

Bonyolultságelmélet jegyzet

Készítették Grolmusz Vince előadásai alapján a 2025/26. évi hallgatók

(Nem hivatalos lektorátlan verzió)

2026-1-28

Contents

1	Kommunikációs játékok	3
2	NP-n túl	10
2.1	Polinomialis hierarchia	10
2.2	PSPACE teljesség	13
3	Interaktív bizonyítások	18
3.1	Zero Knowledge Proofs	24
4	Véletlen bonyolultság osztályok	26
5	Boole-hálózatok	35
6	Párhuzamos algoritmusok	40
7	VLSI chip tervezés	44
8	Rendező hálózatok	46
9	Döntési fák	48
10	Pszudovéletlen számok	49
11	Kolmogorov bonyolultság	53

1 Kommunikációs játékok

Cél: van két játékos, akik bármit ki tudnak számolni gyorsan, de egymás között nehezen kommunikálnak.

Definíció 1.1

Kommunikációs játék

Adott $f : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}$ és $x, y \in \{0, 1\}^n$. A ismeri x -et, de y -t nem, B ismeri y -t, de x -et nem. Ki akarják számolni $f(x, y)$ -t. A költség az A és B között (bármely irányban) kommunikált bitek száma.

Akkor tekintjük $f(x, y)$ -t kiszámoltnak, ha az egyik játékos ismeri $f(x, y)$ -t, és a másik játékos tudja, hogy az egyik tudja.

Definíció 1.2

Protokoll költsége

A P protokoll mellett f költsége a legrosszabb (x, y) input páron $\kappa_{P(f)}$.

Megjegyzés

Megkövetelhetnénk, hogy mindketten tudják $f(x, y)$ -t, ez 1 bit különbséget jelentene csak legfeljebb.

Definíció 1.3

Protokoll

A közös számolási módszer szabályait, hogy mikor ki, és milyen bitet küld protokollnak nevezzük. (Ez az algoritmus megfelelője több játékos esetén.)

Példa

Legyen f tetszőleges, ekkor A elküldheti x -et B-nek, aki „ingyen” kiszámolja $f(x, y)$ -t. Ennek a költsége n .

Példa

ID-függvény

Legyen

$$\text{ID}(x, y) = \begin{cases} 1, & \text{ha } x = y \\ 0, & \text{ha } x \neq y \end{cases}$$

Ekkor a fenti P protokollal $\kappa_P(\text{ID}) = n$ teljesül.

Definíció 1.4

Kommunikációs bonyolultság

$\kappa(f)$ a $\kappa_P(f)$ -ek minimuma az összes f -et kiszámoló P protokollon.

Tétel 1.5

$$\kappa(\text{ID}) = n.$$

Ennek a bizonyításához kell a következő definíció és tétel.

Definíció 1.6

Kommunikációs mátrix

Az f kommunikációs mátrixa az az $M_f \in \{0,1\}^{2^n \times 2^n}$, amelynek sorai x -szel, oszlopai y -nal vannak indexelve, és az x -hez tartozó sor y -hoz tartozó oszlopában $f(x, y)$ szerepel.

Megjegyzés

A továbbiakban a \log mindig a 2-es alapú logaritmust jelenti.

Tétel 1.7

Mehlhorn–Schmidt

$\kappa(f) \geq \log r(M_f)$, ahol $r(M_f)$ az M_f mátrix rangját jelöli.

Proof: Legyen P egy adott protokoll. Tegyük fel, hogy A kezd. Ekkor A kommunikál egy bitet. Ez rögzített P protokoll mellett bizonyos x -ekre 0, bizonyos x -ekre 1. Ezzel az M_f mátrixot két részre bontja: az egyik részben azon sorok vannak, amelyekre 0-t mond, a másikban azok, amelyekre 1-et. Ezek közül az egyik sorrangja $\geq \frac{1}{2}(M_f)$.

Ezt ismételjük addig, amíg A lép. Amikor B lép, akkor ugyanez elismételhető oszloprangra, de egy mátrix sor- és oszloprangja megegyezik. Ha x és y olyan, hogy minden lépésnél a nagyobb rangú részmátrixot adják meg, akkor k lépés után a részmátrix rangja $\geq 2^{-k}r(M_f)$.

Tegyük fel, hogy a k . lépésben vége van a játéknak. Ekkor szimmetriaokokból feltehető, hogy A tudja $f(x, y)$ -t, és B tudja, hogy A tudja. Mivel A tudja $f(x, y)$ -t, az így kapott részmátrix minden sora homogén, azaz vagy csupa 0-t, vagy csupa 1-et tartalmaz. Ha pedig egy sor nem homogén, akkor A nem tudhatja biztosan $f(x, y)$ -t. Hasonlóan, az, hogy B tudja biztosan $f(x, y)$ -t, az azzal ekvivalens, hogy a kapott részmátrix minden oszlopa homogén.

Mivel homogén részmátrix rangja 1, az előbbi egyenlőtlenség szerint $1 \geq 2^{-k}r(M_f)$, azaz $2^k \geq r(M_f)$ fog teljesülni minden olyan (x, y) párra, amelyeket P k lépésben számol ki.

□

Következmény 1.8

Innen könnyen kijön, hogy $\kappa(\text{ID}) = n$, ugyanis $M_{\text{ID}} = I_{2^n}$, és $r(I_{2^n}) = 2^n$, tehát $n \leq \kappa(\text{ID})$ a Mehlhorn–Schmidt-tétel miatt. Másrészt láttuk, hogy $\kappa(f) \leq n$ minden f -re, így $\kappa(\text{ID}) = n$.

Megjegyzés

Felső becslés nem ismeretes $\kappa(f)$ -re. Lovász és Suchs nevéhez fűződő sejtés szerint $\exists c > 0 : \kappa(f) \leq \log^c(r(M_f))$. Tudjuk, hogy $c > 2$ kell hogy teljesüljön. Ismert továbbá, hogy $\kappa(f) \leq r(M_f)$.

Következmény 1.9

$\text{DISJ}(x, y) = \chi_{\{x \cdot y = 0\}}$, a halmazdiszjunktági feladat. Akkor erre is $\kappa(\text{DISJ}) = n$.

Proof of Következmény: Elemszám szerint rendezve az n elemű halmaz részhalmazait a sorokban, és a komplementereiket az oszlopokban

$$M_{\text{DISJ}} = \begin{bmatrix} 1 & * & * & \dots \\ 0 & 1 & * & \dots \\ 0 & 0 & 1 & \dots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

felsőháromszög alakú, vagyis $\kappa(\text{DISJ}) = n$

□

Definíció 1.10

Nemdeterminisztikus kommunikációs bonyolultság

Alíz ismeri x -et, Bob ismeri y -t, E.T. ismeri mindkettőt, és f -et is. Utóbbi meg akarja győzni a játékosokat, hogy tudja. Ezt egy bizonyítással teszi, amit függetlenül A-nak, és B-nek is el kell fogadnia. Egy fix E.T. által az (x, y) párra adott bizonyítás hossza, amikor azt akarja bizonyítani, hogy $f(x, y) = 1$ legyen $\kappa_1^{\text{E.T.}}(f(x, y))$. Legyen továbbá

$$\kappa_1^{\text{E.T.}}(f) := \max_{\{x, y: f(x, y) = 1\}} \kappa_1^{\text{E.T.}}(f(x, y)),$$

végül

$$\kappa_1(f) = \min_{\text{E.T.}} \kappa_1^{\text{E.T.}}(f)$$

a legjobb E.T. által a legrosszabb esetben adott bizonyítás hossza. Hasonlóan definiáljuk a $\kappa_0(f)$ -et is.

Megjegyzés

$\max \kappa_0(f), \kappa_1(f) \leq \kappa(f)$ teljesül, hiszen reprodukálhatja az adott esetben a protokoll által megszabott kommunikációját

Példa

Ha $x \neq y$, akkor az $(i, x_i = 0)$ pár (ahol $y_i = 1$) megadása $\log(n) + 1$ bit hosszú, és bizonyítja, hogy az ID feladat nem teljesül. Egyenlőségre nem látszik kapásból hasonló jó bizonyítás.

Tétel 1.11

Az ND kommunikációs bonyolultság jellemzése fedő téglalapokkal

$\kappa_1(f)$ az a legkisebb t szám, hogy M_f egyesei lefedhetők 2^t darab csupa 1-es részmátrixsal

Megjegyzés

M_f -et már ismerjük, a kommunikációs mátrix. A tételben részmátrix alatt az oszlopok, és sorok egy-egy részhalmazait kiválasztva, a metszetekből álló részt értjük. Figyelem, ez nem feltétlenül egy összefüggő téglalap!

$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

-ben az első és utolsó sor, és oszlopok által meghatározott rész is egy ilyen csupa egyes részmátrix.

Következmény 1.12

Láttuk, hogy $M_{ID} = I_{2^n}$, ezt pedig csak úgy fedhetjük le csupa 1-es téglalapokkal, ha külön-külön kiválasztjuk az átlóelemeket. Következik, hogy $\kappa_1(ID) = n$.

Proof of Tétel 1.11:

$$(\kappa_1(f) \leq t)$$

Tekintsük a fedő téglalapokat. Alíz-nak van egy sora, Bob-nak egy oszlopa. A protokollban megállapodnak a 2^t darab fedőmátrix egy sorrendjében. E.T. bizonyítása az lesz, hogy hanyadik részmátrixban van az (x, y) metszet, ez t bittel kódolható, leellenőrzi, hogy benne van-e az adatuk, és mivel ez csupa egyesből áll, így szükségszerűen $f(x, y) = 1$. %feltesszük, hogy E.T. nem hazudik?

$$(\kappa_1(f) \geq t)$$

Legyen

$$H_\alpha = \{(x, y) : A\text{-nál } x, B\text{-nél } y \text{ van, és } \alpha \text{ üzenetet hallják, akkor elfogadják a bizonyítást}\}.$$

Ha $(x_1, y_1), (x_2, y_2) \in H_\alpha$, akkor $(x_1, y_2), (x_2, y_1) \in H_\alpha$, hiszen az α bizonyítást Alíz elfogadta (x_1, y_1) -re, az ő nézőpontjából semmi nem különbözteti meg a szituációt attól, mintha (x_1, y_2) lenne a felállítás, ezt pedig Bob is elfogadja, hiszen számára (x_1, y_2) , és (x_2, y_2) ugyanolyan, és ez utóbbit elfogadta α -ra. Következik, hogy minden α -ra H_α megfelel egy részmátrixnak. Ha E.T. legfeljebb t bitből bizonyítani tudja, hogy $f(x, y) = 1$, ez szolgáltat lazannyt és 2^t darab csupa egyes részmátrixot.

□

Randomizálva azonban gyorsan is lehet a következő Simon és Rabin nevéhez fűződő protokollal. A generál egy véletlen p prímet $\in \{1, \dots, n^2\}$ (ahol $\log x, \log y \leq n$), és elküldi az $(x \bmod p, p)$ üzenetet, B pedig leellenőrzi, hogy $x \equiv y \bmod p$ teljesül-e, és ezt mondjuk százszor megismétlik.

Ha egyszer is az teljesül, hogy inkongruensek, akkor az eredeti számok sem lehettek egyenlőek, ha mindig kongruensek, és mégsem egyenlőek, akkor százszor teljesült az, hogy $p | x - y \neq 0$.

$A \leq 2^n$ számoknak legfeljebb n darab prímosztója lehet, és n^2 -ig nagyjából $\pi(n^2) \sim \frac{n^2}{2 \log(n)}$ darab prím van. Annak a valószínűsége, hogy egyszer teljesül a kongruencia

$$\mathbb{P}(p|x-y) \leq \frac{n}{\frac{n^2}{2\log(n)}} = \frac{2\log n}{n} \rightarrow 0.$$

Egy kommunikáció $4\log n$ bitet küld, ergo összesen $400\log n$ bitnyi kommunikáció történik.

Nem (teljesen) triviális protokollok:

Példa

Tekintsünk egy fagráfot, aminek van két részfája. Kérdés, hogy az n csúcsú T fa T_1, T_2 részfáinak van-e közös csúcsa. Alíz kapja T_1 -et, Bob T_2 -t értelemszerűen, és mindketten ismerik T -t. Ez eldönthető lenne a DISJ játék speciális eseteként, de adunk egy okosabb protokollt.

Alíz megmondja T_1 egy tetszőleges v csúcsát (ez ugye $\log n$ bit kommunikáció). Majd Bob kiszámolja T_2 -ben a v -hez legközelebbi w csúcsot, mivel fában egyértelmű út van két csúcs között, ez értelmes. Ezt visszaküldi Alíznek, ellenőrzi, hogy $w \in T_1$, ha igen, ez metszetbeli, és készen vagyunk, ha nem, akkor azt mondja, hogy a két fa diszjunkt. Ugyanis, ha a legközelebbi w pont nem része a fának, de egy további u pont része lenne T_1 -nek, az uw szakasz T_2 -ben van, az uv szakasz pedig T_1 -ben, vagyis u közelebb van v -hez, mint w .

Példa

Most Alíz és Bob két részgráfot kap egy G gráfból úgy, hogy G_A független csúcsokból áll, G_B pedig egy teljes részgráf. Kérdés, hogy van-e metszet?

Világos, hogy ha van, legfeljebb 1 pontból állhat.

1. Alíz megnézi, hogy van-e legalább $\frac{n}{2}$ fokú v csúcs a gráfjában, ha igen, akkor $(1, v)$ -t küldi el, ha nem, 0-t.
2. Bob megnézi, hogy van-e $< \frac{n}{2}$ fokú w csúcs G_B -ben, ha igen, $(1, w)$ -t küld, ha nincs, 0-t.

Ezek után Bob tudja, hogy G_A v -ből, és a nem-szomszédaiából áll, ez legfeljebb $\frac{n}{2}$ csúcsból áll, és iteratíven folytathatjuk ezt az eljárást amíg lehet. Ha Bob talál egy kis fokszámú w csúcsot, akkor az ő gráfjának a többi csúcsa ennek a szomszédai közül kerül ki, és ismét rekurzíven folytatható az eljárás. Mi történik, ha mindketten 0-t küldenek? Alíz gráfjában minden csúcs kisebb mint $\frac{n}{2}$ fokú, G_B -ben pedig minden csúcs legalább $\frac{n}{2}$ fokú, ez a két feltétel kizárja egymást, így a két gráf diszjunkt. Addig ismételtetik a fenti lépést, amíg nem mondanak mindketten nullát. Egy lépés $\log n + 1$ bit, és $\log n$ lépésben persze kimerítik a gráfot, vagyis $O(\log^2 n)$ bitre van összesen szükség.

Tétel 1.13

Aho–Ullman–Yanakakis

Minden f -re

$$\kappa(f) \leq (2 + \kappa_0(f))(2 + \kappa_1(f)).$$

Lemma 1.14

Ha M egy $0-1$ mátrix, H egy azonosan nulla részmátrixa, H sorai alkossák az A , oszlopai a B mátrixot, ekkor $\rho(A) + \rho(B) \leq \rho(M)$, ahol $\rho(M)$ a sor/oszloppermutációval képezhető legnagyobb négyzetes felsőháromszög részmátrix méretét jelöli, aminek a főátlója csupa 1-ből áll.

Proof of Lemma: A lemma azon múlik, hogy A és B -t külön-külön mozgathatjuk, a csupa nulla metszet nem fog változni, és a másik mátrixhoz nem nyúltunk hozzá, diszjunkt sorokból/oszlopokból áll. Egy permutációval megfelelő helyre visszük A -ban a maximális U_A felsőháromszög mátrixot, ezt B -ben is elvégezve (U_B) kapunk egy $\begin{bmatrix} U_B & \\ 0 & U_A \end{bmatrix}$ felsőháromszöget M -ben.

$$\left[\begin{array}{c|c|c} & B_1 & \\ \hline A_1 & 0 & A_2 \\ \hline & B_2 & \end{array} \right] \rightarrow \left[\begin{array}{c|c|c|c|c} & B_1 & B_1 & & \\ \hline & B_1 & U_B & & \\ \hline A_1 & 0 & 0 & U_A & A_2 \\ \hline A_1 & 0 & 0 & A_2 & A_2 \\ \hline & B_2 & B_2 & & \end{array} \right]$$

□

Proof of Aho–Ullman–Yanakakis: Világos, hogy $\rho(M_f) \leq r(M_f)$, és $\log \rho(M_f) \leq \kappa_1(f)$ teljesülnek, mert egy csupa 1 főátlójú felsőháromszög mátrix teljes rangú, illetve Tétel 1.11 miatt.

Indukcióval belátjuk, hogy $\kappa(f) \leq (2 + \log \rho(M_f))(2 + \kappa_0(f))$. Ha $\rho(M_f) = 1$, akkor nem is kell kommunikálni, mert egy ilyen mátrixban vagy csak egyesek állnak, vagy pontosan egy sorában vagy oszlopában vannak egyesek. Az általános lépésben tekintsük a kommunikációs mátrix nullásainak a fedését $2^{\kappa_0(f)}$ darab csupa nulla részmátrixszal. Alíz megnézi, hogy fedie-e az ő x inputjának egy részét olyan csupa 0 részmátrix, hogy a hozzá tartozó sorokból alkotott A mátrixra $\rho(A) \leq \frac{\rho(M_f)}{2}$, ha igen, akkor elküldi az $(1, \text{a csupa nulla részmátrix sorszáma})$ üzenetet, ez legfeljebb $1 + \kappa_0(f)$ bit kommunikáció, ha nincs ilyen részmátrix, akkor 0-t küld. Bob hasonlóan megnézi, hogy van-e az y -jához olyan fedő csupa 0 mátrix, amely oszlopaihoz tartozó B mátrixra $\rho(B) \leq \frac{\rho(M_f)}{2}$, ha igen $(1, \text{a fedő mátrix sorszáma})$, ha nincs ilyen, akkor pedig 0-t küld.

Mi történik, ha mindketten 0-t küldenek?

Akkor $f(x, y) = 1$, hiszen ha 0 lenne, akkor a metszetüket lefedné egy csupa 0 részmátrix, de az eddigi kommunikáció szerint az ezen fedőmátrixhoz tartozó sorok, és oszlopok ρ értékei összesen többet adnak, mint $\rho(M_f)$, ellentmondásban a lemmánkkal.

□

Definíció 1.15

Kommunikációs bonyolultságok

$f \in \text{P}^{\text{CC}}$, ha $\exists c > 0 : \kappa(f) \leq \log^c n$.

$f \in \text{NP}^{\text{CC}}$, ha $\exists c > 0 : \kappa_1(f) \leq \log^c n$.

$f \in \text{co-NP}^{\text{CC}}$, ha $\exists c > 0 : \kappa_0(f) \leq \log^c n$.

A fenti tétel következményeként adódik, hogy $\text{P}^{\text{CC}} = \text{NP}^{\text{CC}} \cap \text{co-NP}^{\text{CC}}$.

Láttuk továbbá, hogy $\text{NP}^{\text{CC}} \neq \text{co-NP}^{\text{CC}}$, mert ID benne van a jobb oldalon, de a balban nincs.

$P^{CC} \neq \text{co-NP}^{CC}$ szintén az ID miatt (így $P^{CC} \neq NP^{CC}$ is teljesül).

2 NP-n túl

2.1 Polinomialis hierarchia

Definíció 2.1

Polinomiális reláció

Azt mondjuk, hogy $P(x, y_1, y_2, \dots, y_l)$ egy polinomiális reláció, ha $\exists c$ úgy, hogy $\forall i : |y_i| \leq |x|^c$ és $P(x, y_1, \dots, y_l)$ kiszámolható $|x|$ -ben polinomimális időben.

Definíció 2.2

Σ_i

Tetszőleges L nyelvre $L \in \Sigma_i \Leftrightarrow \exists P(x, y_1, \dots, y_i)$ polinomiális reláció úgy, hogy $x \in L \Leftrightarrow \exists y_1 \forall y_2 \exists y_3 \dots Q y_i$ úgy, hogy $P(x, y_1, y_2, \dots, y_i)$ teljesül. Ahol Q a következőképpen van definiálva:

$$Q = \begin{cases} \forall & \text{ha } i \text{ paros} \\ \exists & \text{ha } i \text{ páratlan} \end{cases}$$

Definíció 2.3

Π_i

Tetszőleges L nyelvre $L \in \Pi_i \Leftrightarrow \exists P(x, y_1, \dots, y_i)$ polinomialis relacio úgy, hogy $x \in L \Leftrightarrow \forall y_1 \exists y_2 \forall y_3 \dots \tilde{Q} y_i$ úgy, hogy $P(x, y_1, y_2, \dots, y_i)$ teljesül. Ahol \tilde{Q} a következőképpen van definiálva:

$$\tilde{Q} = \begin{cases} \forall & \text{ha } i \text{ páratlan} \\ \exists & \text{ha } i \text{ paros} \end{cases}$$

Példa

Pár nevezetes bonyolultsági osztály amit már ismerünk:

1. $\text{NP} = \Sigma_1$
2. $\text{co-NP} = \Pi_1$
3. $\text{P} = \Sigma_0 = \Pi_0$

Megjegyzés

1. Minden i -re $\Sigma_i \subseteq \Sigma_{i+1}$. Valasszuk úgy a polinomimalis relaciot hogy az utolso valtozotol ne fuggjon.
2. Minden i -re $\Pi_i \subseteq \Pi_{i+1}$. Valasszuk úgy a polinomimalis relaciot hogy az elso valtozotol ne fuggjon.
3. Minden i -re $\Pi_i \subseteq \Sigma_{i+1}$.
4. Minden i -re $\Sigma_i \subseteq \Pi_{i+1}$.

Ezen osztályokat a következő hierarchiával tudjuk vizuálisan jellemezni.

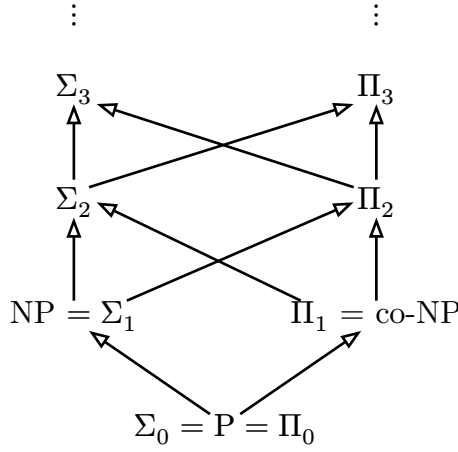


Figure 1: Polinomiális hierarchia vizualizáció

Definíció 2.4

Polynomialis Hierarchia

$$\text{PH} = \bigcup_{i=1}^{\infty} \Sigma_i = \bigcup_{i=1}^{\infty} \Pi_i$$

Definíció 2.5

$$\text{INDEPENDENT} := \{(G, m) : \alpha(G) \geq m\}$$

azaz, azon G grafok és m számok parosai, melyekre G függetlenségi száma nagyobb mint m .

Definíció 2.6

$$\text{EXACT_INDEPENDENT} := \{(G, m) : \alpha(G) = m\}$$

azaz, azon G grafok és m számok parosai, melyekre G függetlenségi száma pont m .

Állítás 2.7

$$\text{EXACT_INDEPENDENT} \in \Sigma_2$$

Proof: $\exists H \subseteq V(G)$ független csucshalmmmaz és $|H| = m$
 $\forall H' \subseteq V(G)$ csucshalmazra, ahol $|H| = m + 1$ mar H' összefuggo.

□

Megjegyzés

A letezest (\exists) és a mindent (\forall) nem kell polinomialis időben számolni, csak a H -t és H' -t kell polinomialis időben ellenőrizni.

Tétel 2.8

Ha $\exists i \geq 1$ amire $\Sigma_i = \Pi_i$, akkor $\Sigma_{i+1} = \Pi_{i+1}$, amiből tovább következik, hogy $\text{PH} = \Sigma_i = \Pi_i$. Azt mondjuk, hogy a polinomialis hierarchia *összeomlik* az i -edik szintre.

Proof: Mivel tudjuk, hogy $\Sigma_i \subseteq \Sigma_{i+1}$, ezért elég azt belátnunk, hogy $\Sigma_{i+1} \subseteq \Sigma_i$ és ezzel belátjuk, hogy $\Sigma_i = \Sigma_{i+1}$. Hasonló módon be tudjuk látni hogy $\Pi_i = \Pi_{i+1}$.

Legyen $L \in \Sigma_{i+1}$ tetszőleges nyelv, bizonyítsuk be hogy $L \in \Sigma_i$. Mivel $L \in \Sigma_{i+1}$, ezért letezik egy P polinomialis relacio, melyre

$$x \in L \Leftrightarrow \exists y_1 \forall y_2 \exists \dots Q y_{i+1} P(x, y_1, \dots, y_i).$$

Tovabba, letezik egy $L' \in \Pi_i$ nyelv, melyre

$$x \in L \Leftrightarrow \exists y_1 : (x, y_1) \in L'.$$

Figyelem, itt csak annyi történt hogy beillesztettük egy extra y_1 változót a letezés (\exists) kvantorral a Π_i definíció ele, így kaptunk egy definíciót Σ_{i+1} -re.

Mivel $\Sigma_i = \Pi_i$, ezért $L' \in \Sigma_i$, tehát letezik egy polinomialis relacio S úgy, hogy

$$x \in L \Leftrightarrow \exists y_1 \exists y_2 \forall \dots Q y_{i+1} S(x, y_1, \dots, y_i).$$

Csoportosíthatjuk y_1 -et és y_2 -t.

$$x \in L \Leftrightarrow \exists (y_1, y_2) \forall \dots Q y_{i+1} S(x, (y_1, y_2), \dots, y_i).$$

A jobboldalon i darab kvantor van és pont abban a sorrendben mint ahogy kell lenniük Σ_i definíciójához. Tehát beláttuk, hogy ha $L \in \Sigma_{i+1}$ akkor $L \in \Sigma_i$.

□

Tétel 2.9

Savitch

Ha $f(n) \geq n$, akkor

$$\text{NSPACE}(f(n)) \subseteq \text{DSPACE}(f^2(n))$$

Proof: Legyen $L \in \text{NSPACE}(f(n))$ egy tetszőlegese nyelv, a célunk megmutatni, hogy L felismerhető egy determinisztikus Turing-géppel $f^2(n)$ tárbán.

Figyeljük meg, hogy ha egy Turing-gép t tárat használ futása alatt, akkor legfeljebb $O(2^{c \cdot t})$ különböző konfigurációba kerülhet.

Tudjuk, hogy van egy nemdeterminisztikus Turing-gép mely felismeri az L nyelvet, tehát a konfigurációs gráfban van út a kezdőállapotból a reprezentáns elfogadó állapotok csúcsába. Mivel legfeljebb $2^{c \cdot f(n)}$ konfiguráció van, ezért egy elfogadó út hossza legfeljebb $2^{c \cdot f(n)}$.

Ha tudunk mutatni egy determinisztikus Turing-gépet ami el tudja dönteni egy gráfban, hogy adott s és t csúcsok között van-e út $O(\log^2 n)$ tárbán, akkor a konfigurációs gráfra alkalmazva $O(f^2(n))$ méretű tárat használó eljárást adnánk L felismerésére.

Megmutatjuk, hogy $O(\log^2 n)$ tárbán el tudjuk dönteni, hogy s és t között megy-e út egy adott G gráfban. Legyen $\text{st-conn}(k, s, t)$ az algoritmus ami eldönti, hogy legfeljebb k hosszú út van-e s és t között. Nyilván ha van olyan $u \in V(G)$ csúcs mely s -ből elérhető legfeljebb $k/2$ hosszú úton és u -ból t elérhető legfeljebb $k/2$ hosszú úton, akkor s -ből t is elérhető legfeljebb k hosszú úton.

Tehát a következőképpen néz ki a rekurziónk:

$$\begin{cases} \text{st-conn}(0, s, t) = (s \stackrel{?}{=} t) \\ \text{st-conn}(1, s, t) = (st) \stackrel{?}{\in} E(G) \\ \text{st-conn}(k, s, t) = \exists? u \in V(G) : \text{st-conn}(k/2, s, u) \wedge \text{st-conn}(k/2, u, t). \end{cases}$$

Látszik, hogy a rekurzió mélysége $O(\log n)$ és mindegyik rekurzív hívásban csak a függvény argumentumait kell tárolnunk amiket bitekben $O(\log n)$ tárban meg tudjuk oldani. Tehát az st-conn algoritmus $O(\log^2 n)$ tárban működik, és ezzel készen is vagyunk, mivel

$$(\log(2^{c \cdot f(n)}))^2 = (c \cdot f(n) \cdot \log 2)^2 = c^2 \cdot f^2(n) = O(f^2(n)).$$

□

Következmény 2.10

$$\text{NPSPACE} = \text{PSPACE}$$

Proof: Polinom négyzete polinom.

□

2.2 PSPACE teljesség

Definíció 2.11

PSPACE teljesség

Azt mondjuk, hogy L PSPACE teljes, ha $L \in \text{PSPACE}$ és $\forall L' \in \text{PSPACE}$ nyelvre $L' \leq L$. Tehát L' visszavezethető L -re polinomiális időben.

Definíció 2.12

tqbf – Totally Quantified Boolean Formula

Azt mondjuk, hogy φ egy teljesen kvantifikált Boole-formula, ha olyan alakba írható, hogy

$$\varphi = Q_1 x_1 Q_2 x_2 \dots Q_l x_l f(x_1, x_2, \dots, x_l),$$

ahol $Q_i \in \{\forall, \exists\}$ és x_i Boole változók és $f(x_1, \dots, x_l)$ egy konjunktív normál formula (CNF).

Példa

$$\varphi = \forall x \exists y \exists z ((x \vee z) \wedge y)$$

Ez a formula igaz.

Megjegyzés

Egy tqbf vagy igaz vagy hamis. Nem olyan mint egy CNF ahol az a kérdés hogy van-e helyes behelyettesítése, hanem magát a kvantálás megválaszolja, hogy a formula igaz vagy hamis.

$TQBF := \{\varphi, \text{ ahol } \varphi \text{ egy tqbf és } \varphi = \text{true}\}.$

Azaz TQBF az igaz teljesen kvantifikált Boole formulák nyelve.

Tétel 2.14

A TQBF nyelv PSPACE teljes.

Proof:

1. $TQBF \in PSPACE$

Végezzünk teljes indukciót a kvantorok számára. Ha $(n - 1)$ kvantoros tqbf ellenőrzését el tudjuk végezni $\text{poly}(n)$ tárban, akkor n kvantoros tqbf ellenőrzésénél csak 1-el több bitet kell tárolnom a kvantor típusára és meg x_n értékét.

2. $\forall L \in PSPACE : L \propto TQBF$

Röviden megemlítjük, hogy ebben az esetben nem tudjuk azt a trükköt eljátszani amivel bizonyítottuk, hogy $SAT \in NPC$, mert a Turing-gép összes szabályos lépését leíró formula hossza már bőven nem polinomiális lesz. Ez a trükk azért működött a SAT feladatnál, mert ott polinomiális időben kellett ellenőriznünk, itt viszont a tárnak kell polinomiálisnak lennie.

Az ötlet, hogy újra felhasználjuk az st-conn feladatot. Ha fel tudjuk írni az st-conn feladatot mint egy polinomiális méretű tqbf a kezdő csúcsra és az elfogadó csúcsok reprezentására a konfiguráció gráfra, akkor készen lennénk. Mivel bármilyen polinomiális tárban felismerhető nyelvet át tudunk írni polinomiális időben egy polinomiálisan hosszú tqbf-re.

Nézzük mit ad a köztes csúcs trükk amit már használtunk a Savitch tétel bizonyításában:

$$\text{st-conn}(k, s, t) \Leftrightarrow \exists u \in V : \text{st-conn}(k/2, s, u) \wedge \text{st-conn}(k/2, u, t).$$

A probléma ezzel a felírással, hogy bár feleztük a k paramétert, de a formula hossza nőtt, tehát összességében nem értünk el érdembeli javulást.

A trükk az, hogy kihasználjuk az univerzális kvantort (\forall), hogy ne kelljen dupláznunk a formula méretét:

$$\text{st-conn}(k, s, t) \Leftrightarrow \exists u \in V : \forall (x, y) \in \{(s, u), (u, t)\} : \text{st-conn}(k/2, x, y).$$

Ha az L nyelvet t tárban felismerte egy Turing-gép, akkor legfeljebb $2^{c \cdot t}$ konfigurációja van. Tehát a konfigurációs gráfnak legfeljebb $2^{c \cdot t}$ csúcsa van. Mivel a jobboldali formula mérete polinomiális és k értékét mindig felezzük, ezért a végső formula mérete polinomiális lesz.

□

Definíció 2.15

Generalized Geography játék

Legyen (G, u) egy rendezett pár, ahol G egy irányított gráf és $u \in V(G)$ a gráf egy adott csúcsa. A játékot Alíz és Bob játssza a következő szabályok alapján:

- Alíz kezd az u csúcsból.
- Alíz és Bob felváltva lépnek.
- A jelenlegi csúcsból csak belőle kifelé menő élrel keresztül szabad lépni a következő csúcsba.
- Már látogatott csúcsba tilos lépni.
- Ha a soron következő játékosnak már nincs szabályos lépése, akkor az ellenfél nyer.

Megjegyzés

Ez a játék az általánosítása az ország-város játéknak, ahol felváltva sorolunk városokat azzal a megkötéssel, hogy a következő város azzal a betűvel kezdőthet amivel az előző végződött és az veszít aki már nem tud várost mondani.

Definíció 2.16

Generalized Geography osztály

$GG = \{(G, u) : \text{Alíznek van nyerő stratégiája } u\text{-ból indulva}\}.$

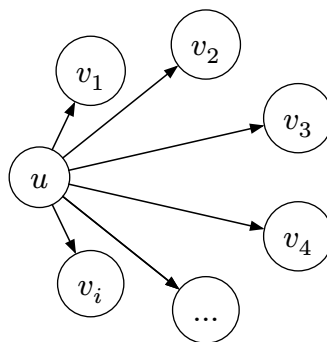
Tétel 2.17

A GG nyelv PSPACE teljes.

Proof:

1. $GG \in PSPACE$

Végezzünk teljes indukciót a leghosszabb út hosszára. Ha polinomiális tárban el tudjuk dönteni, hogy Bob-nak nincsen stratégiája legfeljebb $(n - 1)$ hosszú útra, akkor tudjuk, hogy Alíznek van nyerő stratégiája.



Nézzük meg u -nak az összes ki-szomszédjára, hogy Bob-nak nincs nyerő stratégiája. Mivel minden szomszédra polinomiális tárban eldönthetjük, és a tárat újra tudjuk használni, ezért az egész feladatot el tudjuk dönteni polinomiális tárban.

2. $TQBF \propto GG$

A bizonyítás ezen részén azt kell belátnunk, hogy ha kapunk egy teljesen kvantifikált Boole-formulát, akkor arra tudunk adni egy irányított gráfot, amiben pontosan akkor van nyerő stratégiája Alíznek, ha a tqbf igaz. Legyen például a tqbf a következő:

$$\varphi = Q_1 x_1 Q_2 x_2 \dots Q_l x_l f(x_1, x_2, \dots, x_l).$$

Az irányított gráfot két részből fogjuk felépíteni: kvantifikált értékadások (bal), és ellenőrzés (jobb). Az értékadás részben mindegyik x_i változóra létrehozunk egy kétirányú elágazást, ahol a balra vezető út azt jelenti, hogy x_i igaz, míg a jobbra vezető út azt, hogy x_i hamis.

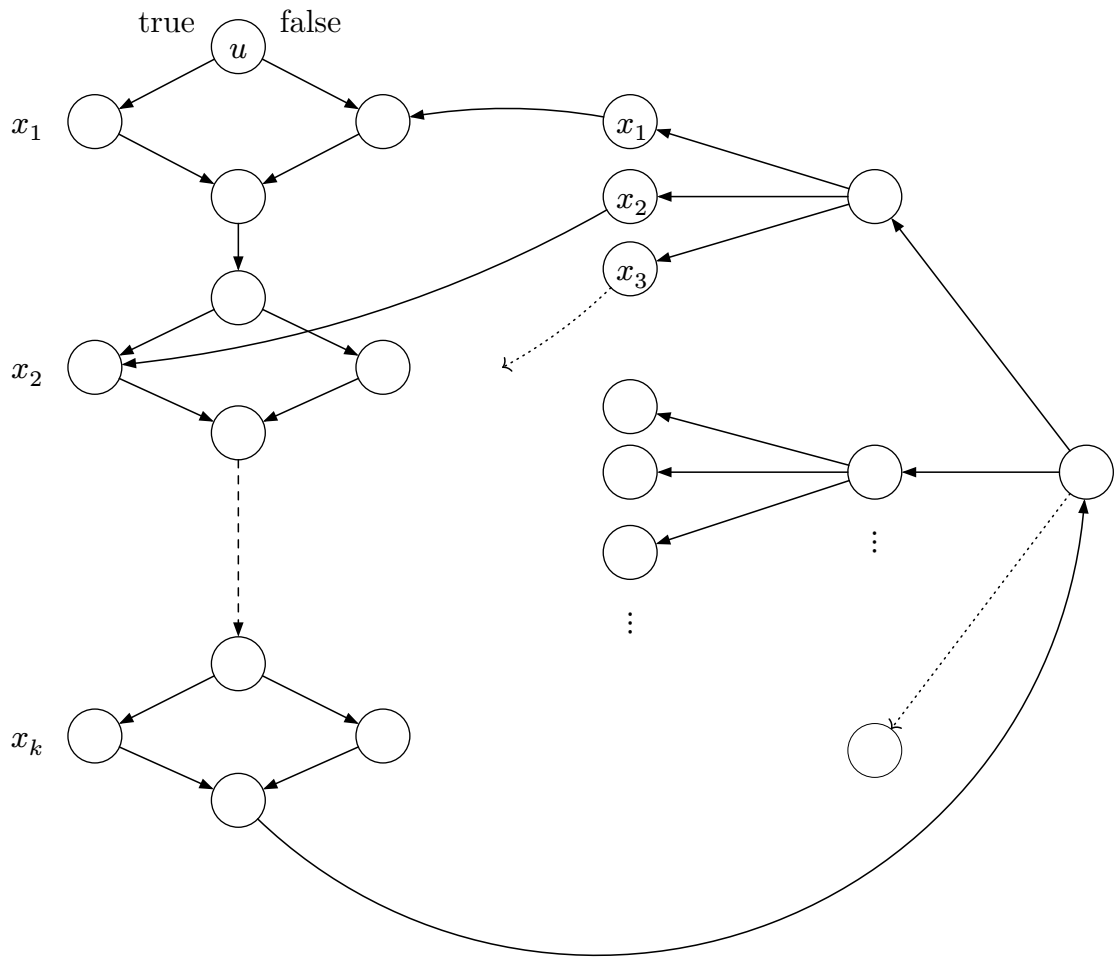
Az értékadásokat az alapján írjuk le, hogy éppen hogyan vannak kvantifikálva. Ha egzisztenciális kvantort (\exists) látunk, akkor Alízt kényszerítjük lépésre, ha univerzális kvantort (\forall), akkor Bobot kényszerítjük lépésre. Ha nem pontosan felváltva szerepelnek a kvantorok, akkor adunk az ellenfélnek egy triviális lépést ahol nincs választása csak előre menni.

Például, ha egymás után van $\forall x_i$ és $\forall x_{i+1}$, akkor egymás után kéne lépni kettőt Bobnak, de ezt a szabályok nem engedik. Ezért azt csináljuk hogy Bob lép egyet, azután beszáruunk egy választás nélküli direkt élt a következő választási lehetőséghez, ezzel kényszerítve Alízt és visszaadva Bobnak a lépés lehetőségét.

A második részben, ahol ellenőrizzük a formulát, úgy írjuk fel a gráfot hogy a gyűjtő csúcsba lépést Bobra kényszerítjük, így Alíz jön soron. Alíz rámutat egy blokk-ra ahol ő tudja hogy minden változó hamis. Így Bob bármelyik változót választja a blokkból az hamis lesz. Azt hogy egy változó hamis úgy mutatjuk meg, hogy miután Bob kiválasztotta, adunk Alíznek egy ingyen lépést és utána bekötjük a gráf első felébe oda ahol választottuk x_i értékét. Ha x_i tagadása szerepel a blokkban akkor abba a csúcsba kötjük be ami azt reprezentálja, hogy x_i hamis, különbe abba ami azt hogy x_i igaz.

Mostmár könnyű meggondolni, hogy ebben az irányított gráfban az, hogy Alíznek van nyerő stratégiája az ekvivalens azzal, hogy a tqbf igaz. Mivel az hogy Alíznek van nyerő stratégiája pont azt jelenti hogy amikor Alíz nyer akkor létezik (\exists) olyan lépés, hogy Bob bármit lép (\forall) még úgy is Alíz fog nyerni.

A következő ábra talán jobban elmagyarázza az érvelést.



□

3 Interaktív bizonyítások

Példa

Interaktív protokoll gráf nem izomorfizmusra.

Artúrnak vagy két gráfja G és G' melyekről el szeretné dönteni, hogy izomorfak-e. Artúr csak egy buta halandó ember, akkor is ha király, csak polinomiális idejű algoritmust tud lefuttatni a fejében. Szerencsére Merlin okosabb mint Artúr és a saját mágiájával bármit ki tud számolni a fejében egy lépés alatt, viszont nem feltétlenül mond mindig igazat.

Artúr a két gráf közül kiválaszt egy gráfot, permutálja a csúcsok számozását, és megkérdezi Merlintől, hogy melyik gráfot mutatja most éppen. Amire Merlin megmondja, hogy G vagy G' a mutatott gráf.

Ha a két gráf izomorf, akkor Merlinnek sincs esélye kitalálni melyik gráfot mutatja éppen Artúr, és így legfeljebb tippelhet.

Tehát Artúr megkérdezi Merlint a fent leírt módon 100-szor, hogy a jelenleg mutatott gráf melyik. Ha Merlin mindegyik alkalommal jól válaszolt akkor vagy nem izomorf a két gráf és így Merlinnek egyértelmű melyik mutatja Artúr, vagy végig tippelt és így $\frac{1}{2^{100}}$ eséllyel mindig pont jót mondott.

Azaz, a gráf nem izomorfizmusra a fent leírt protokoll egy interaktív bizonyítás.

Definíció 3.1

Interaktív protokoll

Azt szeretnénk eldönteni, hogy egy adott w szóra és L nyelvre $w \stackrel{?}{\in} L$.

A bizonyítást Merlin és Artúr együtt fogják végezni. Merlin bizonyít, míg Artúr ellenőrzi Merlin bizonyításait. Artúr egy randomizált Turing-gép, míg Merlin bármit ki tud számolni az input alapján egy lépés alatt.

Először Merlin szól és mond egy polinomiális hosszú üzenetet. Erre Artúr Merlin üzenete és w függvényében polinomiálisan sok véletlen számot felhasználva válaszol. Ezt az interakciót megismételi a két fél amíg Artúr el nem szánja magát és vagy elfogadja a w szót vagy elutasítja.

Azt mondjuk, hogy a protokoll elfogadja az L nyelvet, ha $w \in L$ esetén van olyan Merlin, hogy Artúr legalább $1 - \frac{1}{2^{|w|}}$ valószínűséggel elfogadja w -t, és $w \notin L$ esetén minden Merlin esetén Artúr legfeljebb $\frac{1}{2^{|w|}}$ valószínűséggel fogadja el w -t, azaz téved.

Definíció 3.2

IP osztály

$$\text{IP} := \{L : w \in L\text{-et interaktív protokollal lehet bizonyítani}\}$$

Példa

$\text{NP} \subseteq \text{IP}$

Proof of példa: $w \in L \in \text{NP}$, tehát létezik egy polinomiális tanu $w \in L$ -re. Ha pont ezt a polinomiális tanut válaszolja Merlin akkor az egy jó egy lépéses protokoll.

Tétel 3.3

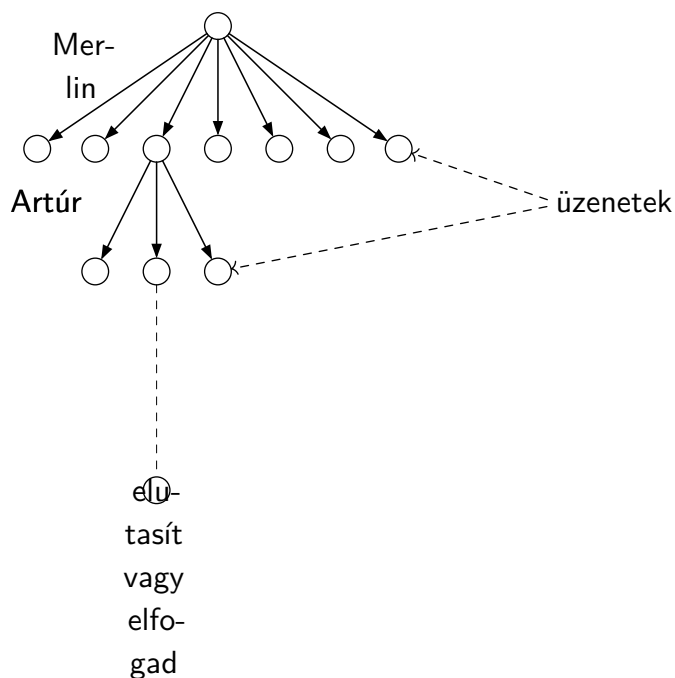
$$\text{IP} \subseteq \text{PSPACE}$$

A bizonyítás előtt bevezjük a következő segítő fogalmat.

Definíció 3.4

Protokoll fa

A protokoll fa egy fix L nyelvre és fix w szóra az összes lehetséges Artúr–Merlin interakciót ábrázolja egy faként a következő módon.



Megjegyzés

Habár egy szinten exponenciálisan sok csúcs lehet, ezt a gráfot be tudjuk járni polinomiális tárbán, mert polinomiális mélységű.

Proof of Tétel 3.3: Képzeljük el az interaktív bizonyítást protokoll fáját. Tudjuk, hogy minden levélben Artúr elfogad vagy elutasít, mert végetért a bizonyítás.

Az a célunk, hogy kiszámoljuk, hogy a gyökérben mekkora valószínűséggel fogadja el Artúr az adott bizonyítást a protokoll mentén. Ehhez az összes csúcsra kiszámoljuk, hogy onnan mekkora valószínűséggel fogadja el Artúr a bizonyítást.

A levelekben tudjuk, hogy 0 vagy 1 a valószínűség, mert ott már vagy elfogadta vagy elutasította.

Merlinnek az a célja, hogy maximalizálja az elfogadási valószínűséget, míg Artúr súlyozottan átlagolja valószínűségeket, mivel Artúr válaszai egy érmedobás után szólnak.

Ezért, ha Artúr válaszol akkor vegyük a súlyozott átlagát a lehetséges válaszoknak az elfogadási valószínűségét, ha viszont Merlin válaszol akkor a maximális elfogadási valószínűséget adjuk meg.

Bár egy csúcsnak lehet exponenciálisan sok gyereke, a protokoll fa mélysége polinomiális. Ezért rekurzívan ki tudjuk számolni a gyökér elfogadási valószínűségét a gyökerelektől kezdve polinomiálisan tárban.

□

Tétel 3.5

Shamir

$$\text{IP} = \text{PSPACE}$$

A bizonyításhoz először bevezetünk pár segédfogalmat és bizonyítunk valamit róluk. A tétel bizonyítása és segéd állítások az eredeti cikk alapján lettek feldolgozva, ami elérhető a következő linken: <https://dl.acm.org/doi/pdf/10.1145/146585.146609>

Definíció 3.6

Egyszerű tqbf

Azt mondjuk, hogy a φ teljesen kvantifikált Boole-formula *egyszerű*, ha minden x_i változójára igaz, hogy x_i kvantálásának helye és előfordulásának helye között legfeljebb egy darab univerzális kvantor (\forall) van.

Példa

A következő formula egy egyszerű tqbf:

$$\varphi = \forall x_1 \forall x_2 \exists x_3 [(x_1 \vee x_2) \wedge \forall x_3 (x_2 \wedge x_3 \wedge x_3)].$$

A következő formula nem egy egyszerű tqbf, mivel a kékkel jelölt x_1 változó kvantálása és második használata (negáltja) között kettő pirossal jelölt univerzális kvantor (\forall) is van.

$$\varphi = \forall x_1 \forall x_2 [(x_1 \wedge x_2) \wedge \forall x_3 (\bar{x}_1 \wedge x_3)]$$

Lemma 3.7

Minden teljesen kvantifikált Boole-formula egyszerű alakra hozható polinomiális időben.

Proof: Ha az eredeti tqbf nem egyszerű akkor van egy első x_i változó melynek a kvantálása és használata között több mint egy univerzális kvantor van. Ebben az esetben nézzük meg az első használatát x_i , mely már sérti a feltételt és cseréljük le x_i -t $\exists x_i^1 [(x_i \wedge x_i^1) \vee (\bar{x}_i \wedge \bar{x}_i^1)]$.

Tehát x_i kvantálása utáni első univerzális kvantor után létrehozunk egy új változót, x_i^1 , mely értéke pont x_i .

Ezeket a módosításokat addig csináljuk amíg egyszerű formulához nem jutunk .

□

Definíció 3.8

Formula aritmetizáltja

Egy teljesen kvantifikált Boole-formulához rendelt aritmetizáltja egy aritmetikai kifejezés, melyet úgy kapunk, hogy a formulában a következő előfordulásokat a megadott párjukra cseréljük le:

$$\begin{aligned} \text{True} &\mapsto 1, & \text{False} &\mapsto 0, & x &\mapsto x \\ x \vee y &\mapsto x + y, & x \wedge y &\mapsto x \cdot y, & \bar{x} &\mapsto (1 - x) \end{aligned}$$

A kvantorok viszont a következőképpen cseréljük le:

$$\forall x(\dots) \mapsto \prod_{x \in \{0,1\}} (\dots) \quad \exists x(\dots) \mapsto \sum_{x \in \{0,1\}} (\dots)$$

Megjegyzés

Az aritmetizált értéke egy egész szám.

Nyilván látszik, hogy a teljesen kvantifikált Boole-formula pontosan akkor igaz, ha az aritmetizáltja nem nulla.

Példa

$$\varphi = \forall x_1 \exists x_2 [(x_1 \wedge x_2) \vee \exists x_3 (\bar{x}_2 \wedge x_3)].$$

Ennek a tqbf-nek aritmetizáltja a következő:

$$f = \prod_{x_1 \in \{0,1\}} \sum_{x_2 \in \{0,1\}} \left[(x_1 \cdot x_2) + \sum_{x_3 \in \{0,1\}} (1 - x_2) \cdot x_3 \right].$$

Figyeljük meg, hogy φ aritmetizáltja nem feltétlenül egy polinomiális méretű szám, a következő példa mutatja hogy 2^{2^n} nagyságú is lehet:

$$\prod \prod \dots \prod (\dots) \geq 2^{2^n},$$

ahol a belső (...) kifejezés ≥ 2 és n darab produktum szerepel egymás mellett.

Állítás 3.9

Ha $0 < f < 2^{2^n}$, $f \in \mathbb{Z}$, akkor $\exists p$ prím úgy, hogy $2^n < p < 2^{2^n}$ és $f \not\equiv 0 \pmod{p}$.

Proof: Tegyük fel, hogy $f \neq 0$. Ha $f \equiv 0 \pmod{p_i}$ minden $2^n < p_i < 2^{2^n}$, akkor a Kínai maradék tétel miatt $f \equiv 0 \pmod{\prod p_i}$.

A prímszám tétel azt állítja hogy ha $\pi(x)$ jelöli a az x -ig terjedő prímszámok számát, akkor

$$\pi(x) \sim \frac{x}{\log(x)}.$$

A mi esetünkben 2^n és 2^{2^n} közé eső prímek száma a következő:

$$\frac{2^{2^n}}{\log(2^{2^n})} - \frac{2^n}{\log(2^n)} = \frac{2^{2^n}}{2n} - \frac{2^n}{n} \geq 2^n.$$

Ebből az következik, hogy a 2^n és 2^{2^n} közötti prímek szorzata legalább $(2^n)^{2^n} = 2^{n2^n} \geq 2^{2^n}$. Viszont $f \leq 2^{2^n}$, ezért nem lehet 0 modulo egy nagyobb szám mint 2^{2^n} . Ezzel ellentmondásra jutottunk és emiatt valóban létezik egy ilyen prím.

Ha viszont $f = 0$, akkor bármilyen prím jó, mert $f \equiv 0 \pmod{p}$ bármilyen p -re.

□

Definíció 3.10

Azt mondjuk, hogy f aritmetizálnak a funkcionális formája $f(x_1)$, amit úgy kapunk, hogy f -ben szereplő első Π_{x_1} avagy Σ_{x_i} jelet eltörlünk és így már a kifejezés függ x_1 értékétől és egy függvényt kapunk.

Példa

Tekintsük a következő igaz tqbf-et:

$$\varphi = \forall x_i [\bar{x}_1 \vee \exists x_2 \forall x_3 (x_1 \wedge x_2) \vee x_3].$$

Ennek a tqbf-nek az aritmetizáltja a következő:

$$f = \prod_{x_1 \in \{0,1\}} \left[(1 - x_1) + \sum_{x_2 \in \{0,1\}} \prod_{x_3 \in \{0,1\}} (x_1 \cdot x_2 + x_3) \right],$$

melynek értéke 2. Ezen aritmetizált funkcionális formája a következő:

$$f(x_1) = \left[(1 - x_1) + \sum_{x_2 \in \{0,1\}} \prod_{x_3 \in \{0,1\}} (x_1 \cdot x_2 + x_3) \right].$$

Egy mindenható Merlin persze egyből megmondja, hogy $f(x_1) = x_1^2 + 1$.

Egy tqbf aritmetizáltja exponenciálisan nagy fokszerű polinom is lehet, például

$$\varphi = \forall x_1 \forall x_2 \dots \forall x_n (x_1 \vee x_2 \vee \dots \vee x_n)$$

aritmetizáltja

$$f = \prod_{x_1 \in \{0,1\}} \prod_{x_2 \in \{0,1\}} \dots \prod_{x_n \in \{0,1\}} (x_1 + x_2 + \dots x_n),$$

melynek funkcionális formájában 2^{n-1} -ed fokon fog szerepelni $(x_1 + c)$ tag. Nyilván ilyen komplikált polinomokat Artúr nem tud kezelni.

Állítás 3.11

Ha φ egy egyszerű tqbf, akkor az aritmetizáltjának a funkcionális formája $f(x_1)$ egy polinom melynek foka legfeljebb lineáris φ méretében.

Proof: Mivel φ egy egyszerű tqbf, ezért x_1 kvantálása és használata között legfeljebb egy univerzális kvantor lehet. A polinom fokát csak a szorzások befolyásolják és ebből legfeljebb egy lehet x_1 előtt ami duplázza x_1 fokát.

□

Proof of $IP = PSPACE$: Azt bizonyítjuk hogy interaktív protokollt tudunk adni a TQBF nyelv felismerésére. Az előbb láttuk hogy minden tqbf hozható egyszerű tqbf alakra, továbbá azt is hogy egy tqbf pontosan akkor igaz, ha aritmetizáltja nem nulla. Azt is láttuk, hogy ha $f \neq 0$, akkor van egy intervallumon egy prím amire $f \not\equiv 0 \pmod{p}$.

Tehát a következőben csak arra adunk interaktív protokollt, hogy egy egyszerű tqbf aritmetizáltja $f \not\equiv 0 \pmod{p}$.

Interaktív protokoll

A protokoll azt fogja bizonyítani, hogy $f \not\equiv 0 \pmod{p}$, ahol f már φ -nek az aritmetizáltja.

1. Először Merlin elküldi $f \pmod{p}$ értékét Artúrnak és a funkcionális formáját egyszerű polinom alakra hozva $f(x_1)$.
2. Ha Merlin egy \prod törlésével kapta a függvényt, akkor Artúr ellenőrzi, hogy $f(0) \cdot f(1) \equiv f \pmod{p}$, ha viszont \sum törlésével kapta akkor azt ellenőrzi, hogy $f(0) + f(1) \equiv f \pmod{p}$.

Miután Artúr ellenőrizte f -et és $f(x)$ -et, véletlenül választ $\xi \in \{0, 1, \dots, p-1\}$ számot és behelyettesíti ξ -t és megkapja az $f(\xi)$ kifejezést. Merlinnek elküldi $f(\xi)$ -t miután leszedte az első kvantort.

3. Erre Merlinnek meg kell adnia egyszerű polinom alakra hozva a funkcionális formáját.
4. Artúr ellenőrzi, hogy visszakapja-e $f(\xi)$ -t a 0 – 1 behelyettesítéssel és összeggel/szorozattal, a letörölt kvantortól függően.
5. A protokoll így folytatódik tovább míg ki nem ürül a kifejezés.

Ha Merlin becsületesen játszik, akkor Artúr mindig elfogad.

Ha viszont Merlin csal, akkor csak ott van értelme csálnia hogy f értékéről hazudik. Ekkor viszont $f(x)$ polinomot is meg kell hamisítani vagy különben egyből lebukna Artúr egyszerű ellenőrzésével. Tehát Merlin f helyett f' -t mond.

Feltéve hogy $f \not\equiv f' \pmod{p}$, akkor

$$\mathbb{P}[(f - f')(\xi) = 0] \approx \frac{1}{2^n},$$

azaz annak a valószínűsége, hogy a két polinom értéke pont megegyezik ξ -ben exponenciálisan kicsi. Mivel $p > 2^n$ különböző szám lehet ξ és $f - f'$ fokszáma lineáris n -ben.

Következik, hogy annak a valószínűsége, hogy a sok csalás után megússza Merlin $\frac{1}{2^n}$.

□

3.1 Zero Knowledge Proofs

Definíció 3.12

Zero Knowledge Proof

Egy zero knowledge proof (ZK) két szereplő interakciója: Merlin (a bizonyító) és Artúr (az ellenőrző).

Egy ZK-nak a következő három tulajdonságot kell teljesítenie:

1. Ha a bizonyítandó állítás igaz, akkor egy őszinte Artúrt meg tudja győzni egy őszinte Merlin.
2. Ha az állítás hamis, akkor még csalással sem tudja Merlin meggyőzni Artúrt, legfeljebb egy kis valószínűséggel téved Artúr.
3. Ha az állítás igaz, akkor Artúr nem tud meg semmi extra információt azon kívül hogy igaz az állítás Merlin bizonyításából.

Megjegyzés

Az első két feltétel tehát csak annyit jelent, hogy ez egy interaktív bizonyítás. A lényeg pont az hogy Artúr nem tud meg semmi mást azon kívül hogy igaz az állítás.

Példa

Gráf nem izomorfizmus

Adott két gráf G_1, G_2 és el szeretnénk dönteni, hogy $G_1 \neq G_2$.

Artúr újra címezi a csúcsokat és megmutatja az egyik gráfot. Erre Merlinnek meg kell mondania, hogy G_1 -et vagy G_2 -t mutatta fel Artúr.

Példa

Hamilton kör létezése

Adott egy G gráf, mely ismert Merlin és Artúr számára is. Merlinnek be kell bizonyítania, hogy ő ismer egy Hamilton-kört G -ben.

1. Merlin újra címezi G csúcsait egy permutációval, ezzel létrehoz egy H gráfot, mely persze izomorf G -vel.
2. Fölírja H éleit egy-egy lapra és fejjel lefelé elhelyezi őket az asztalra és mostmár nem nyúlhat hozzájuk addig amíg Artúr meg nem engedi.
3. Most Artúr a következő kettő eset közül választ:
 - Merlin, bizonyítsd be, hogy H valóban izomorf G -vel.
 - Merlin mutasd meg a Hamilton-körödet H -ban.
4. • Ha azt kell bizonyítania Merlinnek, hogy $G \simeq H$, akkor felfordítja az összes lapot és elmondja a permutációt amivel újra címkézte a csúcsokat. Ezzel Artúr meg tud győződni, hogy valóban izomorf a két gráf.
 - Ha viszont H -ban kell mutatnia Merlinnek egy Hamilton-kört, akkor felfedi csak azokat az éleket amik Hamilton-kört alkotnak.

Proof: Az előző bizonyítás valóban egy zero knowledge proof, mivel ha Merlin ismert egy Hamilton-kört G -ben, akkor erről meggyőződött Artúr. Ha Merlin nem ismert egy Hamilton-kört G -ben, akkor még Artúr kérdése előtt el kellett döntenie, hogy vagy egy valid izomorf gráfot

kreál vagy egy valid Hamilton kört, ami nem izomorf G -vel. Mivel Merlin nem tudja előre Artúr kérdését ezért mindig $1/2$ eséllyel lebukik.

Másrésről, Artúr nem tudott meg semmi többet a Hamilton körről mint, hogy létezik, mivel akkor is amikor Merlin mutat neki egy Hamilton-kört akkor is csak H -ban mutat fel n élet ami n csúcsot tartalmaz, de azt még a halandó Artúr is tudja, hogy egy Hamilton kör n élből és n csúcsból áll.

□

4 Véletlen bonyolultság osztályok

Volt, hogy $L \in \mathbf{NP}$ pontosan akkor, ha létezik polinomiális tanu minden $w \in L$ -re.

Definíció 4.1

RP

$L \in \mathbf{RP} \iff \exists T$ Turing gép amely $w \in L$ esetén legalább $1/2$ eséllyel elfogad és $w \notin L$ esetén 1 valószínűséggel elutasít.

Ezt mind várhatóan polinomiális időben.

Megjegyzés

A következő táblázat ilusztrálja az esetek valószínűségét:

Válasz	Igaz	Hamis
Valóság		
Igaz	$\geq 1/2$	$\leq 1/2$
Hamis	0	1

Megjegyzés

Ha $L \in \mathbf{RP}$, akkor feltehető, hogy polinomiális ideig működik az őt elfogadó Turing-gép.

Proof of Megjegyzés: RP definíciójából a várható lépésszám polinomiális, tehát $\mathbb{E}(t(w)) \leq |w|^c$ valamilyen $c > 0$ -ra.

A Markov egyenlőtlenség alapján:

$$\mathbb{P}(t(w) \geq 8|w|^c) < \frac{1}{8}.$$

Kezdjük el futtatni a T Turing-gépet mely a definícióban szerepel. Ha T leáll $8 \cdot |w|^c$ lépésen belül, akkor azt válaszoljuk amit T . Ha viszont több mint $8 \cdot |w|^c$ lépésig fut, akkor elutasítunk.

Figyeljük meg, hogy ha $w \notin L$, akkor nem tudunk tévedni, mivel vagy leáll az adott idő alatt és elutasítunk, vagy nem áll le és akkor is elutasítunk.

Ha viszont $w \in L$, akkor bajban vagyunk, ezért kétszer futtatjuk T -t az előző feltételekkel és csak akkor utasítjuk el w -t, ha mindkét alkalommal elutasítanak. Annak a valószínűsége, hogy elfogadjuk w -t a fenti logika szerint legfeljebb $\frac{1}{2} - \frac{1}{8} = \frac{3}{8}$, mert ha kevesebb mint $8|w|^c$ ideig fut, akkor $\frac{1}{2}$ eséllyel elfogadjuk, ha tovább fut akkor elutasítjuk, aminek az esélye legfeljebb $\frac{1}{8}$.

Tehát annak az esélye, hogy egyszer futtatva elfogadjuk $1 - \frac{3}{8} = \frac{5}{8}$. Ha kétszer futtatjuk, akkor

$$\left(\frac{5}{8}\right)^2 < \frac{1}{2}.$$

□

Megjegyzés

Az a sejtés, hogy $\mathbf{RP} = \mathbf{P}$. Ezt még nem sikerült bizonyítani, de nem sokan lepődnének meg ha valóban így lenne.

Megjegyzés

Érthetően $\mathbf{P} \subseteq \mathbf{RP} \subseteq \mathbf{NP}$.

Definíció 4.2

\mathbf{P}/f

\mathbf{P}/f azon nyelveket jelöli, melyekre létezik egy Turing-gép, ami elfogadja a (w, a) párt polinomiális időben, ahol a egy legfeljebb $f(|w|)$ méretű szó amit megsúgunk és csak az input hosszától függ.

Megjegyzés

Az előző definícióban fontos, hogy a sugás kizárólag az input hosszától függ, tehát minden k hosszú inputra ugyanazt a sugást adjuk meg a Turing-gépnek.

Megjegyzés

$\mathbf{P}/1 \stackrel{?}{=} \mathbf{P}$, ahol $\mathbf{P}/1$ azt jelenti, hogy mindegyik input hosszhhoz kizárólag egy betűt/bitet súghatunk.

A válasz negatív, sőt $\mathbf{P}/1$ -ben nem is mindegyik nyelv rekurzív, nem hogy polinomiális időben felismerhető nyelvek.

A bizonyításhoz tekintsünk egy tetszőleges nem rekurzív nyelvet a $\{0, 1\}$ ábécé fölött, ilyen nyilván van mert majdnem mindegyik nyelv nem rekurzív. Mivel megszámlálhatóan sok Turing-gép van és mindegyik Turing-gép legfeljebb egy nyelvet ismer fel és megszámlálhatatlanul sok nyelv van adott ábécé fölött, ezért majdnem minden nyelv nem rekurzív.

A $\{0, 1\}$ ábécé fölötti L nyelvet felfoghatjuk úgy mint a természetes számok egy részhalmazát $L \subseteq \mathbb{N}$, ahol egy szó egy természetes szám bináris alakja. Eltekintve attól hogy mit kezdünk a kezdő nullásokkal...

Legyen K az a nyelv, amiben a $\overbrace{11111 \dots 11111}^x$ szavak vannak, ahol $x \in L$. Tehát K egy unáris nyelv ahol pontosan az x hosszú szavak vannak, ahol $x \in L$.

Ez a nyelv nyilván nem rekurzív, mert L sem az. Viszont a hossza az inputnak egyértelműen meghatározza x -et, így az egy bites sugás $x \in L$ kérdésre a válasz.

Definíció 4.3

\mathbf{P}/poly

$$\mathbf{P}/\text{poly} = \bigcup_{k=1}^{\infty} \mathbf{P}/n^k$$

$$\mathbf{P}/\log = \bigcup_{k=1}^{\infty} \mathbf{P}/k \cdot \log n$$

$$L^=n = L \cap \Sigma_0^n$$

Tehát L -nek a pontosan n hosszú szavai.

$$\mathbf{RP} \subseteq \mathbf{P}/\text{poly}$$

Proof: Legyen $L \in \mathbf{RP}$, azt kell belátnunk, hogy $L \in \mathbf{P}/\text{poly}$. Legyen továbbá $n = |w|$ és tegyük fel, hogy $\Sigma_0 = \{0, 1\}$.

Ugye azt kell belátnunk, hogy van polinomiális hosszú sűgás, amivel már el tudja dönteni egy Turing-gép egy szó tagságát, legyen ez a polinom p . Így feltehetjük, hogy L -et (\mathbf{RP} értelemben) felismerő T Turing-gép egy $p(n)$ hosszú véletlen bitsorozatot használ.

Tehát egy $r \in \Sigma_0^{p(n)}$ véletlen bitsorozat meghatározza T működését.

Azt mondjuk, hogy S jó $L^=n$ -re, ha $\forall w \in L^=n$ -re $\exists r \in S$ úgy, hogy T elfogadja a (w, r) párost. Szavakban annyit jelent, hogy S bitsorozatok halmaza, melyben van olyan bitsorozat amellyel T elfogadja w -t.

Nyilván $S = \Sigma_0^{p(n)}$ jó, mivel ebben benne van az összes $p(n)$ hosszú bitsorozat. A továbbiakban ezt a halmazt szeretnénk szűkíteni.

Tudjuk, hogy adott $w \in L^=n$ -re annak a valószínűsége, hogy T elutasítja w -t véletlen r -el az legfeljebb $1/2$.

Ha S pontosan k darab véletlen $p(n)$ hosszú bitsorozatból áll, akkor annak a valószínűsége, hogy T elutasítja w -t minden $r \in S$ bitsorozatra az legfeljebb $1/2^k$.

Így annak a valószínűsége, hogy $\exists w \in L^=n$, hogy T elutasítja w -t minden $r \in S$ bitsorozatra, az

$$< \frac{|L^=n|}{2^k} \leq \frac{2^n}{2^k}.$$

Ha $k = n + 1$, akkor az előző legfeljebb $1/2$. Mivel ez a valószínűség kisebb 1-nél ezért kell léteznie egy olyan bitsorozatnak a k közül ami jó az összes w -re.

Tehát adott n hosszú w inputra sűgni fogunk egy k bitsorozatból álló S halmazt, amit képzelhetünk úgy mint k darab $p(n)$ bitsorozat konkatenálva, mert az osztály definíciójában csak egy bitsorozatot sűghatunk. Látjuk, hogy a sűgás mérete $k \cdot p(n)$, ami polinomiális n -ben.

Végül lefuttatjuk mind a k darab $p(n)$ hosszú bitsorozatra T -t.

□

Definíció 4.7

Ritka nyelv

Azt mondjuk, hogy L ritka, ha $\exists c > 0$ úgy, hogy $\forall n \ |L^n| \leq n^c$.

Tétel 4.8

Mahaney

Ha létezik egy ritka L nyelv ami **NP**-teljes, akkor $\mathbf{P} = \mathbf{NP}$.

Definíció 4.9

Unáris nyelv

Azt mondjuk, hogy az L nyelv unáris, ha $L \subseteq \{1\}^*$, azaz olyan szavakból áll melyekben csak 1-es szerepel. (pl.: 1111 vagy 11)

Tétel 4.10

Berman

Ha létezik egy unáris L nyelv ami **NP**-teljes, akkor $\mathbf{P} = \mathbf{NP}$.

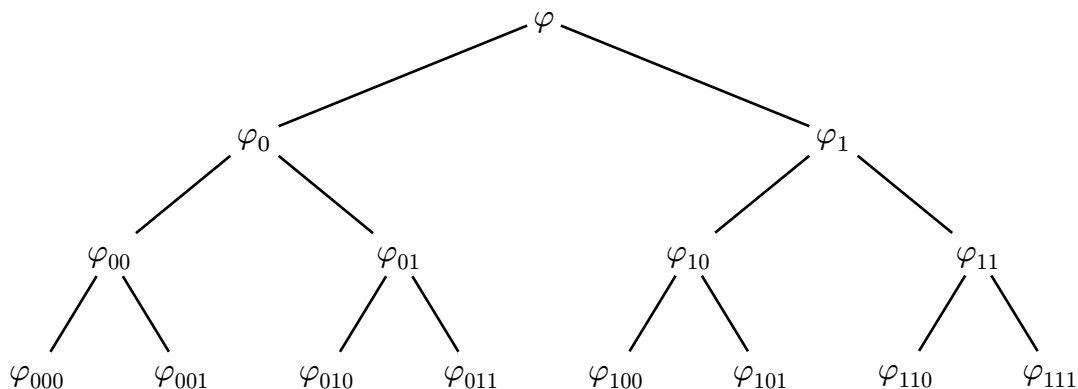
Proof: Mivel $L \in \mathbf{NPC}$ ezért a SAT nyelv visszavezethető L -re.

Legyen φ egy CNF, mivel $\text{SAT} \propto L$, ezért $\exists f : \varphi \mapsto f(\varphi)$ leképezés.

Játszuk azt a játékot, hogy minden lépésben az egyik változóját a φ CNF-nek lefixáljuk, például az első lépésben lefixáljuk az első változóját φ -nek úgy, hogy $\varphi_{x_1=1} = \varphi_1$ avagy $\varphi_{x_1=0} = \varphi_0$. Hasonlóképpen a következő szinteken a következő változókat fixáljuk.

Ezzel a játékkal a következő fát alkotjuk. Itt minden levélben már csak egy 0 vagy 1 érték szerepel. Ennek a fának a mélysége n ha n változós a φ CNF.

Figyelem, hogy lehet egy közbülső csúcsnak is már fix az értéke, például ha már lefixáltunk egy-egy változót mindegyik klózban igazra.



Érthetően $\varphi \in \text{SAT}$, ha van igaz értékadása, azaz a fenti fában van olyan levél melynek értéke 1.

Azt mondom, hogy egy csúcs kielégíthető a CNF fában, ha van 1-es csúcs az alatta lévő részfában.

□

Lemma 4.11

Ha mélységi bejárást (DFS) végzek a CNF fában, azaz addig megyek ameddig nem találok 1-es vagy bejárom az egész fát, akkor megállásig vagy $\leq n$ darab kielégíthető csúcson mentem át vagy egyen sem.

Proof: Ha bejártam az egész fát akkor nem találtam egy 1-es csúcsot sem, tehát egyik csúcs sem kielégíthető az egész fában tehát nem járhattam be egyet sem.

Ha viszont találtam egy 1-es csúcsot akkor az 1-es csúcs szülein végigmentem, ezek mind kielégíthetők. Mivel DFS-t végzek ezért nem is mentem át másik kielégíthető csúcson, mert akkor lementem volna egészen addig a csúcsig alatta ami 1-es.

□

Proof of Berman folytatása: Minden φ CNF-hez hozzárendelek egy $f(\varphi)$ szint. A színeim két csoportba fognak tartozni: $f(\varphi) \in L$ vagy $f(\varphi) \notin L$.

Miközben járom be a CNF-fát DFS-el, számolom a $f(\varphi)$ színeket. Ha egy szint már $(n + 1)$ -szer láttam, akkor az előző lemmából tudom, hogy ott már nem lehet 1-es csúcs alatta, ezért nem megyek tovább.

Mivel polinomiálisan sok szín van, mert a visszavezetés polinomiális, és mindegyik szint legfeljebb n -szer látogatok meg, ezért legfeljebb $n \cdot \text{poly}(n)$ csúcson mentem át.

Ennyi csúcs látogatása után már meg agyok győződve, hogy $\varphi \in \text{SAT}$ vagy sem. Tehát adtam egy polinomiális algoritmust SAT-ra, tehát $\mathbf{P} = \mathbf{NP}$.

□

Tétel 4.12

Fortune

Ha létezik egy ritka A nyelv és $\overline{\text{SAT}} \propto A$, akkor $\mathbf{P} = \mathbf{NP}$.

Itt a $\overline{\text{SAT}}$ azt a nyelvet jelöli amiben azon konjunktív normál formák vannak, melyeket nem lehet kielégíteni.

Proof: Mivel A ritka, ezért $|A^n| \leq n^c$, ekkor ugyanazt az érvelést követve, mint a Tétel 4.10 bizonyításában, ha nem találtunk kielégítő értékadást a fában DFS-el, akkor legfeljebb $n \cdot n^c$ lépést tettünk. Ha viszont találtunk kielégítő értékadást, akkor még gyorsabban megtaláltuk mint az előző esetben.

□

Definíció 4.13

Azt mondjuk, hogy a B nyelv $P/c \log n$ időben visszavezethető az A nyelvre, ha van olyan Turing-gép, mely $c \log n$ méretű súgással $|w|$ -ben polinomiális időben ki tudja számolni $f(w)$, amelyre $w \in B \iff f(w) \in A$.

Jelölésben $B \underset{P/c \log n}{\propto} A$.

Következmény 4.14

Ha létezik egy ritka A nyelv és $\overline{\text{SAT}} \underset{P/c \log n}{\propto} A$, akkor $\mathbf{P} = \mathbf{NP}$.

Proof: A T Turing-gép tippelje meg az $a_{|w|}$ sugást, ha a T gépg több mint $(n+1)|A|^n$ lépést tesz meg akkor leállítjuk és tippelünk egy új sugást.

□

Definíció 4.15

$\mathbf{NP}/f(n)$ azon L nyelvek osztályát jelöli, melyekre van egy olyan T nem determinisztikus Turing-gép, mely $w \in L$ -et egy $f(|w|)$ méretű sugással elfogadja.

Lemma 4.16

Ha létezik egy ritka A nyelv és $A \in \mathbf{NP}$, akkor $\overline{A} \in \mathbf{NP}/c \log n$.

Proof: Legyen a w inputra a $|w|$ hosszú szavak száma A -ban a sugás, azaz $l = |A^{|w|}|$ és ezt az l -et sugjuk, ami $c \cdot \log n$ méretű, mert A ritka.

A T nem determinisztikus Turing-gép (nem determinisztikusan) legenerálja az összes $|w|$ hosszú szót és mindegyikre leellenőrzi, hogy eleme-e A -nak, így megkapjuk $A^{|w|}$ -t.

Ezután végigmegy $A^{|w|}$ összes w_i elemén és leellenőrzi, hogy $w_i \stackrel{?}{=} w$. Ha van olyan i , melyre $w_i = w$, akkor $w \in A \Rightarrow w \notin \overline{A}$, különben $w \in \overline{A}$. Ezzel így akkor felismertük \overline{A} -t a sugással.

□

Proof of Mahaney: Ha $A \in \mathbf{NPC}$ és A ritka, akkor $\text{SAT} \propto A$ és $\overline{\text{SAT}} \propto \overline{A}$, sőt ez a két visszavezetés ugyanazzal az f -el megy.

A Lemma 4.16 miatt, ha A ritka és $A \in \mathbf{NP}$, akkor $\overline{A} \in \mathbf{NP}/c \log n$. Tehát $\overline{\text{SAT}} \propto \overline{A} \in \mathbf{NP}/c \log n$, ami miatt $\overline{\text{SAT}} \in \mathbf{NP}/c \log n$.

Mivel $A \in \mathbf{NPC}$, ezért $\overline{\text{SAT}} \underset{P/c \log n}{\propto} A$ és így a Következmény 4.14 miatt $\mathbf{P} = \mathbf{NP}$.

□

Definíció 4.17

BPP

BPP azon L nyelvek osztályát jelöli, melyekre létezik egy randomizált T Turing-gépt, hogy $w \in L$ esetén $> \frac{2}{3}$ eséllyel elfogad és $w \notin L$ esetén $< \frac{1}{3}$ eséllyel fogad el. És a futásidő várhatóan polinomiális.

A nevét a *Bounded error Probabilistic Polynomial* kifejezésből kapja.

Megjegyzés

A következő táblázat ilusztrálja az esetek valószínűségét:

Válasz	Igaz	Hamis
Valóság		
Igaz	$\geq 2/3$	$\leq 1/3$
Hamis	$\leq 1/3$	$\geq 2/3$

Megjegyzés

A BPP definíciójában is hasonlóan le lehet cserélni a várhatóan polinomiális időt konkrét polinomiális időre a Megjegyzés 4.2 érvelésével.

Megjegyzés

A sejtés az, hogy $\mathbf{P} = \mathbf{BPP}$.

Megjegyzés

Nyilván $\mathbf{P} \subseteq \mathbf{RP} \subseteq \mathbf{BPP}$.

Definíció 4.18

ZPP

$$\mathbf{ZPP} = \mathbf{RP} \cap \mathbf{coRP}$$

Megjegyzés

A fenti definíció ekvivalens azzal, hogy olyan L nyelvek vannak \mathbf{ZPP} -ben, melyekre van olyan T probabilisztikus Turing-gép, mely hiba nélkül felismeri L -et. És ezt mind várhatóan polinomiális időben teszi meg.

Az osztály a nevét a Zero (error) Probabilistic Polynomial kifejezésből kapja.

Megjegyzés

A tanult Schwartz-Zippel lemma érvén azt kapjuk, hogy a polinom különbség feladat \mathbf{coRP} -beli. Azaz, azon L nyelv, melyben (p, q) polinom párosok vannak, ahol $p \neq q$.

Állítás 4.19

Ha a T probabilisztikus Turing-gép azonosan egyenletes eloszlású független biteket használ, akkor ekvivalens definíciókat kapunk az előzőkkel.

Proof: Nem bizonyítjuk.

□

$$\mathbf{BPP} \in \Sigma_2 \cap \Pi_2$$

Proof: Legyen $L \in \mathbf{BPP}$, ekkor tudjuk, hogy létezik egy T probabilsztikus TG ami $\frac{1}{2^n}$ valószínűséggel vét csak hibát $x \in L$ eldöntésekor.

Tehát T véletlen biteket használ. Legyen S_x azon véletlen bitek halmaza, melyre ha $r \in S_x$ és $x \in L$ akkor $T(x, r)$ elfogad.

Mivel a hiba korlátolt ennél a TG-nél, ezért $|S_x| \geq (1 - 2^{-n})2^m$, mert a hiba valószínűsége $|S_x|/2^m$, mert m hosszú véletlen biteket használhat a TG.

Legyen \mathbb{F}_2^m az m bites számok halmaza.

Állítás 4.21 -et alkalmazva látszik, hogy

$$x \in L \iff \exists r_1, r_2, \dots, r_k \in \mathbb{F}_2^m : \forall z \in \mathbb{F}_2^m \text{-re} \\ T \text{ elfogadja } x\text{-et legalább egy } z + r_i\text{-re}$$

ami pont a definíciója annak, hogy $L \in \Sigma_2$.

Tehát beláttuk, hogy $\mathbf{BPP} \subseteq \Sigma_2$. Továbbá, mivel $\mathbf{BPP} = \mathbf{coBPP}$ és $\mathbf{co}\Sigma_2 = \Pi_2$ így

$$\mathbf{coBPP} \subseteq \mathbf{co}\Sigma_2 \implies \mathbf{BPP} \subseteq \Pi_2.$$

□

Állítás 4.21

Ha $|S_x| \geq (1 - 2^{-n})2^m$ akkor $\exists r_1, \dots, r_k \in \mathbb{F}_2^m$, ahol $k = \lceil \frac{m}{n} \rceil + 1$, úgy hogy

$$\bigcup_{i=1}^k (S_x + r_i) = \mathbb{F}_2^m.$$

Proof: Célunk megmutatni, hogy

$$\mathbb{P} \left[\bigcup_{i=1}^k (S_x + r_i) \neq \mathbb{F}_2^m \right] < \frac{1}{2}.$$

Legyen $z \in \mathbb{F}_2^m$. Ekkor $z + r$ is véletlen vektor, mert egy véletlen vektor eltoltja is véletlen. Így

$$\mathbb{P}_r[z \notin S_x + r] = \mathbb{P}_r[z + r \notin S_x] \leq 2^{-n}.$$

Ha veszünk k darab független véletlen $r_1, \dots, r_k \in \mathbb{F}_2^m$ vektort, akkor

$$\mathbb{P} \left[z \notin \bigcup_{i=1}^k (S_x + r_i) \right] \leq (2^{-n})^k = 2^{-nk}.$$

Továbbá

$$\mathbb{P} \left[\exists z \notin \bigcup_{i=1}^k (S_x + r_i) \right] \leq 2^{-nk} \cdot 2^m \leq \frac{1}{2},$$

ahol az utolsó egyenlőtlenséget pont $k = \lceil \frac{m}{n} \rceil + 1$ választásával kapjuk. Tehát

$$\mathbb{P} \left[\forall z \in \bigcup_{i=1}^k (S_x + r_i) \right] \geq \frac{1}{2},$$

ami azt jelenti, hogy tudunk választani r_1, \dots, r_k véletlen vektorokat, hogy S_x -et ezekkel eltolva már fedjük \mathbb{F}_2^m -et.

□

5 Boole-hálózatok

Definíció 5.1

Boole-hálózat

A \mathcal{C} egy Boole-hálózat, ha egy irányított körmentes gráf (DAG), aminek csúcsai címkézettek. Továbbá, a következő speciális csúcsokkal rendelkezik:

- input csúcsok: $u : d_{\text{in}}(u) = 0$
- output csúcsok: $u : d_{\text{out}}(u) = 0$
- belső csúcsok: minden más.

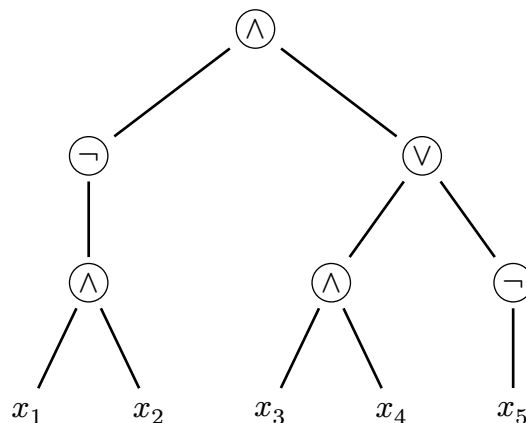
A belső csúcsok mindegyikén adott egy Boole függvény, melynek változói a beéleken keresztül kapott csúcsok értékei.

A \mathcal{C} Boole-hálózat inputjaira írunk biteket és az kiszámolja a belső csúcsokon lévő Boole-függvények segítségével az értékeket, addig ameddig el nem érnek az értékek az output csúcsokba. Az input csúcsok értéke az input, az output csúcsok értéke az output.

Megjegyzés

A következő ábra illusztrálja, hogy hogyan képzelhetünk el egy Boole-hálót. Figyelem, az ábrázolt Boole-hálózat egy fa, viszont lehetnek kereszt élek is, azaz irányítatlan körök, csak irányított körök tilosak.

A legalján vannak az inputok, a legtetején az output. A legalsó szinttől egyesével felfele számoljuk ki a szintek értékeit az előző szint segítségével.



Megjegyzés

Angolul Boolean-circuit-nek hívják a Boole-hálózatokat. Létezik olyan is, hogy Boolean-network de az teljesen más.

Definíció 5.2

A \mathcal{C} Boole-háló mérete a csúcsainak száma.

Definíció 5.3

A \mathcal{C} Boole-háló mélysége a leghosszabb $s-t$ út hossza, ahol s egy input csúcs és t egy output csúcs.

Megjegyzés

Minden $f : \{0, 1\}^n \rightarrow \{0, 1\}$ Boole függvényt ki lehet számolni 3 mély, exponenciális méretű, Boole-hálózattal, ami csak a \wedge, \vee, \neg függvényeket alkalmazza a belső csúcsokon.

Proof: Ismert, hogy minden Boole-függvény felírható konjunktív normál forma alakba. Tehát, a Boole-hálózat legalsó szintjén az inputokat csoportosítjuk klózokba \vee jelekkel, ahol kell beillesztünk egy \neg jelet, ha $\neg x_i$ szerepel a klózban. Az alkotott klózokat meg behúzzuk az egyetlen output csúcsba, amiben egy \wedge jel szerepel, tehát össze vagyoljuk a klózokat.

□

Tétel 5.4

Adott $\{0, 1\}$ fölötti L nyelvre $L \in P/\text{poly} \iff \exists \mathcal{C}_1, \dots, \mathcal{C}_n$ Boole-hálók, hogy \mathcal{C}_i az i inputon működik és $\exists c > 0 : \mathcal{C}_i$ mérete legfeljebb i^c és $w \in L$ -et eldönti a $\mathcal{C}_{|w|}$ háló.

Proof:

\Leftarrow : $\forall n : |w| = n$ a sűgás legyen C_n . Mivel C_n mérete polinomiál ezért C_n leírása is polinomiális és tudunk alkotni egy Turing-gépet, amely polinomiális futásidő alatt tudja szimulálni C_n Boole-háló működését.

\Rightarrow : Ha $L \in P$, akkor létezik olyan T Turing-gép, ami L -et eldönti polinomiális időben. Feltehető, hogy T egy egyszalagos Turing-gép, mert tudjuk, hogy minden Turing-gép szimulálható egy egyszalagossal, persze négyzetese lassítással, de ez még belefér a polinomiális futásidőbe.

Csináljunk úgy, mintha a T egyetlen szalagjára írt input olyan mint C_n

Tehát minden mezőre egy lépésről a másikkra egy $O(1)$ bemenetű Boole-függvény dönt el az új állapotot.

Mivel T polinomiálisan hosszú ideig fut, ezért legfeljebb polinomiális mélységű az így kapott háló.

Ha $L \in P/\text{poly}$, akkor az első lépésben a szalag úgy néz ki, hogy van a szokásos input a szalagon balról jobbra, és ettől az inputtól balra odaírjuk még a polinomiális sűgást.

□

Következmény 5.5

Ha kiderülne, hogy G3C-re nincs öt felismerő polinomiális méretű Boole-háló, akkor $P \neq NP$. Mivel így $P/\text{poly} \neq NP$ és $P \subseteq P/\text{poly}$.

Definíció 5.6

$$\text{PARITY}(x_1, x_2, \dots, x_n) = \sum_{i=1}^n x_i \pmod{2}$$

$$\text{MAJORITY}(x_1, x_2, \dots, x_n) = \begin{cases} 1 & \text{ha } \sum x_i > n/2 \\ 0 & \text{különben} \end{cases}$$

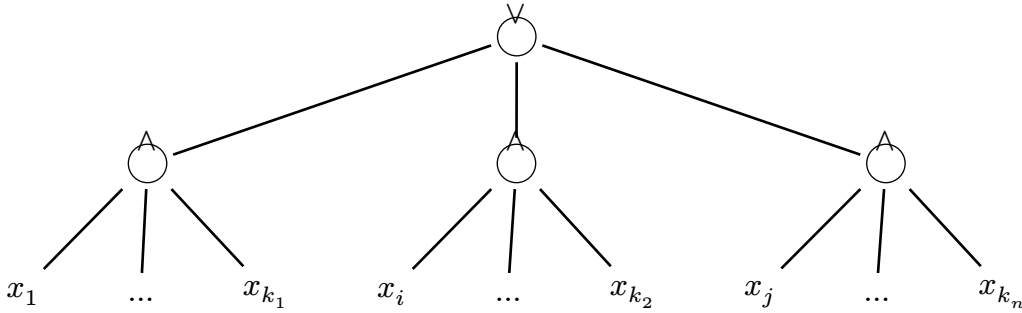
$$\text{THRESHOLD}_k(x_1, x_2, \dots, x_n) = \begin{cases} 1 & \text{ha } \sum x_i > k \\ 0 & \text{különben} \end{cases}$$

$$\text{MOD}_m(x_1, x_2, \dots, x_n) = \begin{cases} 1 & \text{ha } \sum x_i \not\equiv 0 \pmod{m} \\ 0 & \text{különben} \end{cases}$$

Megjegyzés

2-mély Boole-hálókra exponenciális méret kell a PARITY kiszámolására.

Proof: Mivel a PARITY egy Boole-függvény ezért fel lehet írni konjunktív normál formába, így ezt a konjunktív normál formát ki tudjuk számolni Boole-hálóval. A következő ábra illusztrálja hogy hogyan nézhet ki egy CNF hálója.



Mivel azon n hosszú bemenetek száma melyekre 1-et ad a PARITY-t kiszámoló háló 2^{n-1} , azaz

$$|\text{PARITY}_n^{-1}(1)| = 2^{n-1},$$

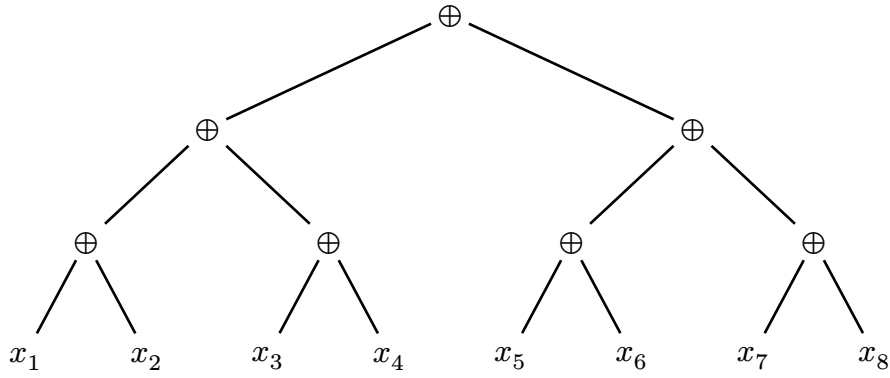
ezért nem lehet kevesebb mint 2^n csúcsa a hálónak.

□

Megjegyzés

A PARITY nyelvire $\log n$ -nél mélyebb Boole-hálóra nem megy a szuper-polinomiális alsó becslés.

Proof: Jelöljük a mod 2 összeadást \oplus -al, azaz a bináris XOR műveletet. Ekkor, ezekkel a \oplus kapukkal tudunk alkotni egy $\log n$ mély Boole-hálót, aminek a mérete $O(n)$ a következő módon:



□

Megjegyzés

A PARITY nyelvet $\log n / \log \log n$ mélységben is ki lehet számolni a \wedge, \vee, \neg jelekkel.

Proof: TODO

□

Állítás 5.7

$$\log_{\log n} n = \frac{\log n}{\log \log n}$$

Proof: A logaritmus alap átváltás formula révén

$$\log_a b = \frac{\log_c a}{\log_c b} \implies \log_{\log n} n = \frac{\log n}{\log \log n}.$$

□

Tétel 5.8

Yao–Håstad–Smolensky

A PARITY-t kiszámoló \wedge, \vee, \neg kapukat használó k -mély Boole-háló mérete legalább

$$2^{\frac{1}{2}n^{\frac{1}{2k}}}.$$

Proof: TODO

□

Megjegyzés

A bizonyítás a következő cikk alapján hangzott el előadáson: <https://dl.acm.org/doi/pdf/10.1145/28395.28404>

Tétel 5.9

$$\mathbf{BPP} \subseteq \mathbf{P}/\text{poly}$$

Proof: **TODO**

□

6 Párhuzamos algoritmusok

Példa

Feladat: **input:** $a_1, \dots, a_n \in \mathbb{N}$, **output:** $\max a_i$

n^2 processzorral meg tudunk adni egy $O(1)$ algoritmust.

Csinálunk egy b auxiliary tömböt. Minden $i \neq j$ index párosra egy processzorral kiszámoljuk $a_i < a_j$ -t. Ha $a_i < a_j$, akkor $b[i] = *$. Egy lépés alatt az n^2 processzor elvégzi az összes összehasonlítást. A végén csak a maximális elemnél nem lesz $*$.

Csak akkor szeretne ugyanoda írni két processzor ha ugyanazt akarják írni, így ez nem lesz probléma.

Példa

Feladat: **input:** $a_1, \dots, a_n \in \mathbb{N}$, **output:** $\max a_i$

n processzorral meg tudunk adni egy $O(\log \log n)$ algoritmust.

Bontsuk fel az n elemet \sqrt{n} darab \sqrt{n} méretű részre. Ha kiszámoltuk mindegyik rész maximumját, akkor már csak a kapott \sqrt{n} maximum maximumját kell kiszámolnunk, amit n processzorral tudunk $O(1)$ időben.

Legyen $T(n)$, hogy mennyi idő n elem maximumát kiszámolni n processzorral. Most láttuk, hogy $T(n) = T(\sqrt{n}) + 1$.

Tegyük fel, hogy $n = 2^k$. Ekkor $T(n) = T(2^k) = T(2^{\frac{k}{2}})$. Legyen $S(k) = T(2^k)$. Ekkor $S(k) = S(\frac{k}{2}) + 1$, innen már egyből látszik, hogy $S(k) = O(\log k)$.

Mivel $k = \log n$ és $T(n) = S(k)$ ezért $T(n) = O(\log \log n)$.

Definíció 6.1

PRAM

A PRAM-nek van n processzora és mindegyik processzor egy külön RAM, amit processzornak hívunk, és van egy közös (globális) memóriája. A közös memóriát egész számokkal indexelünk és M -nek jelöljük, tehát a memória cellái: $\dots, M[-2], M[-1], M[0], M[1], M[2], \dots$

A közös memóriába tetszőleges egész számok írhatóak, tehát $\forall i M[i] \in \mathbb{Z}$.

Egy számítási lépés a következő három szakaszból áll:

1. Mindegyik processzor olvashat legfeljebb 1 cellát M -ből.
2. Mindegyik processzor számolhat.
3. Mindegyik processzor írhat legfeljebb 1 cellát m -be.

A közös memóriába konkurens írásokat hogyan oldunk fel?

- Illegális egyszerre több processzornak ugyanabba a globális memória címbe írnia.
- Csak akkor megengedett ha a konkurens írások megegyeznek.
- Mindegyik processzornak van egy prioritása és a konkurens írásoknál a legmagasabb prioritású processzor írása marad.

A PRAM inputja M -ben adott és ide is várjuk az output-ot.

A PRAM egy lépését annak számoljuk, hogy a processzorok párhuzamosan olvasnak, számítanak, és írnak egyet.

Megjegyzés

Jelölje egy adott feladat megoldására szükséges időt p processzorral t_p . Ekkor $\frac{t_1}{p} \leq t_p$.

Proof: Tegyük fel, hogy $pt_p < t_1$, ez azt jelenti, hogy ha egy processzorral szimuláljuk p processzor párhuzamos működését az gyorsabb lenne mint t_1 , de pont úgy definiáltuk t_1 -et, hogy ez a minimális idő hogy 1 processzorral megoldjuk a feladatot.

□

Következmény 6.2

Az előző miatt egy exponenciális algoritmushoz legalább exponenciálisan sok processzor kell, hogy polinomiális idejű algoritmust kapjunk.

Definíció 6.3

NC

Ha egy L feladatot $n^{O(1)}$ processzorral $\log^{O(1)} n$ időben ki tudok számolni, akkor az L feladat NC-ben van (*Nick's Class*).

Példa

Két n hosszú vektor skaláris szorzata n processzorral $\log n$ időben.

$$a, b \in \mathbb{N}^n, \text{ kell } a \cdot b \in \mathbb{N}.$$

Proof: $a_i b_i$ szorzatok $O(1)$. Összeadások mint egy teljes bináris fa $\Rightarrow \log n$

□

Példa

Mátrix-mátrix szorzás nkm processzorral $O(\log m)$ időben.

$A \in \mathbb{N}^{n \times m}$, $B \in \mathbb{N}^{m \times k}$, kell $AB = C \in \mathbb{N}^{n \times k}$.

Proof: A mátrix-mátrix szorzás pontosan nk darab skaláris szorzás, ahol C_{ij} -t pontosan úgy kapom, hogy A -nak az i -edik sorát és B -nek a j -edik oszlopát skalárisan szorzom.

Mivel mindegyik cellájára a C eredmény mátrixnak tudok szolgáltatni m darab processzort és egy cella értékét két m hosszú vektor skaláris szorzata határozza meg, ezért $O(\log m)$ időben készen vagyok.

□

Példa

Positív élsúlyokkal rendelkező G gráfban minden kezdő csúcsra és minden végső csúcsra a legrövidebb út keresése n^3 processzorral $O(\log n \log \log n)$ időben. (All Pairs Shortest Paths)

Proof: Definiáljuk a következő műveletet két mátrix között $A \oplus B$, ahol

$$(A \oplus B)_{i,j} := \min_{k=1,\dots,n} \{a_{ik} + b_{kj}\}.$$

Nyilvánvaló, hogy egy ilyen műveletet ki tudunk számolni $O(\log \log n)$ időben, ha van n^3 processzorunk, mert minden eredmény cellára adunk n processzort és ott már n elem minimumát n processzorral $O(\log \log n)$ időben el tudjuk végezni.

Vegyük észre, hogy $A^n := \overbrace{A \oplus A \oplus \dots \oplus A}^n$ pont azt adja meg, hogy milyen hosszú két csúcs között a legrövidebb út. Ezt már láttuk algoritmuselméleten a Floyd–Warshall algoritmusnál.

Figyeljük meg, hogy A^n -et ki tudjuk számolni $O(\log n)$ művelettel úgy hogy kiszámoljuk a kettő hatványokat sorrendben: $A^1, A^2, A^4, A^8, \dots, A^n$.

Tehát összesen $O(\log n)$ darab műveletet kell elvégeznünk, melyek mind $O(\log \log n)$ időben mennek, tehát $O(\log n \log \log n)$ a végső idő.

□

Emlékezzünk vissza, hogy 2 mély és exponenciális méretű Boole hálózattal ki tudunk számolni bármilyen Boole függvényt. Ennek a következménye, hogy exponenciálisan sok processzorral minden Boole függvényt ki tudunk számolni $O(1)$ időben.

Csánky tétele TODO

Teljes párosítás páros gráfban TODO

Állítás 6.4

Adott n számot tudunk rendezni n processzorral $O(\log^2 n)$ időben.

Proof: Az összefésülő rendezést fogjuk párhuzamosan megcsinálni. Tehát m elemet úgy rendezek, hogy az első felét és a második felét külön-külön rendezem és utána összefésülöm.

Hogyan fésülök össze két m hosszú rendezet sorozatot $O(\log m)$ időben $O(m)$ processzorral?

Állítsunk egy-egy processzort minden elemre a két m hosszú sorozatból. Mindegyik processzor megkeresi, hogy a hozzá rendelt elem hol lenne a másik tömbben bináris kereséssel. Ha az egyik sorozatban az i -edik elem a másik sorozatban a j -edik elem lenne, akkor az összefésült sorozatban az $(i + j - 1)$ -edik lesz. Tehát be is szűrjük a megfelelő helyére az elemet.

Tehát minden elemet be tudok szűrni egyszerre $O(\log m)$ időben az összefésült sorozatba.

Továbbá az összefésülő rendezés mindig felez és így $O(\log n)$ szintje lesz, és minden szinten összesen n elemet kell összefésülni, tehát az összefésülések költsége $O(\log n)$ minden szinten. Tehát eredményként $O(\log^2 n)$ lépés alatt tudtunk rendezni egy sorozatot n processzossal.

□

7 VLSI chip tervezés

Megjegyzés

VLSI = Very Large Scale Integration

Definíció 7.1

VLSI chip

A VLSI chip egy négyzetrácson van definiálva a következő tulajdonságokkal:

- a drótok legalább 1 távolságra vannak egymástól
- csak négyzetrács vonalait követhetik a drótok
- egy dróton, egy lépésben, egy bit mehet
- a drótok Boole függvényeket kiszámoló kapukat kötnek össze
- a drótok akár processzorokat is összeköthetnek
- egy kapun áthaladás egy időegységet vesz igénybe
- a kapuk a csomópontokban vannak

Tétel 7.2

Thompson

Ha A jelöli a chip területét és T a számoláshoz szükséges időt, akkor $A \cdot T \geq n^2$ az $ID(x, y)$ célfüggvény kiszámolásához.

Példa

$O(n)$ időben $O(n)$ területen

TODO

Példa

$O(\log n)$ időben $O(n^2)$ területen

TODO

Proof of Tétel 7.2:

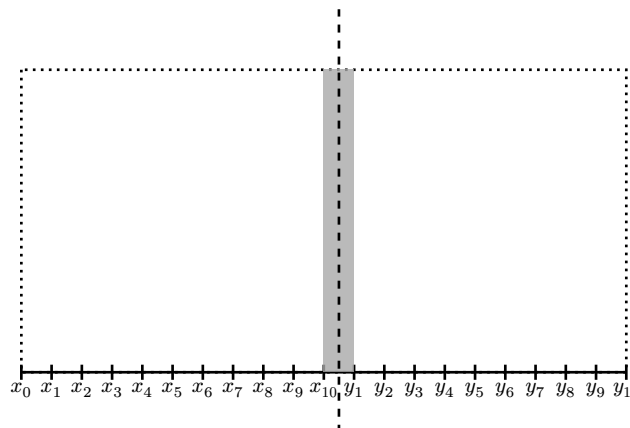


Figure 2: Thompson segédábra

A Mehlhorn–Schmidt tétel miatt tudjuk, hogy $ID(x, y)$ kiszámolásához kell legalább n bit közlés. Tehát a segédábrában jelzett középső szaggatott vonalat az egész számítás alatt összesen legalább n bitnek kell kereszteznie.

Jelölje d a drótok számát amik keresztezik az ábrában szürkével sátirozott területet. Az egész számítás alatt összesen $d \cdot T$ bitet tudunk akkor közölni ennyi dróton. Az előbb tárgyaltuk, hogy ennek legalább n -nek kell lennie, szóval $d \cdot T \geq n$, azaz legalább $\frac{n}{T}$ drót keresztezi ezt a területet.

Hasonló megfontolással állíthatjuk, hogy az x_9 és x_{10} közötti területet legalább $\frac{n-1}{T}$ drót keresztezi, mert a köztük húzható vonalat legalább $n - 1$ bit keresztezi, hogy ki lehessen számolni $ID(x_1 \dots x_{n-1}, y_1 \dots y_{n-1})$ -et.

A keresztező drótok alsó becslését felírva az összes ilyen csíkra és összegezve azt kapjuk, hogy az egész alaptréületet legalább

$$2 \sum_{i=0}^n \frac{n-i}{T} = 2 \frac{n(n-1)}{2T} = \frac{n(n-1)}{T}$$

drót keresztezi. Továbbá az alaptréület legalább annyi mint ahány drót keresztezi, így beláttuk, hogy

$$A \geq \frac{n(n-1)}{T} \implies A \cdot T \geq n^2.$$

□

8 Rendező hálózatok

Definíció 8.1

Rendező hálózatok

TODO

Lemma 8.2

0–1 lemma

Egy rendező háló pontosan akkor rendez jól, ha minden $\{0, 1\}^n$ sorozatot jól rendezi.

Proof: TODO

□

Tétel 8.3

Batcher

Tudunk adni egy rendező hálót, ami $O(\log^2 n)$ párhuzamos időben és $O(n \log^2 n)$ összehasonlítással rendez egy n hosszú sorozatot.

Proof: Az összefésülő rendezést fogjuk megvalósítani rendező hálózattal és egy kis trükkel.

1. Rekurzívan rendezzük az első és második felét külön-külön az inputnak.
2. Rendezzük a páros indexű elemeket és a páratlan indexűeket külön-külön.
3. Rakjuk helyre a $(2k - 1, 2k)$ párokat $k = 1, \dots, \frac{n}{2}$ -re.

Nézzük meg a következő ábrát jobb intuícióért.

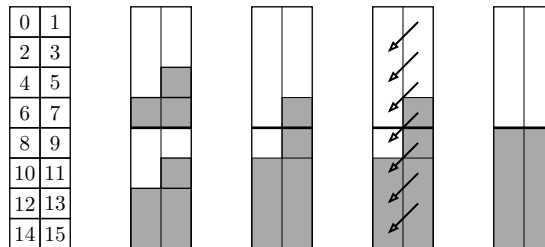


Figure 3: Batcher sort sematikus ábra

Ha megértettük hogy hogyan működik a Batcher sort, akkor már csak azt kell belátnunk hogy valóban jól rendez és elemeznünk a futásidejét.

Kezdjük korrektséggel. A Lemma 8.2 értelmében elég $\{0, 1\}^n$ sorozatokra elemezni. Szerencsére valaki pont úgy rajzolta fel a sematikus ábrát, hogy a szürke cellák reprezentálják az 1-es elemeket, míg a világos cellák a 0-as elemeket.

A rendező háló korrektsége praktikusán adja magát az ábrából. Az első két lépés után már majdnem jól van rendezve a sorozat, annyi diszkrepanciával, hogy a páratlan elemeknél legfeljebb 2-vel több 1-es lehet, mint a párosoknál. Ezért még végigmegyünk az összes egymás melletti páros-páratlan páron és helyre rakjuk őket és készen vagyunk.

A futásideje a hálónak is könnyen látható. Legyen $T(n)$ az n hosszú inputon használt összehasonlításoknak a száma. Ekkor $T(n) = 2T(\frac{n}{2}) + S(n)$, ahol $S(n)$ azt jelöli, hogy hány összehasonlítás kell kettő rendezett tömb rendezésére.

Látszik, hogy $S(n) = 2S(\frac{n}{2}) + n$, amiről tudjuk, hogy $S(n) = O(n \log n)$.

Innen már evidens, hogy $T(n) = O(n \log^2 n)$.

Megjegyzés**AKS**

Létezik egy úgynevezett Ajtai–Kolmós–Szemerédi rendező háló, ami $O(\log n)$ párhuzamos időben és $O(n \log n)$ összehasonlítással rendez egy n hosszú sorozatot.

9 Döntési fák

TODO

definicio

Döntési fa rendezésre $\geq n \log n$ mély

algo konvex burok keresesre

algebrai dontesi fa def

n csucs kozul konvex burkat megkeresni linearis teszt fv-el $\Omega(n \log n)$ mely algebrai dontesi faval

Milnor–Thom tetel

Egyszeru dontesi fa def

zarkozott Boole fv def

graf of eldontese zarkozott

teljes csucs letese zarkozott

$D(f), D_0(f), D_1(f)$ def

peldak ezek ertekeire: OR, teljes csucs

Tardos tetel: $D(f) \leq D_0(f) \cdot D_1(f)$

Ha f egy N elemű halmazon veszi fel az 1 értéket, és $N=2^k M$, ahol M már páratlan, akkor az f -et kiszámoló bármely egyszerű döntési fa mélysége legalább $n-k$.

Szimmetrikus Boole fv

Minden nem konstans szimmetrikus Boole fv zarkozott

Gyenge szimmetria

Aanderaa–Karp–Rosenberg sejtes: Minden nem konstns monoton gyengen szimmetrikus Boole fv zarkozott

10 Pszeudovéletlen számok

TODO

Pszeudovéletlen sorozatok

Példa

Eltolás regiszter

Az eltolás regiszterek a következő módon vannak definiálva. Legyen $f : \{0, 1\}^n \rightarrow \{0, 1\}$ egy könnyen számolható függvény. Egy n bites a_0, a_1, \dots, a_{n-1} magból kiindulva, kiszámoljuk a $a_n, a_{n+1}, a_{n+2}, \dots$ biteket az

$$a_k = f(a_{k-1}, a_{k-2}, \dots, a_{k-n})$$

rekurzióval. Az „eltolás regiszter” név abból a tulajdonságból származik, hogy csak $n + 1$ bitet kell eltárolnunk: miután eltároltuk $f(a_0, \dots, a_{n-1})$ -t a_n -ben, nincs szükségünk többé a_0 -ra és eltolhatjuk a_1 -t a_0 -ba, a_2 -t a_1 -be stb. Legfontosabb eset, mikor f egy lineáris függvény a 2-elemű test felett, a továbbiakban ezzel foglalkozunk.

Bizonyos esetekben az eltolás regiszter által generált bitek véletlennek látszanak, legalábbis egy darabig. Természetesen az a_0, a_1, \dots sorozatban egy idő után ismétlődik valamely n egymás utáni bit, és innentől periodikus lesz a sorozat. Ennek viszont nem kell a_{2^n} előtt bekövetkeznie, és valóban, választható olyan lineáris függvény, melyre a sorozat periódusa 2^n .

A problémát az jelenti, hogy a sorozatnak van más rejtett szerkezete is a periodicitáson kívül. Valóban, legyen

$$f(x_0, \dots, x_{n-1}) = b_0x_0 + b_1x_1 + \dots + b_{n-1}x_{n-1}$$

(ahol $b_i \in \{0, 1\}$). Tegyük fel, hogy nem ismerjük a b_0, \dots, b_{n-1} együtthatókat, viszont az eredményül kapott sorozatnak ismerjük az első n bitjét (a_n, \dots, a_{2n-1}). Ekkor tekintsük a következő, 2-elemű test feletti lineáris egyenletrendszert n egyenlettel és n ismeretlennel:

$$\begin{aligned} b_0a_0 + b_1a_1 + \dots + b_{n-1}a_{n-1} &= a_n \\ b_0a_1 + b_1a_2 + \dots + b_{n-1}a_n &= a_{n+1} \\ &\vdots \\ b_0a_{n-1} + b_1a_n + \dots + b_{n-1}a_{2n-2} &= a_{2n-1} \end{aligned}$$

Ekkor, ha egyértelműen meghatároztuk a b_i -ket, akkor meg tudjuk mondani minden további a_{2n}, a_{2n+1}, \dots elemét a sorozatnak.

Előfordulhat persze, hogy az egyenletrendszer megoldása nem egyértelmű, mert az egyenletek összefüggenek. Például, ha a $0, 0, \dots, 0$ magból indultunk, akkor az egyenletek nem mondanak semmit. Megmutatható viszont, hogy egy véletlen magból indulva az egyenletek pozitív valószínűséggel meghatározzák a b_i -ket. Tehát a sorozat első $2n$ elemének ismeretében a többi „nem látszik véletlennek” már egy olyan megfigyelő számára sem, aki egy viszonylag egyszerű (polinom idejű) számítást akar csak elvégezni.

Példa

Lineáris kongruencia-generátor

A gyakorlatban legfontosabb pszeudovéletlenszám-generátorok a lineáris kongruencia generátorok. Egy ilyen generátor három pozitív egész paraméterrel van megadva (a, b és m). Az X_0 magból kiindulva, melyre $0 \leq X_0 \leq m-1$, a generátor az X_1, X_2, \dots egészeket a

$$X_i = aX_{i-1} + b \pmod{m}$$

rekurzióval számolja. Eredményként használhatjuk az X_i -ket, vagy pl. a középső bitjeiket.

Megjegyzés

Az derül ki, hogy ezen generátorok által kapott sorozatok is megjósolhatók polinom idejű számolással, polinom darab sorozatelem felhasználásával. Ám ezen algoritmusok nagyon bonyolultak, így gyorsaságuk és egyszerűségük miatt a lineáris kongruencia generátorok jók a legtöbb gyakorlati alkalmazásban.

Példa

Utolsó példaként nézzük a kettes számrendszerbeli alakját pl. a $\sqrt{5}$ -nek:

$$\sqrt{5} = 10.001111000110111\dots$$

Megjegyzés

Ez a sorozat eléggé véletlenszerűnek látszik. Természetesen nem használhatjuk mindig ugyanazt a számot, de választhatunk mondjuk egy n -bites a egészet és ekkor legyen a kimenet $\sqrt{a} - \lfloor \sqrt{a} \rfloor$.

Sajnos ez a módszer is "feltörhető" nagyon bonyolult (viszont polinom idejű) algoritmikus számelméleti módszerekkel.

Definíció 10.1

NEGL(n)

Azt mondjuk, hogy egy $f : \mathbb{Z}^+ \rightarrow \mathbb{R}$ függvény elhanyagolható, ha minden fix k -ra $n \rightarrow \infty$ esetén $n^k f(n) \rightarrow 0$.

Megjegyzés

Szavakkal, f gyorsabban tart 0-hoz mint bármelyik polinom reciproka.

Jelöljük ezt (a nagy O -hoz hasonlóan) úgy, hogy

$$f(n) = \text{NEGL}(n).$$

Figyeljük meg, hogy egy elhanyagolható függvény bármilyen polinommal megszorozva elhanyagolható marad, azaz

$$n^r \cdot \text{NEGL}(n) = \text{NEGL}(n)$$

minden fix r esetén.

Definíció 10.2

Generátor

Tekintsünk egy polinom időben számítható $G : \{0, 1\}^* \rightarrow \{0, 1\}^*$ függvényt, melyre feltesszük, hogy $|G(x)|$ csak $|x|$ -től függ és $|x| < |G(x)| < |x|^c$ egy c konstansra. Egy ilyen függvényt hívunk *generátornak*.

Definíció 10.3

Teszt

Legyen \mathcal{A} egy polinom idejű randomizált algoritmus (Turing-gép), amely bármely z bemenetre kiszámít belőle egy $\mathcal{A}(z)$ bitet (a kimenetnél a 0 jelentése, hogy „nem véletlen”, míg az 1 jelentése, hogy „véletlen” volt a bemenet).

Definíció 10.4

Sikeres teszt

Rögzítsünk le egy $n \geq 1$ számot. Válasszuk x -et egyenletesen $\{0, 1\}^n$ -ből és y -t egyenletesen $\{0, 1\}^N$ -ből, ahol $N = |G(x)|$. Feldobunk egy érmét és az eredménytől függően vagy $G(x)$ -et vagy y -t adjuk meg \mathcal{A} -nak bemenetként. Akkor hívjuk \mathcal{A} -t sikeresnek (ezekre a véletlen választásokra nézve), ha vagy $G(x)$ volt a bemenet és 0 a kimenet, vagy pedig y volt a bemenet és 1 a kimenet.

Tehát helyesen felismerte a teszt, hogy a valódi véletlen számot adtuk-e vagy a generáltat.

Definíció 10.5

Biztonságos véletlenszám-generátor

Egy G generátor (biztonságos) pszeudovéletlenszám-generátor, ha minden polinom idejű randomizált \mathcal{A} algoritmusra, melynek bemenete egy z 0-1 sorozat és kimenete egy $\mathcal{A}(z)$ bit, annak valószínűsége, hogy \mathcal{A} sikeres, legfeljebb $\frac{1}{2} + \text{NEGL}(n)$.

Megjegyzés

Ez a feltétel azt jelenti, hogy $G(x)$ átmegy minden „értelmes” (polinom időben számolható) teszten abban az értelemben, hogy annak valószínűsége, hogy a teszt felismeri, hogy $G(x)$ nem valódi véletlen, elhanyagolhatóan nagyobb csak $\frac{1}{2}$ -nél.

Állítás 10.6

Ha $P = NP$, akkor nincs biztonságos polinomiális idejű véletlenszám-generátor.

Definíció 10.7

Egyirányú függvény

Egy $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$ függvényt egyirányúnak nevezünk, ha

1. létezik egy $c \geq 1$ konstans, melyre $|x|^{\frac{1}{c}} < |f(x)| < |x|^c$;
2. $f(x)$ számolható polinom időben;
3. minden \mathcal{A} polinom idejű randomizált algoritmusra, mely 0-1 sorozatokat számol 0-1 sorozatokból, és egy véletlen y sorozatra, mely egyenletesen véletlenül van választva $\{0, 1\}^n$ -ből, teljesül:

$$\mathbb{P}[f(\mathcal{A}(f(y))) = f(y)] = \text{NEGL}(n).$$

Definíció 10.8

Egyirányú permutáció

Az egyirányú permutáció egy olyan egyirányú függvény, mely bijekció és $|f(x)| = |x|$ minden x esetén.

Tétel 10.9

Tegyük fel, hogy f egyirányú permutáció. Ekkor ebből alkotható egy g biztonságos véletlenszám-generátor.

Kandidátusok egyirányú függvényre

Négyzetgyökvonás prím és összetett modulussal, szorzás

11 Kolmogorov bonyolultság

Kolmogorov bonyolultság def

Kolm bony jól definált

Lemma: `print "x"`

Tétel: A Kolm bony csak konstans erejű függ a TG-tól (invariancia)

Tétel: A kolm bony nem rekurzív

Tétel: Kozelithetlenség

Tétel: Keves olyan szó van aminek a Kolm bonyolultsága kicsi

Kor: 99%-a a szavaknak Kolm bony $\geq n - 8$

Informatikusan véletlen sorozat

Számítógép nem tud informatikusan véletlen sorozatot generálni

Formális axióma rendszer mérete def

Tétel: Véges méretű axióma rendszerben legfeljebb véges sok m -re lehet bizonyítani, hogy $K(x) > m$.

