

BONYOLULTSÁGELMÉLET

ELŐADÓ: GAZDAG ZSOLT

KÉSZÜLT ...

- IVÁN SZABOLCS KORÁBBI ELŐADÁSFÓLIÁI, VALAMINT
- M. SIPSER: INTRODUCTION TO THE THEORY OF COMPUTATION ÉS
- C. PAPADIMITRIOU: SZÁMÍTÁSI BONYOLULTSÁG C. KÖNYVEI ALAPJÁN

KURZUS INFORMÁCIÓK

Előadó: Gazdag Zsolt

- Elérhetőség:

- Coospace PM; a tárgy Coospace fóruma
- gazdag@inf.u-szeged.hu
- Irinyi ép. III. lh. 1. em.

A kurzus **teljesítésének** feltételei:

- Teljesített gyakorlat
- Sikeres géptermi Coospace teszt
 - kifejőς és teszt kérdések
 - alap összefüggések, definíciók, állítások, és ezek alapján kikövetkeztethető egyszerű kérdések és feladatok
 - a teszt kérdéseknél rossz válaszért pontlevonás
 - sikeres a teszt ha legalább 40%-os; ha a teszt sikertelen, akkor a vizsga érdemjegye elégtelen
- A sikeres tesztet megírók megajánlott jegyet kapnak az alábbiak szerint:
 - 40%-60% - elégéges (2)
 - 60%-80% - közepes (3)
 - 80%-100% - jó (4)
- A sikeres tesztet megírók jelentkezhetnek szóbeli vizsgára, amennyiben a megajánlott jegynél jobbat szeretnének.

Tananyag: a Coospace-re felkerülnek az előadás fóliák és egyéb segédanyagok

MIRŐL LESZ SZÓ?

- **Kiszámíthatóság elmélet**

- Algoritmus modelleket definiál
- Azt vizsgálja, hogy ezek univerzálisak-e (minden megoldható velük ami algoritmikusan megoldható)

- Eldönthetetlenséget bizonyít

- **Bonyolultságelmélet**

- Azt vizsgálja, hogy egy probléma milyen erőforrásigénnyel oldható meg

- Ezek alapján osztályokba sorolja a problémákat

- Keresi egy adott osztályban a legnehézebben megoldható problémákat

- **Megjegyzés:** Az első előadás egy betekintés ezen téma körökbe, a felhasznált fogalmakat tárgyalni fogjuk részletesebben is



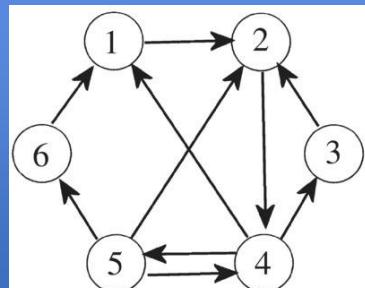
KISZÁMÍTHATÓSÁG ELMÉLET – MEGOLDHATÓ PROBLÉMÁK

- David Hilbert azt gondolta, hogy minden probléma megoldható
 - Hilbert 23 problémája (1900, Párizs)
- Mit jelent az, hogy egy probléma megoldható?
 - Azt hogy van egy olyan algoritmus, ami a probléma minden bemenetére kiszámolja a helyes választ
- A kiszámíthatóság- és bonyolultságelméletben jellemzően eldöntési problémákkal foglalkozunk:
 - A bemenetre a válasz bináris: igen/nem, true/false, 1/0
- Eldöntési probléma például az ELÉRHETŐSÉG:
 - Bemenet: egy $G = (V, E)$ irányított gráf, ahol $V = \{1, \dots, n\}$
 - Kérdés: Vajon van-e út G -ben 1-ből n -be?

KISZÁMÍTHATÓSÁG ELMÉLET – ELDÖNTHETŐ PROBLÉMÁK

- A megoldható eldöntési problémákat **eldönthetőnek** nevezük
- **ELÉRHETŐSÉG** eldönthető:
 - Bemenet: $G = (V, E)$ irányított gráf, ahol $V = \{1, \dots, n\}$
 - Legyen $S = \{1\}$
 - Amíg $n \notin S$ és $S \neq \emptyset$
 - Vegyünk ki egy u csúcsot S -ból;
 - Vegyük be S -be azon u -val szomszédos csúcsokat, melyek még nem szerepeltek S -ben
 - Ha $n \in S$, akkor kimenet: *igen*, egyébként kimenet: *nem*

Ez egy polinom időigényű algoritmus



$S = \{1\};$
 $S = \{1, 2\}, u = 1, S = \{2\};$
 $S = \{2, 4\}, u = 2, S = \{4\};$
 $S = \{4, 3, 5\}, u = 4, S = \{3, 5\};$
 $S = \{5\}, u = 3, S = \{5\};$
 $S = \{5, 6\}, u = 5, S = \{6\}$
Kimenet: *igen*

KISZÁMÍTHATÓSÁG ELMÉLET – HILBERT 10-IK PROBLÉMÁJA

- Hilbert 10-ik problémája (H10):
 - Bemenet: egy p egész együtthatós polinom, x_1, \dots, x_n változókkal
 - Kérdés: van-e az x_1, \dots, x_n változóknak olyan egész értékei, melyekre
$$p(x_1, \dots, x_n) = 0?$$
 - Például ha $p = 2x_1 - 3x_1x_2 + 2$, akkor *igen* a válasz a kérdésre
 - Mert pl. $p(2,1) = 0$
 - De például ha $p = 2x - 1$, akkor *nem* a válasz a kérdésre
 - Mert nincs olyan i egész szám, melyre $2i - 1 = 0$
- Vajon eldönthető-e H10?
- Tudjuk, hogy a \mathbb{Z} -feletti szám n -esek felsorolhatók, mondjuk a $(0, \dots, 0)$ -val kezdve (hogyan?)

KISZÁMÍTHATÓSÁG ELMÉLET – HILBERT 10-ES

- Tetszőleges $(i, j, k) \in \mathbb{Z}^3$ esetén legyen (i', j', k') a számhármasok felsorolásában az (i, j, k) -t közvetlenül követő
- Tekintsünk a következő **algoritmust**:
- Bemenet: **egy tetszőleges** $p(x_1, x_2, x_3)$ polinom
 - Legyen $(i, j, k) = (0, 0, 0)$
 - Amíg $p(i, j, k) \neq 0$
 - Legyen $(i, j, k) = (i', j', k')$
 - Kimenet: *igen*
- Eldönti ez az algoritmus H10-et?
 - Egyszerűen minden „jó” bemeneten **helyes válasszal megáll**
 - Másrészt a „rossz” bemeneteken **nem áll meg**
 - Például a $2x_1 + 0x_2 + 0x_3 - 1$ polinomon nem áll meg
- Tehát **nem dönti el a problémát** (ahhoz az kellene, hogy minden bemeneten megálljon helyes válasszal)

KISZÁMÍTHATÓSÁG ELMÉLET – BEVEZETÉS

- Van másik algoritmus ami eldönti a H10 problémát?
 - Nincs (Matiyasevich, 1970)
- Egy tetszőleges E eldöntési probléma B bemenetét szokás pozitívnak nevezni ha B -re a válasz igen E -ben, és negatívnak, ha nem a válasz rá
- Félig eldönthetőnek (vagy rekurzívan felsorolhatónak) nevezzük azokat a problémákat, melyekhez van olyan algoritmus, ami pozitív bemenetekre helyes válasszal megáll, negatív bemenetekre pedig helyes válasszal megáll vagy nem áll meg
- Eldönthetők (vagy rekurzívak) azok a félig eldönthető problémák, melyekre van olyan algoritmus, ami minden bemeneten megáll
 - H10 tehát félig eldönthető, de nem eldönthető
 - Ismerünk még ilyet: az elsőrendű logikai formulák kielégíthetetlensége is félig eldönthető

KISZÁMÍTHATÓSÁG ELMÉLET – ELDÖNTHETETLENSÉG

- Az, hogy vannak eldönthetetlen problémák könnyen bizonyítható számossági megfontolások alapján is:
 - Legyen Σ egy ábécé
 - Σ -feletti formális nyelv megszámlálhatatlanul végtelen sok van
 - Tetszőleges $L \subseteq \Sigma^*$ nyelvre a szóprobléma is egy eldöntési probléma:
 - Adott egy tetszőleges $w \in \Sigma^*$ szó; kérdés: vajon $w \in L$ teljesül-e?
 - Tehát az eldöntési problémák megszámlálhatatlanul végtelen sokan vannak
 - Algoritmus viszont megszámlálhatóan végtelen sok van:
 - minden algoritmus tekinthető egy véges hosszú szónak
 - Tehát az algoritmusok sorba rendezhetők, például lexikografikusan
- Ezért ha minden eldöntési problémához hozzárendelünk egy őt eldöntő algoritmust, akkor kell legyen olyan, amelyikhez nem tudunk algoritmust rendelni

KISZÁMÍTHATÓSÁG ELMÉLET – ELDÖNTHETETLENSÉG

- Hogyan lehet bebizonyítani, hogy egy probléma nem eldönthető?
- Kell egy matematikai eszköz
 - Amiről mindenki „elfogadja” hogy amit algoritmikusan el lehet dönten, azt ezzel az eszközzel is el lehet
 - Ilyen eszköz több is van:
 - **Turing-gép** (a véges automaták, veremautomaták általánosítása), Alan Turing 1936
 - **λ -kalkulus** (függvények absztrakcióján és alkalmazásán alapuló formális rendszer), A. Church 1936
 - **RAM gép** (egyszerű programozási nyelv, kb. mint a gépi kód)
 - **Általános nyelvtan** (formális nyelvek, a Chomsky-hierarchia legerősebb nyelvtana), Chomsky 1956
- Church-Turing tézis: a kiszámíthatóság fenti modelljei mind az effektíven (algoritmikusan) kiszámítható problémák osztályát definiálják (tézis és nem téTEL, mert az, hogy „effektíven kiszámítható” egy intuitív fogalom)

KISZÁMÍTHATÓSÁG ELMÉLET – ELDÖNTHETETLENSÉG

- A Church-Turing tézis alapján ha egy probléma nem dönthető el a fenti kiszámítási modellek egyikével, akkor nem dönthető el semmilyen algoritmussal sem
- Az első problémák, melyek eldönthetetlensége kiderült:
 - A Turing-gépek megállási problémája
 - λ -kalkulusbeli kifejezések ekvivalenciája
- Az eldönthetetlenség bizonyítható visszavezetéssel is:
 - Legyen K és L két eldöntési probléma, és tegyük fel, hogy K eldönthetetlen
 - Ha van olyan A algoritmus, ami a K bemenetéből kiszámolja L bemenetét úgy, hogy pozitív bemenethez pozitívat, negatívhoz negatívat rendel, akkor L is eldönthetetlen
 - Például vegyük a következő problémát: adott egy F elsőrendű formula, vajon tautológia-e F (másképp: érvényes-e F)?
 - Erre a problémára visszavezethető a kielégíthetetlenség eldöntése (hogyan?), ami eldönthetetlen
 - Így tehát az érvényesség sem lehet eldönthető

BONYOLULTSÁGELMÉLET – BEVEZETÉS

Itt csak eldönthető problémákat vizsgálunk

- Arra vagyunk kíváncsiak, hogy milyen erőforrásigénnyel dönthető el egy probléma
 - Például az ELÉRHETŐSÉG polinom időben, polinom tárral eldönthető (a korábbi fólián megadott algoritmus polinom idő- és tárígényű kóddal implementálható)
 - Érdekes kérdés, hogy eldönthető-e szublineáris tárfelhasználással
 - Szublineáris: lineárisnál kisebb, például logaritmikus
 - Az ELÉRHETŐSÉG-ről tudjuk, hogy $\log^2 n$ tárral is eldönthető (lásd Savith téTELét később)
 - Jellemző a „time-space tradeoff”
 - Például Savitch algoritmusának időigényére nincs polinom felső korlát

BONYOLULTSÁGELMÉLET – BEVEZETÉS

Miért fontos, hogy ismerjük egy adott probléma komplexitását?

- Mert például ha **felhőben** futtatjuk az algoritmust, akkor a felhasznált processzoridő és tár alapján fizetünk: <https://azure.microsoft.com/en-us/pricing/calculator/>

The screenshot shows the Azure Pricing Calculator interface. At the top, the URL is <https://azure.microsoft.com/en-us/pricing/calculator/>. The main heading is "Cloud Services".
Region: East US
Category: All
Instance Series: All
Virtual machines: 1 x 730 Hours
INSTANCE: A4: 8 Cores, 14 GB RAM, 605 GB Temporary storage, \$0.480/hour
Savings Options:
Pay as you go (selected)
1 year reserved option is not available for your instance selection.
3 year reserved option is not available for your instance selection.
Reservations: 1 year reserved (radio button selected)
\$467.20 Average per month (\$0.00 charged upfront) = \$467.20 Average per month

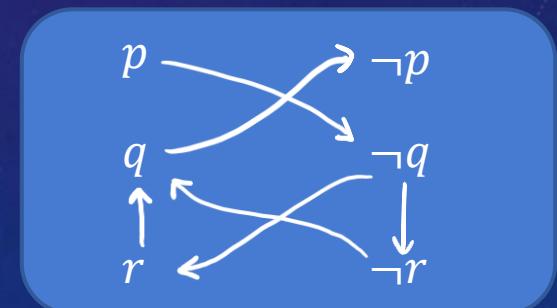
BONYOLULTSÁGELMÉLET – VISSZAVEZETÉS

- Az is érdekelhet minket, hogy két probléma bonyolultsága **hogyan viszonyul egymáshoz**
- Tegyük fel, hogy adottak a következő ítéletkalkulusbeli formulák:

$$p, \neg p \vee \neg q, r \vee q, q \vee \neg r$$

- Kérdés: **kielégíthető-e** ez a formulahalmaz?
- Ahelyett, hogy értékadásokat vizsgálnánk, a következőt is tehetjük:

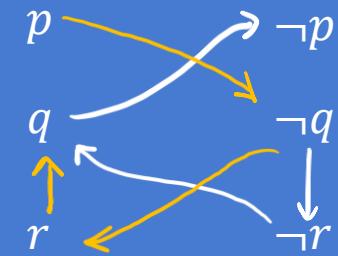
- Definiálunk egy G gráfot $\{p, \neg p, q, \neg q, r, \neg r\}$ csúcsokkal
- A \vee -kapcsolatokból implikációkat csinálunk, az implikációkat pedig éleknek tekintjük és ezeket felvesszük G -be:



- A formulahalmaz **akkor és csak akkor kielégíthetetlen** ha a következő teljesül:
 - Van olyan $\alpha \in \{p, \neg p, q, \neg q, r, \neg r\}$, hogy G -ben van út p -ből α -ba és α -ból az α komplementerébe

BONYOLULTSÁGELMÉLET – VISSZAVEZETÉS

- Mivel a példában van út p -ből $\neg q$ -ba, és onnan q -ba, a $p, \neg p \vee \neg q, r \vee q, q \vee \neg r$ formulahalmaz kielégíthetetlen
 - A gráf csak segít követni a függőségeket: p igaz kell legyen, de akkor $\neg q$ is, amiből következik, hogy r is, és végül q is, ami ellentmondás
- Erről a módszerről már volt szó ma, ez is egy **visszavezetés**: a példánkban a kielégíthetőség eldöntését vezettük vissza az ELÉRHETŐSÉG-re
- Ha **hatékony** (gyorsan, mondjuk polinom időben kiszámítható) a visszavezetés, akkor a visszavezetett probléma, általában, **legfeljebb olyan nehezen** oldható meg, mint az, amire visszavezettünk
- A példában:
 - A gráf megkonstruálása elvégezhető polinom időben a formulahalmaz méretében, és
 - az ELÉRHETŐSÉG is megoldható polinom időben
 - Ezért a formulahalmaz kielégíthetősége is eldönthető polinom időben
- Vajon eldönthető a kielégíthetőség \log^2 tárral is?
 - Igen, de ennek belátásához még hatékonyabb (pl. log táras visszavezetés kell)



BONYOLULTSÁGELMÉLET – NEHEZEN MEGOLDHATÓ PROBLÉMÁK

- A fentiekből következik: amire hatékonyan visszavezetünk nem lehet könnyebben megoldható, mint az amit visszavezetünk
- **A visszavezetést arra is felhasználhatjuk, hogy egy problémáról megmutassuk, hogy nehezen megoldható**
- **A SAT probléma:**
 - Bemenet: egy ítéletkalkulusbeli konjunktív normálformában lévő formula
 - Kérdés: vajon kielégíthető-e ez a formula?
- **A logika kurzuson tanult összes algoritmus a legrosszabb esetben exponenciális ideig fut erre a problémára**
- **A SAT megoldására nem ismert hatékony (polinom idejű) algoritmus**
 - Az a sejtés, hogy ilyen nem is létezik
 - Emiatt a SAT-ra „nehezen megoldható” (ún. NP-teljes, lásd később) problémaként fogunk hivatkozni
- **Ezért ha a SAT-ot hatékonyan vissza tudjuk vezetni egy K problémára, akkor K -ról is bizonyítjuk, hogy nehezen megoldható**

BONYOLULTSÁGELMÉLET – NEHEZEN MEGOLDHATÓ PROBLÉMÁK

- Vegyük például az ELÉRHETŐSÉG következő általánosítását, a HAMILTON-ÚT_{1→n} problémát:
 - Bemenet: egy $G = (V, E)$ irányított gráf, ahol $V = \{1, \dots, n\}$
 - Kérdés: Vajon van-e olyan út G -ben 1-ből n -be ami minden csúcsot pontosan egyszer érint?
 - Tetszőleges F formulához hatékonyan megkonstruálható olyan G_F gráf, hogy F akkor és csak akkor kielégíthető ha G_F -ben van Hamilton út adott két csúcs között (hogy hogyan, azt látni fogjuk később)
 - Ezért a HAMILTON-ÚT_{1→n} nem lehet könnyebben megoldható, mint a SAT
 - Vagyis a HAMILTON-ÚT_{1→n} is egy nehezen megoldható probléma (NP-teljes)
-
- De miért baj az, hogy ha valamire csak exponenciális időigényű algoritmus létezik?

BONYOLULTSÁGELMÉLET – FUTÁSI IDŐ

Az alábbi táblázat megadja, hogy adott futásidejű algoritmus adott számítási kapacitású architektúrán mekkora inputra fut még le **egy napon belül**.

					
C64	1Mflops	1Gflops	5TFlops	34PFlops	Föld bolygó 10EFlops
n	86.4G	8.64×10^{13}	4.32×10^{17}	2.94×10^{21}	8.64×10^{23}
$n \log n$	2.75G	2.1×10^{12}	8.1×10^{15}	4.5×10^{19}	1.17×10^{22}
n^2	300k	9.3M	657M	54G	929G
n^3	4420	44.208	756k	14.3M	95M
2^n	36	46	58	71	79
1.1^n	264	336	426	518	578
$n!$	14	16	19	22	24

Amíg Moore törvénye (nagyjából) igaz, addig a polinomidejű algoritmusok egy-egy újabb év elteltével konstansszor több adattal tudnak adott időn belül elbálni.

BONYOLULTSÁGELMÉLET – 2. ELŐADÁS

ELŐADÓ: GAZDAG ZSOLT

KÉSZÜLT ...

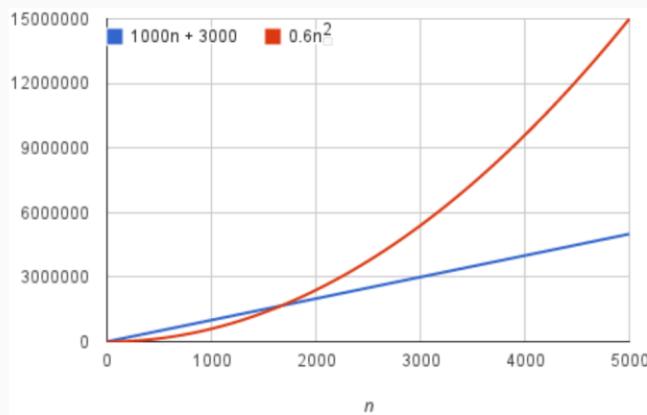
- IVAN SZABOLCS KORÁBBI ELŐADÁSFÓLIÁI, VALAMIINTI
- M. SIPSER: INTRODUCTION TO THE THEORY OF COMPUTATION ÉS
- C. PAPADIMITRIOU: SZÁMÍTÁSI BONYOLULTSÁG C. KÖNYVEI ALAPJÁN

ASZIMPTOTIKUS JELÖLÉSEK

Legyenek $f, g : \mathbb{N} \rightarrow \mathbb{N}$ függvények.

- $f(n) = \mathcal{O}(g(n))$, ha létezik olyan $c > 0$, hogy $f(n) \leq c \cdot g(n)$ minden n -re.
- $f(n) = \Omega(g(n))$, ha $g(n) = \mathcal{O}(f(n))$.
- $f(n) = \Theta(g(n))$, ha $f(n) = \mathcal{O}(g(n))$ és $g(n) = \mathcal{O}(f(n))$.

Ha $f(n) = \mathcal{O}(g(n))$, de $g(n) \neq \mathcal{O}(f(n))$, annak jele $f(n) = o(g(n))$. (és ω hasonlóan.)



$$1000n + 3000 = \mathcal{O}(0.6n^2)$$

ASZIMPTOTIKUS JELÖLÉSEK

Példa

Ha $p(n)$ és $q(n)$ polinomok (pozitív főegyütthatóval), akkor

- $p(n) = \mathcal{O}(q(n))$ pontosan akkor, ha p fokszáma legfeljebb akkora, mint q -é;
- $p(n) = \Theta(q(n))$ pontosan akkor, ha fokszámaik megegyeznek.

Legyen $a \in \mathbb{N}$, $a > 1$. Ekkor $p(n) = o(a^n)$.

Nagyobb fokú polinom jobban nő, exponenciális még jobban.

Példa

Legyen $c \in \mathbb{N}$. Ekkor minden $p(n)$ (pozitív főeggyütthatójú) polinomra

- $\log n^c = \mathcal{O}(\log n)$ és $\log n = o(p(n))$ és
- $p(n) = o(a^{\log n}) = o(a^n)$ minden $a \in \mathbb{N}, a > 1$ esetén

A P OSZTÁLY INFORMÁLIS DEFINÍCIÓJA

Egy probléma a **P osztályba** esik, ha megoldható polinom időigényű (vagy $\mathcal{O}(n^k)$ időigényű) algoritmussal.

Ezt majd precízebben is definiáljuk a tárgyalt algoritmus modellekre

A Cobham – Edmonds tézis

- Egy algoritmus akkor ad **gyakorlatilag kielégítő** megoldást egy problémára, ha polinom időigényű.
- Egy problémát akkor tekintünk **gyakorlatilag megoldhatónak**, ha a **P** osztályba esik.

A COBHAM – EDMONDS TÉZIS

A tézist lehet vitatni:

- Nem veszi figyelemben a konstansokat
 - Egy $10^{23}n$ időigényű algoritmus nyilván nem hatékony
- Nem veszi figyelembe a polinom fokszámát
 - Melyiket használnánk inkább, egy n^{200} időigényű vagy egy $2^{0,0001n}$ időigényű algoritmust?
 - A polinom értékét már $n = 2$ esetben sem tudjuk kiszámolni
 - Az exponenciális függvény értékét viszont akár $n = 100000$ esetén is
- Legrosszabb eset helyett a várható viselkedés vizsgálata is indokolt lehet
 - Például a SAT problémára nem ismert polinom idejű algoritmus
 - Mégis, a SAT Competition nevű versenyeken sokszor többszázezer klózzal kódolt valós életből vett problémákra sikerül hatékony megoldást adni

A COBHAM – EDMONDS TÉZIS

A tézist lehet **vitatni**:

- Néha **közelítő** megoldással is megelégszünk
 - Vegyük például az **UTAZÓÜGYNÖK** problémát:
 - **Adott**: az $1, 2, \dots, n$ városok és bármely két i, j város távolsága: $d_{i,j}$
 - **Feladat**: találunk egy, az összes várost pontosan egyszer érintő legrövidebb körutat.
 - Az **UTAZÓÜGYNÖK** problémára nem ismert polinom időigényű algoritmus
 - **Christofides algoritmusa** viszont $\mathcal{O}(n^3)$ időigényű, és garantáltan talál egy olyan megoldást, ami az optimum $3/2$ -szerese alatt van

Ugyanakkor

- A gyakorlatban előforduló **P**-beli problémák **rendje** kicsi
- A **P** osztály elegáns matematikai elmélethez vezet

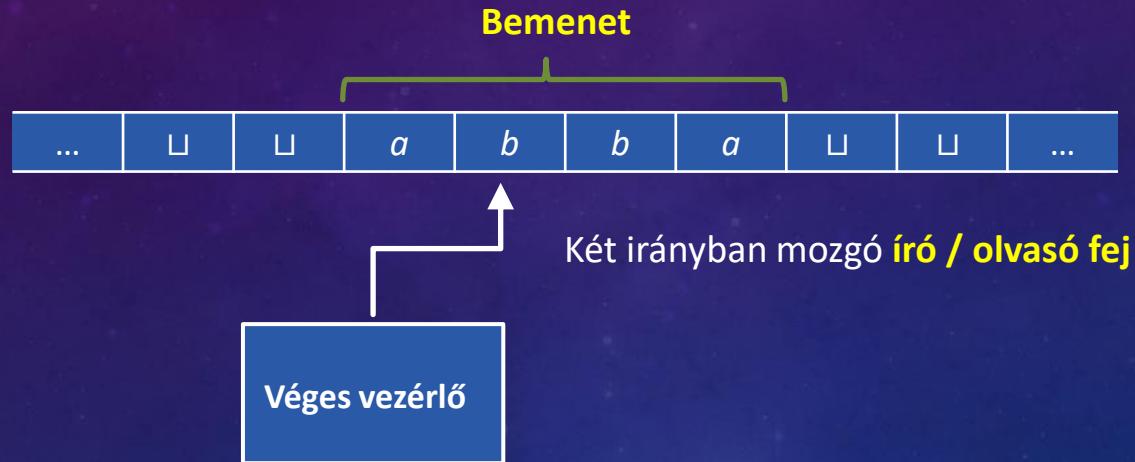
BETŰ, ÁBÉCÉ, SZAVAK, NYELVEK

Emlékeztető a **Formális nyelvek** kurzusról:

- **Ábécé:** véges, nem üres halmaz, elemei a betűk
- Egy Σ ábécé felett **szó:** Σ -beli betűk véges sorozata
 - Egy $u = a_1 \dots a_n$, $a_i \in \Sigma$, szó **hossza** n , jele $|u|$; a benne lévő a betűk száma $|u|_a$
 - Ha $n = 0$, akkor u az **üres szó**; jele ε
 - u **fordítottja:** $u^{-1} = a_n \dots a_1$
- Σ -feletti összes szó: Σ^* ; $\Sigma^+ = \Sigma^* - \{\varepsilon\}$
- Σ -feletti (formális) **nyelv:** Σ^* egy tetszőleges részhalmaza
- Tetszőleges H halmazra, $\mathcal{P}(H)$ a H **részalmazainak halmaza** (azaz, $\mathcal{P}(H) = 2^H$)

A KISZÁMÍTHATÓSÁG MATEMATIKAI MODELLJEI: A TURING-GÉP

Egy Turing-gép felépítése:



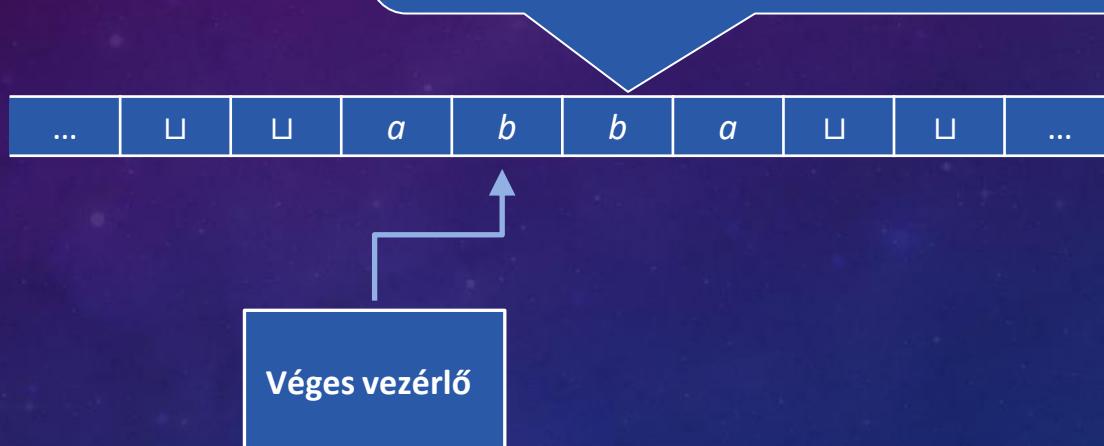
Két irányban
(potenciálisan)
végtelen szalag

- <https://www.youtube.com/watch?v=E3keLeMwfHY>
- <https://www.youtube.com/watch?v=FTSAiF9AHN4>

A TURING-GÉP

$$M = (Q, \Sigma, \Gamma, \delta, q_0, q_i, q_n)$$

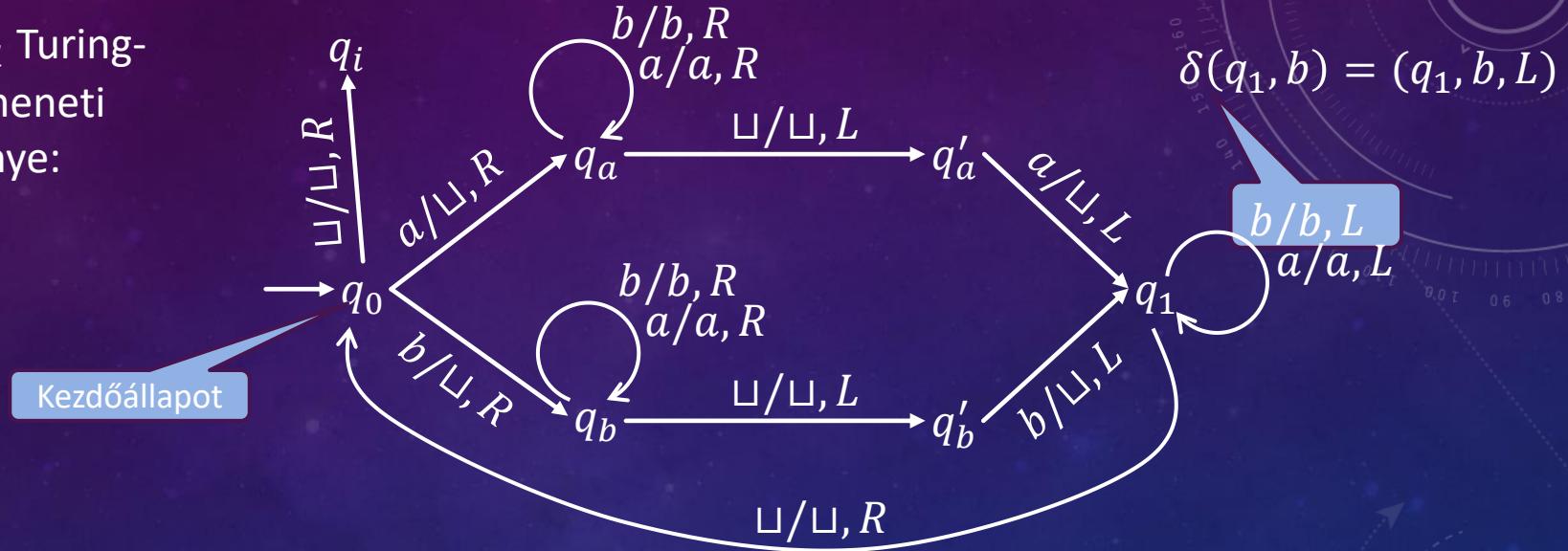
- A bemenet egy Σ ábécé feletti szó
- A szalagon előforduló szimbólumok egy Γ ábécé elemei
 $\Sigma \subset \Gamma$, $\sqcup \in \Gamma - \Sigma$, \sqcup : üres szimbólum, kezdetben ezek vannak a bemenet körül a szalagon



- A gép állapotai egy véges Q halmaz elemei
- Speciális állapotok Q -ban:
 - q_0 - kezdőállapot
 - q_i - elfogadó állapot
 - q_n - elutasító állapot
- A gép „programja” egy $\delta: Q' \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$ függvény, ahol $Q' = Q - \{q_i, q_n\}$

A TURING-GÉP – PÉLDA

Az M_{pal} Turing-gép átmeneti függvénye:



- A be nem rajzolt átmenetek q_n -be vezetnek
- Bemenő jelek: a, b
- Számítás az $abba$ szón:

$\sqcup abba \sqcup \Rightarrow \sqcup bba \sqcup \Rightarrow^3 \sqcup bba \sqcup \Rightarrow \sqcup bba \sqcup \Rightarrow \sqcup bb \sqcup \Rightarrow^2 \sqcup bb \sqcup \Rightarrow \sqcup bb \sqcup \Rightarrow^* \sqcup \Rightarrow \sqcup$

Kezdőkonfiguráció

Konfiguráció (mi van a szalagon, hol van a fej, mi az állapot)

Elfogadó konfiguráció

A TURING-GÉP - KONFIGURÁCIÓ

Az $M = (Q, \Sigma, \Gamma, \delta, q_0, q_i, q_n)$ Turing–gép egy konfigurációja egy hármas, ami leírja, hogy

- mi van a szalag nem csak \sqcup -ot tartalmazó részén,
- hol van a szalagon a fej,
- milyen állapotban van M

Egy konfiguráció megadható

- szemléletesen, mint az előző példában
- vagy uv alakban, ahol $u \in \Gamma^*$, $v \in \Gamma^+$, uv a szalagon lévő szó (tőle balra és jobbra csak \sqcup van), q az M aktuális állapota, a fej pedig v első betűjére mutat

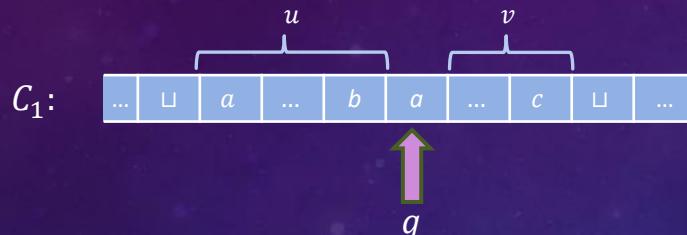
Egy uv konfiguráció

- Kezdőkonfiguráció, ha $v \in \Sigma^+ \cup \{\sqcup\}$, $q = q_0$ és $u = \epsilon$
- Elfogadó konfiguráció, ha $q = q_i$
- Elutasító konfiguráció, ha $q = q_n$
- Az elfogadó és elutasító konfigurációkat megállási konfigurációknak is nevezzük

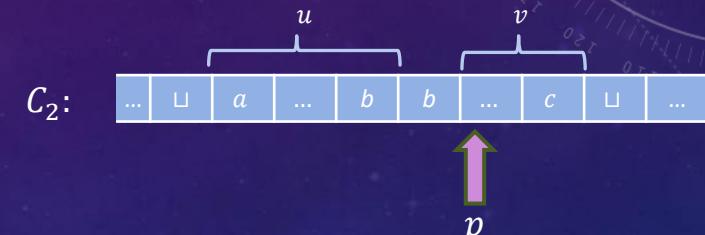
A TURING-GÉP - KONFIGURÁCIÓÁTMENET

Legyen C_1 és C_2 az $M = (Q, \Sigma, \Gamma, \delta, q_0, q_i, q_n)$ Turing-gép két konfigurációja

- C_2 közvetlenül elérhető C_1 -ből (jele: $C_1 \Rightarrow C_2$) ha

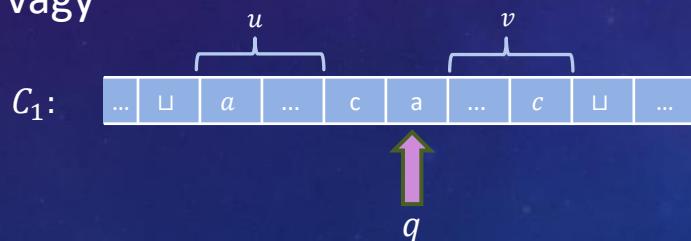


és $\delta(q, a) = (p, b, R)$

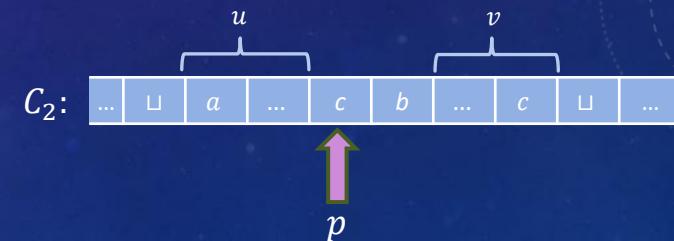


$(a, b \in \Gamma, u \in \Gamma^*, v \in \Gamma^+)$

vagy



és $\delta(q, a) = (p, b, L)$



$(a, b, c \in \Gamma, u, v \in \Gamma^*)$

A TURING-GÉP - KONFIGURÁCIÓÁTMENET

Többlépéses konfiguráció átmenet:

Legyen C, C' az M két konfigurációja

- C' elérhető C -ből, jele $C \Rightarrow^* C'$, ha

- vannak olyan C_1, \dots, C_n konfigurációk ($n \geq 1$) hogy

$$C = C_1 \Rightarrow C_2 \Rightarrow \dots C_{n-1} \Rightarrow C_n = C'$$

- M felismeri az $u \in \Sigma^*$ szót, ha az $q_0 u$ kezdőkonfigurációból elérhető egy elfogadó konfiguráció
- Az M által felismert nyelv:

$$L(M) := \{u \in \Sigma^* \mid M \text{ felismeri az } u\text{-t}\}$$

TURING-FELISMERHETŐ NYELVEK

Egy L nyelv Turing-felismerhető, ha van olyan M Turing gép, hogy $L=L(M)$

- Ha M minden bemeneten meg is áll, akkor L eldönthető

A Turing-felismerhető nyelveket ugyanazok, mint a **rekurzívan felsorolható nyelveknek**

- Ezen nyelvek osztályát RE -vel jelöljük
- Az eldönthető nyelvek osztályát pedig R -rel jelöljük

Itt is (ezek még csak nem is felismerhetőek)

Itt is vannak nyelvek (ezek nem eldönthetőek)
Pl. a Hilbert 10-es

RE

R

TURING-GÉP IDŐIGÉNY

Legyen $f: \mathbb{N} \rightarrow \mathbb{N}$ egy függvény, M pedig egy Turing-gép

- M időigénye $f(n)$, ha M minden n hosszú bemeneten legfeljebb $f(n)$ lépéssel megáll

Határozzuk meg M_{pal} által felismert nyelvet és az időigényét!

- Az M_{pal} működése:
 - Amíg q_0 -ban nem \sqcup -ra mutat a fej
 - Letöröl egy betűt a szalagon lévő szó elejéről és az új állapotban megjegyzi ezt a betűt
 - Elmegy a szó végére (ezt \sqcup jelzi)
 - Visszalép egyet (a fej pozíciója a szó utolsó betűje)
 - Ha a fej pozíóján a szó elején letörölt betű van, akkor letörli az utolsó betűt, és az új állapot q_1 , egyébként elutasítja a bemenetet és megáll
 - q_1 -ben visszamegy a szó elejére (ezt \sqcup jelzi) és q_0 -ba lép
 - Elfogadja a bemenetet
- A felismert nyelv: $L(M_{pal}) = \{uu^{-1} \mid u \in \{a, b\}^*\}$

- Az időigény: Legyen a bemenő szó u , a hossza n .
- M_{pal} u -n akkor lép a legtöbbet, ha elfogadja
- Ekkor a ciklusmag egyszeri végrehatása $O(n)$ lépés
- A ciklusmag legfeljebb $O(n)$ -szer hajtódik végre
- M_{pal} időigénye $O(n^2)$

BONYOLULTSÁGELMÉLET – 3. ELŐADÁS

ELŐADÓ: GAZDAG ZSOLT

KÉSZULT ...

- IVÁN SZABOLCS KORÁBBI ELŐADÁSFOLÍÁI, VALAMINT
- M. SIPSER: INTRODUCTION TO THE THEORY OF COMPUTATION ES
- C. PAPADIMITRIOU: SZÁMÍTÁSI BONYOLULTSÁG C. KÖNYVEI ALAPJÁN

TURING-GÉP VARIÁNSOK

Többszalagos Turing-gép:

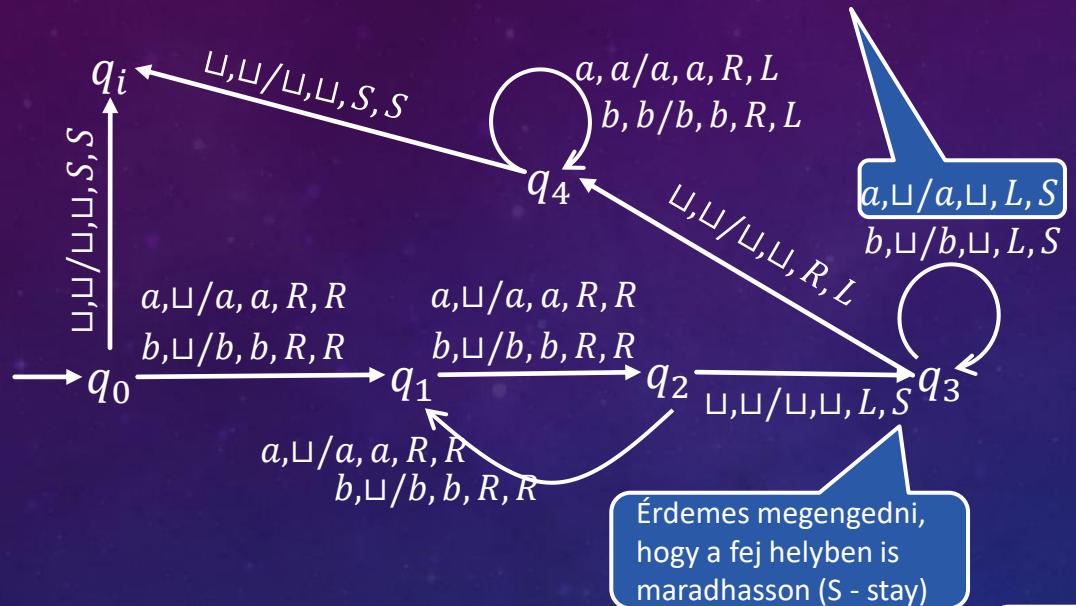
- Turing-gép k (konstans) számú szalaggal
- A szalagok mindegyike rendelkezik egy független író / olvasó fejjel
- A bemenet az első szalagra kerül, a többi szalag üres
- Az átmeneti függvény:

$$\delta: (Q - \{q_i, q_n\}) \times \Gamma^k \rightarrow Q \times \Gamma^k \times \{L, R\}^k$$

- A további komponensek mint az egyszalagos esetben
- Az egyszalagosra tanult definíciók kiterjeszhetők a többszalagos esetre (konfiguráció, konfiguráció-átmenet, felismert nyelv, időigény, stb.)

TÖBBSZALAGOS TURING-GÉP – PÉLDA

Az M_{pal2} kétszalagos Turing-gép átmeneti függvénye:



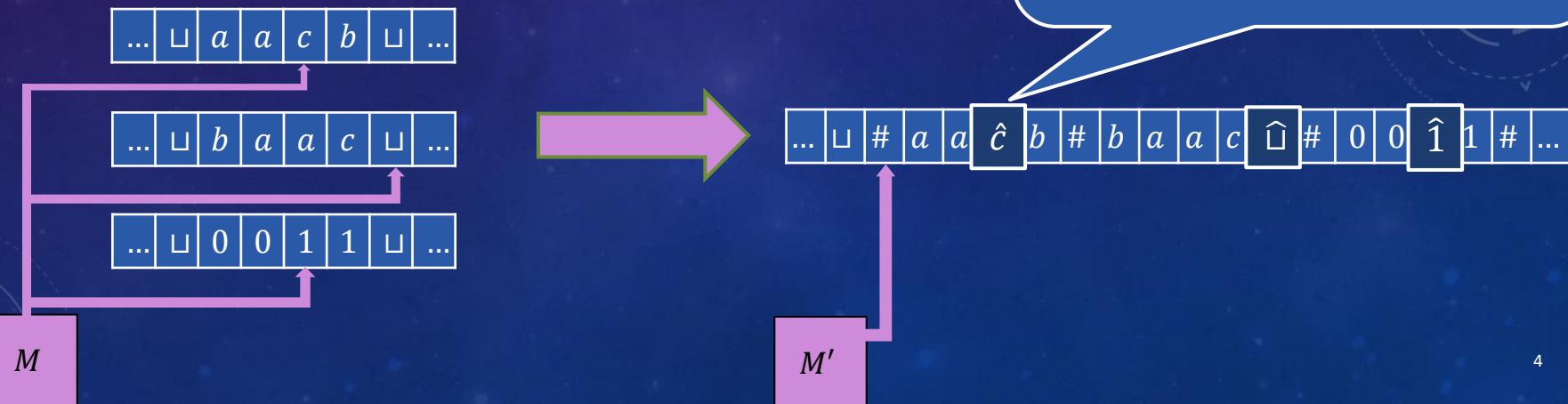
- A be nem rajzolt átmenetek q_n -be vezetnek
- bemenő jelek: a, b
- A felismert nyelv:**
$$L(M_{pal2}) = \{uu^{-1} \mid u \in \{a, b\}^*\}$$
- Időigény:** $O(n)$

M_{pal2} működése:

- Ha üres a bemenet, akkor elfogad, egyébként átmásol 1 betűt a 2-es szalagra és q_1 -be lép
- Ha a fej \sqcup -ra mutat, akkor elutasít, egyébként átmásol egy betűt a 2-es szalagra és q_2 -be lép
- Amíg van betű a 2-es szalagon
 - Átmásol két betűt a 2-es szalagra (ha csak egy betű van, akkor elutasít)
 - q_3 -ban az első szalagon a szó elejére megy, majd q_4 -be lép
 - q_4 -ben az első szalagon jobbra, a másodikon balra lépve összehasonlítja a két szalagon lévő szót
 - Ha megegyeznek, akkor elfogad, egyébként elutasít

TÖBBSZALAGOS TURING-GÉP

- Egy L nyelv $f(n)$ időben eldönthető, ha van olyan $f(n)$ időigényű, akár többszalagos Turing-gép, ami eldönti L -t
 - A P bonyolultsági osztály formálisan: a (Turing-géppel) $O(n^k)$, $k \in \mathbb{N}$, időben eldönthető problémák (nyelvek) osztálya
- Két Turing-gép ekvivalens, ha ugyanazt a nyelvet ismerik fel
- Állítás: minden k -szalagos M Turing-géphez van vele ekvivalens egyszalagos M' Turing-gép
- Bizonyítás (vázlat)
- M' egy szalagon tárolja az M teljes konfigurációját:



TÖBBSZALAGOS TURING-GÉP

- A szimuláció menete egy $a_1 a_2 \dots a_n$ bemeneten:

1. M' előállítja M kezdőkonfigurációját:



2. M' végigmegy a szalagon (számolja a #-okat) és eltárolja a ^-pal megjelölt szimbólumokat az állapotában
3. M' az M átmenetfüggvénye alapján aktualizálja a szalagját (ha M valamelyik szalagján nő a szó hozza, akkor M' -nek az adott ponttól mozgatnia kell a szalagja tartalmát jobbra)
4. Ha M elfogadó vagy elutasító állapotba lép, akkor M' is belép a saját elfogadó vagy elutasító állapotába
5. Egyébként M' folytatja a szimulációt a 2-ik ponttal

- Megjegyzés: M' időigény-romlása négyzetes

NEMDETERMINISZTIKUS TURING-GÉP

A két irányban végtelen szalaggal rendelkező, többszalagos Turing-gép általánosítása

- Az átmeneti függvény:

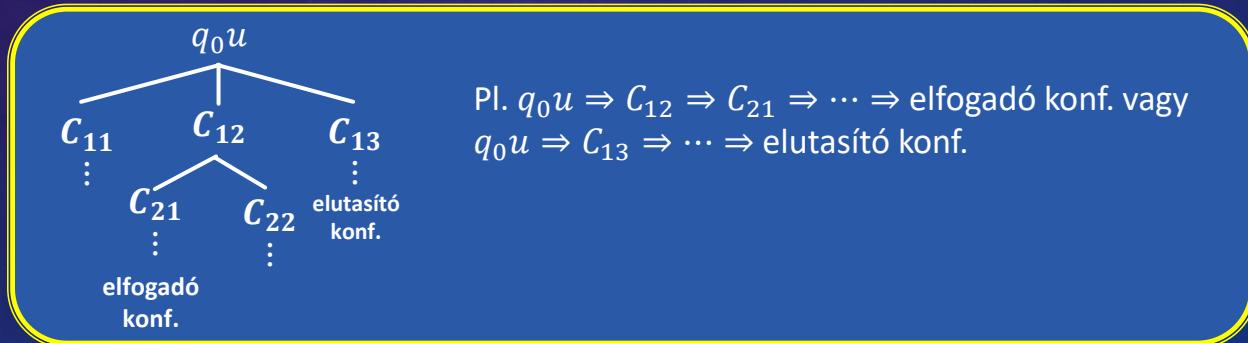
$$\delta: (Q - \{q_i, q_n\}) \times \Gamma^k \rightarrow P(Q \times \Gamma^k \times \{L, R\}^k)$$

$$\text{Azaz: } \delta(q, a_1, \dots, a_k) \subseteq Q \times \Gamma^k \times \{L, R\}^k$$

- Konfiguráció: ugyanaz, mint a determinisztikus esetben
- Konfiguráció átmenet: a determinisztikus esethez hasonlóan
- Például legyen M egy nemdeterminisztikus egyszalagos Turing gép és C_1, C_2 az M két konfigurációja.
 - C_2 közvetlenül elérhető C_1 -ből, jele $C_1 \Rightarrow C_2$, ha az alábbiak egyike teljesül:
 - $C_1 = uqav, C_2 = ubpv$ és $(p, b, R) \in \delta(q, a)$, ahol $a, b \in \Gamma, u \in \Gamma^*, v \in \Gamma^+$
 - $C_1 = ucqav, C_2 = upcbv$ és $(p, b, L) \in \delta(q, a)$, ahol $a, b, c \in \Gamma, u, v \in \Gamma^*$
 - A többlépéses konfiguráció átmenet (\Rightarrow^*) és a felismert nyelv is a determinisztikus esethez hasonlóan definíálható

NEMDETERMINISZTIKUS TURING-GÉP

- Egy M nemdeterminisztikus Turing-gép számítása egy u szón egy ún. számítási fával szemléltethető:
 - A gyökere az M kezdőkonfigurációja az u -n
 - A csúcsai M konfigurációi
 - Két szomszédos csúcs (azaz egy éssel összekötött csúcspár) megfelel M egy lehetséges konfigurációátmenetének (az élek a levelek felé irányulnak)
 - minden csúcsnak annyi gyermeke van ahány nemdeterminisztikus választás létezik az adott konfigurációban

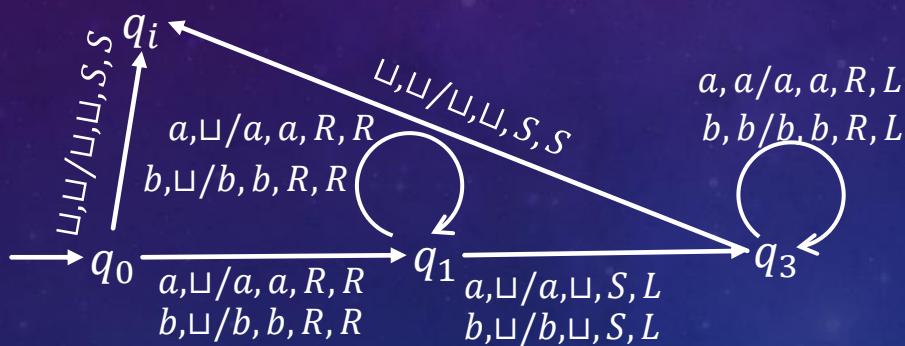


- M elfogadja az u -t ha a számítási fa legalább egy levele elfogadó konfiguráció

NEMDETERMINISZTIKUS TURING-GÉP

- M eldönti az $L \subseteq \Sigma^*$ nyelvet, ha felismeri és minden $u \in \Sigma^*$ szóra az M számítási fája véges és minden levele elfogadó vagy elutasító konfiguráció
- M $f(n)$ időigényű, ha minden $u \in \Sigma^*$ n hosszú szóra a számítási fa legfeljebb $f(n)$ magas
- Az NP bonyolultsági osztály formálisan: azon eldöntési problémák (nyelvek) osztálya, melyek eldönthetők $O(n^k)$, $k \in \mathbb{N}$, időigényű nemdeterminisztikus Turing-géppel

• Példa: M_{pal3}



- A felismert nyelv: $L(M_{pal3}) = \{uu^{-1} \mid u \in \{a, b\}^*\}$
- Időigény: $O(n)$

M_{pal3} működése:

- Ha üres a bemenet, akkor elfogad, egyébként átmásol 1 betűt a 2-es szalagra és q_1 -be lép
- q_1 -ben szétosztja a számítást:
 - Az egyik továbbra is másol q_1 -ben
 - A másik abbahagyja és q_3 -ba lép
- q_3 -ban az első szalagon jobbra, a másodikon balra lépve összehasonlítja a két szalagon lévő szót
- Ha megegyeznek, akkor elfogad, egyébként elutasít

NEMDETERMINISZTIKUS TURING-GÉP

Állítás: Legyen $M = (Q, \Sigma, \Gamma, \delta, q_0, q_i, q_n)$ egy nemdeterminisztikus Turing-gép

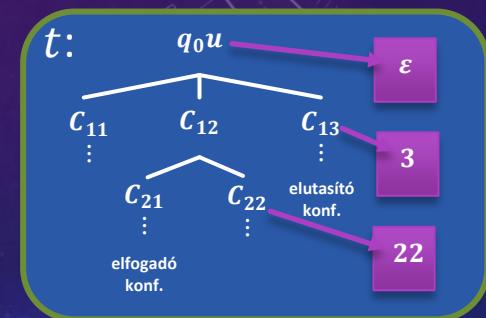
- Megadható egy M -mel ekvivalens M' determinisztikus Turing-gép
- Továbbá, ha M $f(n)$ időigényű, akkor M' $2^{O(f(n))}$ időigényű

Bizonyítás (vázlat)

- Legyen $u \in \Sigma^*$ és tekintsük az M t számítási fáját az u -n
- A t minden csúcsához egyértelműen hozzárendelhető egy $\{1, \dots, d\}^*$ -beli szó, ahol d a t kifokainak maximuma
- M' egy ciklusban az egyik szalagján lexikografikusan felsorolja a $\{1, \dots, d\}^*$ -beli szavakat egészen addig, amíg nem talál t -ben elfogadó konfigurációt:
 - legyen az aktuális szó w ; M' szimulálja a $q_0 u \Rightarrow^* C$ számítást, ahol C a w -vel címkézett konfiguráció t -ben
 - Ha C elfogadó, akkor M' is elfogadó állapotba lép, egyébként átirja w az ōt lexikografikusan követő szóra

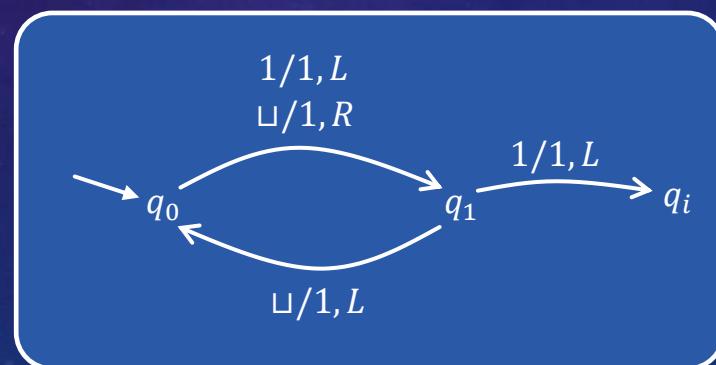
Következmény: Ha egy L nyelv eldönthető nemdeterminisztikus Turing-géppel, akkor eldönthető determinisztikussal is

Megjegyzés: Az a sejtés (még nincs bebizonyítva), hogy nem lehet egy nemdeterminisztikus Turing-gépet determinisztikus Turing-géppel az előbb látottnál hatékonyabban szimulálni



TURING-GÉP - BONYOLULT VAGY EGYSZERŰ

- Bár a Turing-gép egy nagyon egyszerű algoritmus modell, már a vele kapcsolatos legegyszerűbb kérdések megválaszolása is kihívás lehet
- Ezt demonstrálja a „Szorgos Hód” játék:
 - Adott $n \geq 2$ számhoz keressük azt az n állapotú Turing-gépet (n állapot plusz egy amiben megáll), ami bizonyíthatóan a legtöbbet lépi, mielőtt megáll
 - A gépet üres bemenettel kell indítani
 - Szalagszimbólumként csak \sqcup és 1 használható
- A 2 állapotú „Szorgos Hód”
 - Vajon mennyit lép és mennyi 1-est ír a szalagra, amíg megáll:

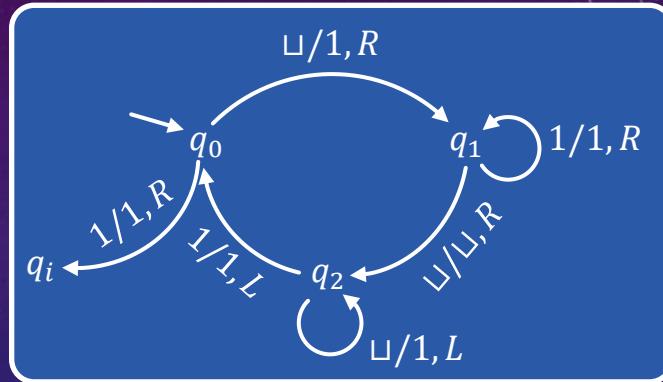


- Válasz: 4 darab 1-es 6 lépés után

TURING-GÉP - BONYOLULT VAGY EGYSZERŰ

A 3 állapotú „Szorgos Hód”

- Vajon mennyit lép és mennyi 1-est ír a szalagra, amíg megáll:
- Válasz: 6 darab 1-es 14 lépés után



A 4 állapotú „Szorgos Hód”

- 13 1-est ír a szalagra 107 lépés után

A legjobb 5 állapotú jelölt:

- 4098 1-est ír a szalagra 47.176.870 lépés után

A legjobb 6 állapotú jelölt:

- $\approx 3.5 \cdot 10^{18267}$ 1-est ír a szalagra $\approx 7.4 \cdot 10^{36534}$ lépés után

ELDÖNTÉSI PROBLÉMA MINT FORMÁLIS NYELV

Jelölés: Tetszőleges D objektumra $\langle D \rangle$ jelöli a D egy tömör elkódolását egy megfelelő szóban

- (D lehet formula, gráf, Turing-gép – lásd később, vagy bármilyen végesen reprezentálható dolog)
- Tömör elkódolás: például ha egy számot binárisan adunk meg (unárisan nem tömör)

Egy P eldöntési probléma megfeleltethető egy

$$L_P := \{\langle I \rangle \mid I \text{ a } P \text{ pozitív bemenete}\} \text{ formális nyelvnek}$$

Például

- A SAT problémának megfeleltethető a $\{\langle \varphi \rangle \mid \varphi \text{ egy kielégíthető ítéletkalkulusbeli konjunktív nf.}\}$ nyelv
- $L_u = \{\langle M, w \rangle \mid w \in L(M)\}$ (M egy Turing-gép w pedig ennek egy bemenete) megfelel azon problémának, ahol a feladat annak eldöntése, hogy egy M TG elfogadja-e a w bemenetet
 - Ezt szokás univerzális nyelvnek hívni
- $L_{\text{átıló}} = \{\langle M \rangle \mid \langle M \rangle \notin L(M)\}$ (M egy Turing-gép) megfelel azon problémának, ahol a feladat annak eldöntése, hogy egy M Turing-gép elfogadja-e saját maga kódját bemenetként

A TURING-GÉPEK ELKÓDOLÁSÁRÓL

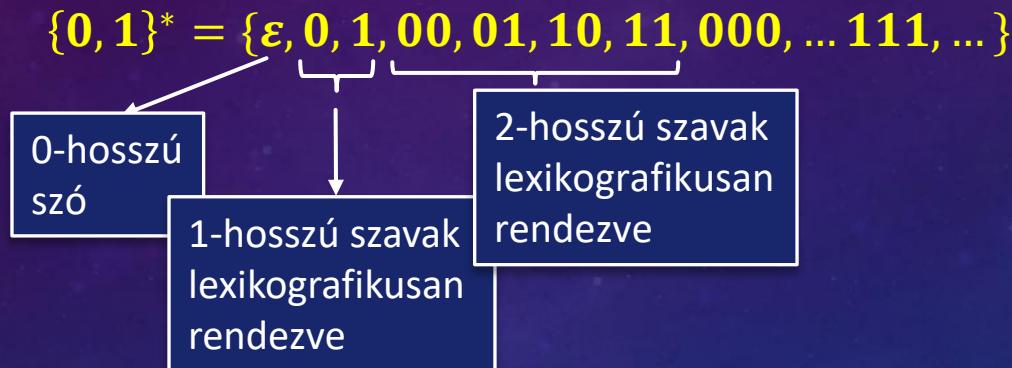
- Megjegyzés: Ebben a részben csak $\{0,1\}$ bemenő ábécével rendelkező Turing-gépeket vizsgálunk
- Az $\langle M, w \rangle$ és $\langle M \rangle$ definíciója:
- Legyen $M = (Q, \{0,1\}, \Gamma, \delta, q_0, q_i, q_n)$ egy Turing-gép
 - Q és Γ minden eleméhez rendelhetünk egy sorszámat
 - a fej L és R irányait tekinthetjük rendre az 1-es és 2-es iránynak
- M minden $\delta(q, a) = (p, b, D)$ átmenete **egyértelműen** elkódolható a következő szóval:

0 ... 0 1 0 ... 0 1 0 ... 0 1 0 ... 0 1 0 ... 0
q sorszáma db 0 a sorszáma db 0 ... az irány sorszáma db 0

- $\langle M \rangle$ kódja: $\langle \delta_1 \rangle 11 \langle \delta_2 \rangle 11 \dots 11 \langle \delta_l \rangle$, ahol $\delta_1, \dots, \delta_l$ az M összes átmenete, $\langle \delta_i \rangle$ az i -ik átmenet kódja
- $\langle M, w \rangle$ kódja: $\langle M \rangle 111w$

A TURING-GÉPEK ELKÓDOLÁSÁRÓL

- A $\{0,1\}^*$ elemei felsorolhatóak, pl. lexikografikusan:



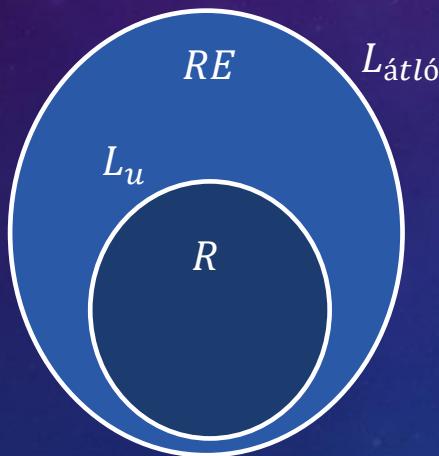
- Jelölés: minden $i \geq 1$ -re,
 - w_i jelöli a $\{0,1\}^*$ halmaz i -ik elemét
 - M_i jelöli a w_i által kódolt Turing-gépet (ha w_i nem kódol Turing-gépet, akkor M_i egy olyan tetszőleges Turing-gép, ami nem fogad el semmit)

AZ RE SZERKEZETE

Emlékeztető:

- R az eldönthető, RE pedig a Turing-felismerhető nyelvek osztálya
- $R \subseteq RE$

A célunk megmutatni, hogy:



$$L_u = \{\langle M, w \rangle \mid w \in L(M)\}$$

$$L_{\text{atló}} = \{\langle M \rangle \mid \langle M \rangle \notin L(M)\}$$

ELDÖNTHETETLEN PROBLÉMÁK

Állítás: $L_{\text{átló}}$ nem ismerhető fel Turing-géppel

Bizonyítás:

- Tekintsük azt a T táblázatot, mely i -ik sorának j -ik oszlopa ($i, j \geq 1$), azaz $T(i, j)$, akkor és csak akkor 1, ha $w_j \in L(M_i)$
- Legyen d a T átlójában olvasható végtelen hosszú bitsztring, és legyen \bar{d} a d bitenkénti komplementere
- Ekkor igazak az alábbi megjegyzések:
 - minden $i \geq 1$ -re, T i -ik sora az $L(M_i)$ nyelv karakterisztikus függvénye
 - \bar{d} az $L_{\text{átló}}$ karakterisztikus függvénye
 - minden Turing-géppel felismerhető, azaz RE -beli nyelv karakterisztikus függvénye megegyezik T valamelyik sorával
 - \bar{d} különbözik T minden sorától
- Ezek alapján $L_{\text{átló}}$ különbözik az összes RE -beli nyelvtől

BONYOLULTSÁGELMÉLET – 4. ELŐADÁS

ELŐADÓ: GAZDAG ZSOLT

KÉSZÜLT ...

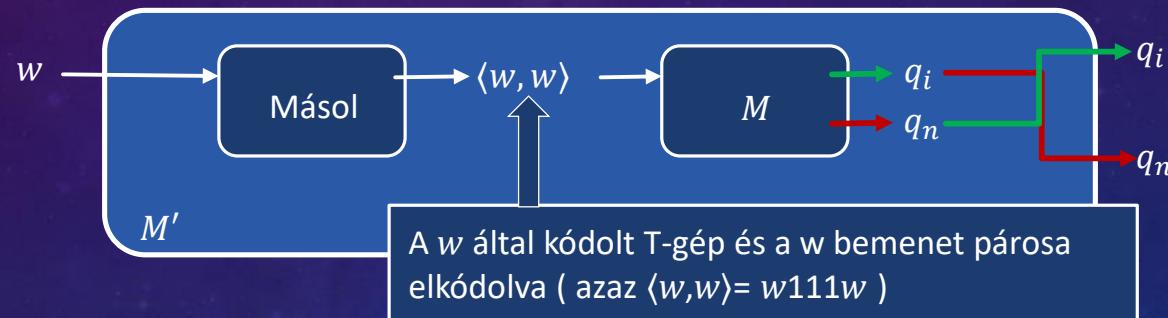
- IVÁN SZABOLCS KORÁBBI ELŐADÁSFÓLIÁI, VALAMINT
- M. SIPSER: INTRODUCTION TO THE THEORY OF COMPUTATION ÉS
- C. PAPADIMITRIOU: SZÁMÍTÁSI BONYOLULTSÁG C. KÖNYVEI ALAPJÁN

ELDÖNTHETETLEN PROBLÉMÁK

Állítás: Az L_u nyelv eldönthetetlen

Bizonyítás:

- Indirekt módon tfh. $L_u \in R$ és legyen M egy L_u -t eldöntő Turing-gép
- Konstruáljuk meg M' -t:



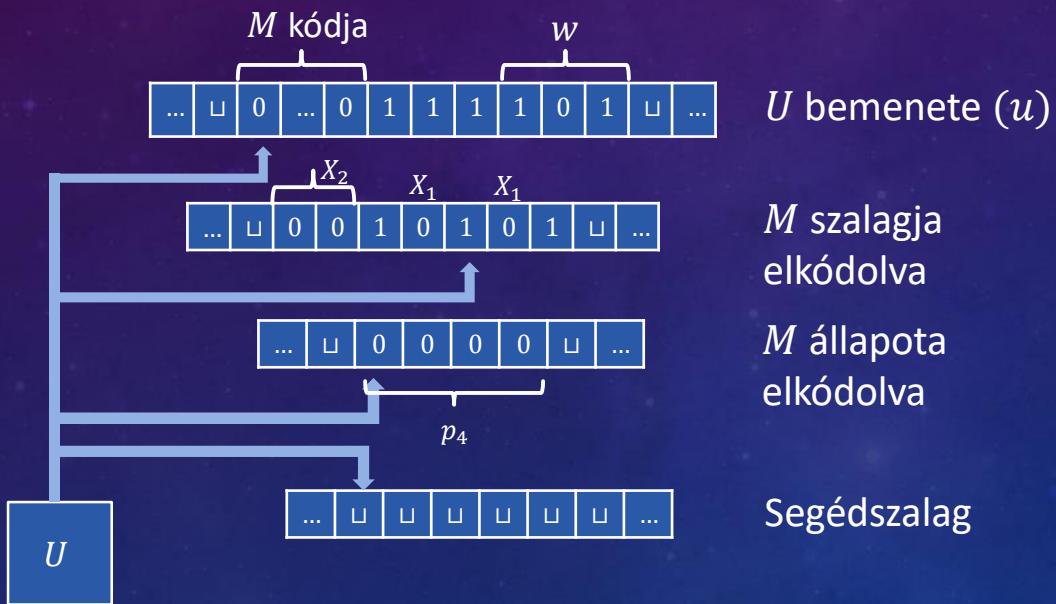
- Az M' elfogadja w -t $\Leftrightarrow M$ nem fogadja el a $\langle w, w \rangle$ szót \Leftrightarrow A w által kódolt Turing-gép nem fogadja el w -t
 - Az első \Leftrightarrow esetében fontos, hogy M minden bemeneten megáll!
- Azaz $w \in L(M') \Leftrightarrow w \in L_{\text{átlo}}$
- Tehát $L(M') = L_{\text{átlo}}$, ami ellentmondás, mert $L_{\text{átlo}}$ -t nem lehet felismerni
- Következik, hogy L_u -t nem lehet eldöntení

ELDÖNTHETETLEN PROBLÉMÁK

Állítás: Az L_u felismerhető Turing-géppel

Bizonyítás:

- L_u -t egy ún. Univerzális Turing-gép ismeri fel:



U (vázlatos) működése:

- Létrehozza M kezdőkonfigurációját elkódolva a 2-es (M szalagja) és 3-as (M állapota) szalagon
- Szimulálja M egy lépését:
 - Leolvassa a második szalagról M aktuálisan olvasott szalagszimbólumát
 - Megnézi, hogy a bemenetén szereplő u szó kódol-e Turing-gépet; ha nem elutasítja a bemenetet
 - Leolvassa a harmadik szalagról M aktuális állapotát
 - Szimulálja M egy lépését (M ott van az 1-es szalagon)
 - Ha M aktuális állapota elfogadó vagy elutasító, akkor U is belép a saját elfogadó vagy elutasító állapotába

Megjegyzés: ha M nem áll meg w -n, akkor U sem áll meg $\langle M, w \rangle$ -n

ELDÖNTHETETLEN PROBLÉMÁK

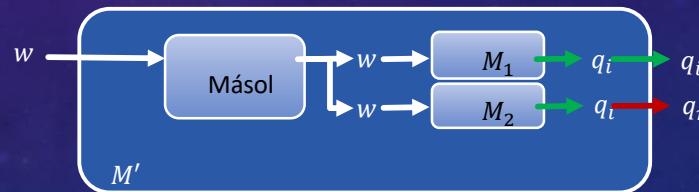
Tehát egy L eldöntési probléma komplementere az az \bar{L} probléma aminek pozitív bemenetei az L negatív bemenetei, negatív bemenetei pedig az L pozitív bemenete i

Jelölés: Ha $L \subseteq \Sigma^*$, akkor L komplementere az $\bar{L} := \{u \in \Sigma^* \mid u \notin L\}$

Állítás: Ha L és \bar{L} is felismerhető Turing-géppel, akkor L eldönthető

Bizonyítás: Legyen M_1 és M_2 rendre az L -t és \bar{L} -t felismerő Turing-gép

- Konstruáljuk meg az M' kétszalagos Turing-gépet:



- M' lemásolja w -t a második szalagjára, majd felváltva szimulálja M_1 és M_2 egy-egy lépését addig, amíg valamelyik elfogadó állapotba lép
- Így M' az L -et ismeri fel, és minden bemeneten meg is áll, azaz M el is dönti L -et

A KOMPLEMENTER PROBLÉMÁK OSZTÁLYA

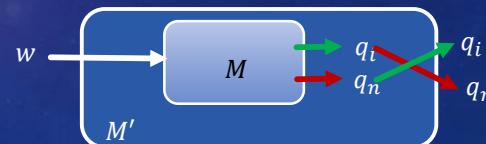
Jelölés: Tetszőleges C problémaosztályra coC jelöli a C -beli problémák komplementereinek az osztályát

Következmény: $RE \neq coRE$

- Tudjuk, hogy $L_u \in RE$, ezért elég belátni, hogy $\overline{L_u} \notin RE$
- Indirekt módon tegyük fel, hogy $\overline{L_u} \in RE$
- Akkor a korábbi állításunk alapján $L_u \in R$
- De azt tanultuk, hogy $L_u \notin R$, tehát $\overline{L_u} \notin RE$

Állítás: $R = coR$

- Legyen $L \in R$ és M egy Turing-gép, ami az L -t dönti el
- Akkor a következő M' \overline{L} -t dönti el:



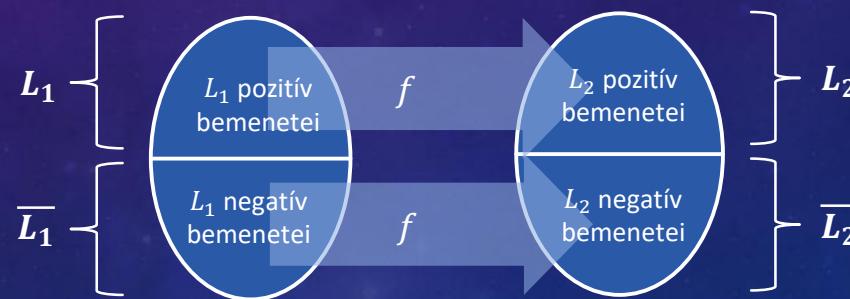
Hasonlóan látható be, hogy $P = coP$

Ugyanakkor az a sejtés, hogy $NP \neq coNP$

VISSZAVEZETÉSEK

- Egy $f: \Sigma^* \rightarrow \Delta^*$ függvényt kiszámíthatónak nevezünk, ha van olyan Turing-gép, ami egy $u \in \Sigma^*$ szóval az első szalagján elindítva úgy áll meg, hogy az utolsó szalagján az $f(u)$ szó van
- Egy $L_1 \subseteq \Sigma^*$ nyelv visszavezethető egy $L_2 \subseteq \Delta^*$ nyelvre, ha van olyan $f: \Sigma^* \rightarrow \Delta^*$ kiszámítható függvény, hogy minden $u \in \Sigma^*$ szóra,

ha $u \in L_1$, akkor $f(u) \in L_2$ és ha $u \in \bar{L}_1$, akkor $f(u) \in \bar{L}_2$



- Azt, hogy L_1 visszavezethető L_2 -re röviden így jelöljük: $(L_1 \leq L_2)$

Megjegyzés: Ha $L_1 \leq L_2$, akkor nyilvánvalóan $\bar{L}_1 \leq \bar{L}_2$ is teljesül

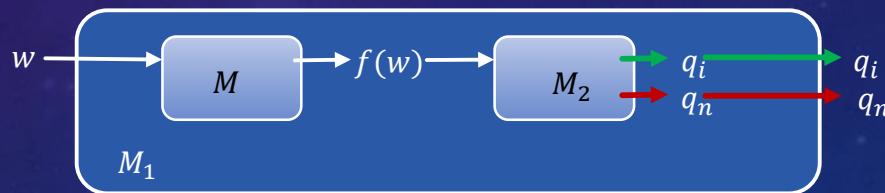
VISSZAVEZETÉSEK

Mit mondhatunk el egy probléma megoldhatóságáról, ha tudjuk, hogy visszavezethető rá egy másik probléma vagy ő vezethető vissza egy másik problémára?

Állítás: Ha $L_1 \leq L_2$ és $L_2 \in RE$ akkor $L_1 \in RE$, továbbá ha L_2 eldönthető, akkor L_1 is az

Bizonyítás:

- Legyen $L_2 \in RE$
- Legyen M_2 az L_2 -t felismerő, M pedig a visszavezetést kiszámító Turing-gép
- Konstruáljuk meg M_1 -et:



- Világos, hogy M_1 az L_1 -et ismeri fel, azaz $L_1 \in RE$
- Ha $L_2 \in R$, akkor M_2 választható olyannak, hogy minden bemeneten megáll
- Akkor M_1 is minden bemeneten megáll, tehát eldönti L_1 -et

Következmény: Ha L_1 eldönthetetlen, akkor L_2 is az

A TURING-GÉPEK MEGÁLLÁSI PROBLÉMÁJA

Az alábbi nyelvet a **Turing-gépek megállási problémájának** nevezik:

$$L_h = \{\langle M, w \rangle \mid M \text{ megáll } w\text{-n}\}$$

Eldöntési problémaként megfogalmazva:

- Adott egy M Turing-gép és M egy w bemenete
- Kérdés: Megáll-e M a w -n?

Fontos: egy Turing-gép csak a q_i vagy q_n állapotok valamelyikében tud megállni

Állítás: L_h eldönthetetlen

Bizonyítás: Korábbi téTEL ALAPJÁN elég megmutatni, hogy $L_u \leq L_h$

- Tetszőleges M Turing-gépre, legyen M' az alábbi Turing-gép
 - M' tetszőleges u bemeneten a következőket teszi:
 1. Futtatja M -et u -n (úgy, ahogy azt az univerzális Turing-gép teszi)
 2. Ha M q_i -be lép, akkor M' is q_i -be lép
 3. Ha M q_n -be lép, akkor M' egy végtelen ciklusba lép

A TURING-GÉPEK MEGÁLLÁSI PROBLÉMÁJA

Bizonyítás (folyt.)

- Belátható, hogy

1. Az $f: \langle M \rangle \rightarrow \langle M' \rangle$ leképezés egy kiszámítható függvény

2. Tetszőleges M, w Turing-gép — bemenet párosra:

$$\langle M, w \rangle \in L_u \Leftrightarrow M \text{ megáll } w\text{-n } q_i\text{-ben} \Leftrightarrow \text{Az } M' \text{ megáll } w\text{-n} \Leftrightarrow \langle M', w \rangle \in L_h$$

- Tehát az M' konstrukciója az L_u visszavezetése L_h -ra
- Mivel L_u eldönthetetlen, következik, hogy L_h is az

Megjegyzés: Visszavezetések megadásakor jellemzően csak azon szavakat vizsgáljuk, amelyek kódolnak valamilyen objektumot

- Pl. a fenti esetben nem foglalkoztunk azzal, hogy f mit rendeljen egy olyan szóhoz, ami nem kódol Turing-gépet (ami egyébként egy könnyen kezelhető eset)

A TURING-GÉPEK MEGÁLLÁSI PROBLÉMÁJA

Állítás: $L_h \in RE$

Bizonyítás: Korábbi téTEL alapján elég megmutatni, hogy $L_h \leq L_u$

- Tetszőleges M Turing-gépre, legyen M' az alábbi Turing-gép:
 - M' tetszőleges u bemeneten a következőket teszi:
 1. Futtatja M -et u -n
 2. Ha M q_i -be vagy q_n -be lép, akkor M' q_i -be lép
 - Belátható, hogy
 1. Az $f: \langle M \rangle \rightarrow \langle M' \rangle$ leképezés egy kiszámítható függvény
 2. Tetszőleges M, w Turing-gép — bemenet párosra,
$$\langle M, w \rangle \in L_h \Leftrightarrow \text{Az } M \text{ megáll } w\text{-n} \Leftrightarrow \text{Az } M' \text{ elfogadja } w\text{-t} \Leftrightarrow \langle M', w \rangle \in L_u$$
 - Tehát M' konstrukciója az L_h visszavezetése L_u -ra
 - Következik, hogy $L_h \in RE$

MEGÁLLÁS AZ ÜRES SZÓN

Láttuk, hogy az nem eldönthető, vajon megáll-e egy Turing-gép egy tetszőleges szón

- Az vajon eldönthető, hogy megáll-e egy konkrét szón, mondjuk az üres szón?
- A „szorgos hód” játék óta tudjuk, hogy ez sem lehet egyszerű feladat

Állítás: $L_\varepsilon \notin R$, ahol $L_\varepsilon = \{\langle M \rangle \mid M \text{ megáll az } \varepsilon \text{ bemeneten}\}$

Bizonyítás: Mivel $L_h \notin R$, egy korábbi állítás alapján elég megmutatni, hogy $L_h \leq L_\varepsilon$

- Tetszőleges M, w Turing-gép — bemenet párosra, legyen M' az alábbi Turing-gép:
 - M' tetszőleges u bemeneten a következőket teszi:
 1. Megnézi, hogy $u = \varepsilon$ teljesül-e; ha nem, akkor q_n -be lép;
 2. Egyébként az első szalagra írja a w -t és futtatja rajta M -et;
 3. Ha M q_i vagy q_n állapotba lép, akkor M' belép a saját elfogadó állapotába.
 - Belátható, hogy
 1. Az $f: \langle M, w \rangle \rightarrow \langle M' \rangle$ leképezés egy kiszámítható függvény
 2. Tetszőleges M, w Turing-gép — bemenet párosra,
$$\langle M, w \rangle \in L_h \Leftrightarrow \text{Az } M \text{ megáll } w\text{-n} \Leftrightarrow \text{A } M' \text{ elfogadja } \varepsilon\text{-t} \Leftrightarrow \langle M' \rangle \in L_\varepsilon$$
 - Tehát M' konstrukciója az L_h visszavezetése L_ε -ra
 - Következik, hogy $L_\varepsilon \notin R$

LEGALÁBB EGY SZÓ ELFOGADÁSA

Láttuk, hogy nem lehet eldönteni, hogy egy Turing-gép megáll-e az üres szón

- Hasonlóan látható be, hogy nem eldönthető az, hogy elfogad-e egy konkrét szót
- De az vajon eldönthető-e, hogy **elfogad-e egyáltalán bármit?**

Állítás: $NE_{TG} \notin R$, ahol $NE_{TG} = \{\langle M \rangle \mid L(M) \neq \emptyset\}$

Bizonyítás: Mivel $L_u \notin R$, egy korábbi állítás alapján elég megmutatni, hogy $L_u \leq NE_{TG}$

- Tetszőleges M, w Turing-gép — bemenet párosra, legyen M' az alábbi Turing-gép:
 - M' tetszőleges u bemeneten a következőket teszi:
 1. Megnézi, hogy $u = w$ teljesül-e; ha nem, akkor elutasítja a bemenetet;
 2. Egyébként futtatja M -et u -n (azaz w -n);
 3. Ha M q_i -be vagy q_n -be lép, akkor M' is rendre belép a saját elfogadó vagy elutasító állapotába.
- Belátható, hogy
 1. Az $f: \langle M, w \rangle \rightarrow \langle M' \rangle$ leképezés egy kiszámítható függvény
 2. Tetszőleges M, w Turing-gép — bemenet párosra,
 $\langle M, w \rangle \in L_u \Leftrightarrow$ Az M elfogadja w -t \Leftrightarrow A M' elfogadja w -t $\Leftrightarrow M'$ elfogad legalább egy szót $\Leftrightarrow \langle M' \rangle \in NE_{TG}$
- Tehát M' konstrukciója az L_u visszavezetése E_{TG} -re
- Következik, hogy $NE_{TG} \notin R$

AZ ÜRES NYELVET FELISMERŐ TURING-GÉPEK NEM FELISMERHETŐK

Legyen $E_{TG} = \overline{NE_{TG}}$

- Tehát E_{TG} azon Turing-gépek kódjait tartalmazza, melyek nem fogadnak el egyetlen szót sem (esetleg olyan szavakat, melyek nem is kódolnak Turing-gépet)
- Megmutatjuk, hogy $E_{TG} \notin RE$, azaz annak eldöntésére, hogy egy Turing-gép nem fogad el semmit még félig-eldöntő algoritmus sem adható
- Ehhez elég megmutatni, hogy $NE_{TG} \in RE$ (mert ha E_{TG} és NE_{TG} is RE -beli lenne, akkor mindenki előnnyel lenne, ami ellentmondás)

Állítás: $NE_{TG} \in RE$

- Bizonyítás: Legyen N az alábbi Turing-gép
- N egy $\langle M \rangle$ szóval a bemenetén, ahol M egy tetszőleges Turing-gép, a következőt teszi
- minden $i = 1, 2, 3, \dots$ számra
 - Futtatja M -et i lépésig a w_1, \dots, w_i szavak mindegyikén (emlékeztető: w_i a $\{0,1\}$ -feletti szavak felsorolásában az i -ik)
 - Ha eközben M elfogad, N is elfogadó állapotba lép
- Így N akkor és csak akkor fogadja el $\langle M \rangle$ -et, ha M elfogad legalább egy szót
- Tehát N az NE_{TG} nyelvet ismeri fel

A TURING-GÉPEK EKVIVALENCIÁJA NEM FELISMERHETŐ

Legyen $EQ_{TG} = \{\langle M_1, M_2 \rangle \mid M_1 \text{ és } M_2 \text{ ekvivalens Turing-gépek}\}$

Állítás: $EQ_{TG} \notin RE$

Bizonyítás: Mivel $E_{TG} \notin RE$, korábbi eredményünk alapján elegendő megmutatni, hogy $E_{TG} \leq EQ_{TG}$

- Tetszőleges M Turing-géphez legyen
 - $M_1 := M$ és
 - M_2 egy tetszőleges Turing-gép, ami az üres nyelvet ismeri fel
- Világos, hogy
 1. Az $f: \langle M \rangle \rightarrow \langle M_1, M_2 \rangle$ leképezés egy kiszámítható függvény és
 2. tetszőleges M Turing-gépre

$$\langle M \rangle \in E_{TG} \Leftrightarrow M \text{ nem fogad el egy szót sem} \Leftrightarrow M \text{ ekvivalens } M_2\text{-vel} \Leftrightarrow \langle M_1, M_2 \rangle \in EQ_{TG}$$

- Tehát M_1 és M_2 konstrukciója E_{TG} visszavezetése EQ_{TG} -re
- Következik, hogy $EQ_{TG} \notin RE$

RICE TÉTELE

Eldönthető egyáltalán bármilyen kérdés a Turing-gépek által felismert nyelvekkel kapcsolatban?

- Egy ilyen kérdés **triviális** ha minden Turing-gép által felismerhető nyelvre *igen* a válasz vagy mindenre *nem* a válasz
- Valójában mindenre **két triviális kérdést** tudunk megfogalmazni egy tetszőleges M Turing-géppel kapcsolatban:
 - Vajon M RE -beli nyelvet fogad-e el? (triviális, hogy a válasz erre *igen*) és az, hogy
 - vajon M RE -n kívüli nyelvet fogad-e el? (triviálisan, hogy erre a válasz *nem*)
- **Nem triviális** kérdések például: vajon az M által felismert nyelv
 - üres-e?
 - véges-e?
 - reguláris-e?

Rice tétele: A Turing-gépek által felismert nyelvekkel kapcsolatos összes nem triviális kérdés **eldönthetetlen**

BONYOLULTSÁGELMÉLET – 5. ELŐADÁS

ELŐADÓ: GAZDAG ZSOLT

KÉSZÜLT ...

- IVÁN SZABOLCS KORÁBBI ELŐADÁSFÓLIÁI, VALAMINT
- M. SIPSER: INTRODUCTION TO THE THEORY OF COMPUTATION ÉS
- C. PAPADIMITRIOU: SZÁMÍTÁSI BONYOLULTSÁG C. KÖNYVEI ALAPJÁN

TOVÁBBI ELDÖNTETLEN PROBLÉMÁK – A POST MEGFELELKEZÉSI PROBLÉMÁJA

Legyen Σ egy tetszőleges ábécé és $u_1, \dots, u_n, v_1, \dots, v_n \in \Sigma^+$ ($n \geq 1$)

- A $D = \left\{ \frac{u_1}{v_1}, \dots, \frac{u_n}{v_n} \right\}$ halmazt **dominókészletnek** nevezik
- Az $\frac{u_{i_1}}{v_{i_1}} \dots \frac{u_{i_m}}{v_{i_m}}$ ($m \geq 1, 1 \leq i_1, \dots, i_m \leq n$) dominósorozat a D **egy megoldása**, ha
$$u_{i_1} \dots u_{i_m} = v_{i_1} \dots v_{i_m}$$

Példa: a $\left\{ \frac{b}{ca}, \frac{a}{ab}, \frac{ca}{a}, \frac{abc}{c} \right\}$ dominókészlet egy megoldása: $\frac{a}{ab} \frac{b}{ca} \frac{ca}{a} \frac{a}{ab} \frac{abc}{c}$

A **PMP** probléma

- Adott egy D dominókészlet
- Van-e D -nek megoldása?

Állítás: *PMP* Turing-felismerhető, de nem eldönthető

- Félíg eldönthető, mert a potenciális megoldások felsorolhatók és tesztelhetők
- Nem eldönthető, mert vissza lehet vezetni rá az L_u problémát

TOVÁBBI ELDÖNTHETETLEN PROBLÉMÁK – A CF NYELVTANOK EGYÉRTELMŰSÉGE

ECF probléma: Adott egy G CF nyelvtan, döntsük el, hogy G egyértelmű-e

- **Emlékeztető:** egy $G = (N, \Sigma, P, S)$ CF nyelvtan az S kezdő nemterminálisból kiindulva vezet le Σ -feletti (terminális szimbólumokból álló) szavakat a P -beli szabályok segítségével
- minden P -beli szabály $A \rightarrow \alpha$ alakú, ahol $A \in N$ egy nemterminális és $\alpha \in (N \cup \Sigma)^*$
- G **egyértelmű**, ha az általa generált nyelv minden szavának pontosan egy baloldali levezetése van G -ben

Állítás: Az *ECF* probléma eldönthetetlen

Bizonyítás: Visszavezetjük a *PMP* problémát (ami eldönthetetlen) az *ECF* komplementerére

- Ebből már következik, hogy *ECF* is eldönthetetlen
- Legyen $D = \left\{ \frac{u_1}{v_1}, \dots, \frac{u_n}{v_n} \right\}$ egy dominókészlet
- Konstruáljuk meg a következő CF nyelvtant $G_D := (\{S, A, B\}, \Sigma \cup \Delta, P \cup \{S \rightarrow A, S \rightarrow B\}, S)$, ahol
 - $\Delta = \{a_1, \dots, a_n\}$, $\Sigma \cap \Delta = \emptyset$
 - $P = \{A \rightarrow u_1 A a_1, \dots, A \rightarrow u_n A a_n, A \rightarrow \varepsilon\} \cup \{B \rightarrow v_1 B a_1, \dots, B \rightarrow v_n B a_n, B \rightarrow \varepsilon\}$

TOVÁBBI ELDÖNTHETETLEN PROBLÉMÁK – A CF NYELVTANOK EGYÉRTLEMŰSÉGE

Megmutatjuk, hogy az $f: \langle D \rangle \rightarrow \langle G_D \rangle$ konstrukció valóban egy visszavezetés

- Ha $\frac{u_{i_1}}{v_{i_1}} \dots \frac{u_{i_m}}{v_{i_m}}$ a D egy megoldása
 - Akkor $u_{i_1} \dots u_{i_m} = v_{i_1} \dots v_{i_m}$
 - A G_D konstrukciója miatt érvényesek az alábbi levezetések: $S \Rightarrow A \Rightarrow^* u_{i_1} \dots u_{i_m} a_{i_m} \dots a_{i_1}$ és $S \Rightarrow B \Rightarrow^* v_{i_1} \dots v_{i_m} a_{i_m} \dots a_{i_1}$
 - Azaz ugyanannak a szónak van két különböző baloldali levezetése, azaz G_D nem egyértelmű
- Most tegyük fel azt, hogy G_D nem egyértelmű
 - Akkor van olyan $w \in L(G_D)$ szó, aminek van két különböző baloldali levezetése
 - Az egyik levezetés $S \rightarrow A$ -val a másik pedig $S \rightarrow B$ -vel kell kezdődjön
 - Viszont w alakja csak a következő lehet: $w = xy$, ahol $x \in \Sigma^*$ és $y \in \{a_1, \dots, a_n\}^*$
 - Ezért a fenti két levezetésben a szabályok ugyanolyan sorrendben kell alkalmazásra kerülniük, amiből következik, hogy D -nek van megoldása

TOVÁBBI ELDÖNTHETETLEN PROBLÉMÁK – A CF NYELVTANOK EKVIVALENCIÁJA

Állítás: Tetszőleges G_1 és G_2 CF nyelvtanokra eldönthetetlen, vajon $L(G_1) \cap L(G_2) \neq \emptyset$

- Legyen D egy dominókészlet, G_A, G_B a G_D két rész-nyelvtana rendre az A -t és B -t tekintve kezdőszimbólumnak
- Legyen $G_1 = G_A$ és $G_2 = G_B$
- Ekkor D -nek pontosan akkor van megoldása, ha $L(G_A) \cap L(G_B) \neq \emptyset$

Következmény: Tetszőleges G_1 és G_2 CF nyelvtanokra az alábbi problémák mindegyike eldönthetetlen:

- $L(G_1) = L(G_2)$

Azaz a **CF nyelvtanok ekvivalenciája is eldönthetetlen**

- $L(G_1) = \Gamma^*$ valamely Γ ábécére
- $L(G_1) \subseteq L(G_2)$

- Jelölje L_A és L_B rendre $L(G_A)$ -t és $L(G_B)$ -t
- Belátható, hogy $\overline{L_A}$ és $\overline{L_B}$ is CF nyelvek, így a CF nyelvek unióra való zártsága alapján $\overline{L_A \cap L_B}$ is CF nyelv
- Legyen G_1 az $\overline{L_A \cap L_B}$ -t, G_2 pedig a $(\Sigma \cup \Delta)^*$ nyelvet generáló CF nyelvtan
- Ekkor $L(G_1) = L(G_2)$ azt jelenti, hogy $L_A \cap L_B = \emptyset$ (azaz D -nek nincs megoldása).
- Mivel az utóbbi egyenlőség eldönthetetlen, kapjuk, hogy $L(G_1) = L(G_2)$ is az

TÁRBONYOLULTSÁG

A tárbonyolultság vizsgálatához ún. off-line Turing-gépeket használunk:

- A bemenetet (ami az első szalagon van) csak olvassa
- Ha van kimenet, akkor azt az utolsó szalagra írja, és nem olvashatja vissza

A felhasznált tárba csak a munkaszalagokon (ezek a bemeneti és kimeneti szalagok közötti szalagok) felhasznált cellák száma számít bele:

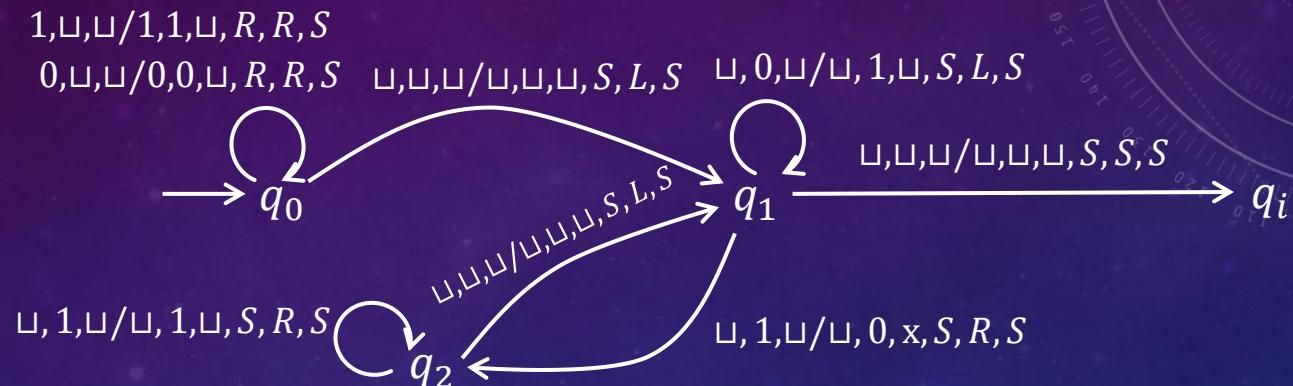
- Legyen $f: \mathbb{N} \rightarrow \mathbb{N}$ egy függvény
- Egy M (off-line) Turing-gép $f(n)$ tárigényű, ha minden n hosszú bemenetre a munkaszalagokon felhasznált cellák száma legfeljebb $f(n)$

Megjegyzések:

- Szublineáris tárbonyolultság is érdekes: pl. az ELÉRHETŐSÉG $\log^2 n$ tárral is megoldható, ahol n a csúcsok száma (lásd Savitch tétele később)
- A tár újra felhasználható (ellenben az idővel)
- Egy lineáris táras Turing-gép képes a kimenetére írni a bemenet méretében exponenciálisan sok betűt (lásd következő példa)

TÁRBONYOLULTSÁG - PÉLDA

Tekintsük a következő háromszalagos Turing-gépet, aminek a bemenő ábécéje $\{0,1\}$:



Működés egy u bemeneten:

- q_0 -ban a második szalagra másolja az u -t ($O(n)$ lépés)
 - q_1 -ben amíg a második szalagon nem csupa 0 van
 - A második szalagon jobbról-balra átírja a 0-t 1-esekre $O(n)$ lépés
 - Ha 1-es lát, átírja 0-ra, a kimenetre (harmadik szalag) ír egy x -et és q_2 -be lép
 - q_2 -ben a második szalagon a szó végére megy ($O(n)$ lépés)
 - Ha szó elejére ér, megáll elfogadó állapotban

Időigény: $O(n)$ szorozva a kimenet hosszával, ami az u , mint bináris szám, decimálisan

- az időigény tehát $O(n2^n)$

Tárigény: $O(n)$

TÁRRA L KAPCSOLATOS BONYOLULTSÁGI OSZTÁLYOK

- PSPACE-szel jelöljük a
 - polinom (azaz $O(n^k)$, $k \in \mathbb{N}$) tárral,
- L-lel pedig a logaritmikus (azaz $O(\log n)$) tárral
 - determinisztikus Turing-géppel eldönthető problémák osztályát (n a bemenet mérete)
- Hasonlóan, NPSPACE-szel jelöljük a
 - polinom tárral,
- NL-lel pedig a logaritmikus tárral
 - nemdeterminisztikus Turing-géppel eldönthető problémák osztályát
- Triviálisan:
 - $L \subseteq \text{PSPACE}$
 - $\text{PSPACE} \subseteq \text{NPSPACE}$ és $L \subseteq \text{NL}$
 - $\text{NP} \subseteq \text{NPSPACE}$

NEMDETERMINISZTIKUS LOGARITMIKUS TÁR

Állítás: ELÉRHETŐSÉG \in NL

- Legyen M egy nemdeterminisztikus Turing-gép és tegyük fel, hogy M bemenetén egy $G = (\{1, \dots, n\}, E)$ irányított gráf reprezentációja van (mondjuk a szomszédsági mátrixszal reprezentálva, tehát a bemenet mérete n^2)
- M a következőt teszi
 - Ráírja az 1 számot (azaz az 1 csúcs sorszámát binárisan) a második szalagra
 - Ráírja a 0-t a harmadik szalagra
 - Amíg a harmadik szalagon n -nél kisebb bináris szám van
 - Legyen u a második szalagon lévő (binárisan tárolt) csúcs
 - M a szomszédsági mátrixban nemdeterminisztikusan keres egy u -ból elérhető v csúcsot, és v -t (bináris alakban) a második szalagra írja
 - Ha v az n szám bináris alakja, akkor M elfogadja a bemenetet, egyébként növeli a harmadik szalagon lévő számot binárisan eggyel
 - Elutasítja a bemenetet
- M tárfelhasználása: a bemenetet csak olvassa, kimenet most nincs, tehát a második és harmadik szalagon felhasznált cellák száma számít, ami $O(\log n)$)

SAVITCH TÉTELE

Állítás: ELÉRHETŐSÉG eldönthető egy $O(\log^2 n)$ táras determinisztikus Turing-géppel

- Először kiterjesztjük az ELÉRHETŐSÉG problémát
- Legyen $G = (\{1, \dots, n\}, E)$ egy irányított gráf, $1 \leq i \leq n$, $u, v \in \{1, \dots, n\}$, és tekintsük a következő Boole-függvényt:
 - $\text{MAXELÉR}(G, u, v, i) = \text{igaz} \Leftrightarrow G\text{-ben van legfeljebb } i \text{ hosszú út } u\text{-ból } v\text{-be}$
- $\text{MAXELÉR}(G, u, v, i)$ kiszámítható egy $O(\log^2 n)$ tárigényű N (determinisztikus) Turing-géppel:
 - $\text{MAXELÉR}(G, u, v, i) = \text{igaz} \Leftrightarrow \text{ha } i = 1 \text{ és } u = v \text{ vagy } (u, v) \in E \text{ VAGY}$
$$i > 1 \text{ és } \exists w \in V: \text{MAXELÉR}\left(G, u, w, \left\lfloor \frac{i}{2} \right\rfloor\right) = \text{igaz} \wedge \text{MAXELÉR}\left(G, w, v, \left\lceil \frac{i}{2} \right\rceil\right) = \text{igaz}$$
 - A számításhoz N -nek legfeljebb $\log n$ mélységű rekurziót kell alkalmaznia, és a rekurzió minden szintjén $O(\log n)$ méretű adatot kell eltárolnia
 - Tehát N tárigénye $O(\log^2 n)$

Készen vagyunk, mert G -ben pontosan akkor elérhető 1-ből n , ha

$$\text{MAXELÉR}(G, 1, n, n) = \text{igaz}$$

Egy u -ból v -be tartó út hossza az úton szereplő élek száma

KONFIGURÁCIÓS GRÁF

Legyen M egy $f(n)$ tárigényű nemdeterminisztikus Turing-gép és w az M egy n hosszú bemenete

- M konfigurációs gráfja w -n egy olyan $G_{M,w}$ irányított gráf, melynek
 - A csúcsai M lehetséges konfigurációi w -n, és
 - $G_{M,w}$ -ben C_1 csúcsból (konfigurációból) pontosan akkor vezet egy él C_2 csúcsba (konfigurációba) ha $C_1 \Rightarrow C_2$
- $G_{M,w}$ csúcsainak száma legfeljebb $2^{df(n)}$ egy alkalmas, M -től függő d konstansra
 - Mert $f(n)$ cellára $2^{cf(n)}$ -féle szót lehet írni (c a szalagszimbólumok száma)
 - $f(n)$ pozícióban lehet a fej, és
 - Konstans sok állapot van
- Feltehetjük, hogy $G_{M,w}$ -ben csak egy elfogadó konfiguráció van (normalizáljuk M -met pl. úgy, hogy mielőtt elfogad, letörli a szalagját)
 - Jelölje $c_{kezdő}$ és $c_{elfogadó}$ rendre a kezdő- és elfogadó konfigurációkat

SAVITCH TÉTELÉNEK KÖVETKEZMÉNYE

Legyen M egy $f(n)$ tárigényű nemdeterminisztikus Turing-gép és w az M egy n hosszú bemenete

- Ahhoz, hogy eldöntsük, vajon M elfogadja-e w -t elegendő a Savitch-tétel bizonyításában szereplő N Turing-géppel kiszámolni az $\text{MAXELÉR}(G_{M,w}, c_{kezdő}, c_{elfogadó}, 2^{d \cdot f(n)})$ értékét
- Láttuk, hogy N tárigénye $O(\log^2 2^{d \cdot f(n)}) = O(f^2(n))$

Következmény:

- Ha egy nyelv eldönthető $f(n) \geq \log n$ tárigényű nemdeterminisztikus Turing-géppel, akkor eldönthető $O(f^2(n))$ tárigényű determinisztikussal is

Következmény:

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

BONYOLULTSÁGELMÉLET – 6. ELŐADÁS

ELŐADÓ: GAZDAG ZSOLT

KÉSZÜLT ...

- IVÁN SZABOLCS KÖRÁBBI ELŐADÁSFÓLIÁI, VALAMIINT
- M. SIPSER: INTRODUCTION TO THE THEORY OF COMPUTATION ÉS
- C. PAPADIMITRIOU: SZÁMÍTÁSI BONYOLULTSÁG C. KÖNYVEI ALAPJÁN

KÉTVERMES AUTOMATA

Formális nyelvek órán tanultunk néhány **modellt** nyelvek felismerésére:

Véges automata:



pontosan a **reguláris** nyelveket ismeri fel

Veremautomata:



pontosan a **környezetfüggetlen** nyelveket ismeri fel

Mi történik ha a veremautomatának adunk még egy vermet?



Kétvermes automata

Működése: adott állapotból, adott bemenő jel (input szimbólum vagy ϵ) és legfelső veremjelek hatására új állapotba lép, és a vermek tetején lévő betűk helyére újakat (akár többet is) ír

KÉTVERMES AUTOMATA

Állítás: minden M egyszalagos Turing-géphez megadható egy ekvivalens (ugyanazt a nyelvet felismerő) V kétvermes automata

- Alapötlet:



V úgy tárolja M szalagkonfigurációját, hogy a fejtől balra lévő szó van az első veremben és a fej pozíóján kezdődő szó van a második veremben úgy, hogy a fejtől balra lévő és a fej pozíóján kezdődő betűk vannak legfelül



V verem-konfigurációja

KÉTVERMES AUTOMATA

V működése (vázlat)

- Tegyük fel, hogy a $w\$$ szó van az V bemenetén (az egyszerűség kedvéért feltesszük, hogy az inputot egy egyedi „end marker”, ez lesz a $\$$)
 - V bemásolja w -t az első verebe
 - Ezután az első veremből átmásolja w -t a másodikba, ezzel előállítva M kezdőkonfigurációját
 - Ezután V szimulálja M egy lépését a következőképpen:
 - V ismeri M állapotát (a saját állapotában el tudja tárolni azt), legyen ez q , és az M által olvasott betűt, mert ez van a második verem tetején, legyen ez X (ha a második veremben veremvégjel van legfelül, akkor $X = \sqcup$)
 - Így V végre tudja hajtani M egy lépését:
 - Ha M X -et Y -ra írja át és jobbra lép, akkor M Y -t ír az első verem tetejére és nem ír semmit X helyére a másodikban (spec. esetekre azért figyelni kell: veremvégjelet ne vegyük ki a másodikból, \sqcup -t ne írunk az elsőbe ha az üres)
 - Ha M X -et Y -ra írja át és balra lép, akkor kiveszi az első verem tetejét, mondjuk Z -t, és kicseréli X -et ZY -ra a másodikban (Z lesz felül); ha az első verem üres, akkor egy \sqcup kerül az Y fölé
 - Ha M új állapota elfogadó állapot, akkor V is elfogadó állapotba lép, egyébként pedig szimulálja az M következő lépését

ÚJRA AZ EKVIVALENCIAPROBLÉMÁRÓL

Hogyan változik a vizsgált automaták **ekvivalenciaproblémájának** bonyolultsága?

- **A véges automaták ekvivalenciaproblémája eldönthető**

- Legyen M_1 és M_2 két véges automata és M a két automata direkt-szorzat konstrukciójával kapott automata
- M végállapotai megadhatók úgy, hogy

M állapothalmaza az M_1 és M_2 állapothalmazainak direkt szorzata, az átmenetfüggvénye pedig az állapotok első komponenseiben az M_1 a másodikban pedig az M_2 átmeneteit szimulálja

$$L(M) = \left((L(M_1) - L(M_2)) \cup (L(M_2) - L(M_1)) \right) \text{ teljesüljön}$$

- M_1 és M_2 pontosan akkor ekvivalensek, ha $L(M) = \emptyset$
- $L(M) = \emptyset$ pedig eldönthető, elég tesztelni, hogy a **kezdőállapotából elérhető-e végállapot** (ez az **ELÉRHETŐSÉG** probléma)
- A **veremautomaták** pontosan a CF nyelveket ismerik fel
 - Ezért a veremautomaták ekvivalenciaproblémája **nem eldönthető**
- A **kétvermes automaták** Turing-ekvivalensek, tehát az Ő ekvivalencia-problémájuk még csak **nem is félíg eldönthető**

A RAM GÉP

RAM gép

- ▶ rendelkezik végtelen sok **regiszterrel**
- ▶ a regiszterek típusai: **Input**, **Output** és **Working** (munka-)regiszterek, minden típus regiszterei 0-tól indexeltek
- ▶ minden regiszter egy nemnegatív egész számot tartalmaz
- ▶ címzési módok:
 - ▷ k – konstans
 - ▷ $I[k]$, $O[k]$, $W[k]$ – direkt
 - ▷ $W[W[k]]$, $W[I[k]]$ stb. – indirekt

- ▶ **program:** **utasítások** egy véges sorozata
- ▶ minden **utasítás** egy **sorszám** és egy **elemi művelet**

(egy sorszám legfeljebb egy utasításhoz tartozhat)

A RAM GÉP

- elemi műveletek:
 - ▷ $X := Y$ – értékadás
 - X itt egy direkt vagy egy indirekt címzés, pl. $O[7]$ vagy $W[I[3]]$
 - Y két címzés összege vagy különbsége (ha $a < b$, akkor $a - b$ eredménye 0)
 - ▷ JZ r, ℓ – ha $r == 0$, ugrás az ℓ . sorszámú utasításra
 - ▷ HALT, ACCEPT, REJECT – megállás
- a programszámláló (PC) pozitív egész, 0-ról indul
 - ▷ a gép **egy lépésben** végrehajtja azt az utasítást, melynek sorszáma a PC tartalma
 - ▷ a feltételes ugrást kivéve minden lépés után eggyel nő a PC
 - ▷ ha nincs ilyen sorszámú utasítás, a számítás megáll
- a számítás tetszőleges pillanatában csak véges sok regiszter értéke nem 0
ez azért jó, mert akkor lehet „igazi” architektúrán is szimulálni

A RAM GÉP

► inicializálás:

- ▷ az input az $I[1], I[2], \dots, I[m]$ számsorozat, ahol $m = I[0]$ írja le az inputban szereplő „tényleg használt” regiszterek számát
 - pl. 6, 1, 4, 2, 8, 5, 7, ... esetén az input az 1, 4, 2, 8, 5, 7 sorozat
- ▷ a többi regiszter tartalma 0

► az output

- ▷ ACCEPT/REJECT, ha a megállás egy ACCEPT/REJECT utasítás végrehajtásakor következett be
- ▷ az $O[1], O[2], \dots, O[k]$ számsorozat, ahol $k = O[0]$ írja le az outputban szereplő „használt” regiszterek számát

Az M RAM gép az a_1, \dots, a_m inputon számított outputját $M(a_1, \dots, a_m)$ jelöli; ha M nem áll meg ezen az inputon, annak jele $M(a_1, \dots, a_m) = \nearrow$.

FELTÉTELEK, UGRÁSOK

Persze a példáinkban, konstrukcióinkban nem minden közvetlenül ezekből az elemi műveletekből rakunk össze: a következő pár fólián látunk néhány példát, hogyan lehet RAM gépen strukturáltan programozni

if $L \leq R$ **goto** Y

1. $X := L - R$
2. **if** $X == 0$ **goto** Y ha $L > R$, itt nem ugrunk el Y -ra, hanem „simán megyünk tovább” 3-ra
ahol X egy másra nem használt munkaregiszter.

Feltétel nélküli ugrás, $==$, $<$, $>$, \geq relációk szerinti feltételek hasonló módon

az ilyen alakú **if**-**goto** feltételes ugrás ugyan nem 1, de **konstans** sok lépés

STACK MEMÓRIA, FÜGGVÉNYHÍVÁSOK

Ha egy SP munkaregisztert (pl $W[0]$ -t) kinevezünk **stack pointernek**, és függvényeinket mindig **rögzített paraméterszámmal** definiáljuk, akkor lokális változókat használó rekurzív (most: „önmagukat hívó”) függvényeket is írhatunk a megszokott módon

- ▶ $W[SP] := k$, ahol k az a programsor, ahová a függvényhívás után vissza kell térnünk
 - ▶ SP egyesével növelése mellett a paramétereket sorban bemásoljuk $W[SP]$ -be
 - ▶ ugunk a függvény belépési pontjára
-
- ▶ a függvény a verem tetején megtalálja a paramétereit, a saját lokális változóit ezek tetejére hozza létre
 - ▶ visszatéréskor a megfelelő értékkel (paraméterszám+lokális változók száma) csökkenti SP-t és a megfelelő, a veremben letárolt címre ugrik vissza
- Ha az argumentumokat már kiszámoltuk, akkor a call+return is **konstans sok** lépés

STRUKURÁLT PROGRAMOZÁS

while F **do** (R)

1. if $F == 0$ goto 4
2. R
3. goto 1

a ciklusok időigénye függ attól, hogy hányszor is fut le a ciklus

do (R) **while** F

1. R
2. if $F == 0$ goto 4
3. goto 1

ha legfeljebb X -szer fut a ciklus és a magja legfeljebb Y lépés,
akkor az egész $\max X \cdot Y$ lépésekben lefut

for $i := 1 \dots n$ **do** (R)

1. $i := 1$ kényelmi okokból olhasható változóneveket használunk $W[42]$ -like nevek helyett
2. while $i \leq n$ do (
3. R
4. $i := i + 1$)

ARITMETIKA

Legmagasabb bit

```
function HIGH(n)
    if n == 0 then return 0
    a := 1
    while a + a ≤ n do a := a + a
    return a
```

pl. $\text{HIGH}(42) = 32$

a lépésszám $\log n$ -nel arányos!

Paritás

```
function ODD(n)
    a := 0
    while a ≠ n do
        n := n - a
        a := HIGH(n)
    if a == 1 then return 1
    return 0
```

minden iterációban eggyel kevesebb lesz az 1-es n -ben (bináris rep)

$\Rightarrow \max \log n$ -szer fut le

a ciklusmag $\log n$ -nel arányos lépésszámot igényel

ez így $\log n + \log n = \log^2 n$ -nel arányos lépésszám lesz

ARITMETIKA

Felezés

Az előző kettőhöz hasonlóan szintén $\log^2 n$ lépében megvan: rendre kiszámoljuk a legmagasabb bitet, kivonjuk, és a felét hozzáadogatjuk egy akkumulátorhoz

Szorzás

```
function MULT(a,b)      kb mint az általános papíron szorzás, gyorsabb, mint b-szer összeadni a-t
    r := 0
    while b > 0 do
        if ODD(b) then r := r + a
        b := b/2
        a := a + a
    return r
```

ez max $\log b$ -szer fut le
ez $O(\log^2 b)$ lépés
ez is $O(\log^2 b)$ lépés
ez $O(1)$ lépés

Ciklusmag összesen $O(\log^2 b)$ lépés, $O(\log b)$ -szer fut \Rightarrow ez így lefut $O(\log^3 b)$ lépében
ezzel szemben b -szer összeadni a -t $O(b)$ lépés lenne, ami sokkal több

Ami fontos: pszeudokódban / RAM gépen ha számokkal dolgozunk, melyeknek az értéke legfeljebb N , azokat lehet összeadni, kivonni, szorozni, osztani $\log^k N$ lépében valamilyen k konstansra (ezt úgy hívják, hogy a lépésszám polilogaritmikus N -ben), de például hatványozni már nem

AZ INPUT MÉRETE ÉS A FUTÁSIDŐ: DEFINÍCIÓK

Futásidő egy adott inputon: lépésszám

Az M RAM gép **időigénye az a_1, \dots, a_m inputon** a megállásig végrehajtott utasítások száma, ha M megáll a_1, \dots, a_m -en; egyébként a gép nem áll meg és időigénye végtelen

Az input mérete

Az a_1, \dots, a_m **input mérete** az a_i számok **bináris reprezentációinak** hosszának összege (azaz $\sum_{i=1}^m (1 + \lfloor \log a_i \rfloor)$, de általában csak $\sum_{i=1}^m \log a_i$ -ként fogjuk írni, nagyságrendben ugyanaz a kettő, azzal, hogy akkor $\log 0 := \log 1 := 1$). Általában n jelöli az input méretét.

A gép időkorlátja

Az M RAM gép **időkorlátja** az $f(n) : \mathbb{N} \rightarrow \mathbb{N}$ függvény, ha tetszőleges n méretű inputon legfeljebb $f(n)$ lépésekben megáll.

Az **időkorlát** és **időigény** kifejezéseket egymás szinonimáiként fogjuk használni

A RAM GÉP ÉS TURING-GÉP EKVIVALENCIÁJA

Állítás: minden $f(n)$ időigényű R RAM gép szimulálható egy $O(f(n)^3)$ időigényű M ötszalagos Turing-géppel

- **Megjegyzés:** az, hogy szimulálható egyenes következménye a Church-Turing tézisnek
- M a következőképpen működik
 - A **bemenetén** megkapja az R input regisztereiben lévő számokat, valamint az R utasításait megfelelően elkódolva; a bemenetet csak olvassa
 - A **második szalagján** tárolja R munkaregisztereit, a **harmadikon** pedig R programszámlálóját (PC), ami kezdetben 0
 - A **negyedig szalag** egy segédszalag a számításokhoz
 - Az **utolsó szalagra** írja R kimeneti regisztereinek értékét

A RAM GÉP ÉS TURING-GÉP EKVIVALENCIÁJA

- R egy lépésének szimulációja
 - M megnézi a harmadik szalagon, hogy melyik utasítást kell szimulálnia;
 - Ha ez értékadás ($X := Y$), akkor megkeresi az Y -ban szereplő regiszterértékeket, és a segédszalagon kiszámolja Y értékét; ezután a aktualizálja az X által megadott regiszter értékét a második szalagon, majd a harmadik szalagon növeli eggyel a PC értékét
 - Ha ez feltételes ugrás ($JZ r, \ell$), akkor kiértékeli r -t, és ha az 0, akkor ℓ értékét írja a harmadik szalagra; egyébként növeli a harmadik szalagon lévő értéket 1-gyel
 - Ha ez HALT, ACCEPT, REJECT, akkor M is megáll: HALT és ACCEPT esetében q_i -ben, egyébként q_n -ben
- M időigénye
 - Belátható, hogy R a működése során legfeljebb annyi munkaregisztert használ, amennyit lép, és minden regiszterben lévő szám legfeljebb $O(f(n))$ biten (cellán) tárolható
 - Tehát R egy lépésének szimulációja $O(f(n)^2)$ lépésben elvégezhető
 - Mivel $f(n)$ lépést kell szimulálni, M időigénye $O(f(n)^3)$

A RAM GÉP ÉS TURING-GÉP EKVIVALENCIÁJA

Állítás: minden $f(n)$ időigényű M Turing-gép szimulálható egy $f(n)$ időigényű R RAM géppel

- Feltesszük, hogy M egyetlen szalaggal rendelkezik, és az olyan, hogy csak **jobbra végtelen**; ismert, hogy a Turing-gépek ezzel a megszorítással is univerzálisak
- Legyen $M = (Q, \Sigma, \Gamma, \delta, q_0, q_i, q_n)$, ahol $Q = \{q_0, \dots, q_l\}$, $q_{l-1} = q_i$, $q_l = q_n$, és $\Gamma = \{X_0, \dots, X_k\}$, $X_0 = \sqcup$
 - Legyen $D_\Gamma = \{(i_1, i_2, \dots, i_n) \mid n \in \mathbb{N}, 0 \leq i_j \leq k, j = 1, \dots, n\}$
 - Ekkor minden $X_{i_1} \dots X_{i_n}$ Γ -feletti szónak megfeleltethető a (i_1, i_2, \dots, i_n) D_Γ -beli szám n -es
 - Olyan R -et vázolunk, ami tetszőleges $w = X_{i_1} \dots X_{i_n}$ Σ -feletti szóra akkor és csak akkor ad ACCEPT kimenetet az i_1, i_2, \dots, i_n bemenetre, ha $w \in L(M)$
 - R -ben
 - $W[0]$ tárolja M állapotának sorszámát
 - $W[1]$ tárolja M fejének pozícióját
 - A $W[2], W[3], \dots$ munkaregiszterek tárolják az M szalagján szereplő betűk indexeit
 - R -nek M minden q_r állapot - X_s szalagszimbólum párosához, $r < l - 1$, van egy $K_{r,s}$ kód részlete

A RAM GÉP ÉS TURING-GÉP EKVIVALENCIÁJA

- $K_{r,s}$ vázlata:
 - Ha $\delta(q_r, X_s) = (q_t, X_u, Right)$, akkor R
 - $W[0]$ -t átírja t -re, $W[W[1]]$ -et átírja u -ra és $W[1]$ -et növeli 1-gyel
 - Ha $\delta(q_r, X_s) = (q_t, X_u, Left)$, akkor hasonlóan, kivéve, hogy most $W[1]$ -et csökkenti 1-gyel (ha $W[1] = 0$, akkor $W[1]$ marad 0)
- R a következőképpen működik:
 - Bemásolja a bemenetet a $W[2], W[3], \dots$ munkaregiszterekbe
 - $W[0]$ -ba 0-t és $W[1]$ -be 2-t ír (ez reprezentálja M kezdőkonfigurációját, amikor az M a q_0 kezdőállapotban van és a fej a bemenet első betűjére mutat, amit $W[2]$ tárol)
 - Amíg $W[0]$ -ban nem $l - 1$ vagy l van, R szimulálja M egy lépését:
 - Legyen r, s rendre a $W[0]$ és $W[1]$ regiszterekben lévő szám
 - $R r$ -ből és s -ből kiszámolja a $K_{r,s}$ kód részlet kezdő utasításának sorszámát és végrehajtja ezt a kódot
- Időigény: M egy lépésének szimulálása konstans sok lépés (r, s kiszámítása csak M méretétől függ, ami konstans)
 - Így R időigénye $O(f(n))$

BONYOLULTSÁGELMÉLET – 7. ELŐADÁS

ELŐADÓ: GAZDAG ZSOLT

KÉSZÜLT ...

- IVÁN SZABOLCS KORÁBBI ELŐADÁSFÓLIÁI, VALAMINT
- M. SIPSER: INTRODUCTION TO THE THEORY OF COMPUTATION ÉS
- C. PAPADIMITRIOU: SZÁMÍTÁSI BONYOLULTSÁG C. KÖNYVEI ALAPJÁN

RAM GÉPEK TÁRBONYOLULTSÁGA

Ha egy Ram-program esetében csak a **használt regiszterek számát** vennénk alapul, **irreálisan alacsony** tárigény-fogalomhoz jutnánk

- Mert bármennyi regiszter tartalmát bele lehet kódolni egyetlen regiszterbe
- Ezért a tárigényt a következőképpen definiáljuk:

Regiszterenként

Egy regiszter igénye egy adott inputon: a benne a program futása során tárolt **legnagyobb** szám bináris alakjának a **hossza**.

(Azaz $1 + \lfloor \log a \rfloor$, ha ez a maximális érték a , kivéve, ha $a == 0$, mely esetben 1).

Példa

Ha a $W[7]$ regiszterben a program futása során a legnagyobb érték **20**, akkor **ennek a regiszternek** a tárigénye 5 (a $20 = 10100_2$ string hossza).

Tárigény egy adott inputon

A teljes program tárigénye egy adott inputon: az összes, a futás során **használt** regiszter tárigényének összege

RAM GÉPEK TÁRBONYOLULTSÁGA

A két memóriamodell

Amint egy cellát megcímez a program (direkt vagy indirekt címzéssel, írásra vagy olvasásra), Ő **és az összes megelőző cella az adott típusban használtnak minősül.**

(Tehát pl. ha egyszer értéket adunk $W[100]$ -nak, akkor $W[0]$ -tól $W[100]$ -ig minden egyik cella „használt” lesz.

Ha annyit tesz a programunk, hogy $W[100]$ értékét 20-ra állítja, akkor tárigénye 105: $W[0]$ -tól $W[99]$ -ig minden egyik regiszterben ott egy 0, amihez kell 1-1 bit, $W[100]$ -ban a 20-hoz pedig kell még 5, összesen 105.

Kivéve a következő esetet:

- ▶ ha a program az **input** regisztereket **csak olvassa**,
- ▶ az **output** regiszterekbe pedig **csak ír**, ráadásul stream módban: először $O[0]$ kap értéket, majd $O[1]$, $O[2]$ stb. eldöntési problémáknál nem használjuk O -t

akkor az input és output regiszterek tárigényét nem kell bevenni a tárigénybe. (Ez az ún. „**offline**” vagy „lyukszáagos” eset.)

(Ez megfelel annak a memóriamodellnek, mikor a hívó fél biztosítja az I/O területet: az inputot nem írhatjuk át, az outputot pedig egy output stream formájában kapjuk.)

A RAM GÉP ÉS A BONYOLULTSÁGI OSZTÁLYOK

A program tárígye

A program tárígye az $f(n) : \mathbb{N} \rightarrow \mathbb{N}$ függvény, ha bármely, legfeljebb n méretű inputon legfeljebb $f(n)$ tárat használ.

- Egy RAM-program idő- és tárígyének definíciója alapján a tanult (determinisztikus) bonyolultsági osztályok (pl. P, L, PSPACE) **ugyanúgy definiálhatók**, mint a Turing-gépek esetében
- Láttuk, hogy hogyan szimulálható RAM géppel egy Turing-gép
 - Ezt felhasználva **az R és RE osztályok is definiálhatók** RAM gépekkel is
 - Sőt, visszavezetéseket is tudunk RAM gépekkel is definiálni
- A továbbiakban viszont, ha más nem mondunk, a Turing-gépet fogjuk kiszámítási modellként használni

TOVÁBBI IDŐ- ÉS TÁRBONYOLULTSÁGI OSZTÁLYOK

- $\text{TIME}(f(n))$ -nel (rendre, $\text{NTIME}(f(n))$ -nel) jelöljük az $\mathcal{O}(f(n))$ időigényű determinisztikus (rendre, nemdeterminisztikus) Turinggéppel eldönthető problémák osztályát
 - Tehát $\text{P} = \bigcup_{k \in \mathbb{N}} \text{TIME}(n^k)$ és $\text{NP} = \bigcup_{k \in \mathbb{N}} \text{NTIME}(n^k)$
- $\text{SPACE}(f(n))$ -nel (rendre, $\text{NSPACE}(f(n))$ -nel) jelöljük az $\mathcal{O}(f(n))$ tárigényű determinisztikus (rendre, nemdeterminisztikus) Turinggéppel eldönthető problémák osztályát
 - Tehát $\text{PSPACE} = \bigcup_{k \in \mathbb{N}} \text{SPACE}(n^k)$ és $\text{NPSPACE} = \bigcup_{k \in \mathbb{N}} \text{NSPACE}(n^k)$
- $\text{EXP} = \bigcup_{k \in \mathbb{N}} \text{TIME}(2^{n^k})$
 - Tehát EXP az exponenciális időben megoldható problémák osztálya

HIERARCHIA TÉTELEK

- $\text{TIME}(f(n)) \subseteq \text{SPACE}(f(n))$
 - Mert egy $f(n)$ időigényű Turing-gép legfeljebb $f(n)$ méretű tárat tud felhasználni
- Ha $f(n) \geq \log n$, akkor $\text{SPACE}(f(n)) \subseteq \text{TIME}(k^{f(n)})$, valamely alkalmas $k \in \mathbb{N}$ konstansra
 - Láttuk, hogy egy $f(n)$ tárigényű M Turing-gép konfigurációs gráfjának mérete egy n hosszú bemeneten legfeljebb $2^{df(n)}$ egy alkalmas, M -től függő d konstansra
 - Tehát ha M futás közben konfigurációt ismételne, akkor végtelen ciklusba esne, ezért a konfigurációk száma egy felső korlát a lépésszámra
- **Megengedett függvények:** Az $f : \mathbb{N} \rightarrow \mathbb{N}$ monoton növő függvény megengedett, ha létezik olyan determinisztikus $\mathcal{O}(f(n))$ tárigényű és $\mathcal{O}(n + f(n))$ időigényű Turing-gép, ami az $1^n \mapsto 1^{f(n)}$ függvényt számítja ki
 - Azaz n darab egyesből $f(n)$ darab egyest készít, közben nem használ indokolatlanul sok tárat, sem időt. A polinomok, \exp , \log függvények stb. amik „realisztikusan” előjönnek tár/időigényként, mind ilyenek

HIERARCHIA TÉTELEK

Időhierarchia tétele: Ha $f(n) \geq n$ megengedett függvény, akkor

$$\text{TIME}(f(n)) \subsetneq \text{TIME}(f(2n+1)^3)$$

- Ötlet: Legyen $g(n) \geq n$ megengedett függvény és

$$H_g = \{\langle M, w, \rangle \mid M \text{ legfeljebb } g(|w|) \text{ lépésekben elfogadja } w\text{-t}\}$$

- Az $L_u \in \text{RE}$ bizonyításban látott univerzális Turing-gépet lehet módosítani úgy, hogy adott w -re $O(g(|w|)^3)$ időben
 - kiszámolja a $g(|w|)$ értéket,
 - Szimulálja M -et w -n és
 - figyeli, hogy M legfeljebb $g(|w|)$ -t lép-e
- Tehát $H_g \in \text{TIME}(g(n)^3)$
- Ugyanakkor belátható, hogy $H_g \notin \text{TIME}\left(g\left(\left\lfloor \frac{n}{2} \right\rfloor\right)\right)$ (nem triviális, nem bizonyítható)

Tárhierarchia tétele: Ha $f(n)$ megengedett függvény, akkor

$$\text{SPACE}(f(n)) \subsetneq \text{SPACE}(f(n) \log f(n))$$

ALAPVETŐ ÖSSZEFÜGGÉSEK

$$\text{NL} \subseteq \text{P} \subseteq \text{PSPACE} \subseteq \text{EXP}(\subsetneq \text{R})$$

- $\text{NL} \subseteq \text{P}$, mert egy logtáras (akár nemdeterminisztikus) Turing-gép konfigurációs gráfjának mérete polinomiális a bemenet méretének a függvényében
- $\text{P} \subseteq \text{PSPACE}$, hiszen korábban már láttuk, hogy $\text{P} \subseteq \text{NP} \subseteq \text{NPSPACE} = \text{PSPACE}$
- $\text{PSPACE} \subseteq \text{EXP}$, mert egy polinom táras Turing-gép konfigurációs gráfjának mérete exponenciális a bemenet méretének a függvényében
- Az a sejtés, hogy a fenti tartalmazások mindegyike valódi
- Megjegyzés:
 - Az időhierarchia tétel miatt $\text{P} \subsetneq \text{EXP}$ (mert $\text{P} \subseteq \text{TIME}(2^n) \subsetneq \text{TIME}\left((2^{2n+1})^3\right) \subseteq \text{EXP}$)
 - A tárhierarchia tétel miatt $\text{L} \subsetneq \text{PSACE}$ (mert $\text{L} \subseteq \text{SPACE}(n) \subsetneq \text{SPACE}(n \log n) \subseteq \text{PSPACE}$)
- $\text{EXP} \subsetneq \text{R}$, mert nem minden probléma dönthető el exponenciális időben
 - Például a Presburger aritmetika egy állításáról el lehet döntenи, hogy igaz-e vagy sem, de nincs rá exponenciális időigényű algoritmus

POLINOM IDŐBEN VERIFIKÁLHATÓSÁG

P tartalmazza a gyakorlatban is hatékonyan megoldható problémákat, de milyen problémák vannak NP-ben?

- Minden NP-beli L problémára a következő jellemző: létezik egy olyan polinom idejű T nemdeterminisztikus Turing-gép ami következőt tudja
 - T az L minden I bemenetére nemdeterminisztikusan generálja I egy lehetséges M megoldását
 - Fontos, hogy I minden lehetséges megoldása generálható legyen
 - Determinisztikus számítással leellenőrzi, hogy M valóban megoldása-e I -nek (ezt úgy is mondjuk, hogy ellenőrzi vajon M egy tanúja annak, hogy M egy pozitív bemenet)

Formálisan: Azt mondjuk, hogy egy L nyelv polinom időben verifikálható, ha van olyan $K \in P$ nyelv és k szám, hogy

$$L = \{x \mid \exists y (x, y) \in K \text{ és } |y| = O(|x|^k)\}$$

Bemenet

Megoldás

A Megoldás
polinom időben
verifikálható

y x -hez képest
"nem túl nagy"

K polinom időben
eldönthető
determinisztikusan

Állítás: Egy nyelv akkor és csak akkor NP-beli, ha polinom időben verifikálható

POLINOM IDŐBEN VERIFIKÁLHATÓSÁG - PÉLDÁK

- **SAT polinom időben verifikálható** (tehát NP-beli)
 - mert egy T Turing-gép egy tetszőleges φ konjunktív normálformára
 - Nemdeterminisztikusan legenerál minden egyes potenciális I tanút (I nem más, mint φ változóinak egy értékadása)
 - Polinom időben ellenőrzi, hogy I kielégíti-e φ -t
 - Ha igen, akkor elfogadja a bemenetet, ha nem, akkor elutasítja
 - Ekkor T számítási fájában pontosan akkor lesz az egyik levél egy elfogadó konfiguráció, ha φ -nek van egy kielégítő értékadása, azaz pontosan akkor ha φ kielégíthető
 - Tehát T pontosan akkor fogadja el φ -t ha az kielégíthető
- **A HAMILTON-ÚT_{1→n} is polinom időben verifikálható**
 - mert egy T Turing-gép egy tetszőleges G n csúcsú irányított gráfra
 - Nemdeterminisztikusan legenerál minden lehetséges n csúcsból álló csúcssel sorozatot
 - Polinom időben ellenőrzi, hogy ezek Hamilton utat alkotnak-e 1-ből n -be

POLINOM IDEJŰ VISSZAVEZETÉSEK

Milyen visszavezetésre van szükségünk, ha NP-beli problémák bonyolultságát szeretnénk összehasonlítani?

- Az **eddig használt nem jó**, mert attól, hogy $L_1 \leq L_2$ és L_2 könnyen megoldható még nem következik, hogy L_1 is könnyen megoldható (hiszen a visszavezetés lehet nehezen, mondjuk exponenciális időben kiszámítható)
- Olyan visszavezetés kell, ami feltehetőleg **nem elég erős**, hogy minden NP-beli problémát megoldjon
- Mivel valószínűleg $P \neq NP$, a **polinom időben kiszámítható** visszavezetések megfelelnek a céljainknak

Egy L_1 nyelv **polinom időben visszavezethető** az L_2 nyelvre (jele: $L_1 \leq_p L_2$), ha

- $L_1 \leq L_2$ és
- a visszavezetéshez használt függvény kiszámítható egy **polinom időkorlátos determinisztikus Turing-géppel**

PÁROSÍTÁS \leq_p SAT

PÁROSÍTÁS

- ▶ Input: $G = (V, E)$ gráf.
- ▶ Output: van-e G -ben teljes párosítás?

A visszavezetés

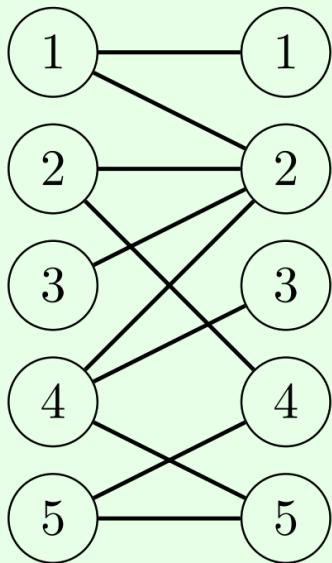
- ▶ minden $(u, v) \in E$ élhez rendelünk egy $x_{u,v}$ logikai változót.
- ▶ Intuíció: $x_{u,v}$ akkor 1, ha (u, v) párosításbeli él, 0 egyébként.
- ▶ minden u csúcsra felírjuk, hogy legalább egy rá illeszkedő élt kiválasztunk...
- ▶ ... és azt is, hogy legfeljebb egyet.

Belátható, hogy

- G -ben pontosan akkor van teljes párosítás ha az így kapott formula kielégíthető
- A konstrukció polinom időben elvégezhető

PÁROSÍTÁS \leq_p SAT

Példa



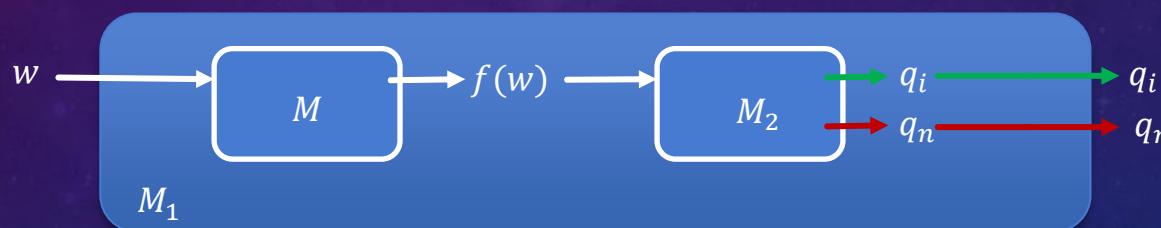
$$\begin{aligned} & (x_{1,1} \vee x_{1,2}) \wedge (x_{2,2} \vee x_{2,4}) \wedge x_{3,2} \wedge (x_{4,2} \vee x_{4,3} \vee x_{4,5}) \\ & \wedge (x_{5,4} \vee x_{5,5}) \wedge x_{1,1} \wedge (x_{1,2} \vee x_{2,2} \vee x_{3,2} \vee x_{4,2}) \\ & \wedge x_{4,3} \wedge (x_{2,4} \vee x_{5,4}) \wedge (x_{4,5} \vee x_{5,5}) \\ & \wedge (\neg x_{1,1} \vee \neg x_{1,2}) \wedge (\neg x_{2,2} \vee \neg x_{2,4}) \wedge (\neg x_{4,2} \vee \neg x_{4,3}) \\ & \wedge (\neg x_{4,2} \vee \neg x_{4,5}) \wedge (\neg x_{4,3} \vee \neg x_{4,5}) \wedge (\neg x_{5,4} \vee \neg x_{5,5}) \\ & \wedge (\neg x_{1,2} \vee \neg x_{2,2}) \wedge (\neg x_{1,2} \vee \neg x_{3,2}) \wedge (\neg x_{1,2} \vee \neg x_{4,2}) \\ & \wedge (\neg x_{2,2} \vee \neg x_{3,2}) \wedge (\neg x_{2,2} \vee \neg x_{4,2}) \wedge (\neg x_{3,2} \vee \neg x_{4,2}) \\ & \wedge (\neg x_{2,4} \vee \neg x_{5,4}) \wedge (\neg x_{4,5} \vee \neg x_{5,5}) \end{aligned}$$

Definíció: Tetszőleges \mathbb{C} problémaosztályra, \mathbb{C} zárt a polinom idejű visszavezetésre nézve, ha bármely $L_2 \in \mathbb{C}$ és L_1 problémák esetén, $L_1 \leq_p L_2$ -ből következik, hogy $L_1 \in \mathbb{C}$

POLINOM IDEJŰ VISSZAVEZETÉSEK

Állítás: P és NP zártak a polinomidejű visszavezetésre nézve

- Legyen M a visszavezetést kiszámoló és M_2 az L_2 -t eldöntő Turing-gép
- Konstruáljuk meg az M_1 Turing-gépet a következőképpen:



- Korábban láttuk, hogy M L_1 -et dönti el; polinom időigényű is:
 - legyen M időigénye p_1 , M_2 -é pedig p_2 (p_1 és p_2 polinomok)
 - ha w n hosszú, akkor $f(w)$ legfeljebb $p_1(n)$ hosszú lehet
 - ezért az M_1 időigénye $O(p_1(n) + p_2(p_1(n)))$, ami szintén polinom
- Továbbá ha M_2 nemdeterminisztikus, akkor M_1 is az, és ha M_2 determinisztikus, akkor M_1 is az

Példa: Mivel $SAT \in NP$ és $\text{PÁROSÍTÁS} \leq_p SAT$, kapjuk, hogy $\text{PÁROSÍTÁS} \in NP$

NP-TELJESSÉG

Legyen \mathbb{C} egy problémaosztály

- Egy L probléma \mathbb{C} -nehéz (a polinom idejű visszavezetésre nézve), ha minden $L' \in \mathbb{C}$ esetén $L' \leq_p L$
- Ha még $L \in \mathbb{C}$ is teljesül, akkor $L \in \mathbb{C}$ -teljes

Állítás: Legyen L egy NP-teljes probléma; ha L megoldható polinomidőben determinisztikus Turing-géppel, azaz $L \in P$, akkor $P = NP$

- Mivel $P \subseteq NP$, elég megmutatni, hogy $NP \subseteq P$
- Legyen $L' \in NP$ egy tetszőleges probléma; ekkor $L' \leq_p L$ (mert L NP-teljes),
- De akkor $L' \in P$ (mert P zárt a polinomidejű visszavezetésre nézve)

Megjegyzés: Az NP-teljes problémák a legnehezebben megoldhatók az NP osztályon belül (a fenti állítás alapján az nem lehet, hogy egy NP-teljest megoldok „könnyen”, de van olyan NP-beli, amit nem tudok megoldani „könnyen”)

BONYOLULTSÁGELMÉLET – 8. ELŐADÁS

ELŐADÓ: GAZDAG ZSOLT

KÉSZÜLT ...

- IVÁN SZABOLCS KORÁBBI ELŐADÁSFÓLIÁI, VALAMINT
- M. SIPSER: INTRODUCTION TO THE THEORY OF COMPUTATION ÉS
- C. PAPADIMITRIOU: SZÁMÍTÁSI BONYOLULTSÁG C. KÖNYVEI ALAPJÁN

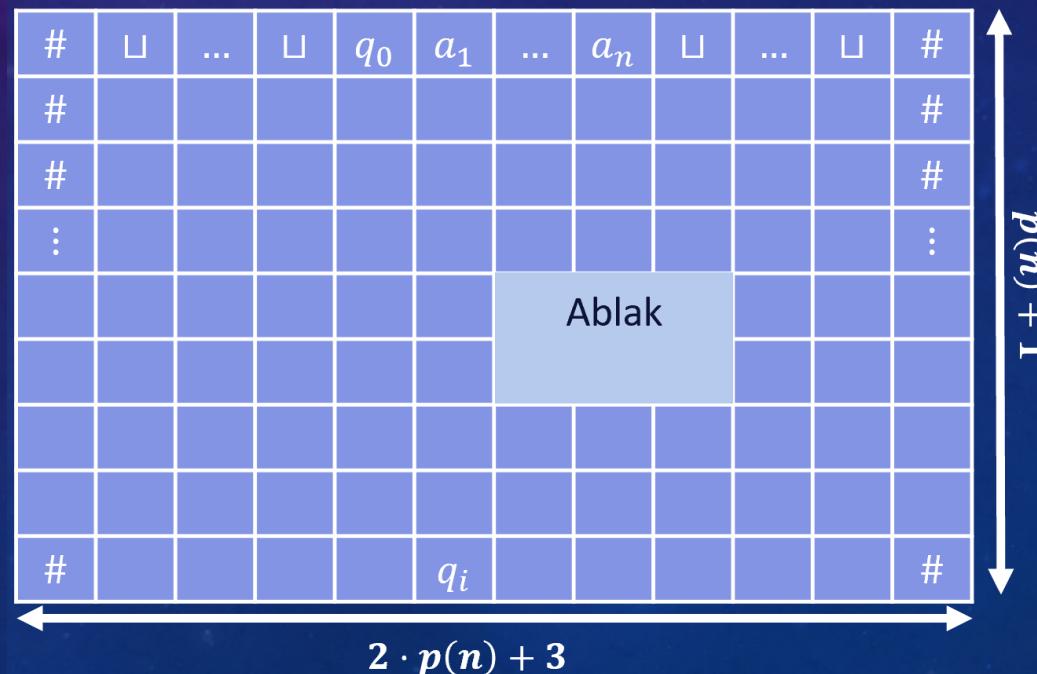
AZ ELSŐ NP-TELJES PROBLÉMA: A SAT

Cook-tétele: SAT NP-teljes

- Korábban láttuk, hogy $SAT \in NP$
- **SAT NP-nehéz:** megmutatjuk, hogy tetszőleges $L \in NP$ -re, $L \leq_p SAT$
 - Legyen $L \in NP$ és $M = (Q, \Sigma, \Gamma, \delta, q_0, q_i, q_n)$ egy L -et $p(n)$ polinom időben eldöntő nemdeterminisztikus Turing-gép
 - Legyen $w = a_1 \dots a_n \in \Sigma^*$
 - Polinom időben megkonstruálunk egy φ_w KNF-et úgy, hogy
 - $w \in L \Leftrightarrow M$ elfogadja w -t $\Leftrightarrow \varphi$ kielégíthető $\Leftrightarrow \langle \varphi_w \rangle \in SAT$
 - **Megjegyzés:**
 - M minden konfigurációjában legfeljebb $2p(n)$ darab cella nem \sqcup
 - M minden (nemdeterminisztikus) számításának legfeljebb $p(n) + 1$ egymást követő konfiguráció után el kell fogadnia vagy el kell utasítania a bemenetet

AZ ELSŐ NP-TELJES PROBLÉMA: A SAT

- M számítása w -n leírható egy T táblázattal:
 - T első sora M kezdőkonfigurációja w -n
 - T két egymást követő sora M két egymást követő konfigurációja (a két sor közötti különbség belefér egy 2×3 -as ablakba)
 - Egy ilyen ablak **legális**, ha „kompatibilis” δ -val
- T elfogadó, ha egyik sora elfogadó konfiguráció (feltesszük, hogy az ez utáni sorok megegyeznek ezzel a sorral)

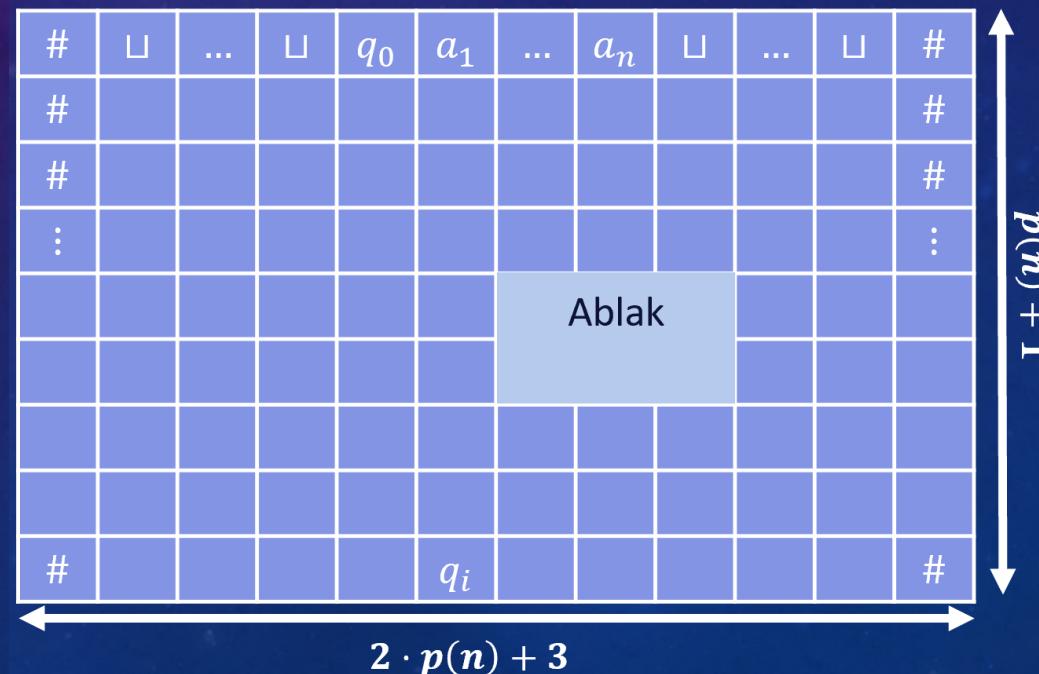


AZ ELSŐ NP-TELJES PROBLÉMA: A SAT

- Legyen φ_w a következő KNF
- φ_w ítéletváltozói $p_{i,j,s}$ alakúak, melynek jelentése: T i -ik sorának j -ik cellájában s szimbólum van, ahol $s \in C := Q \cup \Gamma \cup \{\#\}$
- $\varphi_w = \varphi_0 \wedge \varphi_{start} \wedge \varphi_{move} \wedge \varphi_{accept}$

$$\varphi_0 = \bigwedge_{\substack{1 \leq i \leq p(n)+1 \\ 1 \leq j \leq 2p(n)+3}} \left(\left(\bigvee_{s \in C} p_{i,j,s} \right) \wedge \left(\bigwedge_{s,t \in C, s \neq t} (\neg p_{i,j,s} \vee \neg p_{i,j,t}) \right) \right)$$

T(i, j)-ben van valamilyen betű
T(i, j)-ben nincs két különböző betű



AZ ELSÓ NP-TELJES PROBLÉMA: A SAT

- $\varphi_w = \varphi_0 \wedge \varphi_{start} \wedge \varphi_{move} \wedge \varphi_{accept}$

$$\varphi_{start} = p_{1,1,\#} \wedge p_{1,2,\sqcup} \wedge \dots \wedge p_{1,2p(n)+3,\#}$$

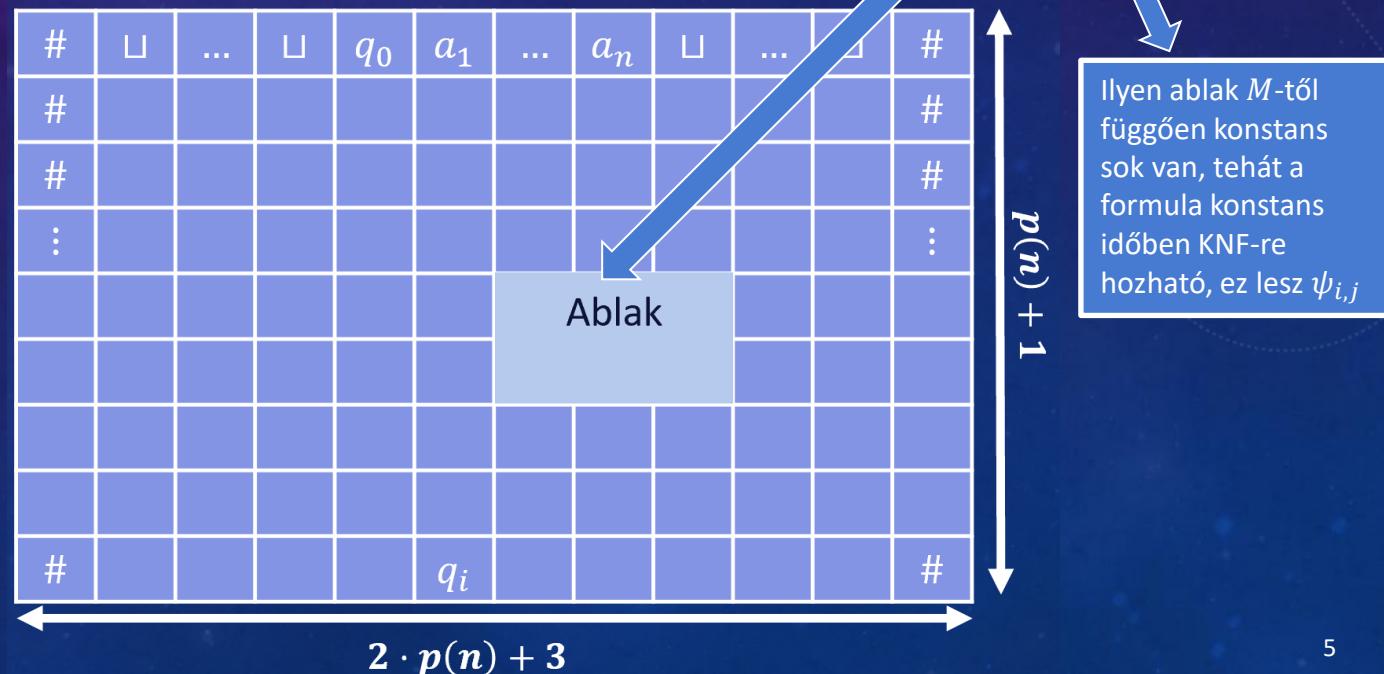
$$\varphi_{accept} = \bigvee_{j=2}^{2p(n)+2} p_{p(n)+1,j,q_i}$$

$$\varphi_{move} = \bigwedge_{\substack{1 \leq i \leq p(n) \\ 2 \leq j \leq 2p(n)+2}} \psi_{i,j}, \text{ ahol}$$

$$\psi_{i,j} \equiv \bigvee_{\substack{(b_1, \dots, b_6) \\ \text{legális ablak}}} p_{i,j-1,b_1} \wedge \dots \wedge p_{i+1,j+1,b_6}$$

Belátható, hogy

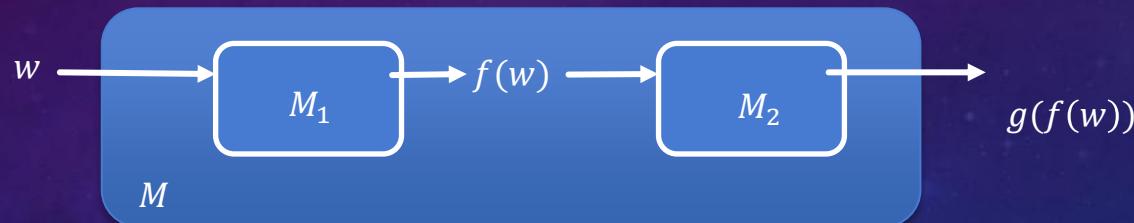
- φ_w egy KNF-ben adott formula
- φ_w polinom időben megkonstruálható
- $w \in L \Leftrightarrow$ van olyan fent vázolt táblázat, aminek utolsó sora elfogadó $\Leftrightarrow \langle \varphi_w \rangle \in \text{SAT}$
- Tehát, a fenti konstrukció L polinom idejű visszavezetése SAT-ra
- Mivel L tetszőleges NP-beli nyelv volt kapjuk, hogy **SAT NP-teljes**



A POLINOM IDEJŰ VISSZAVEZETÉSEK TRANZITÍVAK

Állítás: Ha $L_1 \leq_p L_2$ és $L_2 \leq_p L_3$, akkor $L_1 \leq_p L_3$

- Legyen f az $L_1 \leq_p L_2$, g pedig az $L_2 \leq_p L_3$ visszavezetéshez használt függvény, és legyen M_1 és M_2 rendre az f és g függvényeket kiszámoló polinom idejű Turing-gép
- Konstruáljuk meg az **M Turing-gépet** a következőképpen:



- Tehát M egy tetszőleges w szón először futtatja M_1 -et, majd a kapott $f(w)$ -n futtatja M_2 -t
- Világos, hogy M egy $L_1 \leq L_3$ visszavezetést kiszámoló Turing-gép
- legyen M_1 időigénye p_1 , M_2 -é pedig p_2 (p_1 és p_2 polinomok)
- ha w n hosszú, akkor $f(w)$ legfeljebb $p_1(n)$ hosszú lehet
- ezért az M időigénye $O(p_1(n) + p_2(p_1(n)))$, ami szintén polinom
- Tehát $L_1 \leq_p L_2$

NP-TELJESSÉG BIZONYÍTÁSA

Következmény: Legyen L NP-teljes, $L' \in \text{NP}$ és tegyük fel, hogy $L \leq_p L'$

Akkor L' is NP-teljes

- Mert tetszőleges L'' problémára $L'' \leq_p L$ (hiszen L NP-teljes)
- Tehát $L'' \leq_p L \leq_p L'$, azaz $L'' \leq_p L'$ (mert a polinom idejű visszavezetések tranzitívak)
- Ezért L' is NP-teljes (mert NP-beli, és minden NP-beli visszavezethető rá polinom időben)

Definíció: Tetszőleges $k \geq 1$ számra a $k\text{SAT}$ probléma a következő:

- Adott egy φ zérusréndű KNF, melynek minden tagjában pontosan k (nem feltétlenül különböző) literál szerepel
- Kérdés: kielégíthető-e φ ?

Világos, hogy minden k -ra $k\text{SAT} \in \text{NP}$

- Logika előadáson láttuk, hogy $2\text{SAT} \in \text{P}$
- Megmutatjuk, hogy 3SAT NP-teljes

3SAT NP-TELJES

Megmutatjuk, hogy $SAT \leq_p 3SAT$

- Tetszőleges φ KNF-ben lévő formulához konstruáljuk meg φ' -t a következő táblázat alapján, ahol p, p_1, \dots, p_{n-2} minden új, korábban nem használt ítéletváltozók:

φ egy C klóza	φ' megfelelő klózai
l	$(l \vee l \vee l)$
$l_1 \vee l_2$	$(l_1 \vee l_2 \vee l_2)$
$l_1 \vee l_2 \vee l_3$	$l_1 \vee l_2 \vee l_3$
$l_1 \vee l_2 \vee l_3 \vee l_4$	$(l_1 \vee l_2 \vee p) \wedge (\neg p \vee l_3 \vee l_4)$
$l_1 \vee \dots \vee l_n$ ($n \geq 5$)	$(l_1 \vee l_2 \vee p_1) \wedge (\neg p_1 \vee l_3 \vee p_2) \wedge \dots \wedge (\neg p_{n-2} \vee l_{n-1} \vee l_n)$

Belátható, hogy φ kielégíthető $\Leftrightarrow \varphi'$ kielégíthető

- Tegyük fel, hogy egy A változóértékkedásra $A \models \varphi$ és legyen C a φ egy klóza

3SAT NP-TELJES

- Ha a C klöz $l, l_1 \vee l_2$ vagy $l_1 \vee l_2 \vee l_3$ alakú, akkor A triviálisan kielégíti φ' megfelelő klózait
- Ha $C = l_1 \vee l_2 \vee l_3 \vee l_4$, akkor
 - $A \vDash (l_1 \vee l_2)$ vagy $A \vDash (l_3 \vee l_4)$
 - Terjesszük ki A -t a p változóra a következőképpen
 - Legyen $A(p) = 0$ ha $A \vDash (l_1 \vee l_2)$, és legyen $A(p) = 1$ ha $A \vDash (l_3 \vee l_4)$ (ha minden eset fennáll, akkor $A(p)$ értéke tetszőleges)
 - Ekkor A kielégíti $(l_1 \vee l_2 \vee p)$ -t és $(\neg p \vee l_3 \vee l_4)$ -t is, azaz kielégíti a C -hez megkonstruált $(l_1 \vee l_2 \vee p) \wedge (\neg p \vee l_3 \vee l_4)$ formulát
- Ha $C = l_1 \vee \dots \vee l_n$ ($n \geq 5$), akkor a fentihez hasonlóan kiegészíthetjük A -t úgy, hogy kielégítse φ' megfelelő klózát

A fentiek alapján másik irány könnyen látható: Ha egy A értékadásra $A \vDash \varphi'$, akkor $A \vDash \varphi$

A SAT VÁLTOZATAI

Logika előadáson láttuk, hogy a HORNSAT probléma is P-beli: Adott egy φ Horn-formula (olyan KNF, ahol minden klóz legfeljebb egy pozitív literált tartalmaz); kielégíthető-e φ ?

Most megmutatjuk, hogy bár 2SAT és HORNSAT P-beli, a következő probléma (nevezzük 2-3SAT-nak) NP-teljes: adott egy φ KNF, ahol minden klóz pontosan kettő literált vagy pontosan három negatív literált tartalmaz; kielégíthető-e φ ?

- Ehhez megmutatjuk, hogy $3\text{SAT} \leq_p 2\text{-3SAT}$:
 - Legyen $\varphi \mapsto \varphi'$ a következő leképezés: φ minden C klózára
 - ha C olyan, hogy három negatív literált tartalmaz, akkor vegyük fel C -t φ' -be is
 - Egyébként legyen C' az a klóz, amit úgy kapunk C -ből, hogy benne minden x változót kicserélünk $\neg\hat{x}$ -re, ahol \hat{x} egy új, még nem használt változó, és felvesszük φ' -be C' -t és az $x \vee \hat{x}$ és $\neg x \vee \neg\hat{x}$ klózokat
 - A $\varphi \mapsto \varphi'$ leképezése polinom időben kiszámítható és visszavezetés is:
 - A φ változónak egy tetszőleges A értékadására legyen A' az A kiterjesztése az új változókra úgy, hogy $A' \models \hat{x} \Leftrightarrow A \not\models x$. Belátható, hogy ha $\varphi \models A$ akkor $A' \models \varphi'$
 - Másrészt, ha egy tetszőleges A' -re $A' \models \varphi'$, akkor az $x \vee \hat{x}$ és $\neg x \vee \neg\hat{x}$ klózok miatt A' nem rendeli ugyanazt az értéket x -hez és \hat{x} -hoz. Ebből következik, hogy $A' \models \varphi$
 - Tehát φ' akkor és csak akkor kielégíthető, ha φ' is az

GRÁFELMÉLETI NP-TELJES PROBLÉMÁK

FÜGGETLEN CSÚCSHALMAZ

- Adott: $G = (V, E)$ irányítatlan gráf és egy K szám
- Kérdés: Van-e G -ben K darab olyan csúcs, melyek független csúcshalmazt alkotnak? (Azaz van-e G -ben K darab páronként nem szomszédos csúcs?)

Állítás: FÜGGETLEN CSÚCSHALMAZ NP-teljes

- NP-beli, mert polinom időben verifikálható: tetszőleges K csúcsról polinom időben eldönthető, hogy páronként nem szomszédosak-e
- NP-nehéz, mert 3SAT \leq_p FÜGGETLEN CSÚCSÚCSHALMAZ
 - Legyen $\varphi = (l_{11} \vee l_{12} \vee l_{13}) \wedge \dots \wedge (l_{k1} \vee l_{k2} \vee l_{k3})$, és konstruáljuk meg G_φ -t úgy, hogy minden klózhöz felveszünk egy háromszöget, melynek csúcsai a klóz literáljaival címkézettek
 - Továbbá minden ellentétes literál közé felveszünk egy élt. Végül: legyen $K := k$
- G_φ polinom időben megkonstruálható, valamint φ kielégíthető \Leftrightarrow
 - van olyan A értékadás, ami kielégíti minden klóz legalább egy literálját \Leftrightarrow
 - ezen literálokkal címkézett literálok páronként nem szomszédosak G_φ -ben \Leftrightarrow
- G_φ -ben van k csúcsú független csúcshalmaz

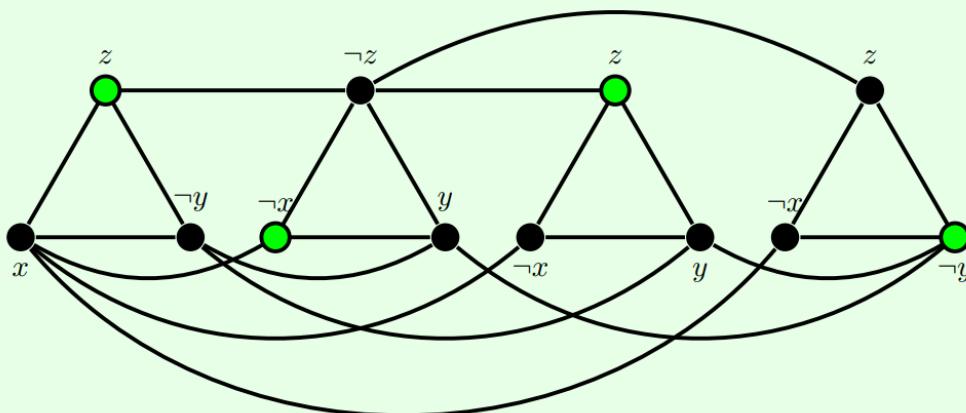
FÜGGETLEN CSÚCSSHALMAZ

Példa

Legyen $\varphi = (x \vee \neg y \vee z) \wedge (\neg x \vee y \vee \neg z) \wedge (\neg x \vee y \vee z) \wedge (\neg x \vee \neg y \vee z)$.

Ekkor a G gráf:

G csúcsai nem színesek, az csak illusztráció!



és $K = 4$.

A zölddel jelölt csúcsok egy négyelemű független csúcshalmazt alkotnak; egy kielégítő kiértékelés pedig: $x = y = 0, z = 1$.

HA VALAMI NEHÉZ, AKKOR AZ ÁLTALÁNOSÍTÁSA IS AZ

Vegyük például az ÁLT-SAT problémát:

- Adott egy φ ítéletkalkulusbeli formula; vajon kielégíthető-e?
- Ez a probléma szintén **NP-teljes**: **NP-beli**, amit hasonlóan lehet belátni, mint a SAT NP-beliségét; az pedig, hogy **NP-nehéz** triviális: bármit, amit a SAT-ra vissza lehet vezetni polinom időben, az ÁLT-SAT-ra is vissza lehet (ugyanaz a konstrukció jó lesz)

Tehát egy **NP-nehéz** probléma általánosítása szintén **NP-nehéz**

Vegyük például a következő problémákat

CSÚCSLEFEDÉS

- Adott: $G = (V, E)$ irányítatlan gráf és egy K szám
- Kérdés: Van-e G -ben K darab olyan csúcs, hogy G minden élének egyik végpontja ezen csúcsok valamelyikére esik?

Állítás: CSÚCSLEFEDÉS NP-teljes (gyakorlaton)

HA VALAMI NEHÉZ, AKKOR AZ ÁLTALÁNOSÍTÁSA IS AZ

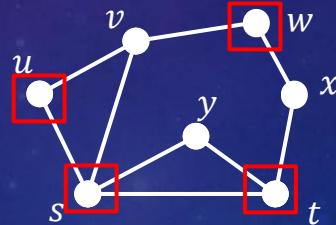
HITTING SET

- Adott egy U halmaz, $T = \{s_1, \dots, s_n\}$, ahol $s_1, \dots, s_n \subseteq U$, és egy K szám
- Kérdés: Van-e olyan K elemű $H \subseteq U$ halmaz (a hitting set) ami tartalmaz minden T -beli halmazból legalább egy elemet?

Állítás: HITTING SET NP-teljes

- Egyszerűtlenül NP-beli, mert polinom időben verifikálható
- Másrész a HITTING SET probléma a CSÚCSLEFEDÉS általánosítása: ha a HITTING SET bemenetként csak olyan T -ket engedünk meg, amiben csak kételemű halmazok szerepelhetnek, akkor a CSÚCSLEFEDÉS problémát kapjuk

Például vegyük a következő gráfot:



- Ebben négy elemű független csúcshalmazt keresni ugyanaz, mint az élek halmazában (ami most a T) keresni egy négy elemű „hitting set”-et:
 - $T = \{\underline{u}, v\}, \{\underline{u}, s\}, \{v, \underline{w}\}, \{\underline{s}, v\}, \{\underline{w}, x\}, \{x, \underline{t}\}, \{\underline{s}, y\}, \{y, \underline{t}\}, \{\underline{s}, t\}\}$
 - Egy négy elemű hitting set T -ben: $H = \{u, s, t, w\}$

HITTING SET ÉS HALMAZ FEDÉS

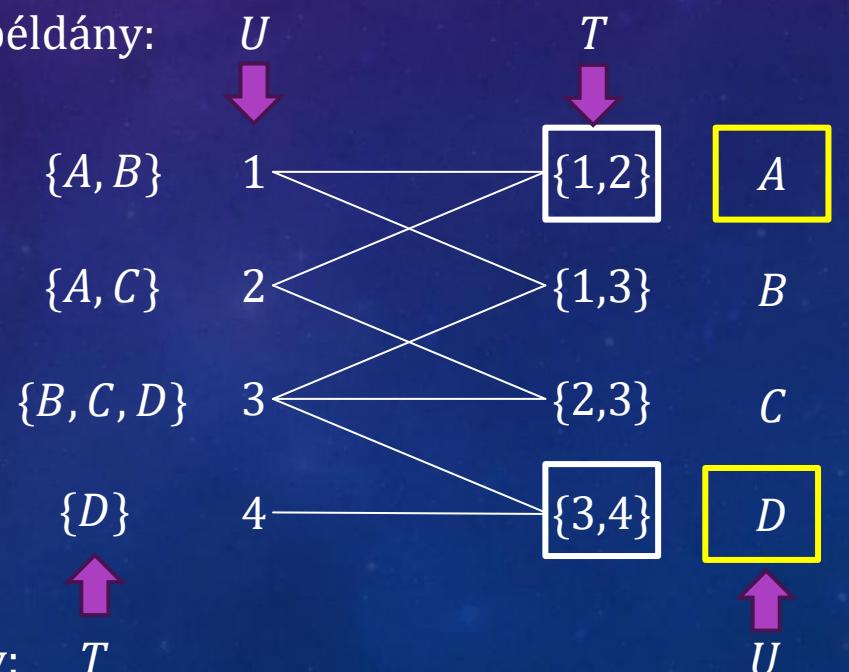
Ugyanazt a problémát többféleképpen is meg lehet fogalmazni

Példa: HALMAZ FEDÉS

- Adott egy U halmaz, U részhalmazainak egy $T = \{s_1, \dots, s_n\}$ sorozata és egy K szám
- Kérdés: Van-e ezen halmazok közül K darab olyan melyek uniója U ?

HALMAZ FEDÉS ugyanaz a probléma, mint a HITTING SET, tehát NP-teljes:

HALMAZ FEDÉS pozitív példány:



HITTING SET pozitív példány: T

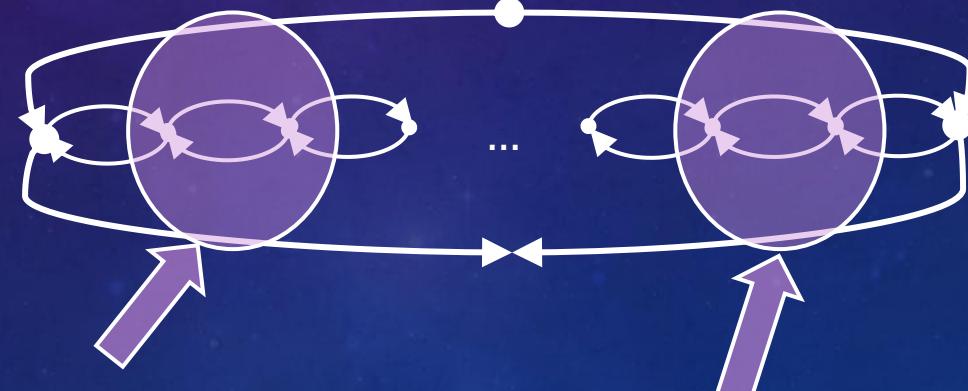
$K = 2$ 15

HAMILTON-ÚT_{1→n} NP-TELJES

Azt már láttuk, hogy HAMILTON-ÚT_{1→n} NP-beli

Megmutatjuk, hogy SAT \leq_p HAMILTON-ÚT_{1→n}

- Legyen $\varphi = c_1 \wedge \dots \wedge c_k$ egy tetszőleges KNF-ben adott formula
- Tegyük fel, hogy a φ -ben használt változók: p_1, p_2, \dots, p_l
- minden p_i változóhoz konstruálunk meg egy részgráfot a következőképpen:

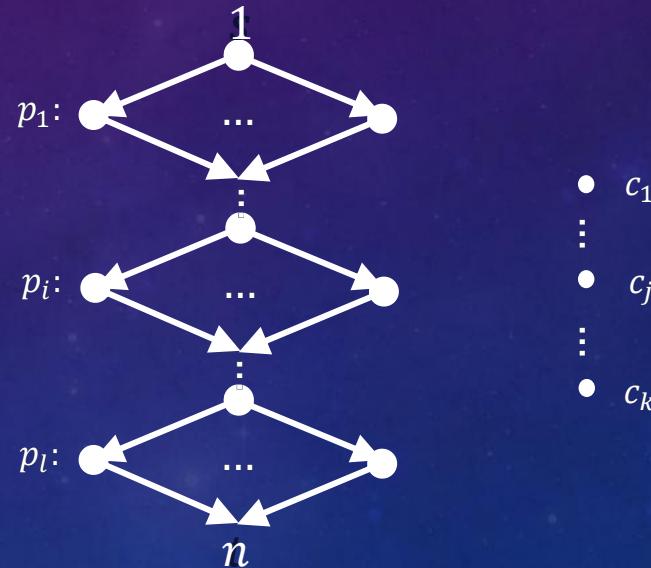


c₁-nek megfeleltetett
csúcspár

c_k-nak megfeleltetett
csúcspár

HAMILTON-ÚT $_{1 \rightarrow n}$ NP-TELJES

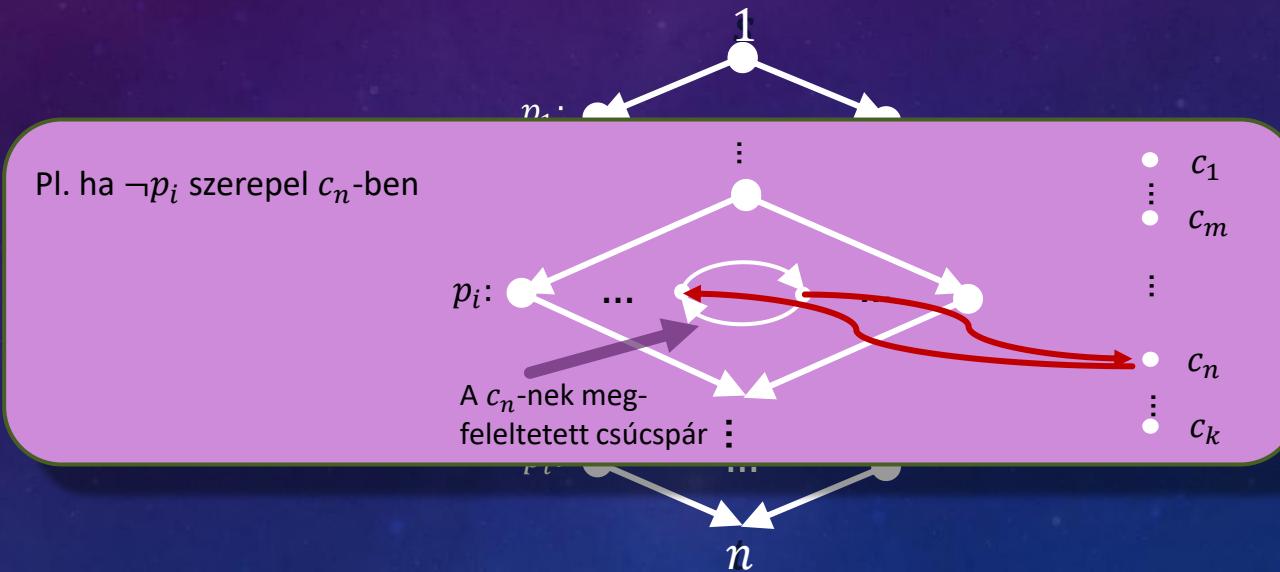
Legyen G_φ az alábbi gráf:



- c_1
- ⋮
- c_j
- ⋮
- c_k

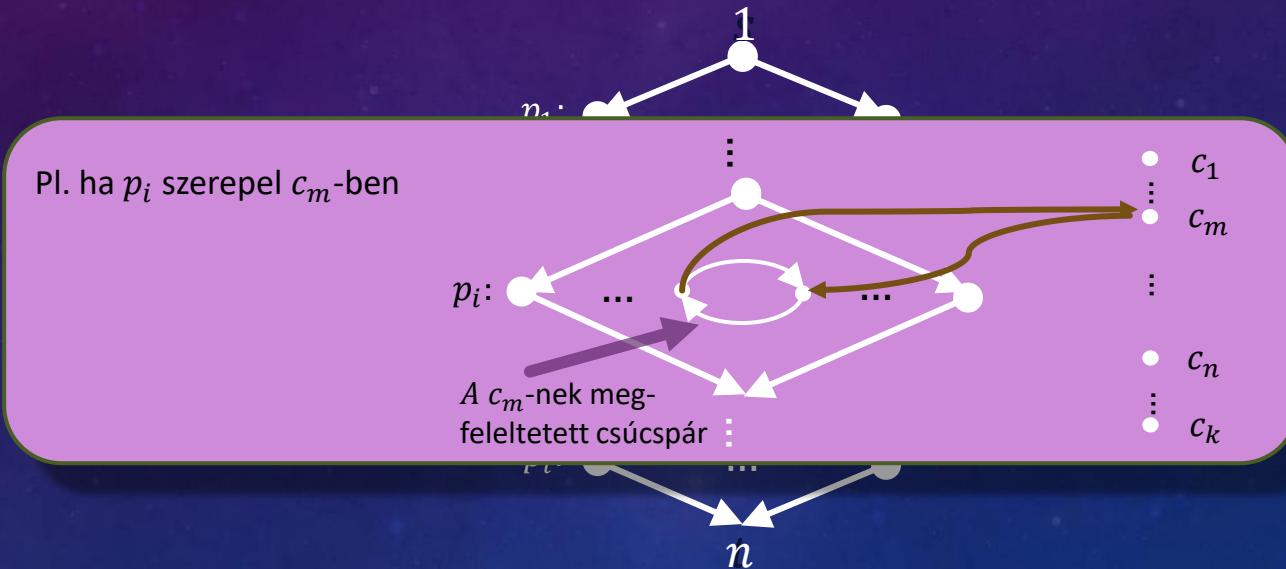
HAMILTON-ÚT $_{1 \rightarrow n}$ NP-TELJES

A klázoknak megfelelő csúcsok meglátogatása a rombuszokból:



HAMILTON-ÚT $_{1 \rightarrow n}$ NP-TELJES

A klázoknak megfelelő csúcsok meglátogatása a rombuszokból:



HAMILTON-ÚT_{1→n} NP-TELJES

- G_φ polinom időben megkonstruálható
- Továbbá, belátható, hogy
 φ kielégíthető $\Leftrightarrow G_\varphi$ -ben van Hamilton-út 1-ből n -be
- Mert
 - minden 1-ből n -be vezető u út meghatározza a φ -beli változók egy A értékkadását:
 - u egy p_i -nek megfelelő részgráf tetején pontosan akkor megy balra, ha $A(p_i) = 1$
 - Ekkor $A \models \varphi \Leftrightarrow u$ Hamilton úttá alakítható

Példa: Konstruáljuk meg G_φ -t, ha $\varphi = (p \vee \neg q) \wedge (\neg p \vee q)$

BONYOLULTSÁGELMÉLET – 9. ELŐADÁS

ELŐADÓ: GAZDAG ZSOLT

KÉSZÜLT ...

- IVÁN SZABOLCS KORÁBBI ELŐADÁSFÓLIÁI, VALAMINT
- M. SIPSER: INTRODUCTION TO THE THEORY OF COMPUTATION ÉS
- C. PAPADIMITRIOU: SZÁMÍTÁSI BONYOLULTSÁG C. KÖNYVEI ALAPJÁN

HAMILTON-ÚT KERESÉS IRÁNYÍTATLAN GRÁFBAN

Legyen HAMILTON-ÚT_{1-n} a következő probléma:

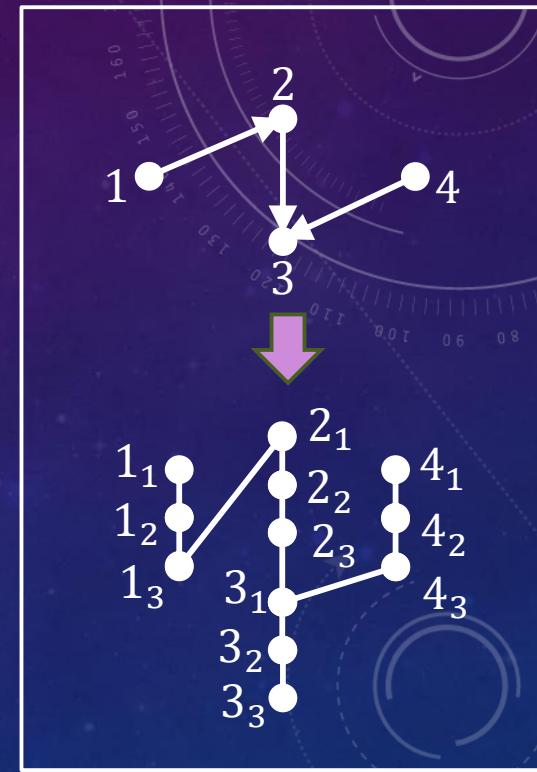
- Adott: $G = (V, E)$ irányítatlan gráf, ahol $V = \{1, \dots, n\}$
- Kérdés: Van-e G -ben Hamilton út 1-ből n -be?

Állítás: HAMILTON-ÚT_{1-n} NP-teljes

- Visszavezetjük rá az irányított változatot
- Tetszőleges $G = (\{1, \dots, n\}, E)$ irányított gráfra legyen $G' = (V', E')$ a következő irányítatlan gráf
 - minden $u \in V$ -re, $u_1, u_2, u_3 \in V'$
 - minden $u \in V$ -re, $\{u_1, u_2\}, \{u_2, u_3\} \in E'$
 - minden $(u, v) \in E$ -re, $\{u_3, v_1\} \in E'$
- G' polinom időben megkonstruálható, és belátható, hogy

G -ben van Hamilton-út 1-ből n -be $\Leftrightarrow G'$ -ben van Hamilton-út 1_1 -ből n_3 -ba

- Megjegyzés: A csúcsok triplázása biztosítja G' -ben azt hogy a G -beli élek az irányításuk elvesztése után is pontosan akkor alkossanak Hamilton-utat, ha G -ben azt alkotnak



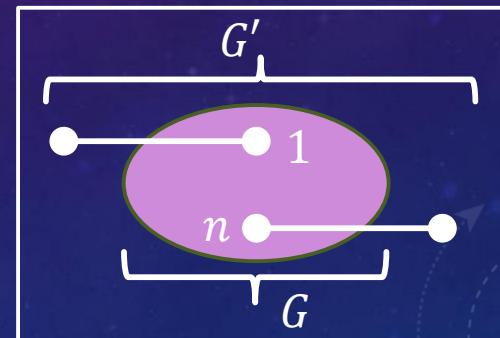
TETSZŐLEGES HAMILTON-ÚT KERESÉSE

Legyen **HAMILTON-ÚT \rightarrow** , a következő probléma:

- Adott: $G = (V, E)$ irányított gráf, ahol $V = \{1, \dots, n\}$
- Kérdés: Van-e G -ben Hamilton út tetszőleges két csúcs között

Hasonlóan, legyen **HAMILTON-ÚT** a **HAMILTON-ÚT \rightarrow** azon változata, ahol a bemenet egy irányítatlan gráf

- Ezek is NP-beliek, és visszavezethető rájuk a megfelelő 1-ből n -be vezető Hamilton-utat kereső probléma. Az irányítatlan esetben pl. így:
- Következik, hogy ezek is NP-teljesek



Hasonlóan NP-teljesek a **HAMILTON-ÚT $_{s \rightarrow t}$** és **HAMILTON-ÚT $_{s-t}$** problémák is

- Itt a bemenet részét képezi két tetszőleges csúcsa a gráfnak, s, t , és ezek között keresünk Hamilton-utat (ezek az 1-ből n -be tartó Hamilton-utat kereső változatok általánosításai)

FESZÍTŐFA KERESÉSE

Egy G irányítatlan gráf feszítőfája a G egy olyan fa részgráfja, ami a G összes csúcsát tartalmazza

- Feszítőfát keresni lehet **polinom időben**, pl. a Kruskal algoritmussal

Legyen **KORLÁTOZOTT FESZÍTŐFA** a következő probléma:

- Adott egy G irányítatlan gráf és egy K szám
- Döntsük el, hogy van-e G -nek olyan feszítőfája, melyben a **csúcsok fokszáma legfeljebb K**

KORLÁTOZOTT FESZÍTŐFA NP-teljes:

- NP-beli: a Turing-gép **nemdeterminisztikusan** legenerálja a G egy G' részgráfját, és **polinom időben ellenőrzi**, hogy G' összefüggő és körmentes-e (azaz fa gráf-e) valamint azt, hogy benne minden csúcs foka legfeljebb K -e
- NP-nehéz, mert a **HAMILTON-ÚT** általánosítása:
 - Egy G gráfban egy olyan feszítőfa, melyben a csúcsok kifoka legfeljebb 2 nem más, mint egy Hamilton-út

HAMILTON-KÖR

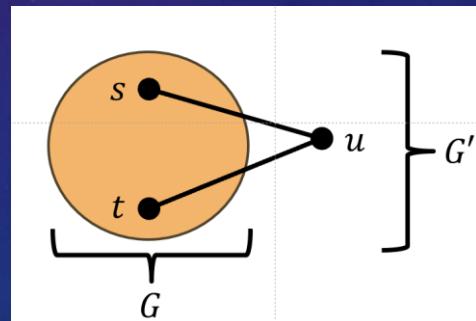
Legyen HAMILTON-KÖR a következő probléma:

- Adott egy G irányítatlan gráf
- Döntsük el, hogy van-e G -ben minden csúcsot pontosan egyszer érintő kör

HAMILTON-KÖR NP-teljes: Az világos, hogy NP-beli,

NP-nehéz is: visszavezetjük rá a HAMILTON-ÚT $_{s-t}$ problémát a következőképpen

- Legyen G egy irányítatlan gráf és s, t a G két csúcsa
- Konstruáljuk meg G' -t az ábrán látható módon:



- G' hatékonyan megkonstruálható, és világos, hogy akkor és csak akkor van benne Hamilton-kör, ha G -ben van Hamilton-út s -ből t -be

AZ UTAZÓÜGYNÖK (TSP) PROBLÉMA

A TSP probléma

- Adott városok $1, \dots, n$ halmaza, és minden i, j városra a két város $d_{i,j}$ távolsága
- A feladat az, hogy találjunk meg az összes várost pontosan egyszer érintő legrövidebb körutat

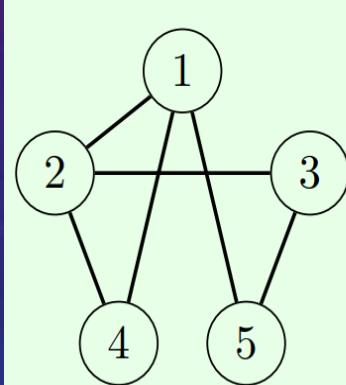
A probléma eldöntési változata a TSP(E) probléma, ahol a bemenet része egy K szám is, és a kérdés az, hogy van-e van-e legfeljebb K költségű körút

- A „brute force” módszer: az összes $\frac{(n-1)!}{2}$ lehetséges körút megvizsgálásával
- Dinamikus programozással: $\mathcal{O}(2^n \cdot n^2)$ időigény
- Nem ismert rá polinomidejű algoritmus

AZ UTAZÓÜGYNÖK (TSP) PROBLÉMA

Állítás: TSP(E) NP-teljes

- NP-beli: A Turing-gép nemdeterminisztikusan generál egy n csúcsot, és polinom időben ellenőrzi, hogy ezek egy legfeljebb K súlyú Hamilton-kört alkotnak-e
- NP-nehéz: HAMILTON-KÖR \leq_p TSP(E)
- Adott n csúcsú G gráfhoz konstruáljuk meg TSP(E) bemenetét a következőképpen:
 - A csúcsok legyenek ugyanazok, mint G -ben
 - $d_{i,j} = 1$, ha (i,j) él volt G -ben, 2 egyébként
 - A célérték: n
 - Ekkor ha van Hamilton-kör G -ben, annak összköltsége n
 - Másrészt, ha van egy legfeljebb n súlyú körutunk, abban minden az n él súlya 1 kell legyen; ezek Hamilton-utat alkotnak G -ben



\Rightarrow

d	1	2	3	4	5
1	0	1	2	1	1
2	1	0	1	1	2
3	2	1	0	2	1
4	1	1	2	0	2
5	1	2	1	2	0

és a célérték $C = 5$.

GRÁFSZINEZÉSI PROBLÉMÁK

3-SZÍNEZÉS

- ▶ **Input:** $G = (V, E)$ (irányítatlan) gráf.
- ▶ **Output:** kiszínezhető-e G helyesen, vagyis adhatunk-e minden csúcsnak egy-egy színt egy háromelemű színhalmazból, hogy szomszédos csúcsok különböző színt kapjanak?

Hasonlóan definiálható tetszőleges k számra a **k -SZÍNEZÉS** probléma

- **2-SZÍNEZÉS** probléma P -ben van, hiszen a gyakorlaton láttuk, hogy $2\text{-SZÍNEZÉS} \leq_p 2\text{SAT}$, 2SAT pedig P -ben van
- A **4-SZÍNEZÉS** probléma triviális síkbarajzolható gráfokra: minden síkbarajzolható gráf színezhető 4 színnel

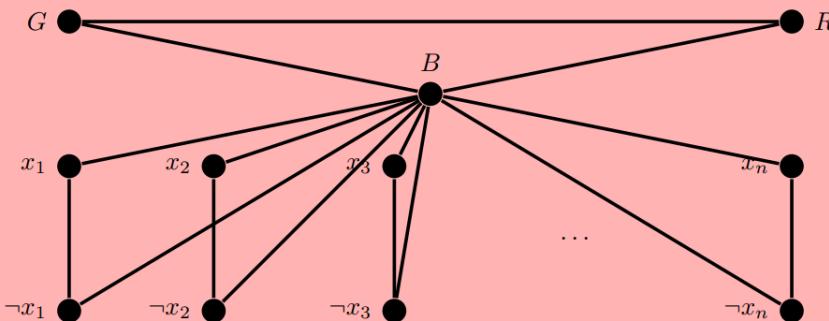


Forrás: Wikipedia

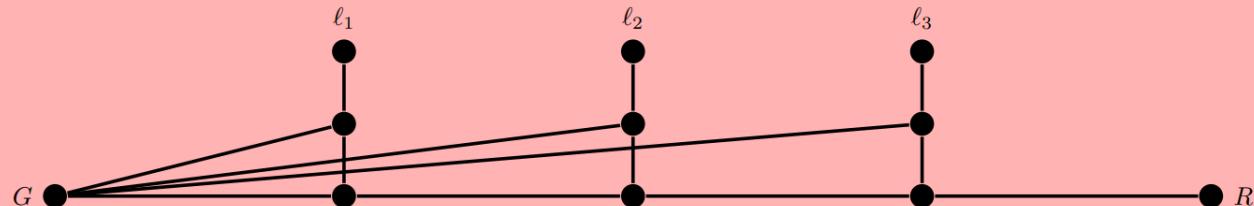
GRÁFSZINEZÉSI PROBLÉMÁK

3-SZÍNEZÉS probléma NP-teljes:

- NP-beli, és
- $3\text{SAT} \leq_p 3\text{-SZÍNEZÉS}$



így minden literál színe R (=„hamis”) vagy G (=„igaz”) színével kell megegyezzen, komplementere pedig egy másikkal \Rightarrow kiértékelés

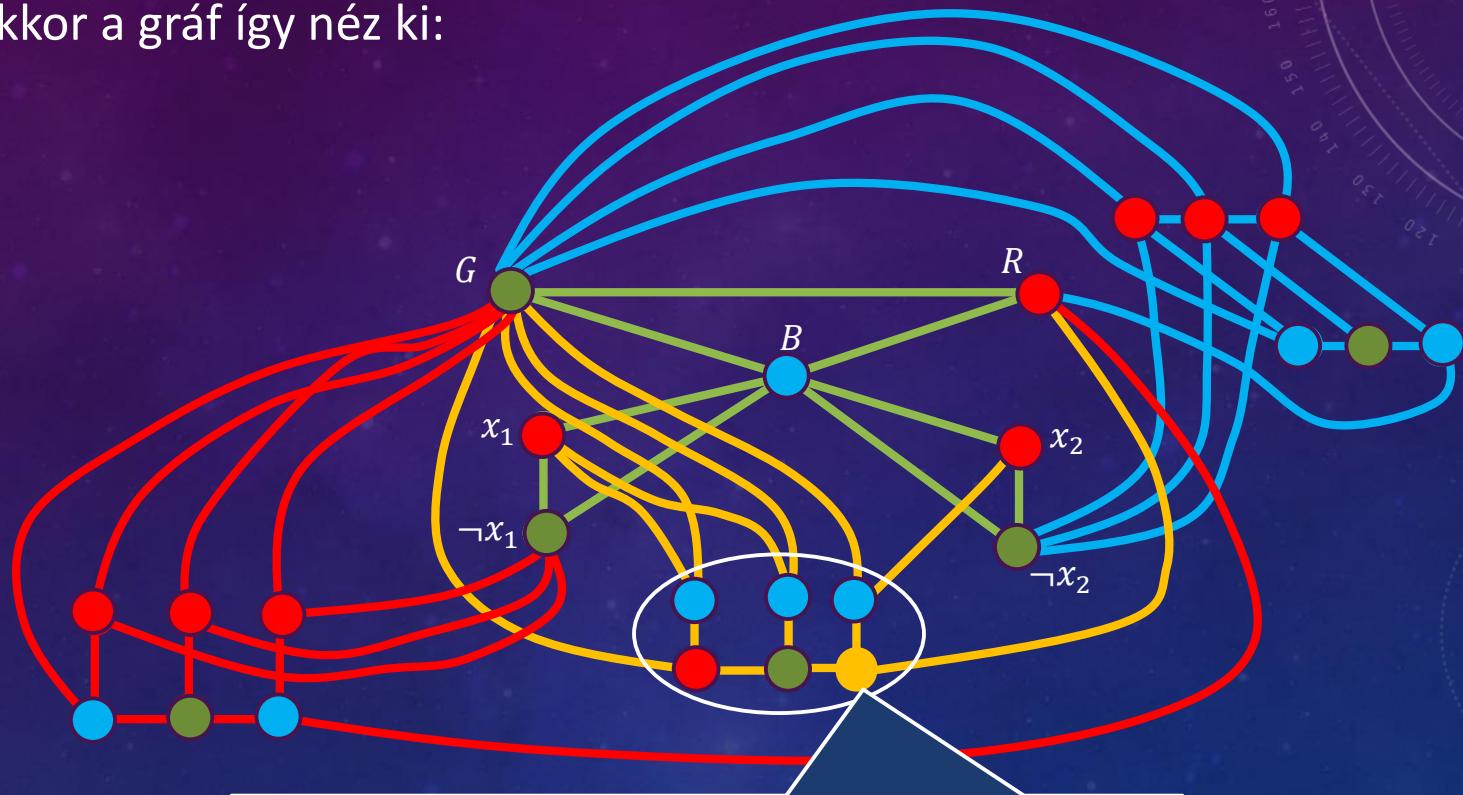


Ellenőrizhető: ha minden literál R színét kapja, a plusz hat csúcsot nem lehet helyesen 3-színezni, ha viszont van köztük G színű, akkor igen

$3\text{SAT} \leq_p 3\text{-SZÍNEZÉS} - \text{PÉLDA}$

Legyen $\varphi = (x_1 \vee x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_1 \vee \neg x_1) \wedge (\neg x_2 \vee \neg x_2 \vee \neg x_2)$

- Akkor a gráf így néz ki:



Ha például az $x_1 = x_2 = 0$ értékkadásnak megfelelően színezünk, akkor a sárga klóznak megfelelő csúcsok közül a felső hármat kékre kell színezni, azaz az alsó csúcsokat nem tudjuk megfelelően színezni

- A gráf nem háromszínezhető $\Rightarrow \varphi$ kielégíthetetlen

NP-TELJESSÉG HALMAZOKRA

Hármasítás

Adott: Három azonos méretű halmaz, F (fiúk), L (lányok), H (házak), és egy $R \subseteq F \times L \times H$ reláció.

Kérdés: Megadható-e az R -beli hármasok egy olyan részhalmaza, melyben minden fiú, lány és ház pontosan egyszer szerepel?

Állítás: HÁRMASÍTÁS NP-teljes (nem bizonyítjuk)

Megjegyzés: PÁROSÍTÁS $\in \text{P}$

HÁRMASÍTÁS az alábbi probléma egy speciális esete:

PONTOS LEFEDÉS HÁRMASOKKAL

Adott: Egy $3m$ elemű U halmaz és 3-elemű részhalmazainak (S_1, \dots, S_n) rendszere.

Kérdés: Kiválasztható m darab S_i úgy, hogy ezek egyesítése U ?

(A kiválasztott S_i -k nyilván diszjunktak.)

Következik, hogy PONTOS LEFEDÉS HÁRMASOKKAL (ami pedig a HALMAZ FEDÉS speciális esete) is NP-teljes

NP-TELJES PROBLÉMÁK SZÁMOKRA

EGÉSZ ÉRTÉKŰ PROGRAMOZÁS

Adott: Egy n -változós, egész együtthatós lineáris egyenlőtlenség-rendszer.

Kérdés: Létezik-e **egész értékű** megoldása?

Ötlet

A HALMAZLEFEDÉS felfogható az EGÉSZ ÉRTÉKŰ PROGRAMOZÁS speciális eseteként:

$$Ax \geq 1, \quad \sum_{i=1}^n x_i \leq K, \quad 0 \leq x_i \leq 1,$$

ahol A oszlopai a halmazrendszer elemeinek felelnek meg.

Azt is meg lehet mutatni, hogy az EGÉSZ ÉRTÉKŰ PROGRAMOZÁS **NP**-ben van.

Az EGÉSZ ÉRTÉKŰ PROGRAMOZÁS probléma **NP-teljes**.

EGÉSZ ÉRTÉKŰ PROGRAMOZÁS NP-TELJES

Példa a visszavezetésre

Pl. ha a HALMAZLEFEDÉS problémában $U = \{1, 2, 3, 4, 5\}$, a halmazaink $S_1 = \{1, 2, 3\}$, $S_2 = \{1, 3, 5\}$, $S_3 = \{1, 4, 5\}$ és $K = 2$, akkor a generált egyenlőtlenségrendszer:

$$x_1 + x_2 + x_3 \leq 2$$

$$x_1 + x_2 + x_3 \geq 1$$

$$x_1 \geq 1$$

$$x_1 + x_2 \geq 1$$

$$x_3 \geq 1$$

$$x_2 + x_3 \geq 1$$

és $0 \leq x_1, x_2, x_3 \leq 1$.

Ha egy megoldásban $x_i = 1$, az „azt jelenti”, hogy S_i -t kiválasztjuk; ha $x_i = 0$, akkor nem (más opció az utolsó feltétel miatt, és mert a változók egészértékűek, nincs).

Így az első feltétel azt mondja, hogy két halmazt választunk ki legfeljebb.

A többi pedig rendre azt, hogy legalább egy olyan halmazt is kiválasztunk, akiben szerepel az 1-es; legalább egy olyat is, akiben szerepel a 2-es, stb.

EGÉSZ ÉRTÉKŰ PROGRAMOZÁS NP-TELJES – MÁSHOGYAN

Előző előadás: a HALMAZ FEDÉS és a HITTING SET gyakorlatilag ugyanazok a problémák; CSÚCSLEFEDÉS a HITTING SET speciális esete

- Következik, hogy a CSÚCSLEFEDÉS felfogható az EGÉSZ ÉRTÉKŰ PROGRAMOZÁS speciális esete is

Így CSÚCSLEFEDÉS könnyen visszavezethető EGÉSZ ÉRTÉKŰ PROGRAMOZÁS-ra:

- Legyen $G = (V, E)$ és K a CSÚCSLEFEDÉS egy bemenete
- Konstruáljuk meg a következő feladatot (legyen $n = |V|$):

Minden u csúcsra: $0 \leq x_u \leq 1$ (a csúcsok egy kiválasztása: $x_u = 1 \Leftrightarrow u$ kiválasztott)

Minden $\{u, v\}$ élre: $x_u + x_v \geq 1$ (minden él legalább egyik végpontját ki kell választani)

$$\sum_{i=1}^n x_i \leq K \quad (\text{legfeljebb } K \text{ csúcs lehet kiválasztva})$$

- Belátható, hogy a fenti feladatnak pontosan akkor van egész értékű megoldása, ha G -ben van K elemű csúcslefedés

NP-TELJES PROBLÉMÁK SZÁMOKRA

RÉSZLETÖSSZEG probléma:

- Adott: a_1, \dots, a_n pozitív egészek és egy K szám
- Kérdés: Kiválasztható-e az a_i -k közül néhány úgy, hogy összegük K legyen?

Állítás: A RÉSZLETÖSSZEG probléma NP-teljes

Az NP-beliség világos, az NP-nehézséget a 3SAT-ról való visszavezetéssel igazoljuk

- Legyen $\varphi = c_1 \wedge \dots \wedge c_n$ egy 3CNF, $c_i = \ell_{i,1} \vee \ell_{i,2} \vee \ell_{i,3}$, a φ -beli változók: x_1, \dots, x_m
 - ▶ Ebből elkészítjük a RÉSZLETÖSSZEG probléma egy példányát, melyben $n + m$ -jegyű (!) számok szerepelnek.
 - ▶ x_i -ből a t_i és f_i számok készülnek:
 - ▷ t_i és f_i első m jegye csupa 0, kivéve az i . jegyet, ami 1-es;
 - ▷ t_i utolsó n jegye közül az $m + k$. akkor 1-es, ha c_k -ban szerepel x_i ,
egyébként 0;
 - ▷ f_i utolsó n jegye közül az $m + k$. akkor 1-es, ha c_k -ban szerepel $\neg x_i$,
egyébként 0.
 - ▶ Továbbá, $a_i = b_i = 0^{m+i-1}10^{n-i}$, minden $i = 1, \dots, n$ esetén
 - ▶ Az eredmény: a $\{f_i, t_i : 1 \leq i \leq m\} \cup \{a_i, b_i : 1 \leq i \leq n\}$ halmaz és a
 $K = 11 \dots 133 \dots 3$ célszám (m darab 1-es, majd n darab 3-as).

RÉSZLETÖSSZEG NP-TELJES

Példa

Ha $\varphi = (x_1 \vee \neg x_2 \vee x_3) \vee (x_1 \vee \neg x_2 \vee x_4) \vee (\neg x_1 \vee x_2 \vee \neg x_4)$:

$$t_1 = 1000110$$

$$f_1 = 1000001$$

$$t_2 = 0100001$$

$$f_2 = 0100110$$

$$t_3 = 0010100$$

$$f_3 = 0010000$$

$$t_4 = 0001010$$

$$f_4 = 0001001$$

$$a_1 = b_1 = 0000100$$

$$a_2 = b_2 = 0000010$$

$$a_3 = b_3 = 0000001$$

$$\underline{K = 1111333}$$

- ▶ t_1 és f_1 közül pontosan az egyiket kell válasszuk (K első jegye miatt);
- ▶ általában t_i és f_i közül is pontosan az egyiket – t_i választása feleljen meg az $x_i = 1$, f_i választása az $x_i = 0$ értékkadásnak;
- ▶ ezzel az $m + j$. jegyek mindegyike 0, 1, 2 vagy 3 lesz $j = 1, \dots, n$ -re, annak függvényében, hogy c_j -ben 0, 1, 2 vagy 3 literál vált igazzá;
- ▶ ha az $m + j$. jegy 3, akkor jó, ha 2, akkor válasszuk ki még mondjuk a_j -t, ha 1, akkor a_j -t és b_j -t is, ha 0, akkor nem tudjuk ezen a jegyen elérni a célszámot

NP-TELJES PROBLÉMÁK SZÁMOKRA

HÁTIZSÁK probléma:

- Adott: N elem mindeneknek w_i súlya és c_i értéke, valamint W és C
- Kérdés: Kiválasztható-e ismétlés nélkül néhány elem úgy, hogy összértékük $\geq C$ és összsúlyuk $\leq W$?

Állítás: A HÁTIZSÁK probléma NP-teljes. Az NP-beliség világos, és nehéz, mert a RÉSZLETÖSSZEG általánosítása

- **Ötlet:** A RÉSZLETÖSSZEG problémában az a_i értékekből rendre készítsünk egy tárgyat $w_i = c_i = a_i$ súlyjal és értékkel, továbbá legyen $C = W = K$, a célszám

Algoritmus a HÁTIZSÁK megoldására (az optimalizálási verzióra):

A következő rekurzív összefüggéssel számítható $T[i, w]$, ami a legfeljebb az első i tárgy felhasználásával egy w kapacitású hátizsákkal elérhető maximális haszon:

$$T[i, w] = \begin{cases} 0 & \text{ha } i = 0 \\ T[i - 1, w] & i > 0, w < w_i \\ \max\{T[i - 1, w], c_i + T[i - 1, w - w_i]\} & i > 0, w \geq w_i \end{cases}$$

Ez ad egy $\mathcal{O}(N \cdot W)$ időigényű algoritmust, ami **nem** polinom, mert az input mérete $n = N(\log w_i + \log c_i) + \log W$.

dinamikus programozás

HÁTIZSÁK PROBLÉMA

Felfoghatjuk úgy is, hogy az algoritmusunk akkor polinomidejű, ha az inputban a súlyokat unárisan reprezentáljuk, vagy ha a súlyok a tárgyak számának egy polinomjával korlátozhatóak. Ez vezet el az algoritmikus bonyolultságelméletben a **pszeudopolinomialitás** fogalmához:

Legyen A egy probléma.

Egy A -t eldöntő algoritmus **pszeudopolinomiális**, ha tetszőleges (a_1, \dots, a_m) inputon a futásideje $\mathcal{O}\left(\left(\sum_{1 \leq i \leq m} a_i\right)^k\right)$, valamely konstans k -ra.

Emlékezzünk vissza: az input **mérete** $\sum_{1 \leq i \leq m} \log a_i$ volt!
tehát nem az input **méretéhez** képest, hanem az input **értékéhez** képest kell polinom legyen.

Ha egy **NP**-teljes probléma eldönthető pszeudopolinomiális algoritmussal, úgy **gyengén NP-teljesnek**, ha pedig unáris változata is **NP**-teljes, akkor **erősen NP-teljesnek** nevezzük.

Unáris változat: amikor a számok unárisan jönnek be az inputra, pl a 4-et 1111 ábrázolja

Ha egy erősen **NP**-teljes problémára van pszeudopolinomiális algoritmus, akkor **P = NP**.

hiszen unáris számábrázolásnál a polinomiális algoritmus ugyanaz, mint a pszeudopolinomiális: méret=érték

BONYOLULTSÁGELMÉLET – 10. ELŐADÁS

ELŐADÓ: GAZDAG ZSOLT

KÉSZÜLT ...

- IVÁN SZABOLCS KORÁBBI ELŐADÁSFÓLIÁI, VALAMINT
- M. SIPSER: INTRODUCTION TO THE THEORY OF COMPUTATION ÉS
- C. PAPADIMITRIOU: SZÁMÍTÁSI BONYOLULTSÁG C. KÖNYVEI ALAPJÁN

PSZEUDOPOLINOMIÁLIS ALGORITMUSOK, ERŐS NP-TELJESSÉG

- ▶ A HÁTIZSÁK probléma gyengén **NP-teljes**.
a dinamikus programozós algoritmus pszeudopolinomiális
- ▶ A TSP(E) probléma viszont erősen **NP-teljes**.
hiszen a HAMILTON-ÚT \leq_P TSP(E) visszavezetésnél minden generált szám 1 vagy 2, ezeket unárisan is könnyű kigenerálni ugyanannyi idő alatt
- ▶ Pozitív egészek faktorizálására van pszeudopolinomiális algoritmus, de nem ismert rá polinomidejű.
Pl. a progalapról is ismert „oszd végig a gyökéig mindenkel” pszeudopolinomiális

A gyengén **NP-teljes** problémák minden példányai gyakorlatilag megoldhatónak tekinthetők, melyekben az inputon érkező számok **értéke** korlátozható az input **méretének** egy **polinomfüggvényével**.

(Pl. a HÁTIZSÁK sok „kisméretű” tárgy esetén.)

RELAXÁCIÓ

A HÁTIZSÁK probléma egészértékű programozási feladatként felírva:

$$\sum_{i=1}^n w_i x_i \leq W$$

$$\sum_{i=1}^n c_i x_i \geq C$$

$$0 \leq x_i \leq 1$$

Egy ILP feladat **LP-relaxációját** kapjuk, ha az egészek helyett a valósak körében oldjuk meg. (Azaz elhagyjuk az „ x_i egész” feltételt.)

Mivel $LP \in P$, a relaxált feladat hatékonyan megoldható.

RELAXÁCIÓ

Néhány optimalizálási problémánál a relaxált feladat optimumából egy „elég jó” megoldást lehet előállítani (de korántsem minden, és persze kérdés, hogy kinek mi az elég jó).

TÖREDÉKES HÁTIZSÁK

- ▶ Mint a HÁTIZSÁK, de a tárgyak törhetők: az i -edik tárgy x_i -ed része, $0 \leq x_i \leq 1$ egy $c_i x_i$ értékű, $w_i x_i$ súlyú tárgy.
ha a tárgyat félbetöröm, feleződik az értéke. Ez pl. homokra igaz, Mona Lisára kevésbé.
- ▶ Erre a feladatba a hatékony **mohó** algoritmus optimális.
először a legjobb ár/súly arányút, aztán a következőt stb, amíg fér, utolsót eltörjük.
- ▶ Kérdés, hogy mennyire jól „közelíti” az eredeti (függvény)problémát?
Erre még visszatérünk, de előbb kell pár fogalom.

APPROXIMÁCIÓ

A következőkben olyan optimalizálási problémákra próbálunk adni hatékony közelítő algoritmusokat, melyeknek eldöntési változata **NP**-nehéz.

Az f függvény minimalizálási/maximalizálási probléma, ha $f(I) = \arg \min_{s \in S(I)} c(s)$ vagy $\arg \max_{s \in S(I)} c(s)$ alakú, ahol $S(I)$ az I inphoz tartozó lehetséges megoldások (nemüres) halmaza, c pedig minden megoldáshoz egy valós költséget/értéket rendelő függvény.

Ha f egy optimalizálási probléma, $\alpha > 0$ egy konstans, A pedig egy olyan polinomidejű algoritmus, melyre $A(I) \in S(I)$ minden I inputra úgy, hogy

$$\forall I : \max \left\{ \frac{c(A(I))}{c(f(I))}, \frac{c(f(I))}{c(A(I))} \right\} \leq \alpha,$$

akkor A egy α -approximáló algoritmus f -re.

azaz: „ha az algoritmus által szállított megoldás értéke legfeljebb α -szor rosszabb az optimumnál”

CSÚCSLEFEDÉS

Emlékeztető

Input: $G = (V, E)$ gráf, $K > 0$ egész.

Output: van-e olyan, legfeljebb K méretű X csúcshalmaz G -ben, melyre igaz, hogy G minden élének legalább az egyik végpontja X -beli?

Optimalizálási változat

Input: $G = (V, E)$ gráf.

Output: egy **legkisebb** lefogó csúcshalmaz G -ben (egy megoldás: egy lefogó csúcshalmaz; költsége: a mérete)

Természetesen ha az optimalizálási változatra lenne egy hatékony módszer, akkor az eldöntésire is. Mivel az eldöntési változat **NP**-teljes, így az optimalizálási változatra sem ismert hatékony algoritmus.

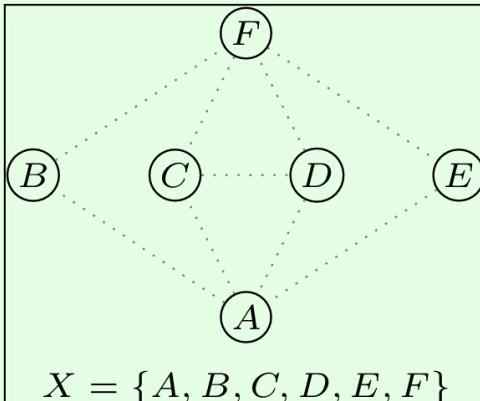
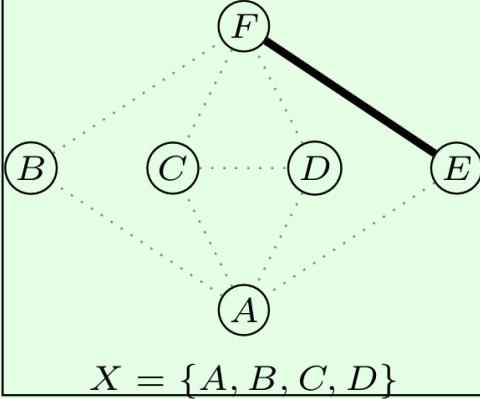
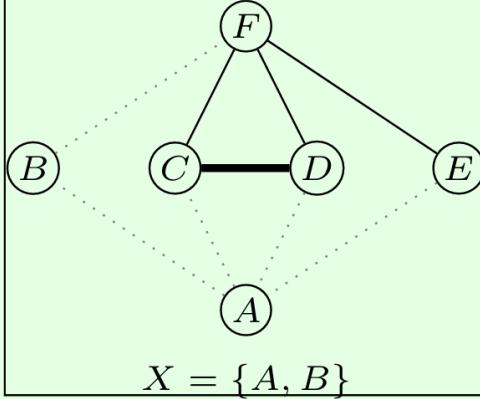
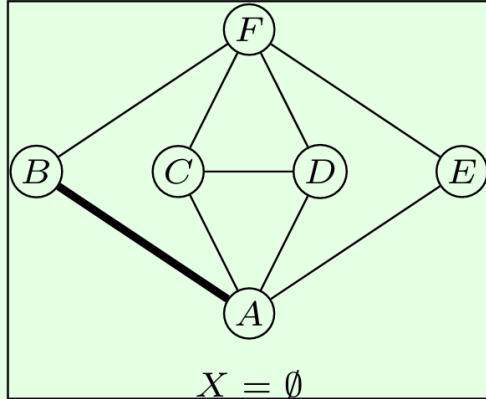
CSÚCSLEFEDÉS

Tekintsük a Csúcslefedés-re a következő egyszerű algoritmust:

- ▶ Legyen $X = \emptyset$.
- ▶ Amíg van él G -ben:
 - ▷ Válasszunk egy **tetszőleges** e élt.
 - ▷ Vegyük be X -be e **mindkét** végpontját.
 - ▷ Vegyük el G -ből az összes, e bármelyik végpontjára illeszkedő élt.
- ▶ Adjuk vissza X -et.

CSÚCSLEFEDÉS

Példa



Így választva az éleket („lexikografikusan”: a legkisebb olyan csúcsot választva, akire még illeszkedik él, azok közül azt, melynek a legkisebb a végpontja) egy 6 méretű csúcshalmazzal fedi le az éleket.
Ha az (A, C) , majd az (F, D) élt választaná ki, akkor egy 4 méretűt kapnánk.
Egy optimum pl. az $\{A, C, F\}$ halmaz.

CSÚCSLEFEDÉS

Az előző fólián szereplő algoritmus egy 2-approximáló algoritmus a Csúcslefedésre.

Ötlet

- ▶ Az algoritmus nyilván egy lefogó csúcshalmazt ad vissza.
- ▶ Ha az X halmazba rendre az e_1, e_2, \dots, e_k élek végpontjai kerültek bele, akkor e_1, \dots, e_k egy párosítás G -ben, mindegyiküknek legalább az egyik végpontját le kell fogni, vagyis a minimum legalább k .
- ▶ Az algoritmus által visszaadott eredmény pedig $2k$.

A Csúcslefedésre a mai napig **nem ismert ennél jobb** approximációs algoritmus.

TSP

A TSP probléma nem közelíthető: **nincs** α -approximáló algoritmus, csak ha $\mathbf{P} = \mathbf{NP}$.

Ötlet

Tegyük fel, hogy van egy α -approximáló A algoritmus TSP-re.

Akkor A -t felhasználva meg tudjuk oldani a Hamilton-kört a következőképpen: a $G = (V, E)$ gráfból készítsük $V \times V$ -n az alábbi $D(G)$ távolság mátrixot:

$$d_{u,v} = \begin{cases} 1 & \text{ha } (u, v) \in E; \\ |V| \cdot \alpha + 1 & \text{egyébként.} \end{cases}$$

Ekkor ha van Hamilton-kör, az optimum $|V|$;

ha nincs, az optimum legalább $|V| \cdot \alpha + 1$.

Ekkor a következő algoritmus:

Ha $A(D(G)) < |V| \cdot \alpha + 1$, fogadjuk el G -t, egyébként utasítsuk el

eldönti a Hamilton-kör problémát, polinomidőben.

Így várhatóan (hacsak nem $\mathbf{P} = \mathbf{NP}$) nincs approximáló algoritmus sem TSP-re. Kereshetünk viszont olyan speciális esetet, amire van.

METRIKUS TSP

A probléma

Input: egy D távolság-mátrix, ami teljesíti a háromszögegyenlőtlenséget:

$$\forall i, j, k : D_{i,j} + D_{j,k} \geq D_{i,k}.$$

Output: a minimális összsúlyú körút súlya.

Bonyolultság

Az eldöntési probléma továbbra is **NP**-nehéz.

(Mert ilyen mátrixot állítottunk elő a Hamilton-kör TSP(E)-re való visszavezetésekor: minden súly vagy 1 volt, vagy 2, ez teljesíti a háromszög-egyenlőtlenséget.)

METRIKUS TSP: 2-APPROXIMÁLÓ ALGORITMUS

Algoritmus

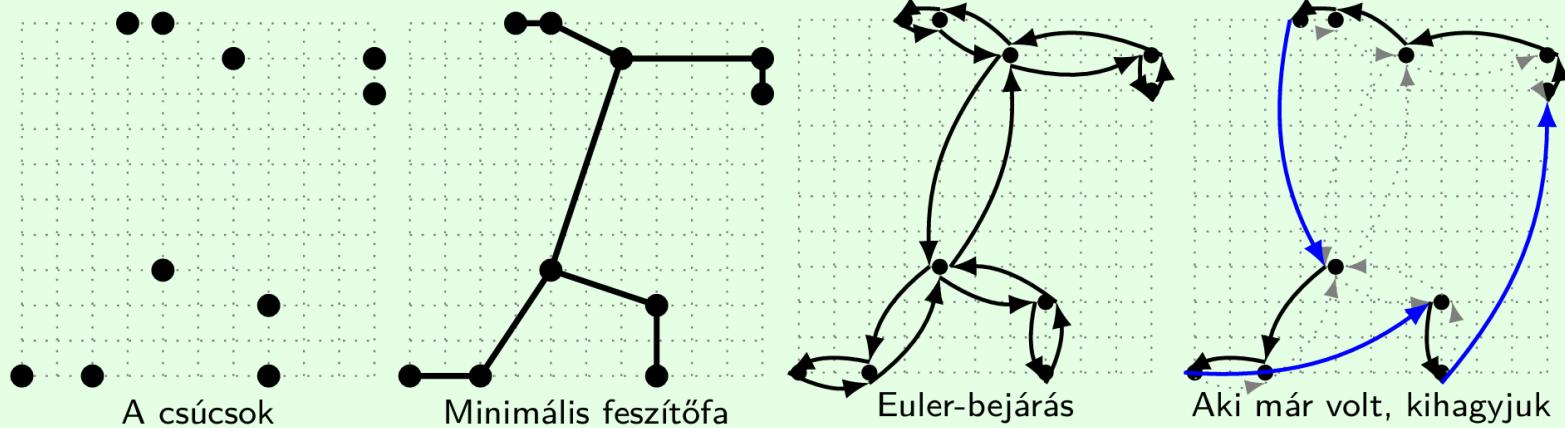
- ▶ Állítsuk elő (Prim vagy Kruskal algoritmusával, például) D egy **minimális feszítőfáját**.
- ▶ Kettőzzük meg a fa éleit.
- ▶ A kapott multigráfnak vegyük egy Euler-körvonalaát.
- ▶ A körvonalból hagyjuk el a korábban már meglátogatott csúcsokat.
- ▶ Adjuk vissza az így kapott kör összsúlyát.

Közelítés

A fenti egy 2-approximáló algoritmus a METRIKUS TSP-re.

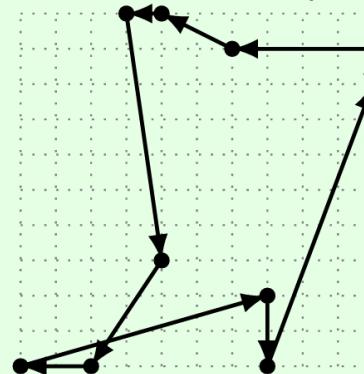
METRIKUS TSP: 2-APPROXIMÁLÓ ALGORITMUS

Példa



Itt a (10, 8) pontból kezdtük el a bejárást, és ennek megfelelően hagytuk el a pontokat.

Az élek hajlítása csak azért van, hogy látsszon a harmadik képen a bejárás; egyenes élekkel a végeredmény:



(nem optimális, pl. azért sem, mert vannak benne egymást keresztező élek, ami javítható lokálisan is.)

MAX-2SAT

A probléma

Input: egy $2CNF$, klózonként pontosan két literállal; a literálok változói különböznek.

Output: egyszerre legfeljebb hány klóz elégíthető ki benne?

Tudjuk, hogy a 2SAT probléma P-beli.

Bonyolultság

Ennek az optimalizálási problémának az eldöntési változata (input egy F $2CNF$ és egy K szám, kielégíthető-e F -ben egyszerre K klóz?) NP-teljes.

Először adunk egy randomizált $\frac{4}{3}$ -közelítő algoritmust, aztán ezt derandomizálva egy determinisztikusat.

RANDOMIZÁLT ALGORITMUS

Random: várható értékben approximáljon

Egy A randomizált algoritmus α -approximáló, ha **várható értéke** legfeljebb α -szor rosszabb, mint az optimális, vagyis:

$$\forall x \max \left\{ \frac{E(A(x))}{f(x)}, \frac{f(x)}{E(A(x))} \right\} \leq \alpha.$$

Max-2SAT

Ebben az esetben tehát egy olyan algoritmust kell adnunk, mely legalább $\frac{3}{4}X$ klózt kielégít az input formulában, ha legfeljebb X elégíthető ki egyszerre.

Ennél többet teszünk: olyan algoritmust fogunk adni, mely (várható értékben) a klózok $\frac{3}{4}$ -ét (tehát nem csak az egyszerre kielégíthető klózokét!) igazzá teszi.

RANDOMIZÁLT ALGORITMUS

Randomizált algoritmus

Állítsunk minden változót egymástól függetlenül $\frac{1}{2} - \frac{1}{2}$ valószínűsséggel igazra vagy hamisra.

Várható érték

Mivel egy klózban két különböző változó van, így annak esélye, hogy a klóz igaz lesz, $\frac{3}{4}$.

A várható érték additivitása miatt így várható értékben a klózok $\frac{3}{4}$ -ét kielégítjük.

DERANDOMIZÁLÁS

Most az előző dián szereplő véletlen algoritmusban csökkentjük a generálandó véletlen bitek számát: **derandomizálunk**.

Determinisztikus algoritmus

Legyenek x_1, x_2, \dots, x_n az input formulában szereplő változók.

- ▶ Először értéket adunk x_1 -nek, majd x_2 -nek, \dots , végül x_n -nek, az i -edik menetben x_i -nek.
- ▶ Az i -edik menetben x_1, \dots, x_{i-1} értéke már rögzítve van.
- ▶ Számítsuk ki $b = 0, 1$ -re, hogy mennyi klóz elégülne ki várható értékben, ha x_i értékét b -re választjuk, x_{i+1}, \dots, x_n értékét pedig $\frac{1}{2} - \frac{1}{2}$ valószínűsséggel, függetlenül választjuk 0-nak ill. 1-nek.
- ▶ Amelyik érték nagyobb lett, arra állítjuk ténylegesen x_i értékét, és folytatjuk a ciklust.

DERANDOMIZÁLÁS

Vegyük észre, hogy a várható értéket determinisztikusan ki tudjuk számítani.

Annak esélye, hogy egy klóz értéke igaz, a következőképp számítható:

- ▶ ha a klózban van olyan literál, melynek értékét 1-re rögzítettük, akkor 1;
- ▶ különben $1 - \frac{1}{2^k}$, ahol $0 \leq k \leq 2$ a klózban levő, még nem rögzített értékű változók száma.

A formulában igazra állított klózok értéke ezek összege.

Azt is be lehet könnyen látni, hogy minden menetben a várható érték nem csökkenhet, így az utolsó menet végére a „várható érték” továbbra is legalább a klózok $\frac{3}{4}$ -e lesz.

MAX-2SAT

Példa

$$\begin{aligned} F = & (x_1 \vee x_2) \wedge (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2) \wedge \\ & (\neg x_1 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge (x_2 \vee \neg x_3) \wedge (x_2 \vee x_4) \wedge \\ & (\neg x_2 \vee x_3) \wedge (\neg x_2 \vee \neg x_3) \wedge (x_3 \vee x_4) \wedge (\neg x_3 \vee \neg x_4) \end{aligned}$$

Ha $x_1 = 0$, a várható értékek: $\frac{1}{2}, \frac{1}{2}, 1, 1, 1, 1$, a többi $\frac{3}{4}$, összesen 9.5;
ha $x_1 = 1$, a várható értékek $1, 1, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}$, a többi $\frac{3}{4}$, összesen 8.5;
így legyen $x_1 = 0$.

Ezek után ha $x_1 = 0$ és $x_2 = 0$, a várható értékek $0, 1, 1, 1, 1, 1, \frac{1}{2}, \frac{1}{2}, 1, 1, \frac{3}{4}$
és $\frac{3}{4}$, összesen 9.5;

ha pedig $x_1 = 0$ és $x_2 = 1$, a várható értékek $1, 0, 1, 1, 1, 1, 1, \frac{1}{2}, \frac{1}{2}, \frac{3}{4}$ és $\frac{3}{4}$,
szintén 9.5. Legyen mondjuk $x_2 = 1$.

... és így tovább...

TÖREDÉKES HÁTIZSÁK

Algoritmus a TÖREDÉKES HÁTIZSÁKra

Erre a változatra a **mohó** algoritmus optimális:

- ▶ rendezük csökkenőbe a tárgyakat $\frac{c_i}{w_i}$ szerint;
- ▶ eszerint a sorrend szerint vegyük be annyi tárgyat teljesen, amennyit csak tudunk;
- ▶ az első tárgyat, ami nem fér be, pedig „törjük” úgy, hogy a töredék megtöltsé a hátizsák maradék kapacitását.

Approximálás?

Adódik a következő ötlet a HÁTIZSÁK problémára:

- ▶ rendezük csökkenőbe a tárgyakat $\frac{c_i}{w_i}$ szerint;
- ▶ eszerint a sorrend szerint vegyük be annyi tárgyat, amennyit csak tudunk.

Ez az algoritmus **nem** α -approximálja a HÁTIZSÁK problémát, semmilyen α konstansra.

HÁTIZSÁK

Példa, amikor nem jól közelít, ha truncoljuk a mohót

Ha pl. két tárgyunk van, $w_1 = 1$, $c_1 = 1$, $w_2 = W$ és $c_2 = W - 1$:

- ▶ az első tárgy fajlagosan értékesebb, mint a második
- ▶ a töredékes változat optimumhelye $(1, \frac{W-1}{W})$, értéke $1 + \frac{(W-1)^2}{W}$
- ▶ a mohó algoritmus lefele kerekítő változata tehát az $(1, 0)$ helyen felvett 1 értéket adja vissza
- ▶ az optimumhely $(0, 1)$, értéke $W - 1$.

Ha tehát W elég nagy, $1 \cdot \alpha < W - 1$ lesz, így az algoritmus ezen változata nem α -approximáló.

HÁTIZSÁK 2-APPROXIMÁLHATÓ

Kis módosítás

A következő algoritmus viszont 2-approximálja a HÁTIZSÁK problémát:

- ▶ Tegyük a tárgyakat fajlagos érték szerint csökkenő sorrendbe:
$$\frac{c_1}{w_1} \geq \frac{c_2}{w_2} \geq \dots \geq \frac{c_n}{w_n}.$$
- ▶ Számítsuk ki a TÖREDÉKES HÁTIZSÁK optimumhelyét:
 $(1, 1, \dots, 1, x, 0, 0, \dots, 0)$, ahol x a k . tárgy (amelyiket el kellett törnünk, mert már nem fért be).
- ▶ Adjuk vissza vagy $\sum_{i=1}^{k-1} c_i \cdot t$, vagy $c_k \cdot t$, amelyik nagyobb.

Tehát: vagy fajlagosan csökkenő sorrendben beteszünk, amit csak lehet, vagy az így éppen kimaradó tárgyat egyedül rakjuk be, amelyik jobb.

HÁTIZSÁK 2-APPROXIMÁLHATÓ

Amiért ez legalább az optimum felét adja:

- ▶ a töredékes hátizsák optima ennek a két értéknek az összegénél kisebb, így a kettő közül a nagyobb legalább a töredékes optimumának fele;
- ▶ a töredékes hátizsák optima (mivel relaxált változat) legalább akkora, mint a 0/1 hátizsáké.

Pár kimaradt részlet

Az algoritmus így akkor nem működik, ha minden tárgy befér egyszerre (akkor az lesz az optimumhely), vagy ha a k . tárgy egyedül se férne be (de az ilyen tárgyakat eleve nem kell felvegyük a listába), de ezek könnyen javíthatók.

BONYOLULTSÁGELMÉLET – 11. ELŐADÁS

ELŐADÓ: GAZDAG ZSOLT

KÉSZÜLT ...

- IVÁN SZABOLCS KORÁBBI ELŐADÁSFÓLIÁI, VALAMINT
- M. SIPSER: INTRODUCTION TO THE THEORY OF COMPUTATION ÉS
- C. PAPADIMITRIOU: SZÁMÍTÁSI BONYOLULTSÁG C. KÖNYVEI ALAPJÁN

HÁTIZSÁK $1 + \epsilon$ -APPROXIMÁXIÓ

Nézzünk egy másik kézenfekvő algoritmust a HÁTIZSÁK probléma közelítő megoldására:

Skálázás ϵ -ra

- ▶ Legyen $w_1, \dots, w_n, c_1, \dots, c_n, W$ a HÁTIZSÁK probléma egy inputja és $\epsilon > 0$ egy valós szám.
- ▶ Oldjuk meg a $w_1, \dots, w_n, c'_1, \dots, c'_n, W$ feladatot dinamikus programozással, ahol $c'_i = \lfloor \frac{c_i}{c_{\max}} \cdot \lfloor \frac{n}{\epsilon} \rfloor \rfloor$, $c_{\max} = \max_i c_i$ és adjuk vissza ennek az eredményét.

Tehát mintha a maximális hasznú tárgy haszna $\lfloor \frac{n}{\epsilon} \rfloor$ lenne, a többi haszna pedig evvel arányosan skálázódna (egészrész véve).

Állítás: rögzített ϵ esetén a fenti feladat megoldása dinamikus programozással egy polinom időigényű $(1 + \epsilon)$ -approximáló algoritmust eredményez

PTAS, FPTAS

A gyakorlatban is azt szeretjük, ha a megrendelő tud mondani egy hibakorlátot, amivel már elégedett (pl. „ha gyorsan ki tudtok számolni egy olyan megoldást, ami csak garantáltan 1%-kal kerül többe, mint az optimum, az már nekem jó”, itt $\epsilon = 0.01$, van akinek kisebb ϵ kell, van akinek nagyobb is jó). Ennek van neve is:

PTAS

Azt mondjuk, hogy az f optimalizálási problémára van **polinomidejű approximációs séma**, avagy PTAS, ha van olyan algoritmus, melynek f inputja és egy $\epsilon > 0$ szám a bemenete és tetszőleges rögzített ϵ -ra $(1 + \epsilon)$ -approximáló polinomidejű algoritmus.

(ϵ esetleg megjelenhet a kitevőben, ami nem annyira szerencsés 1-hez közelí ϵ esetén)

FPTAS

Azt mondjuk, hogy az f optimalizálási problémára van **teljesen polinomidejű approximációs séma**, avagy FPTAS, ha van rá olyan PTAS, mely tetszőleges (x, ϵ) inputra $\text{poly}(|x|, \frac{1}{\epsilon})$ időben fut.

Példa: A HÁTIZSÁK-ra az előző fólián vázolt algoritmus egy FPTAS

Ha egy problémára van FPTAS, azzal tetszőleges inputra tetszőleges pontossággal polinomidőben meg tudjuk határozni az optimumot.

NÉHÁNY TOVÁBBI EREDMÉNY A P ÉS NP OSZTÁLYOKRÓL

Eddig minden **NP**-beli problémánkról vagy azt mutattuk meg, hogy **P**-ben van, vagy azt, hogy **NP**-teljes.

Azonban...

Tétel (Ladner, 1975)

Ha $P \neq NP$, akkor létezik olyan $L \in NP - P$, mely nem **NP**-teljes.

Tehát akkor feltehetően van, ami NP-n „belül”, de P-n „kívül” van

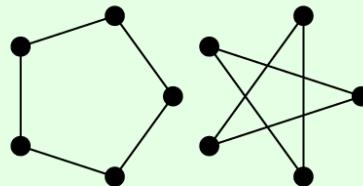
- az ilyen problémákat hívják „**NP-köztes**” (NP-intermediate, NPI) problémának

GRÁF-IZOMORFIZMUS

GRÁF-IZOMORFIZMUS

Input: két gráf.

Output: izomorfak-e?



ez a két gráf pl. egy „igen” példány

Hogy NP-beli, az világos: csak nemdeterminisztikusan meg kell feleltetni egymásnak a csúcsokat és ellenőrizni, hogy ugyanott mennek-e az élek, az „igen” válaszra egy „élartó bijekció” a tanú

FAKTORIZÁLÁS(E)

Input: $N, K > 0$ egészek.

Output: Van-e N -nek K -nál kisebb prímosztója?

A fenti két **NP**-beli probléma egyikéről sem ismert, hogy **P**-beliek-e, sem az, hogy **NP**-teljesek lennének.

mindkettőről az az „elfogadottabb” sejtés, hogy NPI-ben vannak

A CONP OSZTÁLY

$$\text{coNP} = \{A : \overline{A} \in \text{NP}\}$$

Emlékeztető: coC-ben a C-beli problémák komplementerei vannak

Példák

ÉRVÉNYESSÉG

Adott: φ Boole-formula

Kérdés: Érvényes-e, azaz azonosan igaz-e φ ?

HAMILTON-ÚT

Adott: G gráf

Kérdés: Igaz-e, hogy G nem tartalmaz Hamilton-utat?

Az ÉRVÉNYESSÉG és HAMILTON-ÚT problémák coNP-ben vannak.

Ezknél nem az „igen” példányokhoz van „tanú”, hanem a „nem” példányokhoz „cáfolat”

A CONP OSZTÁLY

Egy probléma akkor van

- ▶ **NP**-ben, ha minden igen példányához létezik polinom méretű, polinom időben ellenőrizhető bizonyíték, és csak az igen példányokhoz létezik ilyen bizonyíték.
- ▶ **coNP**-ben, ha minden nem példányhoz létezik polinom méretű, polinom időben ellenőrizhető cáfolat, és csak a nem példányokhoz létezik ilyen cáfolat.

NP \cap **coNP**-ben pedig ezek szerint akkor, ha az „igen” példányokhoz is létezik tanú **és** a „nem” példányokhoz is létezik cáfolat, melyeket ha valaki megad nekünk, gyorsan tudjuk ellenőrizni, hogy tényleg tanú/cáfolat-e

ebből még nem következik, hogy determinisztikusan legyártani is könnyű lenne bármelyiket!

A CONP OSZTÁLY

Állítás: $P \subseteq NP \cap coNP$

- Nyilvánvaló, hogy $P = coP$ (mert ha valamit el tudok dönten polinom időben, akkor a probléma komplementere is eldönthető így az algoritmus *igen/nem* válaszainak felcserélésével)
- Másrészt $coP \subseteq coNP$, hiszen ha lenne egy olyan A probléma, ami benne van $coP - coNP$ -ben, akkor \bar{A} benne lenne P -ben, de nem lenne benne NP -ben (mert ha benne lenne, akkor $\bar{\bar{A}} = A \in coNP$ lenne), ami ellentmondás

Állítás: Legyen \mathbb{C} egy problémaosztály. Egy A probléma akkor és csak akkor \mathbb{C} -teljes (a polinomidejű visszavezetésre nézve), ha \bar{A} $co\mathbb{C}$ -teljes

- Egyszer megint felhasználhatjuk azt, hogy, hogy ha $L_1 \leq_p L_2$, akkor $\overline{L_1} \leq_p \overline{L_2}$, másrészt azt, hogy $co(co\mathbb{C}) = \mathbb{C}$

Következmény:

$\overline{\text{HAMILTON-ÚT}}$ és $\overline{\text{ÉRVÉNYESSÉG}}$ $coNP$ -teljesek (triviálisan
 $\overline{\text{SAT}} \leq_p \overline{\text{ÉRVÉNYESSÉG}}$)

NP ÉS CONP

Állítás: Ha \mathbb{C} zárt a (polinomidejű) visszavezetésre és $A \in \mathbb{C}$ -teljes, akkor

$$\mathbb{C} = \text{co}\mathbb{C} \Leftrightarrow A \in \text{co}\mathbb{C}$$

\Rightarrow : Ha $\mathbb{C} = \text{co}\mathbb{C}$ és $A \in \mathbb{C}$ -teljes, tehát $A \in \mathbb{C}$, akkor nyilván $A \in \text{co}\mathbb{C}$ is

\Leftarrow :

- Ha $A \in \text{co}\mathbb{C}$, akkor $\bar{A} \in \mathbb{C}$
- Ha $A \in \mathbb{C}$ -teljes, akkor $\bar{A} \in \text{co}\mathbb{C}$ -teljes. Tehát tetszőleges $B \in \text{co}\mathbb{C}$ -re $B \leq_p \bar{A}$
- Mivel \mathbb{C} zárt a (polinomidejű) visszavezetésre és $\bar{A} \in \mathbb{C}$, emiatt tetszőleges $B \in \text{co}\mathbb{C}$ szintén \mathbb{C} -ben van
- Tehát $\text{co}\mathbb{C} \subseteq \mathbb{C}$. Könnyen belátható (lásd előző fólián P-re és NP-re), hogy a tartalmazás akkor is fennáll, ha alkalmazzuk minden osztályra a co operátort, azaz $\mathbb{C} \subseteq \text{co}\mathbb{C}$, így a két osztály egyenlő

Következmény: $\text{NP} = \text{coNP} \Leftrightarrow \text{SAT} \in \text{coNP} \Leftrightarrow \text{ÉRVÉNYESSÉG} \in \text{NP}$

Tehát pl. ha a kielégíthetetlenségre lenne NP algoritmus, akkor $\text{NP} = \text{coNP}$

(A rezolúcióról tudjuk, hogy nem NP, túl hosszú lehet egy levezetés)

PSPACE-TELJES PROBLÉMÁK

Emlékeztető: PSPACE a polinom tárás off-line Turing-gépekkel megoldható problémák osztálya

Tekintsük SAT alábbi kierjesztését, amit QBF-nek fogunk nevezni (QBF mint Quantified Boolean Formula):

- Adott egy φ prenex alakú zárt Boole formula; a kérdés: Igaz-e φ ?

Példa:

- $\exists X \forall Y \exists Z ((X \vee Y) \wedge (Y \vee Z) \wedge (\neg Y \vee \neg Z))$ igaz
- $\exists X \forall Y ((X \vee Y) \wedge (\neg X \vee \neg Y))$ nem igaz

Megjegyzés:

- Szokás a QBF-nek egy megszorított változatát tekinteni: φ -ben a kvantorok alternálnak, az első és utolsó kvantor pedig \exists
- Ilyen bemenetekre a QBF felfogható kétszemélyes játékként:
 - Az első játékos választja a páratlan változók értékét, és célja a formula igazzá tétele
 - a második választja a páros változókét, és célja a formula hamissá tétele
- Ilyenkor φ pontosan akkor igaz, ha az első játékosnak van nyerő stratégiája ebben a játékban

QBF PSPACE-TELJES

Állítás: QBF PSPACE-teljes (ez a megszorított változatra is igaz)

Csak azt mutatjuk meg, hogy $\text{QBF} \in \text{PSPACE}$

- Az alábbi A algoritmus QBF-ét dönti el:
 - Egy φ QBF bemenetre
 - Ha φ -ben nincs kvantor, akkor értékeljük ki: ha φ igaz, akkor a kimenetre: *igen*, egyébként a kimenetre: *nem*
 - Ha $\varphi = \exists X\psi$, akkor rekurzívan hívjuk meg A -t ψ -re $X = \text{igaz}$ és $X = \text{hamis}$ értékekkel is; Ha valamelyik esetben *igen* a kimenet, akkor a kimenetre: *igen*, egyébként a kimenetre: *nem*
 - Ha $\varphi = \forall X\psi$, akkor rekurzívan hívjuk meg A -t ψ -re $X = \text{igaz}$ és $X = \text{hamis}$ értékekkel is; Ha minden esetben *igen* a kimenet, akkor a kimenetre: *igen*, egyébként a kimenetre: *nem*
- A tárigénye:
 - A rekurzió mélysége: változók száma
 - Egy szinten tárolandó adat: egy változó igazságértékei
 - Az összes tárigény $O(n \log n)$ φ változóinak számában

FÖLDRAJZI JÁTÉK PSPACE-TELJES

A QBF kétszemélyes játék változatának segítségével további ilyen játékokról ¹³ lehet belátni, hogy PSPACE-teljesek

Legyen például **FÖLDRAJZI JÁTÉK** a következő:

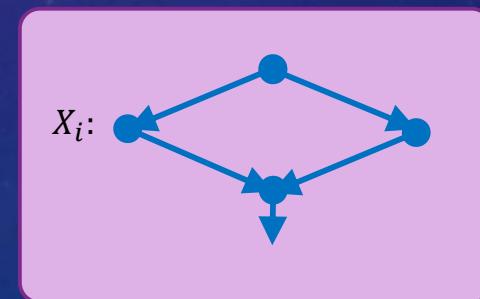
- Adott egy $G = (V, E)$ irányított gráf és $p \in V$
- Kérdés: Van-e nyerő stratégiája a kezdő játékosnak az alábbi játékban:
 - Két játékos felváltva jelöli meg p -ből kiindulva G csúcsait úgy, hogy a következő játékos mindenkor mindenkor csak a legutoljára megjelölt csúcsból elérhető még meg nem jelölt csúcson választhat
 - Az veszít aki nem tud újabb csúcson megjelölni

Állítás: FÖLDRAJZI JÁTÉK PSPACE-teljes

- FÖLDRAJZI JÁTÉK \in PSPACE hasonlóan látható be, mint QBF esetében
- A PSPACE-nehézség belátásához visszavezetjük rá a QBF problémát

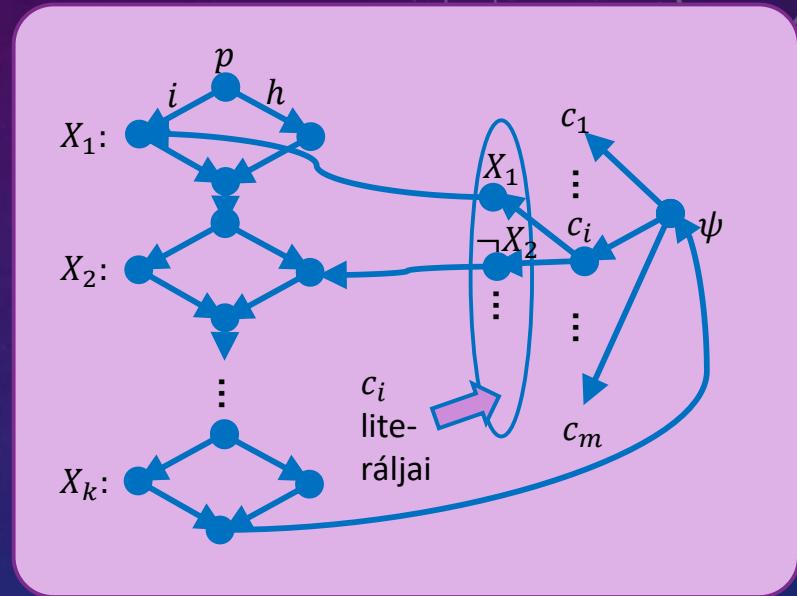
FÖLDRAJZI JÁTÉK PSPACE-NEHÉZ

- Legyen
 - $\varphi = \exists X_1 \forall X_2 \dots \forall X_{k-1} \exists X_k \psi$, ahol $\psi = c_1 \wedge \dots \wedge c_m$
- (QBF ilyen alakú bemenetekre megszorítva is PSPACE-teljes)
- Konstruáljuk meg G_φ -t a következőképpen
 - Először minden X_i -hez elkészítünk egy részgráfot:



FÖLDRAJZI JÁTÉK PSPACE-NEHÉZ

- Ezután legyen G_φ az ábrán látható gráf:
 - A konstrukció polinom időben elvégezhető és belátható, hogy φ igaz $\Leftrightarrow G_\varphi$ -ben van nyerő stratégiája a kezdő játékosnak
- Megjegyzés:** A játékosoknak csak követniük kell a másik játékban való nyerő stratégiájukat (ha van):
 - X_i igazzá tételenek az felel meg, hogy a gráf megfelelő rombuszában balra kell menni, a hamissá tételenek meg az, hogy jobbra
 - Mindig az egyes választja az utolsó változó értékét illetve ennek megfelelően az utolsó rombusz bal vagy jobb oldali csúcsát
 - Ezért az egyes választja a ψ -t, majd a kettes választ egy olyan csúcsot (klózt), amit reményei szerint sikeres hamissá tennie
 - Ha egyesnek van nyerő stratégiája, akkor minden tud választani egy olyan literált, amit a kettes által választott klózban is sikeres igazzá tennie
 - Ekkor kettes nem tud olyan csúcsot választani, ami még nem volt választva



FÖLDRAJZI JÁTÉK PSPACE-NEHÉZ

Konstruáljuk meg G_φ -t ha

$$\varphi = \exists X_1 \forall X_2 \exists X_3 ((X_1 \vee X_2) \wedge (X_2 \vee X_3) \wedge (\neg X_2 \vee \neg X_3))$$

AZ L ÉS NL OSZTÁLYOK

- Emlékeztető:
 - L jelöli determinisztikus Turing-géppel logaritmikus tárral megoldható problémák osztályát
 - NL, pedig a nemdeterminisztikus Turing-géppel logaritmikus tárral megoldható problémák osztályát
- Látni fogjuk, hogy $NL \subseteq P$, ezért a fenti osztályokban már minden visszavezethető mindenre polinom időben, ezért itt logtáras visszavezetést használunk:
- Egy K_1 nyelv logaritmikus tárral visszavezethető egy K_2 nyelvre (jele: $K_1 \leq_l K_2$), ha a $K_1 \leq K_2$ visszavezetés kiszámítható logaritmikus táras determinisztikus (offline) Turing-géppel
- Egy K nyelv NL-nehéz (a log. táras visszavezetésre nézve), ha minden $K' \in NL$ nyelvre, $K' \leq_l K$
 - Ha $K \in NL$ is teljesül, akkor K NL-teljes
 - Állítás: L zárt a logaritmikus tárral való visszavezetésre nézve
 - Következmény: Ha egy K nyelv NL-teljes és $K \in L$, akkor $L = NL$

AZ L ÉS NL OSZTÁLYOK

Állítás: ELÉRHETŐSÉG NL-teljes

- Azt már láttuk, hogy NL-beli
- Legyen K egy NL-beli nyelv; megmutatjuk, hogy $K \leq_l$ ELÉRHETŐSÉG
 - Legyen M egy K -t eldöntő $O(\log n)$ táras nemdeterminisztikus Turing-gép és legyen u az M egy n hosszú bemenete
 - Akkor M egy konfigurációja $c \cdot \log n$ méretű valamely alkalmas c konstansra
 - Legyen G az M konfigurációs gráfja az u -n
 - Legyen s és t rendre az M kezdő- és elfogadó konfigurációja az u -n
- Ekkor $u \in L(M) \Leftrightarrow G$ -ben van út s -ből t -be, tehát G megkonstruálása nem más, mint K visszavezetése ELÉRHETŐSÉG-re
- Ez a visszavezetés logaritmikus tárral kiszámítható
 - Lexikografikusan soroljuk fel az összes $c \cdot \log n$ hosszú szót
 - teszteljük, hogy ez legális konfigurációja-e M -nek, ha igen, akkor a kimenetre írjuk, mint a G egy csúcsát
 - Az élek (konfiguráció párok) hasonlóképpen felsorolhatók, tesztelhetők és a kimenetre írhatók

AZ L ÉS NL OSZTÁLYOK

Korábban kimondtuk, hogy ha egy NL-teljes problémáról kiderül, hogy L-beli, akkor $L = NL$

Következmények:

- Ha $ELÉRHETŐSÉG \in L$, akkor $L = NL$
- $ELÉRHETŐSÉG$ eldönthető $O(\log^2 n)$ tárral determinisztikusan
- $NL \subseteq P$



$ELÉRHETŐSÉG \in NL$ azt jelenti, hogy eldönthető $O(\log n)$ tárral nemdeterminisztikusan; alkalmazzuk Savitch tételét!

- Legyen $K \in NL$ és M egy K -t felismerő logaritmikus táras Turing-gép
- Legyen w az M egy n -hosszú bemenete
- M konfigurációs gráfja a w -n polinom méretű, polinom időben megkonstruálható, és
- ebben a gráfban kell keresni utat a kezdőkonfigurációból az elfogadóba (feltehetjük, hogy egy elfogadó konfiguráció van)
- Ez pedig polinom idő alatt elvégezhető (pl. a Dijkstra algoritmussal)

AZ L ÉS NL OSZTÁLYOK

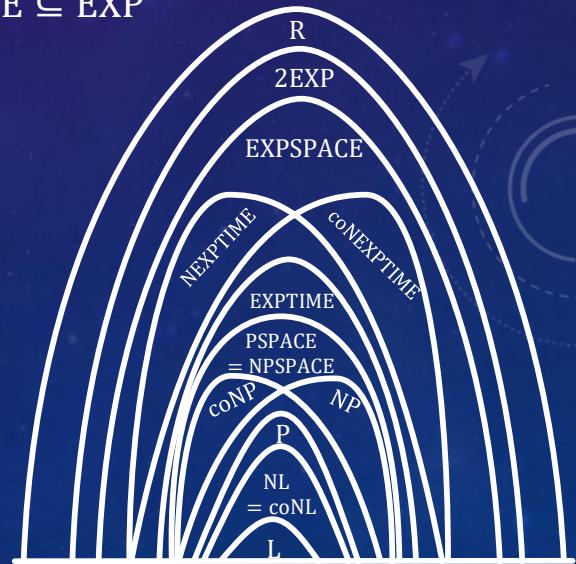
Állítás: $\text{NL} = \text{coNL}$ (az Immerman-Szelepcsényi téTEL)

- Ehhez elég azt megmutatni, hogy $\overline{\text{ELÉRHETŐSÉG}} \in \text{NL}$ (nem bizonyítjuk, messze nem triviális)

Következmény: tetszőleges $s(n) \geq \log n$ függvényre $\text{NSPACE}(s(n)) = \text{coNSPACE}(s(n))$

Összefoglalás: $L \subseteq \text{NL} = \text{coNL} \subseteq P \subseteq NP \subseteq \text{PSPACE} \subseteq \text{PSPACE} \subseteq \text{EXP}$

- Ismert, hogy $\text{NL} \subset \text{PSPACE}$ és $P \subset \text{EXP}$ (idő- és tárhierarchia tételek)
- Az a sejtés, hogy minden tartalmazás valódi



A tanult (és egyéb) osztályok feltehető viszonyai