state_machine.

1. Legyen $Q = \{q_0, q_1\}$, $F = \{q_1\}$ és $\delta(q_i, a) = q_i$ és $\delta(q_i, b) = q_{1-i}$. Milyen nyelvet ismer fel ez az automata? (Megoldás)

2. Adjunk véges automatát, mely az alábbi nyelveket ismeri fel!
a) L = a *baba*-t tartalmazó szavak.
b) L = az *abba*-t nem tartalmazó szavak.
c) L = azon szavak, melyek utolsó előtti betűje b.
d) L = azon szavak, melyekben bármely egymásutáni öt karakterből legalább kettő a.
e) L = magyar (.hu végű) email címek. (Megoldás)

3. Találj ki egy nyelvet, amiről tudod, hogy reguláris-e, de a padtársad nem! (Megoldás)

4. Mutasd meg, hogy van olyan nyelv, amit semely véges automata sem tud felismerni! (Ezeket *nem reguláris* nyelveknek hívjuk.) (Megoldás)

5. Adott két automata, melyek az L_1 ill. L_2 nyelveket ismerik fel. Adj automatát, amely felismeri az
a) $\bar{L}_1$ (azaz L_1 komplementer) nyelvet,
b) $L_1 \cup L_2$ nyelvet,
c) $L_1 \cap L_2$ nyelvet,
d)* $L_1 L_2$ nyelvet. (Azaz olyan szavak, melyek felbonthatók egy L_1 és egy L_2-belire.) (Megoldás)

6.* A kétirányú véges automata abban különbözik a hagyományostól, hogy nem kötelező mindig a következő betűt kérnie, hanem visszamehet az előzőhöz is. (Ez bele van kódolva δ-ba,

hogy mikor merre lép.) Mutasd meg, hogy bármely ilyen automatához létezik egy egyirányú, amely pont ugyanazt a nyelvet ismeri fel! (Megoldás)

7.** Azt mondjuk, hogy x részsorozata y-nak, ha x-et megkaphatjuk y-ból néhány karakter kitörlésével. Adott L nyelv esetén jelölje $SUBSEQ(L)$ azon szavak halmazát, amik valamely L-beli szó részsorozatai. Mutasd meg, hogy tetszőleges L esetén $SUBSEQ(L)$ felismerhető véges automatával, ha
a) a Σ ábécé kételemű,
b) bármely véges Σ ábécé esetén. (Megoldás)

8. Bemenet tizes számrendszerbeli szám. Jegyeit váltakozó előjellel adjuk össze, majd ezt ismételgessük, amíg egyjegyű számot nem kapunk. Mutassuk meg, hogy azon számok nyelve, amikre nullát kapunk, reguláris.
b)** Mi a helyzet, ha csak minden másodikat összeadunk (a többit meg elfelejtjük) és ezt ismételgetjük? (Megoldás)

9. Pumpálós lemma: $\forall L$ *reguláris* $\exists n\ \forall z \in L, |z| \geq n\ \exists u, v \neq \emptyset, w:\ z = uvw$ *és* $|uv| \leq n$ *és* $\forall i\ uv^i w \in L$. (Megoldás)

10. Döntsük el, hogy az alábbi nyelvek regulárisak-e!
a) prímszámok egyes számrendszerben.
b) négyzetszámok egyes számrendszerben.
c) négyzetszámok kettes számrendszerben.
d)* prímszámok kettes számrendszerben. (Megoldás)

11. a) Van-e olyan $S \subset \mathbb{N}$, ami kettes számrendszerben reguláris, de egyesben nem?
b) És fordítva?
c)** És olyan, ami kettesben és hármasban is reguláris, de egyesben nem? (Megoldás)

1.2. NFA

Definíció: Nemdeterminisztikus véges automata: Ugyanaz, mint a determinisztikus, csak egy állapotból többfelé is léphet. (A gráfjában egy csúcsból több azonos indexű él is kiléphet.) Akkor fogad el egy szót, ha van legalább egy olyan futása, ami elfogadja.

12. Mutasd meg, hogy a nemdeterminisztikus véges automaták is csak reguláris nyelveket ismernek fel, azaz minden nemdeterminisztikus véges automatához van egy determinisztikus, ami ugyanazt a nyelvet fogadja el! (Megoldás)

13. Adott L nyelvre L^* azon szavak nyelve, amik előállnak néhány L-beli szó konkatenációjaként. Mutasd meg, hogy ha L reguláris, akkor L^* is! (Megoldás)

14.* Az alternáló véges automata egy nemdeterminisztikus véges automata, melynek minden állapotához hozzá van rendelve **A** vagy **B**. Ez két játékos, akik előre látják az inputot, az **A** jelű állapotoknál **A**, a **B** jelű állapotoknál **B** dönt, hogy melyik állapotba lépjen tovább az automata. **A** célja, hogy az automata elfogadó állapotba érjen a szó végén, míg **B** célja, hogy ne. Akkor fogad el egy szót az automata, ha **A**-nak van rá nyerő stratégiája, az elfogadott

szavak nyelvét pedig „felismeri” az automata. Mutasd meg, hogy az alternáló véges automaták is csak reguláris nyelveket ismernek fel! (Megoldás)

15. Randomizált véges automata: Ugyanaz, mint a nemdeterminisztikus, csak minden éléhez hozzá van rendelve egy valószínűség, ez határozza meg, hogy merre mekkora eséllyel lép. Akkor fogad el egy szót, ha a futásainak $\geq \frac{2}{3}$-a elfogadja és akkor utasítja el, ha a futásainak $\leq \frac{1}{3}$-a fogadja el. Akkor ismer fel egy nyelvet, ha a nyelvbeli szavakat mind elfogadja és a nem nyelvbeli szavakat mind elutasítja (tehát semelyik szóra sem lehet határozatlan).
a)* Mutasd meg, hogy van olyan kétirányú randomizált véges automata, ami felismeri az $L = \{a^n b^n\}$ nyelvet!
b)** Mutasd meg, hogy egyirányú randomizált véges automata csak reguláris nyelveket tud felismerni! (Megoldás)

2. fejezet

Turing-gépek

2.1. Alapok

16. Készíts olyan Turing-gépet, ami a bemenetként kapott $x_1x_2\dots x_m$ sorozatot
a) megkétszerezi ($x_1x_2\dots x_mx_1x_2\dots x_m$).
b) megfordítja ($x_mx_{m-1}\dots x_1$),
c) betűnként megkétszerezi ($x_1x_1x_2x_2\dots x_mx_m$). (Megoldás)

17. Konstruáljunk olyan Turing-gépet, ami
a) az n hosszú, csupa 1-esből álló bemenetre n kettes számrendszerbeli alakját adja vissza,
b) ha a bemenet az n szám bináris alakja, akkor n darab 1-est ír ki (különben azt, hogy „*hibás bemenet*"). (Megoldás)

18. Tegyük fel, hogy van két Turing-gépünk, melyek az f illetve g függvényt számolják ki. Konstruáljunk olyan Turing-gépet, mely az $f \circ g$ függvényt számolja ki. (Megoldás)

19. a) Konstruáljunk olyan Turing-gépet, mely az a^nb^n alakú szavakra $O(n)$ lépésben kiírja, hogy „*igen*", más szavakra végtelen sokáig fut.
b) Konstruáljunk olyan egyszalagos Turing-gépet, mely az a^nb^n alakú szavakra $O(n \log n)$ lépésben kiírja, hogy „*igen*", más szavakra végtelen sokáig fut.
c)* Mutassuk meg, hogy ez egyszalagos gépen $o(n \log n)$ lépésben nem lehetséges. (Megoldás)

20.* Mutassuk meg, hogy egy k szalagos Turing-gép szimulálható egy 2 szalagossal úgy, hogyha az eredeti n lépést tett egy adott inputra, akkor a kétszalagos legfeljebb $O(n \log n)$-et tegyen. (Megoldás)

21. Adj meg egy Turing-gépet, ami bármely x bemenetre pontosan $2^{|x|}$ lépést tesz meg. (Megoldás)

22. Adott egy egész együtthatós, egy változós $p(x)$ polinom. Döntsük el, hogy van-e a $p = 0$ egyenletnek egész megoldása. (Megoldás)

23. Adott a és b természetes számpár inputra polinomiális időben ki tudjuk számolni
a) $a + b$-t,

b) ab-t,
c) $lnko(a, b)$-t. (Megoldás)

24. Adottak a, b és m természetes számok. Előadáson volt, hogy lnko(a, b)-t meg a^b mod m-t ki tudjuk polinomiális időben számolni. Mi a helyzet a/b mod m-mel? (Itt b és m relatív prímek.) (Megoldás)

25. Mutasd meg, hogy ha ki tudjuk számolni $k!$ mod m-et polinom időben minden k, m inputra, akkor polinom időben tudunk faktorizálni is. (Megoldás)

26. Tekintsük azon TG-ek osztályát, amik nem írhatnak a szalagjaikra, hanem csak olvashatnak.
a) Bizonyítsd be, hogy az egyszalagos csakolvasó TG-ek pontosan a reguláris nyelveket döntik el.
b)* Bizonyítsd be, hogy százszalagos csakolvasó TG-pel bármely rekurzív nyelv eldönthető. (Minden szalagján megkapja a szót a gép.) (Megoldás)

27. Hogyan adjunk meg egy gráfot a Turing-gépnek? (Megoldás)

2.2. Rekurzív és rekurzívan felsorolható

28. Legyenek L_1 és L_2 rekurzívan felsorolható nyelvek. Mutasd meg, hogy $L_1 \cup L_2$ és $L_1 \cap L_2$ is rekurzívan felsorolható. (Megoldás)

29. Tegyük fel, hogy L rekurzív és $L' \subset L$. Mutasd meg, hogy L' akkor és csak akkor rekurzív, ha rekurzívan felsorolható és $L \setminus L'$ is rekurzívan felsorolható. (Megoldás)

30. Ha L rekurzívan felsorolható, de nem rekurzív és $L' \subset L$, akkor lehet-e L'
a) rekurzív?
b) co-rekurzívan felsorolható, de nem rekurzív? (Megoldás)

31. Mutasd meg, hogy rekurzívan felsorolható, de nem rekurzív nyelv létezéséből következik Gödel első nemteljességi tétele, miszerint van olyan állítás, mely se nem bizonyítható, se nem cáfolható (bármely, néhány egyszerű követelménynek eleget tevő, rendszerben). (Megoldás)

32. Bizonyítsd be, hogy eldönthetetlen, hogy két TG ugyanazokra a szavakra áll-e meg. (Megoldás)

33. Tegyük fel, hogy a T TG minden inputra megáll, akkor is, ha nem követeljük meg, hogy a szalagon csak véges sok nem *üres* jel legyen. Bizonyítsd be, hogy van olyan n, hogy T sosem megy a szalagon n-nél messzebb. (Megoldás)

34. Ha beugrunk egy fekete lyukba, akkor mialatt számunkra eltelik egy perc, a külső világban eltelik egy örökkévalóság. Tehát ha meg akarjuk tudni, hogy megáll-e egy T Turing-gép, akkor nincs más dolgunk, mint beugrani egy fekete lyukba, és megkérni egy haverunk, hogy ugorjon utánunk, ha leállt T. Ennek vegyük azt az elméleti modelljét, hogy egy *fekete lyukbeli*

univerzális Turing-gép el tudja dönteni egy *hagyományos* Turing-gépről, hogy leáll-e.
a) A fekete lyukbeli megállási probléma is eldönthető a fekete lyukban?
b)** Mutasd meg, hogy van olyan rekurzívan felsorolható L nyelv, hogy a fekete lyukban L rekurzív lesz, de ha csak $x \overset{?}{\in} L$ kérdéseket tehetünk fel a fekete lyukban, akkor a (hagyományos) megállási probléma nem lesz rekurzív, tehát L *gyengébb* orákulum, mint a megállasi probléma. (Megoldás)

2.2.1. Dominók

Definíció: Dominó-probléma: Ki akarjuk parkettázni a síkot egy véges dominó-készlettel, mely tartalmaz egy *Start* elemet, amit muszáj felhasználnunk.

35. Eldönthető-e, hogy kiparkettázható-e a sík a készlettel, ha
a) a dominókat el szabad forgatni 180-kal?
b) a dominókat el szabad forgatni a függőleges tengelyük körül?
(Azaz a függőleges tengelyre tükrözöttjük is a készletben van.)
c) a dominókat el szabad forgatni az *egyik* átlójuk körül?
(Azaz az átlóra tükrözöttjük is a készletben van.) (Megoldás)

36. Mutasd meg, hogy ha a dominóknak nem a szélein, hanem a sarkain vannak a jelek, akkor is eldönthetetlen, hogy kiparkettázható-e a sík. (A találkozásoknál mind a négy jelnek egyeznie kell.) (Megoldás)

37.** Mutassuk meg, hogy van olyan (kötelezően felhasználandó *Start*-dominó NÉLKÜLI) dominókészlet, mellyel a sík kirakható, de nem rakható ki kétszeresen periodikusan (vagyis úgy, hogy alkalmas lineárisan független egész koordinátájú (p, q) és (r, s) vektorokra bármely (x, y) pontot ugyanolyan dominó fed le, mint az $(x + p, y + q)$ és $(x + r, y + s)$ pontot). (Megoldás)

3. fejezet

Tár

Definíció: STconn: Adott n csúcsú (irányított) gráfban el kell dönteni, hogy s és t között van-e út.

38.* Mutasd meg, hogy ha a **DSPACE**$(f(n))$ definícója nem az lenne, hogy hány mezőre lép a Turing-gép, hanem az, hogy hányra ír, akkor **DSPACE**(2)-ben benne lenne az összes rekurzív nyelv. (Megoldás)

39.* a) Mutass egy nem reguláris nyelvet, amit egy kétszalagos, egy szalagot csak olvasó, a másikon $O(\log\log n)$ tárt használó TG eldönt.
b) Bizonyítsd be, hogy egy kétszalagos, egy szalagot csak olvasó, a másikon legfeljebb $o(\log\log n)$ tárt használó TG-pel csak reguláris nyelvet lehet felismerni.
(Erre a TG-nek kell vigyázni, hogy ne használjon több helyet, nincs jelölve a szalag vége!)
(Megoldás)

40. STconn $\in$ **DSPACE**$(\log^2 n)$. (Megoldás)

41. Grid-STconn $\in$ **L** = **DSPACE**$(\log n)$. (Grid-STconn feladat inputja egy $n \times n$-es labirintus és két (nem-fal mező), s és t. Egy input akkor van Grid-STconn-ban, ha s és t között van út.) (Megoldás)

42. Grid-STconn $\in$ **L** = **DSPACE**$(\log n)$. (Grid-STconn feladat inputja egy $n \times n$-es labirintus és két (nem-fal mező), s és t. Egy input akkor van Grid-STconn-ban, ha s és t között van út.) (Megoldás)

43. a) Mutassuk meg, hogy a (,) kételemű ábécé felett a helyesen zárójelezett szavak nyelve **L**-ben van.
b) Mutassuk meg, hogy a (, [,),] négyelemű ábécé felett a helyesen zárójelezett szavak nyelve **L**-ben van.
(Megoldás)

44.** Legyen az $\{a, b, a^{-1}, b^{-1}\}$ négyelemű ábécé felett azon szavak nyelve, amik az $\{a, b\}$ elemek által generált szabad csoportban a triviális elemmel egyenlőek Triv^{a,b}. (Kombinatorikus megfogalmazásban: Ha az egymás mellett álló inverzeket egyszerűsítjük egymással, akkor ezt a lépést mohón ismételgetve az üres szóhoz jutunk-e.) Mutassuk meg, hogy Triv$^{a,b} \in$ **L** = **DSPACE**$(\log n)$. (Megoldás)

45.** Tegyük fel, hogy adva van egy síkgráf az $n \times n$-es rácsra rajzolva, azaz minden csúcsának meg van adva mindkét koordinátája úgy, hogy az élek nem metszik egymást. Irányítsuk meg az éleit odavissza. A cél, hogy az irányított élekhez hozzárendeljük egy $w(uv)$ súlyfüggvényt, amire $w(vu) = -w(uv)$ és minden irányított körre a súlyok összege ne legyen nulla. Meg tudjuk-e ezt csinálni **L**-ben?
(Tehát ha pl. azt akarnánk, hogy az összeg nulla legyen minden körön, akkor jó lenne az azonosan nulla függvény vagy a $w(uv) = u_x - v_x$.) (Megoldás)

46. Adott egy G gráf és egy s kiinduló csúcs.
a) Indukcióval ki tudjuk számolni az s-től d távolságra levő pontok pontos számát **NL**-ben.
b) **NL**-ben el tudjuk dönteni, hogy t benne van vagy nincs benne s komponensében.
c) **NL** = **co-NL**. (Megoldás)

47. Bipartite $\in$ **NL**. (Megoldás)

48. Mutassuk meg a 3-összefüggő gráfok nyelvéről, hogy 3conn $\in$ **NL**. (Megoldás)

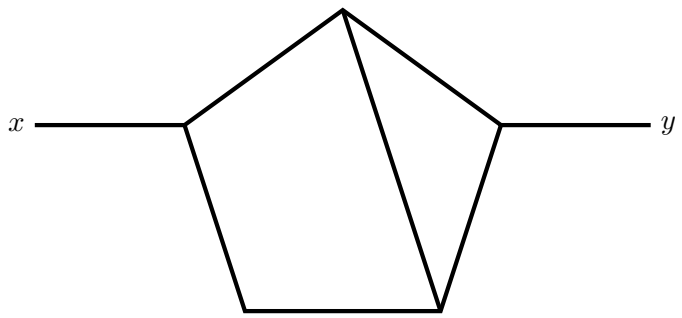
49. Ha **PSPACE** = **PH**($\stackrel{def}{=} \cup\mathbf{\Sigma_k}$), akkor **NL** $\neq$ **NP**. (Megoldás)

50. Mutasd meg, hogy ha **L** = **NL**, akkor **DSPACE**(n) = **NSPACE**(n). (Megoldás)

3.1. Generalized Geography

51. Aciklikus gráfokra is **PSPACE**-teljes a Generalized Geography? Hány élet kell hozzáadnunk egy aciklikus gráfhoz, hogy már **PSPACE**-teljes legyen? (Megoldás)

52. A Generalized Geography játékot irányítatlan gráfokra is hasonlóan definiálhatjuk kétféleképpen is. Az első esetben a játékosok csúcsokat választanak (VUGG), a másodikban pedig éleket (EUGG).
a) Mutasd meg, hogy a kezdő játékosnak akkor és csak akkor van nyerő stratégiája egy G gráf v csúcsából indulva a VUGG játékban, ha minden maximális párosítás tartalmazza v-t.
b) A alábbi gadget felhasználásával mutasd meg, hogy EUGG **PSPACE**-teljes. (Megoldás)



53. Azt mondjuk, hogy egy irányított gráf tömören reprezentált, ha adott egy T Turing-gép, ami u, v bemenetre kiírja, hogy uv él-e $\leq n^k$ időben, ahol k egy fix konstans, pl. a

feladat szempontjából legyen $k = 2$. A SUCCINCT-STCONN nyelv elemei azon T Turing-gépek leírásai, amik olyan gráfot tömörítenek, amiben van st út.
a) Mutasd meg, hogy **PSPACE**-ben eldönthető SUCCINCT-STCONN.
b) Mutasd meg, hogy SUCCINCT-STCONN **PSPACE**-teljes. (Megoldás)

54.** a) A GENERALIZED GEOGRAPHY-nak az a verziója is **PSPACE**-teljes, ha a két játékos külön-külön építi a saját láncát ugyanazon az irányított gráfon. (Bármely csúcsba csak egyikük léphet; mindkettőnek meg van adva, hogy honnan kezd.)
b) Ugyanez irányítatlan gráfokra is **PSPACE**-teljes.
Mj. Azt nem tudom vizsgálták-e ezekben a verziókban, hogy mi van, ha csúcsokat szabad újra használni, csak az éleket nem. (Megoldás)

4. fejezet

NP

55. a) Mutasd meg, hogy **PSPACE**-ben van a 3-színnel színezhető gráfok nyelve.
b) Mutasd meg, hogy **NP**-ben van a 3-színnel színezhető gráfok nyelve.
c) Mutasd meg, hogy ha **P**-ben van a 3-színnel színezhető gráfok nyelve, akkor polinom időben tudunk is keresni egy 3-színnel színezést. (Megoldás)

56. a) Mutasd meg, hogy **NP**-ben van, hogy egy gráfban van-e teljes párosítás.
b) Mutasd meg, hogy ha **P**-ben van, hogy egy gráfban van-e teljes párosítás, akkor polinom időben tudunk is keresni egyet (ha van).
c) Mutasd meg, hogy páros gráfban tudunk teljes párosítást keresni polinom időben.
d)** Mutasd meg, hogy általános gráfban is. (Persze lehet, hogy valaki már tanulta.) (Megoldás)

57. Mutasd meg, hogy
a) **co-NP**-beli,
b)* **NP**-beli,
c)** **P**-beli
a Prímek illetve a Síkbarajzolható gráfok nyelve.
(Ez összesen 6 feladat, van köztük megoldhatatlanul nehéz is.) (Megoldás)

58. a) Mutasd meg, hogy $\mathbf{DTIME}(n^2) \neq \mathbf{DTIME}(n^{10})$.
b)** Mutasd meg, hogy $\mathbf{NTIME}(n^2) \neq \mathbf{NTIME}(n^{10})$. (Megoldás)

59. a) Mutasd meg, hogy van olyan f, amire minden rekurzív nyelv benne van a $\mathbf{DTIME}(f(n))$-ben.
b)* Mutasd meg, hogy van olyan rekurzív f, amire $\mathbf{DTIME}(f(n)) = \mathbf{DTIME}(2^{f(n)})$. (Megoldás)

4.1. NP-teljesség

60. Adva van egy konjunktív normálforma. Szeretnénk eldönteni, hogy megválaszthatjuk-e úgy a változókat, hogy minden klóz tartalmazzon igaz és hamis literált is. Mutassuk meg, hogy annak eldöntése, hogy ezt lehet-e, **NP**-teljes. (Megoldás)

61. **NP**-teljes, hogy egy $2n$ pontú gráfban van-e n-es klikk. (Megoldás)

62.* Mutasd meg, hogy annak eldöntése, hogy egy gráfban van-e Hamilton-út, **NP**-teljes. (Megoldás)

63. Az előző feladatot felhasználva mutasd meg, hogy az alábbi feladatok is **NP**-teljesek.
a) Van-e a gráfban Hamilton-kör?
b) Van-e a gráfban a csúcsoknak legalább a felét tartalmazó kör?
c) TSP: Ha az éleken adott egy metrikus hosszfüggvény (azaz a gráf teljes és teljesül a hosszakra a háromszög egyenlőtlenség), van-e legfejlebb K összhosszú Hamilton-kör? (Megoldás)

64. Adott egy lineáris egyenlőtlenségrendszer.
a) Mutassuk meg, hogy annak eldöntése, hogy van-e egész megoldása, **NP**-nehéz.
b)* Mutassuk meg, hogy ez a feladat **NP**-beli. (Megoldás)

65.* Mutasd meg, hogy **NP**-teljes az n^2 elemű dominókészletek, amikkel lefedhető egy $n \times n$-es négyzet. (Megoldás)

66.* Annak eldöntése, hogy egy Sudokunak van-e megoldása, **NP**-teljes. (Ha $n \times n$-es kisnégyzetek vannak.) (Megoldás)

67. Jelölje k-SAT azon kielégíthet? konjunktív normálformák nyelvét, amelyekben minden klóz legfeljebb k elem?, SAT-k pedig azon konjunktív normálformákét, ahol minden változó legfeljebb k literálban fordul el?. Bizonyítsuk be (egyik volt előadáson), hogy
a) 2-SAT P-beli,
b) 3-SAT NP-teljes,
c) SAT-2 P-beli,
d) SAT-3 NP-teljes. (Megoldás)

68. Annak eldöntése, hogy természetes számok egy adott halmazát szét tudjuk-e osztani két egyenlő részre, **NP**-teljes. (Megoldás)

69. A 3DM problémánál az input egy páros gráf, melyben a felső osztályban minden pont foka 3. Az alsó osztály három színnel van színezve, minden felső osztálybeli pontnak minden színezett részbe egy éle megy. A kérdés, hogy létezik-e olyan „teljes alulról bepárosítás", hogy alul mindenkinek egy, felül mindenkinek 0 vagy 3 a foka. Mutassuk meg, hogy 3DM **NP**-teljes.
(A feladat egy ekvivalens megfogalmazása, hogy adott egy három színnel színezett alaphalmaz és rajta egy 3-uniform hipergráf. A cél egy teljes párosítás keresése, ahol az élek most ugye három eleműek.) (Megoldás)

70.* A Planar*-3SAT problémánál adott egy síkbeli páros gráf, melynek csúcsai egy konjuktív normálformula klózainak és változóinak felelnek meg, két csúcs akkor van összekötve, ha a változó szerepel a klózban. Mutassuk meg, hogy annak eldöntése, hogy igazzá tehető-e, **NP**-teljes. (Megoldás)

71. Mutasd meg, hogy síkgráfra meghatározni a kromatikus számát
a)* **NP**-teljes,
b) de háromszögelt síkgráfra **P**-beli. (Megoldás)

72. Egy G gráf *félkromatikus száma* az a legkisebb k, amire G csúcsainak felét lehet úgy k-színezni, hogy bármely két szomszédos csúcs más színt kapjon. (Csúcsok másik fele színezetlen maradhat.) Mutasd meg, hogy **NP**-teljes eldönteni, hogy egy gráf félkromatikus száma 3-e. (Megoldás)

Definíció: Egy halmazrendszer/hipergráf (csúcsainak) k-színezését többféleképpen definiálhatjuk.
a) (gyenge) k-színezés: minden hiperélben szerepeljen két különböző szín.
b) szivárvány k-színezés: minden hiperélben minden szín különböző legyen.
c) konfliktusmentes k-színezés: minden e hiperélben szerepeljen egy csúcs, aminek a színe az összes többi e-beli színtől különböző.
d) polikromatikus k-színezés: minden hiperélben szerepeljen mind a k szín.

73. Mutasd meg, hogy mind a négy verzió **NP**-teljes minden $k \geq 3$ konstansra. (Megoldás)

74. Mutasd meg, hogy k-uniform hipergráfokra is **NP**-teljesek minden $k \geq 3$ konstansra. (Megoldás)

75.** Annak eldöntése, hogy egy kvadratikus diophantoszi egyenletnek van-e megoldása, **NP**-teljes. (Megoldás)

76.* Egy n résztvevős focibajnokságot k forduló után le kellett állítani a koronavírus miatt.
a)** (nem ismert) **NP**-teljes-e input tabella alapján eldönteni, hogy egy csapat lehet-e még bajnok?
b)* Ha az is az input része, hogy ki kivel játszott, akkor **NP**-teljes. (Megoldás)

5. fejezet

PAD

Definíció: $PAD(L) = \{x0^{n^2} : x \in L\}$ (azaz x mögé írunk még egy csomó nullást és persze $n = |x|$).

77. Mutassuk meg, hogy $\mathbf{DTIME}(n) \neq \mathbf{DTIME}(n^{1.01}) \neq \mathbf{DTIME}(n^2) \neq \mathbf{DTIME}(n^{2.01})$. (Megoldás)

78. Ha $\mathbf{P} = \mathbf{NP}$, akkor $\mathbf{EXP} = \mathbf{NEXP}$. (Megoldás)

79. $\mathbf{P} \cap \mathbf{TALLY} = \mathbf{NP} \cap \mathbf{TALLY} \iff \mathbf{E} = \mathbf{NE}$.
(**TALLY**-val jelöljük az egybetűs nyelvek osztályát, $\mathbf{E} = \mathbf{DTIME}(2^{O(n)})$ és $\mathbf{NE} = \mathbf{NTIME}(2^{O(n)}$.)
(Megoldás)

80. Ha $\mathbf{E} = \mathbf{NE}$, akkor nincs ritka nyelv $\mathbf{NP} \setminus \mathbf{P}$-ben. Azaz ha $\mathbf{DTIME}(2^{O(n)}) = \mathbf{NTIME}(2^{O(n)})$, akkor ha egy $L \in \mathbf{NP}$ nyelvben minden n-re az n hosszú szavak száma $O(n^c)$, akkor $L \in \mathbf{P}$. (Megoldás)

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

6. fejezet

Polinomiális hierarchia és orákulumok

Definíció: $\exists^p L := \left\{ x \in \{0,1\}^* \;\middle|\; \left(\exists w \in \{0,1\}^{\leq p(|x|)}\right) \langle x, w\rangle \in L \right\}$,
és hasonlóan $\forall^p L := \left\{ x \in \{0,1\}^* \;\middle|\; \left(\forall w \in \{0,1\}^{\leq p(|x|)}\right) \langle x, w\rangle \in L \right\}$.
Ezekkel a jelölésekkel $\mathbf{NP} = \exists \cdot \mathbf{P}$ és $\mathbf{co\text{-}NP} = \forall \cdot \mathbf{P}$. Ezen felbuzdulva $\mathbf{\Sigma_k} := \exists \cdot \forall \cdot \exists \ldots \mathbf{P}$ és $\mathbf{\Pi_k} := \forall \cdot \exists \cdot \forall \ldots \mathbf{P}$, ahol összesen k db kvantor áll a $\mathbf{P}$ előtt. Tehát ezekkel a jelölésekkel $\mathbf{NP} = \mathbf{\Sigma_1}$ és $\mathbf{co\text{-}NP} = \mathbf{\Pi_1}$. A polinomiális hierarchia $\mathbf{PH} := \cup \mathbf{\Sigma_k}$.

82. $\mathbf{\Sigma_k} \subset \mathbf{\Sigma_{k+1}} \cap \mathbf{\Pi_{k+1}}$. (Megoldás)

83. $\mathbf{\Sigma_k} \subset \mathbf{PSPACE}$. (Megoldás)

84. $\mathbf{\Pi_k} \subset \mathbf{\Sigma_k} \Rightarrow \mathbf{\Sigma_k} = \mathbf{PH}$. (Megoldás)

85. Tegyük fel, hogy adva van egy konjuktív normál forma, melynek változói x_i-k és y_i-k. Azt szeretnénk eldönteni, hogy minden x_i behelyettesítésre van-e az y_i-knek olyan behelyettesítése, ami kielégítő. Mutassuk meg, hogy ez $\mathbf{\Pi_2}$-ben van. Mi lenne, ha az lenne a kérdés, hogy van-e x_i, hogy minden y_i-re igaz a formula? (Megoldás)

86. Legyen L az $n^2 \times n^2$ méretű olyan általánosított Sudoku állások nyelve, ahol akárhova beírunk egy számot (nem azonos sorba/oszlopba/négyzetbe egy ugyanolyan számmal), marad megoldása.
a) Melyik osztályban van L?
b) Nincs benne egy kisebben is? (Megoldás)

87.* $\mathbf{NP} \subset \mathbf{P/poly} \Rightarrow \mathbf{\Sigma_2} = \mathbf{PH}$. (Megoldás)

88.* $\mathbf{EXP} \subset \mathbf{P/poly} \Rightarrow \mathbf{EXP} = \mathbf{\Sigma_2}$. (Megoldás)

89. Tegyük fel, hogy adott egy játék, ahol két játékos felváltva lép és egy $poly(n)$ méretű táblán játszanak úgy, hogy felváltva elfoglalnak egy mezőt (pl. amőba). Az állás alapján mindig el lehet dönteni $\mathbf{P}$-ben, hogy nyert-e valaki. Tegyük fel, hogy a játék k lépés alatt garantáltan véget ér. Mutassuk meg, hogy azon állások nyelve, amikre kezdőnek van nyerő stratégiája $\in \mathbf{\Sigma_k}$. Sőt. (Megoldás)

Definíció: Az *orákulumos* Turing-gép egy többszalagos $M^?$ Turing-gép, melynek egyik szalagja a kitüntetett kérdező-szalag. Az orákulum egy tetszőleges A nyelv lehet. $M^?$-nek

van egy speciális $q_?$ állapota. Ha ebbe kerül, akkor az orákulum válaszol neki, hogy az éppen akkor a speciális szalagján levő szó benne van-e az A nyelvben.

Az x bemenethez tartozó kimenetet $M^A(x)$-szel jelöljük. Ha **XP** egy olyan nyelv, ami Turing-gépek egy speciális családjával van definiálva, akkor $\mathbf{XP}^A$ jelöli azokat az L nyelveket, amihez van ebből a családból Turing-gép, amihez A orákulumot adva L-et ismeri fel. Ha valamely bonyolultsági osztályhoz van teljes nyelv, akkor gyakran a bonyolultsági osztályt írjuk a kitevőbe, pl. $\mathbf{P}^{\mathbf{NP}}$ azt jelenti, hogy $\mathbf{P}^{\mathrm{SAT}}$. Ha nincs teljes nyelv, akkor is írhatunk bonyolultsági osztályt a kitevőbe, ekkor az új osztály az összes lehetséges nyelv uniója, pl. $\mathbf{P}^{\mathbf{BPP}} = \cup\{\mathbf{P}^L \mid L \in \mathbf{BPP}\}$.

90. Legyen A egy **P**-beli nyelv. Mutassuk meg, hogy $\mathbf{P}^A = \mathbf{P}$. (Megoldás)

91. $\mathbf{P}^{\mathbf{NP}} = \mathbf{P}^{\mathrm{SAT}}$. (Megoldás)

92. Ha $\mathbf{E} = \mathbf{DTIME}(2^{O(n)})$, akkor mi lesz $\mathbf{P}^{\mathbf{E}}$? (Megoldás)

93. $\mathbf{PSPACE}^{\mathbf{PSPACE}} = \mathbf{PSPACE}$. (Megoldás)

94. Mutasd meg, hogy $\mathbf{NP} \cup \mathbf{co\text{-}NP} \subset \mathbf{P}^{\mathbf{NP}}$. (Megoldás)

95. $\mathbf{NP}^{\mathbf{NP} \cap \mathbf{co\text{-}NP}} = \mathbf{NP}$. (Megoldás)

96. $\mathbf{NP}^{\mathbf{NP}} = \mathbf{\Sigma_2}$. (Megoldás)

97.* $\mathbf{BPP}^{\mathbf{BPP}} = \mathbf{BPP}$. (Megoldás)

98. Mutass olyan **XP** bonyolultsági osztályt, amire $\mathbf{XP}^{\mathbf{XP}} \neq \mathbf{XP}$. (Megoldás)

99. $\exists A : \mathbf{P}^A = \mathbf{NP}^A$. (Megoldás)

100.* $\exists A : \mathbf{P}^A \neq \mathbf{NP}^A$. (Megoldás)

101.* Van A nyelv, hogy minden $L \in \mathbf{NEXP}^A$-hoz van $S \subset \mathbb{N}$ végtelen halmaz és $L' \in \mathbf{NP}^A$, amire $L|_S = L'|_S$. (Megoldás)

7. fejezet

Véletlen

102. Adj egy (RAM gépen) $\tilde{O}(n^2)$ idejű randomizált algoritmust, ami eldönti input A, B, C $n \times n$-es egész mátrixokról, hogy $A \cdot B = C$ igaz-e. Melyik osztályban van ez az algoritmus **RP**, **co-RP** és **BPP** közül? (Megoldás)

103. Eldönthetelen egy leírásával adott randomizált, polinom időben futó Turing-gépről, hogy igaz-e, hogy minden x-et vagy legalább $1/2$ eséllyel elfogad, vagy biztosan elutasít. (Megoldás)

104. Mutasd meg például a **BPP**-ről, hogy ekvivalens definíciókat kapunk, ha
a) racionális számok az eloszlások (amik meghatározzák, hogy mikor merre lépünk tovább),
b) csak egy darab állapot van, ami 1/2–1/2 eséllyel lép két másikba,
c) van egy véletlen szalagunk, melynek minden mezője $1/2$ eséllyel 0 és 1,
d) egy olyan pénzt „dobálhatunk", ami valamilyen ismeretlen, konstans p eséllyel lesz fej. (Megoldás)

105.* Mutasd meg, hogyha az eloszlásról csak azt tesszük fel, hogy rekurzív, akkor $\mathbf{BPP} \not\subset \mathbf{EXP}$. (Megoldás)

106. Legyen **PP** azon nyelvek osztálya, amikhez van minden inputra mindig polinomiális időben futó randomizált Turing-gép, ami $< 1/2$ eséllyel hibázik.
a) Mutasd meg, hogy SAT $\in$ **PP**.
b) Mutasd meg, hogy ha a **PP**-t várható futásidővel definiáljuk, akkor az így kapott osztály pontosan a rekurzív nyelvek osztálya lesz. (Megoldás)

107. Hogyan tesztelhetjük le gyoran, hogy q prímhatvány-e? (Megoldás)

108.* Egy adott konjuntív normálformáról valaki elárulja nekünk, hogy pontosan egy megoldása van (mint pl. egy Sudoku feladványnál). Mutassuk meg, hogy ha van polinomiális algoritmus, ami megtalálja a megoldást, akkor $\mathbf{RP} = \mathbf{NP}$. (Megoldás)

109. Egy véletlen R nyelvre egy valószínűséggel $\mathbf{BPP} \subsetneq \mathbf{P}^R$. (Megoldás)

110.* $\mathbf{BPP}^{\mathbf{BPP}} = \mathbf{BPP}$. (Megoldás)

111. Igaz vajon, hogy $\mathbf{RP}^{\mathbf{RP}} = \mathbf{RP}$? (Megoldás)

112. Mutasd meg, hogy ha a **BPP** definícióját úgy változtatjuk meg, hogy $x \in L$-re $> 1/2$ eséllyel elfogadjon és $x \notin L$-re $> 1/2$ eséllyel elutasítson, akkor az így definiált új osztálynak része lesz **NP**. (Megoldás)

8. fejezet

Kolmogorov-bonyolultság

Definíció: Egy U Turing-gép univerzális, ha minden T Turing-gépre $U([T], x) = T(x)$, ahol a $[T]$ a T leírását jelenti. Az x Kolmogorov-bonyolultsága U szerint a legrövidebb program hossza, amire kiírja x-et, azaz $K_U(x) = \min\{|p| \mid U(p) = x\}$. Mostantól azt is feltesszük, hogy az ábécé kételemű. (Az első sorban a vessző azért nem csalás, mert $[T]$-t írhatjuk kiterjesztett binárisban.)

113. Minden U, U'-re van C, hogy $K_U(x) \leq K_{U'}(x) + C$, szóval az U-t ezentúl nem fogjuk kiírni. (Megoldás)

114. Mennyi az alábbi szavak Kolmogorov-bonyolultsága?
a) $0101\ldots01$. ($2n$ hosszú.)
b) $11\ldots1$. (2^n db.)
c) Az n. prímszám. (Megoldás)

115. Igaz-e, hogy ha x az y első néhány jegye, akkor $K(x) \leq K(y)$? (Megoldás)

116. Minden n hosszú x-hez van y, ami legfeljebb egy helyen tér el tőle és $K(y) \leq n - \log n$. (Megoldás)

117. Vegyünk egy jó nagy N számot. Valószínűleg a Kolmogorov-bonyolultsága is jó nagy lesz, azaz $K(N) \geq \log N - O(1)$. És valószínűleg van N-nek egy jó nagy prímosztója, p_k, ami a k. legkisebb prím, azaz k is jó nagy. Most úgy fogjuk elkódolni N-et, hogy leírjuk k-t és N/p-t (kettes számrendszerben). Ez persze $\log k$ illetve $\log(N/p_k)$ bit. Ezekből persze vissza tudjuk fejteni N-et, tehát $K(N) \leq \log k + \log N - \log p_k + O(1)$, ahol az utolsó tag a visszafejtő program leírása. Ezt összevetve a korábbi egyenlőtlenséggel kapjuk, hogy $p_k = O(k)$, ami mintha ellentmondana a prímszámtételnek, ha k elég nagy... (Megoldás)

118. a) Van olyan c, hogy semmilyen x-ről nem bizonyítható, hogy $c \leq K(x)$. (Ha az axiómarendszer axiómái rekurzívan felsorolhatóak és c függhet az őket felsoroló Turing-géptől meg a K definíciójánál használt U-tól.)
b)** Van olyan TG, ami minden x bemenetre kiír egy $L(x) \neq K(x)$ számot, amire $L(x) \leq |x|$ és $\lim_{|x|\to\infty} L(x) \to \infty$? (Nyitott kérdés.) (Megoldás)

119.* Van-e olyan végtelen x sorozat, aminek bármely első n jegyére igaz, hogy $K(x_1 \ldots x_n) \geq n - \log n - C$ valamely C konstansra? (Megoldás)

120. Bizonyítsuk be Kolmogorov-bonyolultsággal, hogy egy véletlen gráfban az átmérő nagy valószínűséggel kettő. (Megoldás)

121. Bizonyítsuk be Kolmogorov-bonyolultsággal, hogy van olyan tournament, amiben nincs $3 \log n$ méretű tranzitív résztournament. (Megoldás)

122.* Bizonyítsuk be Kolmogorov-bonyolultsággal a Lovász lokál lemma alábbi verzióját. Létezik $c > 0$, hogy ha adva van néhány n méretű klóz úgy, hogy minden változó legfeljebb $c2^n/n$ darab klózban szerepel, akkor minden klóz egyszerre igazzá tehető. (Megoldás)

9. fejezet

RAM-gépek

123. Írjunk olyan programot a RAM-gépre, mely adott $x[i]$ bemenetre széthúzza azt, azaz $x[2i]$-be rakja $x[i]$-t, a páratlan indexű rekeszeket pedig nullázza. (Megoldás)

124. Írjunk olyan programot a RAM-gépre, mely adott a pozitív egész számra
a) meghatározza azt a legnagyobb m számot, melyre $2^m \leq a$;
b) kiszámítja a kettes számrendszerbeli alakját (az a szám i. bitjét írja az $x[i]$ rekeszbe);
c) adott a és b pozitív egész számokra kiszámítja a szorzatukat.
Ha a és b számjegyeinek száma k, akkor a program $O(k)$ lépést tegyen $O(k)$ jegy? számokkal. (Megoldás)

125. a) Mutasd meg, hogy van univerzális RAM-program, azaz olyan u program, amire minden p programnak van egy kódja, hogy a kódot írva az első pár negatív regiszterbe, u pontosan ugyanazt adja ki bármely inputra, mint p.
b)* Mutasd meg, hogy olyan is van, ami összesen korlátos sok regisztert használ (az inputon és az outputon kívül) az egész futása alatt (p-től függetlenül és beleértve p kódját is). (Megoldás)

126. Legyen $p(x) = a_0 + a_1 x + \cdots + a_n x^n$ egész együtthatós polinom. Adjunk meg egy RAM-gépet, ami a $x[i] = a_i$ inputra kiszámítja $p^2(x)$ együtthatóit. (Megoldás)

127. Hogyan dönthetjük el RAM-géppel, hogy n szám között szerepel-e két azonos? (Megoldás)

9.1. PRAM

Definíció: Előadáson voltak a PRAM különböző modelljei, az EREW, CREW, CRCW és a P-CRCW, mely rövidítések önmagukért beszélnek, kivéve a két utolsó közti különbséget; a CRCW modellben minden adott mezőbe írónak ugyanazt kell írnia, míg a P-CRCW modellben lehet mást és a legkisebb sorszámú processzor írasa lesz sikeres.

128. Mutassuk meg, hogy két n hosszúságú 0-1 sorozatról n processzorral a P-CRCW modellben $O(1)$ lépésben, az EREW modellben $O(\log n)$ lépésben megállapítható, hogy lexikografikusan melyik a nagyobb. (Megoldás)

129. Mutassuk meg, hogy két n hosszúságú 0-1 sorozatnak, mint kettes számrendszerbeli számnak az összegét ki lehet számítani
a) n^2 processzorral $O(1)$ lépésben a CRCW modellben.
b) n^2 processzorral $O(\log n)$ lépésben az EREW modellben.
c) n processzorral $O(\log n)$ lépésben az EREW modellben. (Megoldás)

130. Mutassuk meg, hogy n darab legfeljebb n hosszúságú 0-1 sorozatnak, mint kettes számrendszerbeli számnak az összegét ki lehet számítani
a) n^3 processzorral $O(\log n)$ lépésben a CRCW modellben.
b) n^2 processzorral $O(\log n)$ lépésben az EREW modellben. (Megoldás)

10. fejezet

Hálózatok

Definíció: Azon Boole-hálózatok a nem-unifom $\mathbf{AC^i}$ hálózatok, amik polinomiális méretűek és $O(\log^i n)$ mélységűek. Az uniform esetben kell egy $O(\log n)$ tárú Turing-gép is, ami elő tudja állítani a család n. tagját, ha a bemenete n). Azok a $\mathbf{NC^i}$ hálózatok, amik $\mathbf{AC^i}$-k és minden kapu befoka legfeljebb 2. Az összes uniója a *Nick's Class*, jele (nem-uniform) **NC**. Ezek a jól párhuzamosítható algoritmusok.
Az előadáson volt/lesz, hogy Parity $\notin \mathbf{AC^0}$. Ha egy nem $\mathbf{AC^0}$-beli függvényt vissza tudunk vezetni valamire, akkor az sem lehet $\mathbf{AC^0}$-ban. Ennek felhasználásával (vagy nélkül) mutassuk meg az alábbi függvényekről, hogy nincsenek $\mathbf{AC^0}$-ban, azaz az outputjuk valamelyik bitje nincs $\mathbf{AC^0}$-ban.

131. $\mathbf{NC^0} \subset \mathbf{AC^0} \subset \mathbf{NC^1} \subset \ldots$ (Megoldás)

132. Compare$(x, y) \in \mathbf{AC^0}$. (Compare$(x, y) = 1$, ha $x \geq y$.) (Megoldás)

133. Sum$_i \in \mathbf{AC^0}$. (Sum$_i(x, y) = x + y$-nak az i. jegye.) (Megoldás)

134. Diff$_i \in \mathbf{AC^0}$. (Diff$_i(x, y) = x - y$-nak az i. jegye, ha $x \geq y$, egyébként 0.) (Megoldás)

135. Parity $\in \mathbf{NC^1}$. (A Parity outputja az inputok *mod* 2 összege.) (Megoldás)

136. Majority $\in \mathbf{AC^1}$. (Majority$(x) = 1$, ha több 1-es van x-ben, mint 0-ás.) (Megoldás)

137.* Majority $\in \mathbf{NC^1}$. (Megoldás)

138. Majority $\notin \mathbf{AC^0}$. (Megoldás)

139. Multiply$(x, y) \notin \mathbf{AC^0}$. (Megoldás)

140. Square $\notin \mathbf{AC^0}$. (Megoldás)

141. Cube $\notin \mathbf{AC^0}$. (Megoldás)

142. Legyen DIVISION-BY-$c(x) = \lfloor \frac{x}{c} \rfloor$. Mit gondolsz, milyen c-kre lesz DIVISION-BY-$c \in \mathbf{AC^0}$? (Megoldás)

143. STCONN $\in \mathbf{AC^1}$. (Megoldás)

144. PERFECT MATCHING $\overset{?}{\in} \mathbf{AC^0}$, azaz el lehet-e dönteni konstans mélységben, hogy van-e teljes párosítás egy gráfban? (Megoldás)

145. STCONN $\overset{?}{\in} \mathbf{AC^0}$, azaz el lehet-e dönteni konstans mélységben, hogy van-e st út egy gráfban? (Megoldás)

146. HAMILTONICITY $\overset{?}{\in} \mathbf{AC^0}$, azaz el lehet-e dönteni konstans mélységben, hogy van-e Hamilton–kör egy gráfban? (Megoldás)

147. Gondoljuk végig, hogy *uniform* hálózatokra $\mathbf{AC^0} \subsetneq \mathbf{NC^1} \subset \mathbf{L} \subset \mathbf{NL} \subset \mathbf{AC^1}$. (Megoldás)

11. fejezet

Döntési fák

148.* Tegyük fel, hogy adott egy n csúcsú teljes páros gráf. Egy lépés megkérdeznünk, hogy két adott csúcsa között megy-e él. Hány kérdésre van szükségünk, hogy
a) eldöntsük ugyanannyi csúcs van-e mindkét osztályában? (n ps)
b) találjunk egy csúcsot, ami a nagyobb osztályból van? (n pl)
A válasz mindkét esetben n-nek valami egyszerű függvénye. (Megoldás)

149. Bizonyítsuk be, hogy a MAJORITY zárkózott. (Megoldás)

150. Mely d-kre zárkózott minden elég nagy n-re, hogy egy n jegyű input szám d-vel oszható-e? (Megoldás)

151. Legyen $f_n = 1$, ha egy n hosszú bitsorozatban van két egymásutáni 1-es. Mennyi $D(f_n)$? (Megoldás)

152. A CYCLE-FREE gráftulajdonság zárkózott ha $n \geq 3$. (Megoldás)

153. Az STCONN gráftulajdonság zárkózott. (Megoldás)

154. a) Egy n csúcsú gráf *skorpió*, ha van egy csúcsa (fej), ami össze van kötve $n-3$ darab csúccsal (lábak) és ezeken kívül még egy másodfokú csúccsal (farok), aminek a másik szomszédja egy elsőfokú csúcs (fullánk). (Tehát a lábak között lehetnek további élek.) $D(\text{„skorpió-e a gráf”}) = O(n)$.
b) Hogyan kapcsolódik ez az Aanderaa-Karp-Rosenberg sejtéshez? (Megoldás)

155. Jelölje $K(x \mid y)$ a legrövidebb program hosszát, ami kiszámitja x-et y-ból. Mutasd meg, hogy

$$\exists P \max_{x,y:f(x)\neq f(y)} \min_{i:x_i\neq y_i} 2^{\max\{K(i|x,P),K(i|y,P)\}} \leq D(f).$$

(Megoldás)

156.** Egy 5 széles *branching program* olyan, mint egy egyszerű döntési fa, de nem fa, csak egy szintezett irányított gráf, ami mindenhol legfeljebb 5 széles. Mutasd meg, hogy a polinom hosszú, 5 széles branching programokkal kiszámítható Boole függvények éppen a (nem-uniform) $\mathbf{NC^1}$-beliek. (Megoldás)

Definíció: Ha $f \not\equiv 1$ egy monoton növő Boole-függvény, akkor legyen a hozzá tartozó szimpliciális komplexus $K_f = \{S : f(S) = 0\}$.

157. Ha K_f nem kontrahálható, akkor f zárkózott. (Megoldás)

158.* Ha f invariáns az inputjának egy ciklikus permutációjára, akkor zárkózott. (Azaz $f(x_1 \dots x_n) = f(x_2 \dots x_n x_1)$ a feltétel.) (Megoldás)

12. fejezet

Interaktív

159. Adj a one-time padhez hasonló algoritmust, ami azt tudja, hogy A tud egy olyan üzenetet küldeni, amit B és C együtt meg tud fejteni, de egyedül egyikük sem. (Megoldás)

12.1. Kommunikációs bonyolultság

Definíció: A kommunikációs játékban A és B játékos is ismeri az $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ függvényt és mindkettőjüknek van egy inputja, x illetve y, amit a másik nem lát. Az a feladat, hogy kiszámolják $f(x,y)$-t. Ehhez biteket küldhetnek egymásnak egy el?re megbeszélt protokoll alapján. Adott f-re a legjobb protokoll esetén a legrosszabb (x,y) párra küldendő bitek számát jelölje $\kappa(f)$.

160. A $\geq$ feladatban A és B is kap egy számot 1-től 2^n-ig és el kell dönteniük, hogy A száma legalább akkora-e, mint B-é. Mennyi $\kappa(\geq)$? (Megoldás)

161. A MED feladatban A és B is $\{1\ldots,n\}$ egy-egy részhalmazát kapja inputnak, az output pedig a két halmaz uniójának mediánja (az uniót multihalmaznak kell tekinteni, vagy tegyük fel, hogy diszjunkt elemeket kapnak). Adj algoritmust, ami megoldja a feladatot
a) $O(\log^2 n)$ lépésben.
b)* $O(\log n)$ lépésben. (Megoldás)

162. Definiáljuk a randomizált kommunikációs bonyolultságot és adjunk gyors algoritmust az identitásra. (Megoldás)

163.** Mutasd meg, hogy minden olyan f-re, aminek a mátrixának bármely két sora különbözik $R^{priv}(f) = \Omega(\log n)$, ahol R^{priv} az előző feladatból a megfelelő definíció. (Megoldás)

164. Egy fa két részfájának diszjunktsága eldönthető
a) $O(\log n)$ bittel.
b)* $\log n + \log\log n + O(1)$ bittel.
c)** Ez éles? (Nem tudom.) (Megoldás)

12.2. Karchmer-Wigerson

Definíció: Karchmer-Wigderson játék: Kommunikációs játék egy verziója. A és B játékos is

ismeri az $f : \{0,1\}^n \to \{0,1\}$ függvényt és mindkettőjüknek van egy inputja, x illetve y, amit a másik nem lát. Azt is tudják, hogy $f(x) = 0$ és $f(y) = 1$. Az a feladat, hogy állapodjanak meg egy i indexben, amire $x_i \neq y_i$. Ehhez biteket küldhetnek egymásnak egy előre megbeszélt protokoll alapján. Adott f-re a legjobb protokoll esetén a legrosszabb esetben (x, y párra) küldendő bitek számát jelölje $KW(f)$.

165. Mutasd meg, hogy $KW(\textsc{Parity}) = \Theta(\log n)$. (Azaz A inputjában páros, B inputjában páratlan sok 1-es szerepel és kell találniuk egy eltérést.) (Megoldás)

166.* Mutassuk meg, hogy ha csak az egyiküknek kell jól tippelnie, akkor is kell $\log n + \Theta(1)$ bit. (Megoldás)

167.* Mutasd meg, hogy $KW(f)$ épp egyenlő a legsekélyebb f-et kiszámító max. 2 befokú Boole-hálózat mélységével. (Megoldás)

168.* Mutassuk meg, hogy ha $f(x) = g(|x|)$ (ahol $|x| = \sum_i x_i$), akkor is elég $O(\log n)$ bit ahhoz, hogy megállapodjanak egy i indexben, amire $x_i \neq y_i$. (Megoldás)

12.3. Zero-knowledge

169. $\{(x, m) : \exists y^2 = x \mod m\} \in \mathbf{PZK}$, azaz adjunk egy gyors zero-knowledge protokollt, amivel bizonyítható, hogy x kvadratikus maradék. (Nem kell a helyességet teljes részletességgel bizonyítani!) (Megoldás)

II. rész

Megoldások

1. fejezet

Véges Automaták

1.1. DFA

Definíció: Véges automata: $M = (Q, \Sigma, \delta, q_0, F)$ véges automatának Q az állapotainak halmaza (véges sok), Σ ábécé felett van értelmezve, δ az átmeneti függvénye (azaz adott betűt olvasva egy állapotban milyen új állapotba kerülünk), q_0 a kezdőállapota, F a végállapotainak halmaza. Azon szavak halmazát, melyeket elolvasva az automata valamelyik végállapotába kerül, $L(M)$-lel jelöljük. Azt is mondjuk, hogy M felismeri az $L(M)$ nyelvet és elfogadja az $L(M)$-beli szavakat. Részletesebben lásd: http://en.wikipedia.org/wiki/Deterministic_finite-state_machine.

1. Amikben páratlan sok a betű van.

4. I. Nyelvből kontinuum sok van, automatából viszont csak megszámlálható sok.
II. Az a^{n^2} egy konkrét nyelv, amiről könnyen belátható, hogy nincs hozzá automata, hiszen a-kat olvasva ciklusba esne, így van egy olyan c, hogy ha a^{n^2}-et elfogadja, akkor a^{n^2+c}-t is.

5. a) Cseréljük meg az elfogadó és az elutasító állapotokat.
b-c) Vegyük a két automata állapotterének szorzatát ($Q_1 \times Q_2$) és azon definiáljuk, hogy hogyan kell továbblépni.
d)* Itt $Q_1 \times P(Q_2)$ lesz az új állapottér, ahol P a hatványhalmazt jelöli.

6. Minden jobbra lépésnél jegyezzük fel, hogy ha az eredeti automata balra átlépne, akkor milyen állapotban térne vissza. Egy ilyenből mindig ki tudjuk számolni a következő ilyet; utána a régit törölhetjük, hogy a memória konstans maradjon.

7. Ez a Higman lemma: https://en.wikipedia.org/wiki/Higman%27s_lemma. A lényeg, hogy a Dickson lemma erősebb verziója igaz: ha valami végesen generált, akkor belőle véges sok is végesen generált. Ezt lehet bizonzítani indukcióval és végtelenes Ramsey-vel. Ezutan indukcióval bizonyítjuk az állítást. Kettőre veszünk valamit, ami nincs benne, ezt kiegészítjük alternálóva, ennél több alternálás nem lehet, $\mathbb{Z}$-re Dickson. Háromra 123123..-ra egészítjük ki. Egy szóra vesszük, amíg csak 1,2 váltakozik, aztán 2,3 váltakozást, 3,1-et stb., ilyenekből nem lehet több, mint 123-as szó hossza, megint lehet Dicksonozni.

Egy másik nehéz feladat: http://en.wikipedia.org/wiki/Star_height_problem.

8. a) Ez pont a 11-es maradék lesz, amit könnyű kiszámolni. Forrás: Szögi Evelin feladata.
b) Bocsi, de én sem tudom...

9. Ez a pumpálós lemma: https://en.wikipedia.org/wiki/Pumping_lemma_for_regular_languages. Lényeg, hogy véges sok állapot miatt előbb-utóbb ugyanabba lép újra a gép.

10. a-b) Trivi pumpálós lemmából. Unáris reguláris nyelvek karakterizációja is egyszerű.
c) 2^{2n} és $(2^n+1)^2$ közé bepumpálhatunk.
d) I. Pumpálás x-et $ax+b$-be viszi (ahol $a=2^k$). Vegyük ennek a transzformációnak az orbitjait mod p. Ha kezdetben p osztja x-et, akkor $p!$ lépés után is osztani fogja (ha $p \neq 2$).
II. Az átmeneti mátrix miatt valamely polinomokra az n hosszú szavak száma $p_1(n)\lambda_1^n + \ldots + p_k(n)\lambda_k^n$, ez nem lehet $2^n/n$.
III. (Borda Bence) Bauer Mihály tétel (Dirichlet speciális esete) szerint végtelen sok $p = k2^n+1$ alakú prím van. Pumpálás után $k2^{n+id}+1$, ami $i = p-1$-re osztható p-vel kis-Fermat miatt.

Egy hasonló feladat: http://cstheory.stackexchange.com/questions/2083/proving-the-set-of-powers-of-2-over-ternary-alphabet-to-be-non-regular/2086#2086.

11. a) Van, pl. 2^n.
b) Nincs.
c) Nem tudok egyszerű bizonyítást, kijön a Cobham Tételből: Let S be a set of non-negative integers and let m and n be multiplicatively independent positive integers. Then S is recognizable by finite automata in both m-ary and n-ary notation if and only if it is ultimately periodic. https://arxiv.org/abs/1801.06704.

1.2. NFA

Definíció: Nemdeterminisztikus véges automata: Ugyanaz, mint a determinisztikus, csak egy állapotból többfelé is léphet. (A gráfjában egy csúcsból több azonos indexű él is kiléphet.) Akkor fogad el egy szót, ha van legalább egy olyan futása, ami elfogadja.

12. Úgy csinálunk nemdeterminisztikus N automatából egy determinisztikus M automatát, hogy az M minden állapota az N lehetséges állapotainak egy részhalmaza. Az a címkéjű él a $(q_1,..,q_k)$-ból a $\{q \mid \exists i q_i \xrightarrow{a} q\}$-ba megy. Azok a $(q_1,..,q_k)$ állapotok elfogadók, amikben van N-beli elfogadó állapot, a $START$ pedig a (q_{START}) állapot.

13. Nemdeterminisztikus automatával könnyű, mert meg tudjuk tippelni, hogy hol érnek véget az L-beli szavak. Ha adott L-et elfogadó determinisztikus M automata, akkor csinálunk belőle egy L^*-ot felismerő nemdeterminisztikus N automatát. Az N állapotai ugyanazok lesznek, csak lesz benne pár plusz él: az összes M-beli elfogadóállapotból vezessen él pontosan ugyanazokba a csúcsokba is, mint a $START$ állapotból.

14. Lehetséges állapotok halmazán csináljunk egy nemdeterminisztikus automatát. Ha **B** jön, akkor összes lehetséges módon lépünk, ha **A** jön, akkor pedig helyesen tippelve a „jó irányba" (ha van ilyen).

15. a) Középről bolyongjunk és számoljuk, hogy mikor melyik végére érünk.
b) Legyen $x \sim y$, ha minden z-re $xz \in L \Leftrightarrow yz \in L$. Ha végtelen sok ilyen osztály lenne, akkor lenne egy x, y pár, ami közel van abban az értelemben, hogy minden állapotban kb. ugyanakkora eséllyel van a randomizált automata miután elolvasta őket. De ekkor minden z-re is kb. ugyanakkorák lesznek az esélyek.

2. fejezet

Turing-gépek

2.1. Alapok

19. a) Lemásoljuk egy másik szalagra az inputot és ellenkező irányból megyünk végig rajtuk.
b) Végigmegyünk rajta és minden második karaktert kitörölve ritkítjuk $\log n$-szer. Ha bármikor eltér a paritása az a-knak és a b-knek, akkor nem egyenlőek, különben meg igen. (Hasonlóan n-et át is konvertálhatjuk unárisból binárisba $\log n$ lépésben.)
c) 3.3-as tetelt itt: https://doi.org/10.1016/0304-3975(85)90165-3.

20. Hennie-Stearns: Two-Tape Simulation of Multitape Turing Machines.

21. A legegyszerűbb megoldás talán egy olvasó + 4 másik szalagos Turing-gép, aminek 4 munkaszalagja unáris, páratlan fázisban $a_1 + a_2$-t írja b_1-re és b_2-re, párosban meg $b_1 + b_2$-t a_1-re és a_2-re.

23. c) Euklideszi algoritmus.

24. Euklideszi algoritmussal először $1/b$-t kiszámoljuk $bp + mq = 1$-et megoldva.

25. Ha ki tudnánk számolni, akkor bináris kereséssel meg tudnánk találni legkisebb k-t, amire $k!$ nem relatív prím m-hez, és ezzel meglenne m egy prímosztója is.
Nem ismert, hogy $\binom{a}{b} \mod m$-et milyen nehéz kiszámítani.
Forrás: http://rjlipton.wordpress.com/2009/02/23/factoring-and-factorials/

26. a) Ez a definícó.
b) Az input ügyes elkódolása után csak arra használjuk a szalagokat, hogy az olvasófej és az input első karaktere közötti távolságot elkódoljuk velük. Meggondolható, hogy egy így elkódolt számot tudunk bármely fix konstanssal maradékosan osztani, illetve két ilyen elkódolt számot összeadni. Ilyen műveletekkel beolvashatjuk az inputot átírva kettes számrendszerbeli számmá, majd szimulálhatjuk a TG működését.

27. Gráfokat megadhatjuk adjacenciamátrixszal vagy éllistával. Fontos, hogy ezek mind átalakíthatók egymásba polinom időben. Azt is jegyezzük meg, hogy az algoritmusunk futásidejét az *input* hosszában mérjük, nem pedig a csúcsszámban! Persze ha csak az érdekel minket, hogy polinomiális-e a futásidő, akkor mindegy melyiket vesszük, de a gyakorlatban azt preferáljuk, ha éllistával van adva a gráf és abban tudunk minél jobb futásidőt. Gráfos feladatoknál az n a csúcsszámot, az m az élszámot jelöli.

2.2. Rekurzív és rekurzívan felsorolható

30. a) Persze, pl. $L' = \emptyset$.
b) Igen, hiszen van olyan rekurzívan felsorolható, amiben benne van az összes 1-gyel kezdődő szó.

31. A bizonyítható állítások rekurzívan felsorolhatók, a cáfolható állítások co-rekurzívan felsorolhatók. Ha L rekurzívan felsorolható és nemigaz a Gödel tétel, akkor minden n-re megpróbáljuk bizonyítani meg cáfolni, hogy n eleme-e L, egyik előbb-utóbb sikerül, tehát L rekurzív.

32. Az egyik TG mindig álljon meg.

33. Kőnig-lemma.

34. a) Nem, mert az átlós bizonyítás relativizálódik.
Hasznos linkek: https://en.wikipedia.org/wiki/Malament%E2%80%93Hogarth_spacetime, https://www.scottaaronson.com/democritus/lec4.html, https://en.wikipedia.org/wiki/Turing_degree.

2.2.1. Dominók

Definíció: Dominó-probléma: Ki akarjuk parkettázni a síkot egy véges dominó-készlettel, mely tartalmaz egy *Start* elemet, amit muszáj felhasználnunk.

35. a) Mindig, sakktáblaszerűen forgatva egyetlen dominót.
b) Elég egy függőleges csíkot kirakni, amit pontosan akkor lehet, ha van egy legfeljebb n hosszú minta, amit tudunk ismételgetni. Ezt $n!$ eset végignézni.
c) Nem, mert feltehető, hogy bal-jobb oldalon levő jelek, meg fenti-lenti jelek csak egymáshoz passzolnak, így vagy egyik dominó sincs elforgatva, vagy mind el van.

36. I. (Padmini Mukkamala) Az eredeti, szélein megjelölt készlet minden darabját törjük négy részre, így minden szélen levő jel két darab sarkára kerül. Az eredeti sarkoknak megfelelő sarkok jele legyen egy új jel (ugyanaz minden dominóra), míg a dominók közepének megfelelő sarkok mind kapjanak egy olyan jelet, ami egyedi, csak ennek a dominónak a négy darabja kapja. Ha például D az eredeti probléma inputjának egy dominója, vágjuk szét négy kis dominóra úgy, hogy ha D oldalain a színek *bal, fent, jobb, lent* voltak, akkor pl a jobb-felső kis dominó színei bal-fenti sarokban *fent*, jobb-fenti sarokban *univerzális*, jobb-lenti sarokban *jobb*, bal-lenti sarokban D-szín.
II. Sakktáblaszerűen az eredeti inputot csináljuk, többi helyre meg univerzálisat rakunk.

37. Megoldás megtalálható az alábbi cikkekben: http://arxiv.org/abs/1003.2801 és http://arxiv.org/abs/cs/0409024.

3. fejezet

Tár

Definíció: STCONN: Adott n csúcsú (irányított) gráfban el kell dönteni, hogy s és t között van-e út.

38. Ugyanaz, mint a 26/b.

39. a) Számok kettes számrendszerben 1-től n-ig egymásután felírva.

40. Savitch tétel megoldásához hasonlóan mutassuk meg, hogy $\mathbf{DSPACE}(d \cdot \log n)$-ben el tudjuk dönteni, hogy van-e 2^d hosszú út.

41. Az s és a t komponensének is keressük meg a legészaknyugatibb pontját.

42. Az s és a t komponensének is keressük meg a legészaknyugatibb pontját.

43. a) Csak meg kell számolni.
b) Az kell, hogy minden nyitóhoz meg tudjuk találni a záró párját. Az kell, hogy ne legyen semely kettő keresztbe.

45. $(v_y - u_y)(v_x + v_y)$ jó, mert az x függvényre alkalmazhatjuk a Green-tételt. Kevésbé felvágósan mondhatjuk azt, hogy az (origó, u, v) háromszögek előjeles területét adjuk össze. Forrás: Raghunath Tewari, Vinodchandran Variyam: Green's Theorem and Isolation in Planar Graphs ContactAdd CommentRSS-Feed, http://eccc.uni-trier.de/report/2010/151/.

46. Immerman-Szelepcsényi tétel.

47. Immerman-Szelepcsényi tétel miatt elég BIPARTITE $\in$ **co-NL**, de arra tanú egy páratlan kör.

48. Próbáljuk végig az összes (legfeljebb) két csúcsot (u, v), hogy őket elhagyva szétesik-e a gráf. Ezt úgy ellenőrizhetjük, hogy minden s, t csúcspárra végigpróbáljuk, hogy egy komponensben lesznek-e $G \setminus \{u, v\}$-ben. Ezt meg tudjuk tenni, mivel STCONN $\in$ **co-NL**.

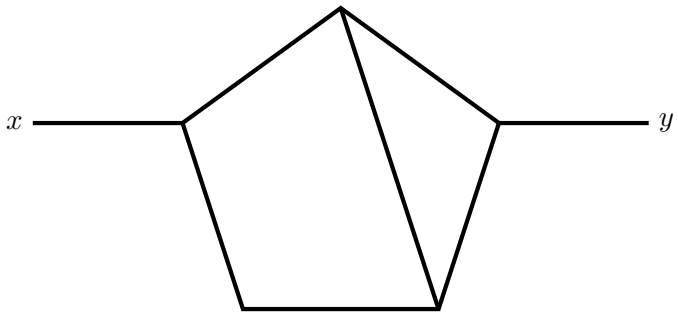
49. Ha **NL** = **NP**, akkor **co-NP** = **co-NL** = **NL** = **NP** (vagy $\mathbf{\Sigma_2} = \mathbf{NP}^{\mathbf{NL} \cap \mathbf{co\text{-}NL}} = \mathbf{NP}$), tehát **NP** = **PH** és **NL** = **PSPACE**, ami ellentmond a tárhierarchiának.

50. PAD-elni kell, de csak fejben, hiszen nincs annyi tár, hogy valóban felírhassuk, amit kéne.

3.1. Generalized Geography

51. Nem, fordított topologikus sorrendben megmondhatjuk melyik csúcsra nyer kezdő.
A konstrukcióban, amivel TQBF-et visszavezetjük Generalized Geography-ra a változós rész utáni élet törölve aciklikus gráfot kapunk, tehát a válasz 1.

52. a) Ha van v-t elkerülő maximális párosítás, annak élein lépkedhet második játékos. Kezdő csak párosításbeli csúcsra léphet, különben lenne javító út. Másik irányra legegyszerűbb, ha felveszünk egy új v' csúcsot, ami csak v-vel szomszédos.
b) A gadgetet be kell rakni minden irányított él helyére. Forrás: Fraenkel, Scheinerman and Ullman: Undirected Edge Geography, Theoretical Computer Science, 1993. https://doi.org/10.1016/0304-3975(93)90026-P



53. a) **NPSPACE**-beli és Savitch tétel.
b) Az általánosított állapottere egy **PSPACE**-es TG-nek pont egy ilyen gráfot ad.

54. Miltzow: Tron, a combinatorial Game on abstract Graphs https://arxiv.org/abs/1110.3211 cikkben 13. oldaltól a), 15. oldaltól b). (De egyébként Nagy Kartal csinált egyszerűbb megoldást rá.)

4. fejezet

NP

55. a) Végigpróbálgatjuk az összes 3-színezést, az aktuális színezés leírása mindig csak $O(n)$ tár, ellenőrzéshez csak végig kell menni az éleken.
b) Tanú egy 3-színezés.
c) Végigmegyünk a nem-éleken és sorban hozzáadjuk a gráfhoz, amelyik nem rontja el a 3-színezhetőséget. Ezt onnan tudjuk, hogy minden él hozzáadása után ellenőrizzük, hogy a gráf 3-színezhető maradt-e. Így végül egy 3-osztályú teljes gráfot kapunk, ezek lesznek a színosztályok.
Fontos típushiba, hogy a 3-színezhetőség tesztelő algoritmusnak nem adhatunk olyan gráfot, aminek néhány csúcsa már ki van színezve!

56. a) Tanú egy teljes párosítás.
b) Egyesével kitöröljük az éleket és megnézzük, hogy van-e még teljes párosítás. Ha nincs, akkor visszarakjuk az élet. Így végül egy teljes párosítás élei maradnak.
c) Opkutból tanult javítóutas módszer ilyen.
d) Edmonds-tétel: https://en.wikipedia.org/wiki/Blossom_algorithm.

57. a) Nem-prímségre tanú egy osztó, nem síkbeliségre egy felosztott K_5 vagy $K_{3,3}$ (Kuratowski-tétel miatt).
b) Prímekre 4.3.5. Tétel Lovász könyvben. Síkgráfokra típushiba, hogy a Fáry tétel szerint van egyenes szakaszokkal is síkbarajzolás és elég a csúcsok koordinátáit megadni, mert ez nem feltétlenül írható le polinom helyen. Egy nem-triviális tétel, hogy minden gráf lerajzolható egy $2n \times n$ méretű rácsra. Ehelyett egy egyszerű megoldás, hogy topológiailag megadjuk, hogy mik a csúcsokbol kimenő élek sorrendjei, melyik él melyik lapon van és a lapokon mi a sorrend. Részletek meggondolandók.
c) Prímekre ez nagyon sokáig nyitott volt, lásd https://en.wikipedia.org/wiki/AKS_primality_test. Síkgráfokra itt találhatók különböző megoldások: https://cstheory.stackexchange.com/questions/24962/what-is-simplest-polynomial-algorithm-for-planarity.

58. a) Az időhierarchia tétel miatty minden rekurzív f-re van olyan rekurzív nyelv, ami nincs $\mathbf{DTIME}(f(n))$-ben. A bizonyítás részletein végigmenve ellenőrizhető, hogy $f(n) = n^2$-re a kapott nyelv $\mathbf{DTIME}(n^{10})$-beli.
b) Legegyszerűbb bizonyítás Zák: http://blog.computationalcomplexity.org/2011/04/new-proof-of-nondeterministic-time.html.

59. a) Vegyük egy felsorolását az összes Turing-gépnek, ami mindig megáll: $T_1, T_2, \ldots$ és legyen $f(n) = \max_{i \le n}$ a T_i lépésszáma legfeljebb n hosszú inputokon. Ha L rekurzív, akkor van egy i, hogy $L = L(T_i)$, tehát $L \in \mathbf{DTIME}(f(n))$. Lásd még *busy beaver* függvény.
b) Hézag tétel: https://en.wikipedia.org/wiki/Gap_theorem.

4.1. NP-teljesség

60. **NP**-beliség nyilvánvaló. Legegyszerűbb megoldás **NP**-nehézségre észrevenni, hogy ez pont ugyanaz, mint a halmazrendszer kétszínezhetősége, ha egyik változó sincs negálva. Tehát mivel ez az **NP**-teljes feladat a speciális esete, ezért ez is **NP**-nehéz. De vissza lehetett vezetni a SAT-ra is többféleképpen; egyik opció volt, hogy minden klózba betesszük ugyanazt az új változót, egy másik meg, hogy minden klózból veszünk mégegy példányt, ahol minden változót negálunk.

61. **NP**-beliség nyilvánvaló. Legegyszerűbb megoldás **NP**-nehézségre felhasználni, hogy INDEPENDENT, azaz azt eldönteni, hogy van-e k méretű független halmaz egy gráfban, **NP**-teljes. Típushiba, hogy $k = n$-et választjuk visszavezetésnél, ezt azonban nem tehetjük meg, mert k az input része volt. Helyes megoldás, hogy ha (G, k) volt az input az INDEPENDENT-nek, akkor abból úgy készítünk egy G' gráfot, hogy G-nek vesszük a komplementerét (hogy a független halmazokból klikkek legyenek) és még hozzáveszünk $2k - n$ izolált pontot, ha $2k \geq n$ (így $k = (n+2k-n)/2$, azaz k pont a G' csúcsszámának a fele), illetve $n-2k$ pontot, akik mindenkivel össze vannak kötve (így $k + n - 2k = (n + n - 2k)/2$, azaz $n - k$ pont a G' csúcsszámának a fele). Mindkét esetben akkor és csak akkor van G'-ben $|V(G')|/2$ méretű klikk, ha G-ben volt k független pont.

63. **NP**-beliség nyilvánvaló mindegyik esetben.
a) A Hamilton-utat vezetjük vissza. Vegyünk hozzá az input G gráfhoz egy új csúcsot, akit mindenkivel összekötünk. Így az új gráfban akkor és csak akkor van Hamilton-kör, ha G-ben van Hamilton-út. Típushiba volt, hogy ha Hamilton-kör van, akkor Hamilton-út is, tehát ez nehezebb. Gondoljátok meg, hogy ezzel a logikával az is **NP**-teljes lenne, hogy egy teljes gráf-e az input. Fontos, hogy a visszavezetésnél (lásd lap tetején), akkor és csak akkor legyen az egyik input megoldás az egyik feladatra, ha a másik input is az a másik feladatra.
b) A Hamilton-kört vezetjük vissza. Vegyünk hozzá az input G gráfhoz $|V(G)|$ izolált pontot, vagy akár vegyük G-t két példányban.
c) A Hamilton-kört vezetjük vissza. Az input G gráf éleire írjunk 1-et, a többi élre pedig 2-t és válasszuk K-t $|V(G)|$-nek. Így csak akokr van n összhosszú Hamilton-kör, ha G-ben volt Hamilton-kör. Típushiba volt, hogy nem olvastátok el a zárójelben szereplő feltételeket...

64. a) A SAT egy inputját könnyű átírni egyenlőtlenségrendszerré, de lehet sok más dolgot is.
b) Meg néhány változó hozz'aadásával elérhető, hogy $Ax = b, x \geq 0$ feladatot kelljen megoldani. Innen Cramer-szabály meg egyéb dolgok kellenek, lásd: http://lara.epfl.ch/web2010/_media/papadimitriou81complexityintegerprogramming.pdf.

65. Következik abból, hogy bármely nemdeterminisztikus TG egy n hosszú inputon „szimulálható" egy dominó készlettel, amit egy determinisztikus TG elő tud állítani ismerve az inputot. Garantálható, hogy minden dominó csak egy helyre passzolhasson, ha mind megindexeljuk (i, j)-vel. Ezután tudjuk egy TG futását szimulálni velük.

67. a) A legfeljebb egy elemű klózokat irtsuk ki, amíg minden klóz kételemű nem lesz. Minden kételemű klóz leírható két következtetésként, pl. $(x \vee y) \leftrightarrow (\bar{x} \to y) \leftrightarrow (\bar{y} \to x)$. Ez megad egy irányított gráfot a literálokon, aminek kétszer annyi éle van, mint klóz. Ha van olyan változó, hogy a negált és a sima literálja egy erősen összefüggő komponensben van, akkor nem megoldható a formula. Vegyük észre, hogy mivel a gráfot szimmetrikus módon definiáltuk, ez azzal ekvivalens, hogy nincs egyikből sem irányított út a másikba. Hasonló okokból ekkor az is teljesül, hogy nem lehet x-ből irányított út y-ba és $\bar{y}$-ba is. Viszont ilyenkor bármely változónak választhatunk tetszőlegesen értéket, majd kitörölhetjük az így értéket kapott változókat, és ezt ismételgethetjük.

b) Nagy klózokat szétvágjuk kisebbekre, pl $(x_1 \vee x_2 \vee x_3 \vee x_4) \leftrightarrow (x_1 \vee x_2 \vee y) \wedge (\bar{y} \vee x_3 \vee x_4)$, ahol az y egy új változó.
c) A legfeljebb egyszer előforduló változókat irtsuk ki, illetve azokat is, amiknek mindkét előfordulása negált vagy mindkettő sima. Tekintsük azt az irányítatlan gráfot, aminek a csúcsai a klózok és akkor van két klóz összekötve, ha van közös változójuk. Így a feladat ekvivalens azzal, hogy egy adott gráf élei megirányíthatóak-e úgy, hogy minden csúcsba vezessen él. Ez visszavezethető egy párosítás feladatra páros gráfban, ahol az egyik osztályban minden csúcs foka 2.
d) Szokott lenni előadáson; Lovászban 4.4.8. Tétel.

68. A SubsetSum feladatot vezetjük rá vissza, ahol azt kell megmondanunk, hogy input a_i számok közül van-e pár, amiknek az összege pont b. Az input számaink legyenek ugyanazok, valamint mégegy szám, a $|2b - \sum_i a_i|$.

69. Visszavezetés a 3-SAT-ról: Minden változónak feleljen meg k piros, k kék és $2k$ zöld pont, mely utóbbiak legyenek szétosztva két k elemű részre: Z_x-re és $Z_{\bar{x}}$-re. Adjunk meg ezen a $4k$ ponton halmazokat úgy, hogy a piros és kék pontok egy teljes párosítása esetén vagy pontosan Z_x vagy pontosan $Z_{\bar{x}}$ van fedve. Minden klóznak feleljen meg egy piros és egy kék pont, amik ugyanabban a három halmazban vannak benne, ahol a harmadik pontok a klózban levő literáloknak felelnek meg. Végül vegyünk fel még egy csomó piros és kék pontot, hogy ne legyen baj a kimaradó zöldekkel.

70. Lichteinstein tétele.

71. a) L. Stockmeyer, Planar 3-colorability is polynomial complete. ACM SIGACT News 5 (1973), 19–25.
b) Tetszőleges szomszédos csúcsokat kiszínezünk, majd veszünk egy élet, aminek mindkét vége ki van színezve, de a harmadik nincs, és azt is kiszínezzük. De van egy egyszerű karakterizáció is: Akkor lesz 3-színezhető egy háromszögelt síkgráf, ha nincs páratlan fokú csúcsa. Ezt úgy lehet bizonyítani, hogy ha minden fok páros, akkor a legkisebb fok 2 vagy 4 és innen indukció. A 2 fokú eset trivi. Ha v 4 fokú, akkor van két szomszédja, pl. u és w, amik nem szomszédosak egymással. Összehúzzuk az u, v, w csúcsokat egy csúccsá, indukcióval színezzük ezt a gráfot. A fokszám-feltétel miatt v másik két eredeti szomszédja ugyanazt a színt fogja kapni, úgyhogy a széthúzásnál u és w őrizze meg a színét, míg v kaphatja a harmadik színt.

72. Ha G a 3col inputja, akkor csináljunk belőle G'-t úgy, hogy hozzáveszünk $|V(G)|$ csúcsot, akik mindenkivel össze vannak kötve.

Definíció: Egy halmazrendszer/hipergráf (csúcsainak) k-színezését többféleképpen definiálhatjuk.
a) (gyenge) k-színezés: minden hiperélben szerepeljen két különböző szín.
b) szivárvány k-színezés: minden hiperélben minden szín különböző legyen.
c) konfliktusmentes k-színezés: minden e hiperélben szerepeljen egy csúcs, aminek a színe az összes többi e-beli színtől különböző.
d) polikromatikus k-színezés: minden hiperélben szerepeljen mind a k szín.

73. a-b-c) Ezek sima gráfokra pont ugyanazok, mint a k-színezhetőség, k-col.
d) Ha G a k-col inputja, akkor legyen $\mathcal{H}$ az hipergráf, hogy G minden éléhez hozzáadunk $k-2$ új csúcsot.

74. b-d) Ezekre jó az előző feladatbeli visszavezetés.
a) Ha G a k-col inputja, akkor legyenek a $\mathcal{H}$ hipergráf csúcsai $V(G) \dot\cup W$, ahol W $k(k-1)$ db új csúcs, $\mathcal{H}$ élei pedig legyenek $\binom{W}{k} \cup \{E(G) + \binom{W}{k-2}\}$, azaz W k-elemű részhalmazai, valamint

G élei kiegészítve $k-2$ darab W-beli csúccsal, az összes lehetséges módon.
c) Ugyanaz, mint a)-ra, csak élhalmaz kisebb, csak $E(G) + \binom{W}{k-2}$-ből áll.

75. http://cstheory.stackexchange.com/questions/14124/is-there-a-natural-problem-on-the-naturals-that-is-np-complete/14147#14147.

76. b) Walter Kern, Daniël Paulusma, The new FIFA rules are hard: complexity aspects of sports competitions, Discrete Applied Mathematics 108 (3), 2001, 317–323, https://doi.org/10.1016/S0166-218X(00)00241-9.

5. fejezet

PAD

Definíció: $PAD(L) = \{x0^{n^2} : x \in L\}$ (azaz x mögé írunk még egy csomó nullást és persze $n = |x|$).

78. PAD-eljunk. Indirekt tegyük fel, hogy van L **NEXP** $\setminus$ **EXP**-ben. Ekkor ez benne van valamilyen k-ra **NTIME**(2^{n^k})-ben. PAD-eljük meg 2^{n^k} darab nullával. Ekkor $PAD(L)$ már **NP**-beli, tehát **P**-beli is, tehát $L \in$ **EXP**, ellentmondás.

80. Legyen $L' = \{(n, k, l, i, b) \mid$ *van* k *darab* n *hosszú* L*-beli szó úgy, hogy az* l*. szó* i*. bitje* $b\}$. Ha $L \in$ **NP**, akkor $L' \in$ **NE**, mert nemdeterminisztikusan felsorolhatjuk a k darab n hosszú szót, $L \in$ **NP** segítségével mindre tudunk tanút is találni. A feltevés szerint ekkor $L' \in$ **E** is teljesül. Most ezt felhasználva megmutatjuk, hogy $L \in$ **P**. Bináris kereséssel megtaláljuk azt a legnagyobb k-t, amire $(n, k, 1, 1, 0) \in L'$ vagy $(n, k, 1, 1, 1) \in L'$; ez lesz az n hosszú szavak száma L-ben. Ezután minden $l \leq k$, $i \leq n$, $b = 0/1$ hármasra megnézzük, hogy (n, k, l, i, b) benne van-e L'-ben. Így mind a k L-beli szót meghatároztuk.

6. fejezet

Polinomiális hierarchia és orákulumok

Definíció: $\exists^p L := \left\{ x \in \{0,1\}^* \;\middle|\; \left(\exists w \in \{0,1\}^{\leq p(|x|)}\right) \langle x, w\rangle \in L \right\}$,
és hasonlóan $\forall^p L := \left\{ x \in \{0,1\}^* \;\middle|\; \left(\forall w \in \{0,1\}^{\leq p(|x|)}\right) \langle x, w\rangle \in L \right\}$.
Ezekkel a jelölésekkel $\mathbf{NP} = \exists \cdot \mathbf{P}$ és $\mathbf{co\text{-}NP} = \forall \cdot \mathbf{P}$. Ezen felbuzdulva $\mathbf{\Sigma_k} := \exists \cdot \forall \cdot \exists \ldots \mathbf{P}$ és $\mathbf{\Pi_k} := \forall \cdot \exists \cdot \forall \ldots \mathbf{P}$, ahol összesen k db kvantor áll a $\mathbf{P}$ előtt. Tehát ezekkel a jelölésekkel $\mathbf{NP} = \mathbf{\Sigma_1}$ és $\mathbf{co\text{-}NP} = \mathbf{\Pi_1}$. A polinomiális hierarchia $\mathbf{PH} := \cup \mathbf{\Sigma_k}$.

85. Akkor trivin $\mathbf{\Sigma_2}$-ban lenne, de az is igaz, hogy $\mathbf{NP}$-beli, mert azt könnyű ellenőrizni, hogy egy KNF-et minden értékadás igazzá tesz-e.

86. $\mathbf{\Pi_2}$-ban van trivi, de igazából $\mathbf{\Sigma_1}$-ben is, mert az összes lehetőséget leellenőrizhetjük.

87. Karp-Lipton tétel.
Ötlet: Először mutassuk meg, hogy ekkor olyan hálózat is van, ami minden kielégíthető SAT formulához ad egy helyes kiértékelést. Aztán mutassuk meg, hogy összeomlana a polinomiális hierarchia.

Ha $\mathbf{NP} \subset \mathbf{P/poly}$, akkor van egy $p(n)$ poly hosszú bitsorozat és T TG, hogy $T(\Psi, p(n)) = 1$ akkor és csak akkor ha $\Psi \in$ SAT. Sőt, olyan T is van, hogy $T(\Psi, p(n)) = 1$ esetén még ki is ad egy x-et, amire $\Psi(x)$ igaz. Most azt fogjuk megmutatni, hogy $\mathbf{\Pi_2} \subset \mathbf{\Sigma_2}$, ebből következik az állítás. Legyen $L \in \mathbf{\Pi_2}$. Ekkor $\Psi \in L$ akkor és csak akkor, ha minden z-re $\Psi_z \in$ SAT, valami $\mathbf{P}$-beli Ψ_z-re. De ez ekvivalens azzal, hogy $\exists p(n) \forall z T(\Psi_z, p(n)) = 1$ és $\Psi_z(x_z)$ igaz, ahol az x_z-t is T adja ki.

88. Meyer tétele (de Karp-Lipton cikkben van).
Előzőhöz hasonlóan. Lenne egy hálózat, ami egy $\mathbf{EXP}$-es számítás bármelyik helyén, bármikor felvett értéket meg tudná mondani. Egy ilyennek a helyességét ellenőrizni is tudjuk $\mathbf{co\text{-}NP}$-ben.

89. Sőt (Borbényi Márton), $\mathbf{P}$-beli, mert csak ennyi opció van konstans körös játékban és mindet végignézhetjük. Ha a tábla exp méretű lenne vagy körönként n-et raknának, akkor csak $\mathbf{\Sigma_k}$-t tudnánk bizonyítani.

Definíció: Az *orákulumos* Turing-gép egy többszalagos $M^?$ Turing-gép, melynek egyik szalagja a kitüntetett kérdező-szalag. Az orákulum egy tetszőleges A nyelv lehet. $M^?$-nek van egy speciális $q_?$ állapota. Ha ebbe kerül, akkor az orákulum válaszol neki, hogy az éppen akkor a speciális szalagján levő szó benne van-e az A nyelvben.

Az x bemenethez tartozó kimenetet $M^A(x)$-szel jelöljük. Ha $\mathbf{XP}$ egy olyan nyelv, ami Turing-gépek egy speciális családjával van definiálva, akkor $\mathbf{XP}^A$ jelöli azokat az L nyelveket, amihez van ebből a családból Turing-gép, amihez A orákulumot adva L-et ismeri fel. Ha valamely bonyolultsági osztályhoz van teljes nyelv, akkor gyakran a bonyolultsági osztályt

írjuk a kitevőbe, pl. $\mathbf{P}^{\mathbf{NP}}$ azt jelenti, hogy $\mathbf{P}^{\mathrm{SAT}}$. Ha nincs teljes nyelv, akkor is írhatunk bonyolultsági osztályt a kitevőbe, ekkor az új osztály az összes lehetséges nyelv uniója, pl. $\mathbf{P}^{\mathbf{BPP}} = \cup\{\mathbf{P}^L \mid L \in \mathbf{BPP}\}$.

90. Futassuk magunk az A-t eldöntő TG-et ahelyett, hogy kérdeznénk tőle.

91. Minden **NP**-beli nyelv visszavezethető SAT-ra, és aztán mint előző.

92. **EXP**.

94. $\mathbf{NP} \subset \mathbf{P}^{\mathbf{NP}}$ nyilvánvaló és $\mathbf{P}^{\mathbf{NP}}$ zárt komplementerre.

95. Bármit kérdeznénk, megtippelhetjük rá a választ és találhatunk rá tanút magunk.
Forrás: http://blog.computationalcomplexity.org/2017/03/np-in-zpp-implies-ph-in-zpp.html alapján.

96. $\mathbf{NP}^{\mathbf{NP}} \supset \mathbf{\Sigma_2}$ definíció szerint. $\mathbf{NP}^{\mathbf{NP}} \subset \mathbf{\Sigma_2}$-hez tippeljük meg összes kérdésre válaszokat, amire van tanú, tippeljük meg, amire nincs, azokra meg ez egyetlen $\mathbf{\Pi_1}$-beli mondat, hogy egyikhez sincs.

97. Legyen orákulumos TG futásideje n^k és hibázzon $< 1/8$ eséllyel. Orákulum nyelvhez minden n-re és k-re van **BPP** alg, ami $< 1/(8n^k)$ eséllyel hibázik. Ez ismételgetéssel kapható bármelyik **BPP**-sből, tehát tudjuk is szimulálni $poly(n^k)$ időben. Ha mindig ezt futtatjuk kérdezés helyett, akkor annak az esélye, hogy valaha is hibázunk kérdeześ szimulálásakor $< 1/8$, tehát az összhiba $< 1/4$.

98. I. **E** vagy **EXP** jó időhierarchia miatt.
II. (Csernák Tamás) Rekurzív felsorolhatók is jók, mert nem zártak komplementerre.

99. I. Ha A **PSPACE**-teljes, ez Savitch tétel.
II. (Ágoston Péter) $A = \mathbf{PH}$, mert ha nemdet k. szintről kérdez, det kérdezhet a $(k+1)$-edikről.

100. Nyelv legyen, hogy van-e A-ban az inputtal megegyező hosszúságú string. Meg lehet csinálni átlós módszerrel, hogy minden **P**-belitől különbözzön.

101. Forrás: Buhrman, Fortnow, Santhanam: Unconditional Lower Bounds Against Advice https://doi.org/10.1007/978-3-642-02927-1_18.

7. fejezet

Véletlen

102. $AB = C$ akkor és csak akkor, ha minden x-re $A(Bx) = Cx$, tehát ez valójában egy n változós lineáris polinomazonosság tesztelés. Konkrétan egy véletlen $x \in \{0,1\}^n$-re teszteljük, hogy $A(Bx) = Cx$ igaz-e. Ha $AB = C$, ez mindig teljesül. Különben legyen $(AB)_{ij} \neq C_{ij}$. Ekkor csak egy x_j értékre lesz $(A(Bx)_i = (Cx)_i$. Tehát legalább $1/2$ eséllyel $A(Bx) \neq Cx$. Ez egy **co-RP**-beli algoritmus, tehát **BPP**-beli is.

103. A megállási problémát vezetjük vissza. Input T-ből készítünk egy T'-t, ami minden x inputra T-t szimulálja a üres inputon $|x|$ lépésig. Ha T nem áll le ezalatt, akkor T' elutasít. Ha T leáll, akkor T' fogadja el $|x|$-et $1/4$ eséllyel.

104. A b) az a) speciális esete.
Ha van egy a) típusú T gépünk, akkor abból úgy készíthetünk egy b) típusú T'-t, hogy ahányszor randomizálni kell, mindig leírjuk egy plusz szalagra, hogy éppen T melyik állapotában kell ezt megtennünk, majd T' egyetlen randomizált állapotával addig sorsolunk, amíg elég nagy pontossággal meg nem közelítjük a racionális számokat. A közelítés szükséges pontosságát, azaz azt, hogy a racionális számot hány bináris „tizedes jegy" pontosságig számoljuk ki, az input hossza határozza meg. Ha T n^c időben fut, akkor elég $1/(2n^c)$ pontossággal közelíteni, így az ebből adódó hiba legfeljebb $(1 - 1/(2n^c))^{n^c} < 1/e^2$, tehát az összhiba legfeljebb $1/3 + 1/e^2 < 1/2$. Ez ismétléssel levihető $1/3$ alá a szokásos módon.
A b) és c) ekvivalenciája nyilvánvaló.
A d) speciális esete a b)-c).
Ha van egy b) típusú T gépünk, akkor abból úgy készíthetünk egy d) típusút, hogy ahányszor a randomizáló állapotba lépnénk, dobjuk fel a pénzt kétszer. Ha fej-íras, lépjünk tovább az egyik állapotba, ha íras-fej, akkor a másikba, míg ha két azonos értéket kapunk, akkor ismételjük ezt meg újra. (Ha ezt olyan sokszor kéne megismételni, hogy túlmennénk a megengedett futásidőn, akkor álljunk le, ennek úgyis kicsi az esélye.)

105. Felismerhetünk egy olyan L nyelvet, ami csak x hosszától függ és annyira ritka, hogy csak 2^{2^t} alakú hosszúságú szavak lehetnek L-ben. Azt, hogy n benne van-e L-ben, ne lehessen **EXP**-ben eldönteni, de 2^{2^n}-ben már igen. Legyen egy állapot, amiből $\sum_{n \in L} 1/n$ eséllyel lépünk egy másikba. Így adott inputra kisebbekre meg tudjuk nézni, hogy benne van-e, így n ismétlés után tudjuk, hogy kb. hány sikert várunk, nagyobbak nem befolyásolják.

106. a) Válasszunk egy véletlen értékadást. Ha jó, akkor fogadjunk el. Ha nem jó, akkor is fogadjunk el $1/2-\varepsilon$ eséllyel és utasítsunk el $1/2+\varepsilon$ eséllyel. Így ha $\Psi \notin$ SAT, akkor $> 1/2$ eséllyel elutasítjuk, míg ha $\Psi \in$ SAT, akkor legalább $1/2^n + (1 - 1/2^n)(1/2 - \varepsilon) > 1/2 + 1/2^{n+1} - \varepsilon$ eséllyel elfogadjuk, tehát ε-t választhatjuk pl. $1/2^{n+2}$-nek.

b) Ezek a Turing-gépek vaóban csak rekuzív nyelvet ismerhetnek fel, hiszen adott inputra addig számolgathatjuk, hogy mekkora eséllyel fogadnak/utasítanak el, amíg valamelyik $> 1/2$ nem lesz. (Tehát a várható futásidőre nem is kell felső korlát.)
A rekurzív nyelvek pedig valóban ebbe a családba tartoznak, hiszen elég szimulálni egy, a nyelvet eldöntő Turing-gépet azzal a módosítással, hogy minden lépés után 50% eséllyel leállunk és véletlenszerűen elfogadunk/elutasítunk. (Tehát a várható futásidő igazából lehet konstans is.)

107. Vegyük q-nak az i. gyökét $i = 1, \ldots, \log q$ értékekre és teszteljük mindről, hogy prím-e.

108. Valiant-Vazirani tétel: https://en.wikipedia.org/wiki/Valiant%E2%80%93Vazirani_theorem.

109. Forrás: Charles H. Bennett, John Gill: Relative to a Random Oracle... https://doi:10.1137/0210008
Ha $L \in \mathbf{BPP}$, akkor olyan Random TG is van, ami az összes inputra összesen $< 1\%$-ot hibázik. Ezzel meg tudjuk mutatni, hogy a véletlen nyelvek 99%-a tartalmazza **BPP**-t, tehát egy valószínűséggel $\mathbf{BPP} \subset \mathbf{P}^L$. Viszont egy valószínűséggel $L \notin \mathbf{BPP}$, tehát egy valószínűséggel $\mathbf{BPP} \neq \mathbf{P}^L$.

110. Az orákulum nyelvhez van **BPP** algoritmus, ami $< 1/2^n$ eséllyel hibázik. Ha ezt futtatjuk rá, akkor annak az esélye, hogy valaha is hibázunk kérdezés szimulálásakor kicsi.

111. $\mathbf{co\text{-}RP} \subset \mathbf{RP}^{\mathbf{RP}}$, szóval következne belőle $\mathbf{co\text{-}RP} = \mathbf{RP}$, jelenleg nem ismert.

112. Tippelünk egy tanút, ha jó elfogadunk. Ha nem, akkor is elfogadunk $1/2 - \varepsilon$ eséllyel, valamint elutasítunk $1/2 + \varepsilon$ eséllyel.
Lásd **PP**: https://en.wikipedia.org/wiki/PP_(complexity)

8. fejezet

Kolmogorov-bonyolultság

Definíció: Egy U Turing-gép univerzális, ha minden T Turing-gépre $U([T], x) = T(x)$, ahol a $[T]$ a T leírását jelenti. Az x Kolmogorov-bonyolultsága U szerint a legrövidebb program hossza, amire kiírja x-et, azaz $K_U(x) = \min\{|p| \mid U(p) = x\}$. Mostantól azt is feltesszük, hogy az ábécé kételemű. (Az első sorban a vessző azért nem csalás, mert $[T]$-t írhatjuk kiterjesztett binárisban.)

114. Ugyanannyi, ezt hívhatjuk akár az n szám Kolmogorov-bonyolultságának.

115. Nem, pl. ha $y = 2^{2^n}$ darab 1-es, akkor annak lesz bonyolult prefixe.

116. Ötlet: Hamming-kód legyen az y. De ebből csak akkor jönne ki simán $n - \log n$, ha n is meg lenne adva. Egyébkent le lehet írni azt, hogy y hányadik Hamming-kódbeli elem az összes (különböző hosszúságú) Hamming-kód közül, ehhez $\sum_i 2^i/i$ kell. Ez indukcióval $O(2^n/n)$, pont jó.

117. Ott a csalás, hogy nem mondtuk meg, hogy hol végződik a k leírása. Ha rendesen számolunk, akkor abból kijön a prímszámtétel egy gyenge verziója.

119. Nincs, mert a szó hossza is információ, ezzel pont $\log n$-et lehet nyerni, ha szerencsénk van. Ezt mégegyszer eljátszva nyerhető még $\log\log n$ stb.
Forrás: Li-Vitányi 2.5.1.

120. Két távoli csúcs leírása $2\log n$, belőlük kimenő élek további $n\log_2 3$ a $2n$ helyett.

121. Tranzitív rész leírása sorrenddel $3\log n \cdot \log n$, míg köztük $\binom{3\log n}{2} = 4.5\log^2 n$ él megy.

122. Ötlet: Induljunk ki egy tetszőleges értékadásból és amíg van, javítsunk meg egy rossz klózt és az ezáltal rosszá tett szomszédokat. Egy javítólépés, hogy egy klóz n változójának új értéket adunk egy Kolmogorov-véletlen x sorozatból. Ha egy javításssorozat túl sokáig tartana, akkor x nem lenne Kolmogorov-véletlen.
Forrás: Robin Moser https://blog.computationalcomplexity.org/2009/06/Kolmogorov-complexity-proof-of-lov.html

9. fejezet

RAM-gépek

9.1. PRAM

Definíció: Előadáson voltak a PRAM különböző modelljei, az EREW, CREW, CRCW és a P-CRCW, mely rövidítések önmagukért beszélnek, kivéve a két utolsó közti különbséget; a CRCW modellben minden adott mezőbe írónak ugyanazt kell írnia, míg a P-CRCW modellben lehet mást és a legkisebb sorszámú processzor írasa lesz sikeres.

129. a) $\mathbf{AC^0}$ hálózat szimulálható CRCW-vel.
b) $\mathbf{NC^1}$ hálózat szimulálható EREW-vel.
c) (Machó Bónis és Nagy Kartal) Jel: C: van carry, mint 1+1, P: propagates, mint 1+0, 0: semmi, mint 0+0. Két egymásutáni átmenete: C*->C, 0*->0, PX->X. Bináris fában kiszámoljuk, hogy 2^k méretű blokkokban hol van carry. majd a gyökértől visszafelé mindenki megmondja a gyerekeinek, hogy van-e carry.

130. a) Következik abból, hogy 2 számot $O(1)$ lépésben össze tudunk adni n^2 procival.
b) 3 számbol csinálunk 2-t $\log_{3/2} n$-szer.

10. fejezet

Hálózatok

Definíció: Azon Boole-hálózatok a nem-unifom $\mathbf{AC^i}$ hálózatok, amik polinomiális méretűek és $O(\log^i n)$ mélységűek. Az uniform esetben kell egy $O(\log n)$ tárú Turing-gép is, ami elő tudja állítani a család n. tagját, ha a bemenete n). Azok a $\mathbf{NC^i}$ hálózatok, amik $\mathbf{AC^i}$-k és minden kapu befoka legfeljebb 2. Az összes uniója a *Nick's Class*, jele (nem-uniform) **NC**. Ezek a jól párhuzamosítható algoritmusok.
Az előadáson volt/lesz, hogy Parity $\notin \mathbf{AC^0}$. Ha egy nem $\mathbf{AC^0}$-beli függvényt vissza tudunk vezetni valamire, akkor az sem lehet $\mathbf{AC^0}$-ban. Ennek felhasználásával (vagy nélkül) mutassuk meg az alábbi függvényekről, hogy nincsenek $\mathbf{AC^0}$-ban, azaz az outputjuk valamelyik bitje nincs $\mathbf{AC^0}$-ban.

131. A méret polinomiális megnövelése árán elérhető, hogy minden kapu befoka kettő legyen. Hogyan változik a mélység? Legfeljebb a fok log-szorosára, ez legfeljebb a méret log-szorosa.

133. Akkor jön helyiértéktúlcsordulás (angolul carry) a k. helyre, ha van $m > m$, hogy $x_m = y_m = 1$ és minden $k < j < m$-re $x_j + y_j \geq 1$. Ezt hálózattal meg lehet csinálni.

134. Visszavezethetjük az összeadásra úgy, hogy $x - y = x + (2^n - 1 - y) + 1 - 2^n$.

137. Következik abból, hogy a KW-je minden szimmetrikus függvénynek $\log n$; lásd 168.

138. Majority-ből kijönne bármilyen Threshold gate ($\sum_i x_i \geq k$ típusú), ezekből páratlanok összeÉSeléséből megvan Parity, ami viszont $\notin \mathbf{AC^0}$.

139. Ha sok nullást közberakva x-be, ezt megszorozzuk 10..010..01..-val, középen megjelenik Parity (meg kicsit arrébb Majority is).

140. $(x+y)^2 - x^2 - y^2$-ben ott lesz xy.

141. $(x+1)^3 - x^3$-ből megvan $3x^2$, amiből $3xy$, amiből 300030003-mal való szorzás ugyanúgy nehéz, mint 1001..-val.
Végét befejezhetjük úgy is (Schwartz Tamás, Borbényi Márton), hogy $3x$-et szorozzuk az 1001../3-mal, amit bele tudunk drótozni a hálózatba.

142. Kettőhatványokra trivi az. Többi számra meg Parity-hez hasonlót (pl. $\sum x_i \mod 3$-at) vissza lehet rá vezetni összeadásból (pl. $\sum x_i \mod 3 = x - \lfloor \frac{x}{3} \rfloor - \lfloor \frac{x}{3} \rfloor - \lfloor \frac{x}{3} \rfloor$).

143. Savitch tételhez hasonlóan felezgetünk.

144. Visszavezetjük a Parity-t. Feleltessük meg a változókat egy gráf n csúcsának, és legyen $x_i x_j$ él, ha $x_i = x_j$.

145. (J. Bird, lásd Furst–Saxe–Sipser: Parity, circuits, and the polynomial–time hierarchy) Visszavezetjük a PARITY-t. Tegyük fel, hogy $x_1 = x_n = 1$ és feleltessük meg a változókat egy gráf n csúcsának úgy, hogy $x_1 = s$ és $x_n = t$.

146. I. Az előző feladat mintájára visszavezetjük a PARITY-t. Feleltessük meg a változókat egy gráf n csúcsának, és legyen x_ix_j él $i < j$-re, ha $x_i = x_j = 1$ és minden $i < k < j$-re $x_k = 0$ VAGY minden $k < i$-re és $k > j$-re $x_k = 0$. Ez tehát vagy egy kör, vagy két kör, paritástól függően. Az összes x_ix_j legyen él, ha $x_i = x_j = 0$. Vegyünk fel még két csúcsot, akiket összekötünk mind az n korábbi csúccsal. Mivel a 0-s részt csak ők kötik össze az 1-es résszel, ezért pontosan akkor van Hamilton–kör, ha az 1-es rész egy körből áll.
II. (Borbényi Márton) Visszavezetjük, hogy annak eldöntése, hogy egy n hosszú 0–1 sorozatban ugyanannyi 1-es van-e, mint 0, az $\notin \mathbf{AC^0}$. (Ezt tudjuk, hiszen a MAJORITY-hoz hasonlóan kijönne belőle a PARITY.) Feleltessük meg a változókat egy gráf n csúcsának, és legyen x_ix_j él, ha $x_i + x_j = 1$.

147. $\mathbf{NC^1} \subset \mathbf{L}$, mert ki tudjuk értékelni a hálózatot: csak log sok szint van.
Az $\mathbf{NL} \subset \mathbf{AC^1}$ meg STCONN $\in \mathbf{AC^1}$ miatt igaz.

11. fejezet

Döntési fák

150. Nem 2-hatványokra, mert bármely d-vel oszthatóra (pl. csupa nullára, ha azt is megengedjük inputnak) a tanú n hosszú.

151. n, kivéve, ha $n \equiv 1 \mod 3$. Bizonyítás megy indukcióval. Egy fontos megfigyelés, hogy ha valaki 1-es, akkor annak a szomszédait is meg kell kérdezni előbb-utóbb, tehát feltehetjük, hogy egyből meg lesznek kérdezve.
Ehhez kapcsolódó cikk sok linkkel: https://arxiv.org/abs/1712.09738.
Egy nehéz feladat cikkből: Van-e 101 részsorozat eldöntése rejtőzködő.

152. Válaszoljunk úgy, hogy feszítőfát kapjunk a végén.

153. I. Válaszoljuk különböző komponensbeliekre azt, hogy nem, kivéve, ha ez a két komponens között az utolsó lehetséges él, akkor igen (hacsak nem az s és a t komponenséről van szó, akkor persze nemmel válaszolunk az utolsóra is).
II. Az st-re azt mondjuk, hogy nem, az ezen kívüli első kérdésre meg, hogy igen. Így két csúcsot összehúztunk és indukciót használhatunk. Ha a „dupla" csúcs egy élét kérdezné az algoritmus, akkor először mindig azt mondjuk, hogy nem, csak másodjára válaszoljunk az indukciós algoritmus szerint.

154. a) Vegyünk egy tetszőleges v csúcsot és kérdezzük végig az összes szomszédját. Ha v foka 0 vagy $n-1$, akkor nem skorpió a gráf. Ha $n-2$, akkor v egyetlen nemszomszédját végigkérdezve megtudjuk, hogy skorpió-e. Különben v se nem fej, se nem fullánk.
Ezután veszünk két-két csúcsot a v-vel szomszédosak és nemszomszédosak közül, és megkérdezzük a köztük levő 6 élet. Ez alapján valamelyikről kiderül, hogy szintén se nem fej, se nem fullánk. Ezt csinájuk, amíg szomszédosakból vagy nemszomszédosakból csak egy marad, majd arra is teszteljük, hogy nem fej vagy fullánk-e.
b) Mutatja, hogy kell a monotonitás. Ismeretlen forrásból: „Originally Aanderaa and Rosenberg conjectured it without monotonicity, as $\Omega(n^2)$, Aanderaa showed this to be false with scorpion. Rivest and Vuillemin proved monotone with $\Omega(n^2)$, then Karp made sharp conjecture."

156. Ez a Barrington-tétel.
Ötlet: Használjuk fel, hogy S_5 nem feloldható!
$\leq$: Indukcióval trivi.
$\leq$: Feltehetjük, hogy az $\mathbf{NC^1}$ hálózatban csak ÉS és NEM kapuk vannak. Minden d mély ilyenhez csinálunk egy 5 széles hálózatot, ami 4^d mély. Bármely szinten belül ugyanazt a változót kérdezzük majd. 5 lehetséges bemenő hely van, 5 kimenő hely, minden input ezeket permutálja. Azt mondjuk, hogy P α-kiszámolja C-t, ha $C(x) = 0$-ra identitás, $C(x) = 1$-re

pedig α lesz a kimenet permutációja.
Lemma 1: Bármely két 5-ciklus konjugált, azaz minden α, β-hoz van γ, hogy $\gamma^{-1}\alpha\gamma = \beta$.
Következmény: Ha α-kiszámolós P van, akkor β-kiszámolós P' is (és legfeljebb 2-vel mélyebb).
Lemma 2: Van α és β 5-ciklus, hogy kommutáturuk is ciklus, azaz $\alpha\beta\alpha^{-1}\beta^{-1} = \gamma$. Például: $\alpha = (12345)$, $\beta = (13542)$. Tehát bármilyen kaput tudunk szimulálni így, kész!

Definíció: Ha $f \not\equiv 1$ egy monoton növő Boole-függvény, akkor legyen a hozzá tartozó szimpliciális komplexus $K_f = \{S : f(S) = 0\}$.

157. (Pituk Sára) Tegyük fel, hogy létezik nem kontrahálható függvény, ami nem zárkózott és vegyük ezek közül azt az f függvényt, ami a lehető legkevesebb helyen vesz fel 0-t, azaz K_f minimális. f-re van $n-1$ mély döntési fa, tekintsünk egyet. Vegyük egy l levelét, ahol $f = 0$, és az ilyenek közül a „legjobboldalibb", azaz minden más 0-levélhez van egy olyan index, aminél l-ben 1 van, de a másik levélben 0. Az l levélhez két input tartozik, attól függően, hogy az utolsó, nem ismert bit 0 vagy 1—előbbit jelölje x_0, utóbbit x_1. Ezeken kívül nincs olyan y input, amire $y \geq x_0$ és $f(y) = 0$, mert ez ellentmondana l választásának. Legyen f' az a fügvény, ami pontosan x_0-ban és x_1-ben tér el f-től. Ha K_f kontrahálható volt, akkor $K_{f'}$ is az lesz, hiszen x_0-ban „kipukkaszthatjuk". De ez ellentmond K_f minimalitásának.
Forrás: Kahn-Saks-Sturtevant tétel, lásd 19. oldal itt: https://arxiv.org/abs/cs/0205031.

12. fejezet

Interaktív

159. B-nél és C-nél is legyen egy titkos kulcs, amit a másik nem ismer: $b, c \in \{0,1\}^n$. Ha A akar küldeni egy titkos üzenetet, azt megteheti az $x \oplus b \oplus c$ kóddal. Tipikus rossz megoldás volt, hogy $x \oplus r$ a kód és B ismeri a páratlan, C meg a páros bitjeit r-nek, de ez nem jó, mert így bármelyikük meg tudja fejteni x felét, mi pedig azt szeretnénk, hogy *semmit* ne tudhassanak meg egyedül x-ről.

12.1. Kommunikációs bonyolultság

Definíció: A kommunikációs játékban A és B játékos is ismeri az $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ függvényt és mindkettőjüknek van egy inputja, x illetve y, amit a másik nem lát. Az a feladat, hogy kiszámolják $f(x,y)$-t. Ehhez biteket küldhetnek egymásnak egy el?re megbeszélt protokoll alapján. Adott f-re a legjobb protokoll esetén a legrosszabb (x,y) párra küldendő bitek számát jelölje $\kappa(f)$.

160. I, A kommunikációs mátrix rangja 2^n, tehát a Mehlhorn-Schmidt tétel alapján nincs jobb a triviális protokollnál.
II, a kommunikációs fában az (i,i) inputpárok mind egy-egy különböző levélbe kell, hogy kerüljenek.

161. a) A mediánjaik közé esik az unió mediánja, ezért elég, ha minden lépésben elküldik egymásnak a mediánjaikat és az aktuális halmazaik méretét, majd akinek kisebb a halmaza, az az elemeinek a felét ki tudja dobni (és másik is ugyanennyit). Így $\log n$ lépés után egyik halmaz egyelemű lesz. Egy másik megoldás, hogy bármely számról $\log n + O(1)$ kommunikációval el tudjuk dönteni merre van tőle a medián és így binárisan kereshetünk.
b) Ügyesen keverjük a fenti két módszert.

162. Egyik definícónál mindenki csak magának tud generálni random biteket, másiknál pedig közös random forrásuk van, pl. aznapi lottó számok. Előbbi esetben $O(\log n)$ bittel megoldható, ha $x \mod p$-t elküldik egymásnak egy random p prímre.
Útóbbiban $O(1)$ bit is elég akár a prímes módszerrel, akár $\langle x, r\rangle \stackrel{?}{\equiv} \langle y, r\rangle \mod 2$-ből.

163. Lásd 19. oldal itt: https://arxiv.org/abs/1007.1841.

164. a) Egyik elküldi bármelyik csúcsát, másik ehhez legközelebbijét.
b) Fa csúcsait számozzuk úgy, hogy első k-t kitörölve minden komponens $< 2n/k$. Ezután elküldjük k hosszát ($\log\log n$ bit), majd k-t, majd $\log(2n/k)$ méretű választ. Forrás: M. Saks and L. Lovasz, "Lattices, mobius functions and communications complexity," FOCS 1988.
c) https://cstheory.stackexchange.com/questions/41524/what-is-the-exact-communication-complexity-of-subtree-disjointness.

12.2. Karchmer-Wigerson

Definíció: Karchmer-Wigderson játék: Kommunikációs játék egy verziója. A és B játékos is ismeri az $f : \{0,1\}^n \to \{0,1\}$ függvényt és mindkettőjüknek van egy inputja, x illetve y, amit a másik nem lát. Azt is tudják, hogy $f(x) = 0$ és $f(y) = 1$. Az a feladat, hogy állapodjanak meg egy i indexben, amire $x_i \neq y_i$. Ehhez biteket küldhetnek egymásnak egy előre megbeszélt protokoll alapján. Adott f-re a legjobb protokoll esetén a legrosszabb esetben $(x, y$ párra) küldendő bitek számát jelölje $KW(f)$.

166. Ha a kommunikáció legfeljebb $\log n - 3$ bit, akkor bármely fix inputra A csak az indexek kevesebb, mint negyedét tippelheti. Kettős leszámlálásból az indexek többsége olyan, hogy az inputok kevesebb, mint felében tippeli A. Ugyanez igaz B-re is, tehát lesz egy olyan index, amit egyikük sem tippel az inputjaik többségében. De ekkor lesz egy olyan inputpár is, amik csak ebben a bitben térnek el, és egyikük sem tippeli soha ezt a bitet az adott inputra.

168. Ha más a pariásuk, kész. Ha paritás megegyezik, de 4-el osztva más a maradékuk, akkor is kész. Így először megkeressük a legkisebb kettő hatványt, amire különböznek, aztán ha már nem fognak, akkor lesz nála kisebb is, amire fognak.

Forrás: Brodal-Husfeldt: A Communication Complexity Proof that Symmetric Functions have Logarithmic Depth https://www.brics.dk/RS/96/1/BRICS-RS-96-1.pdf

12.3. Zero-knowledge

169. Először is feltesszük, hogy x és m relatív prímek, vagy addig futtatjuk az euklideszi algoritmust, amíg azok nem lesznek. Aztán P elküldi r^2-et, majd V választ, hogy r-et akarja-e látni vagy ry-t. Ha x nem négyzetszám, akkor min 50% az esélye, hogy P bajban lesz. Másfelől V nem tud meg mást, mint egy véletlen számot és annak a gyökét.
Ha szigorú definícó szerint akarjuk ez utóbbit megmutatni, akkor P-V kommunikációkat kell V-nek generálnia. A szimuláláshoz vegyük r^2-et és r^2/x-et 50-50% eséllyel, mindkettő eloszlása uniform. Ezután „sorsoljuk”, hogy V azt fogja kérni, amit tudunk.