

# Bonyolultságelmélet

---

Thursday 1<sup>st</sup> December, 2016, 22:35

## Egy algoritmus

- Input: irányított gráf, mondjuk  $G[1..N][1..N]$  szomszédsági mátrixával
- Output: ?

```
function F( $x, y, d$ )  
  if  $d == 0$  then return ( $x == y$ ) OR  $G[x][y]$ ;  
  for  $z = 1 \dots N$  do  
    if F( $x, z, d - 1$ ) AND F( $z, y, d - 1$ ) then return TRUE  
  return FALSE  
return F(1, N,  $\lceil \log N \rceil$ );
```

## A függvény

Az  $f(x, y, d)$  hívás pontosan akkor ad TRUE-t, ha a  $G$  gráfban van  $x$ -ből  $y$ -ba legfeljebb  $2^d$  hosszú út.

- $d == 0$ : legfeljebb egy hosszú út, vagyis  $x == y$  vagy van él  $x$ -ből  $y$ -ba;
- $d > 0$ : ha van  $x$ -ből  $y$ -ba egy legfeljebb  $2^d$  hosszú út, akkor ennek felezőpontjába,  $z$ -be van  $x$ -ből egy legfeljebb  $2^{d-1}$  hosszú út és  $z$ -ből  $y$ -ba is, ez akkor teljesül, ha  $f(x, z, d - 1)$  és  $f(z, y, d - 1)$  is TRUE-val tér vissza.

## A belépési pont

Az  $f(1, N, \lceil \log N \rceil)$  hívás tehát akkor ad TRUE-t, ha van legfeljebb  $2^{\lceil \log N \rceil} \geq N$  hosszú **út 1-ből N-be**, vagyis egy (determinisztikus) algoritmus az ELÉRHETŐSÉG problémára.

## Tárigény

- Az  $f$  függvény egy példányának tárigénye  $\mathcal{O}(\log n)$ , hisz csak az  $x, y, d, z$  változókat deklarálja, mindegyik  $0 \dots N$  közti értéket vesz fel;
- a hívási verem mélysége  $\lceil \log N \rceil = \mathcal{O}(\log n)$ , legfeljebb ennyi példánya van egyszerre  $f$ -nek a memóriában;

tehát a teljes tárigény  $\mathcal{O}(\log^2 n)$ .

Ezzel beláttuk Savitch tételét:

## Tétel

ELÉRHETŐSÉG  $\in$  SPACE( $\log^2 n$ ).

Következmény

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

ha  $f(n) \geq \log n$ .

Ötlet

Az ELÉRHETŐSÉG problémát a nemdeterminisztikus program **konfigurációs gráfján** oldjuk meg.

Következmény

$$\mathbf{PSPACE} = \mathbf{NPSPACE}.$$

Következmény

$$\mathbf{NL} \subsetneq \mathbf{PSPACE}.$$

(Hiszen  $\mathbf{NL} \subseteq \text{SPACE}(\log^2 n) \subsetneq \text{SPACE}(n)$  Savitch tétele és a tárhierarchia tétel szerint.)

# Az Immerman-Szelepcsényi tétel

Egy nemdeterminisztikus algoritmus

- Input: gráf,  $G[1 \dots N][1 \dots N]$  szomszédsági mátrixával.

**function**  $\hat{U}_T(s, t, d)$

$i := s;$

**while**  $d \geq 0$  **do**

**if**  $i == t$  **then return** AC-

CEPT

$j := \text{nd}(N);$

**if** NOT  $G[i][j]$  **then return**

REJECT

$i := j;$

$d := d - 1;$

- A függvény visszaadhat

ACCEPT-et, ha  $s$ -ből  $t$   $d$  lépésen belül elérhető.

- Egyébként mindenképp REJECT-et ad vissza.
- Tárigény:  $\mathcal{O}(\log n)$ .

# Az Immerman-Szelepcsényi tétel

Egy nondeterminisztikus algoritmust hívó algoritmus

- Input: gráf,  $G[1 \dots N][1 \dots N]$  szomszédsági mátrixával.

```
function T-IN-SK( $s, t, prev, k$ )  
   $check := 0$ ;  
  for  $u := 1 \dots N$  do  
    if  $\hat{U}_T(s, u, k - 1)$  then  
       $check := check +$   
1;  
      if  $G[u][t]$  then re-  
turn ACCEPT  
      if  $check == prev$  then  
return REJECT  
  THROW ERROR;
```

- **Ha**  $prev$  az  $s$ -ből legfeljebb  $k - 1$  lépésben elérhető csúcsok száma, akkor:
- ha  $s$ -ből  $t$  legfeljebb  $k$  lépésben elérhető, akkor az eredmény ACCEPT vagy ERROR, lehet ACCEPT;
- ha nem, akkor az eredmény REJECT vagy ERROR, lehet REJECT.

# Az Immerman-Szelepcsényi tétel

Egy  $nd$  algoritmust hívó algoritmust hívó algoritmus

- Input: gráf,  $G[1 \dots N][1 \dots N]$  szomszédsági mátrixával és  $s$  csúcsa.

```
prev := 1;  
for  $k := 1 \dots N$  do  
    current := 0;  
    for  $t := 1 \dots N$  do  
        if T-IN-SK( $s, t, prev, k$ )  
then  
            current := current +  
1;  
    prev := current;  
return prev;
```

- Az algoritmus visszaadja az  $s$ -ből elérhető csúcsok számát, vagy ERROR-t dob; van, mikor a számot adja vissza.
- Mindezt logaritmikus tárigényben.



# Az Immerman-Szelepcsényi tétel

Beláttuk az Immerman-Szelepcsényi tételt:

## Tétel

Van olyan **nemdeterminisztikus** algoritmus, mely **logaritmikus tárban** kiszámítja az input gráf megadott csúcsából elérhető csúcsok számát.

Nemdeterminisztikus függvénykiszámítás: minden szálon vagy a helyes eredményt adja vissza, vagy REJECT-et; továbbá van olyan szál, mikor a helyes eredményt.

# Az Immerman-Szelepcsényi tétel

Következmény

$$\text{NSPACE}(f(n)) = \text{coNSPACE}(f(n)),$$

ha  $f(n) \geq \log n$ .

Ötlet

A konfigurációs gráfban először  $nd$  kiszámítjuk az elérhető konfigurációk számát, aztán  $nd$  mindet megpróbáljuk elérni. Ha annyit értünk el, amennyi csak elérhető és egyik sem elfogadó, akkor **ACCEPT**, különben **REJECT**.

Következmény

$$\mathbf{NL} = \mathbf{coNL}.$$

# A logaritmikusan tárban való visszavezetés

A polinomidejű visszavezetésre nézve  $\mathbf{P}$  minden nemtriviális eleme „ $\mathbf{P}$ -teljes”.

Bevezetünk egy (formailag) „gyengébb” visszavezetést, melyet a  $\mathbf{P}$  osztályon belül alkalmazunk problémák egymáshoz viszonyított nehézségének definiálására.

Az  $f$  függvény az  $A$  eldöntési problémának a  $B$  eldöntési problémára való **logaritmikusan tárigényű** visszavezetése, ha

- $f$  kiszámítható logaritmikusan tárban és
- tetszőleges  $I$  inputra  $A(I) = B(f(I))$ .

Ha létezik  $A$ -nak  $B$ -re való logaritmikusan tárigényű visszavezetése, annak jele  $\mathbf{A} \leq_{\mathcal{L}} \mathbf{B}$ :  $A$  logtárban visszavezethető  $B$ -re.

# A logtáras visszavezetés

Logaritmikusan tárigényű (tehát mindenképp offline) algoritmusnak legfeljebb  $2^{\mathcal{O}(\log n)}$ , azaz polinom sok konfigurációja lehet adott input esetén; mivel a visszavezetés nem eshet végtelen ciklusba, így polinomidőben meg is kell álljon.

Vagyis ha  $A \leq_{\mathcal{L}} B$ , akkor  $A \leq_{\mathcal{P}} B$  is.

Megjegyzés

**Nem ismert**, hogy  $\leq_{\mathcal{L}}$  és  $\leq_{\mathcal{P}}$  egybeesnek-e... (ha igen, akkor  $\mathbf{L} = \mathbf{P}$ )

Legyen  $\mathcal{C}$  problémák egy osztálya. Az  $A$  probléma  **$\mathcal{C}$ -nehéz** a logtáras visszavezetésre nézve, ha minden  $\mathcal{C}$ -beli probléma logtárban visszavezethető  $A$ -ra.

Ha még  $A \in \mathcal{C}$  is, akkor  $A$  egy  $\mathcal{C}$ -teljes probléma a logtáras visszavezetésre nézve.

## Megjegyzés

**Nem ismert**, hogy a logtáras visszavezetésre nézve **NP**-teljes problémák ugyanazok-e, mint a polinomidejű visszavezetésre nézve **NP**-teljesek.

(A **SAT** egy **NP**-teljes probléma a logtáras visszavezetésre nézve is; az összes hatékony visszavezetés, melyet eddig láttunk, egyben logtáras is volt.)

## Tétel

A logtárban való visszavezethetőség tranzitív: ha  $f$  az  $A$ -nak  $B$ -re,  $g$  pedig a  $B$ -nek  $C$ -re való logtáras visszavezetése, akkor az  $f \circ g$  összetett függvény az  $A$ -nak  $C$ -re való logtáras visszavezetése.

## A gond

Legyen  $M_f$  az  $f$ -et,  $M_g$  pedig a  $g$ -t kiszámító program.

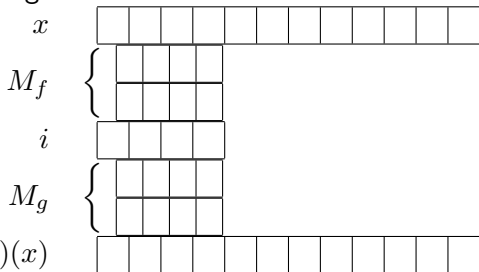
Az nyilván igaz, hogy  $I \in A \Leftrightarrow f(I) \in B \Leftrightarrow g(f(I)) \in C$ .

Csak hogy ha  $M_f(I)$ -t munkaregiszterekbe írjuk, sokkal hosszabb lehet, mint  $\log(|I|)$ !

# A visszavezetés tranzitivitása

## A trükk

Nem tároljuk el  $M_f$  teljes outputját, csak a legutolsóként írt output regiszter sorszámát és a belekerült értéket.



**$M_g$ -nek szüksége van egy nagyobb sorszámú regiszter tartalmára:** folytatjuk  $M_f$  szimulálását.

**$M_g$ -nek szüksége van egy kisebb sorszámú regiszter tartalmára:** előlről kezdjük  $M_f$  szimulálását.

Korábban láttuk, hogy **P** (így **NL**) bármely két nemtriviális problémája hatékonyan visszavezethető egymásra, így a polinomidejű visszavezetésre nézve ezen osztályokon belül a „teljesség” fogalma értelmetlenné válik.

Azt mondjuk, hogy egy probléma **NL/P**-teljes/nehéz, ha a **logtáras** visszavezetésre nézve az.

Tétel

A HÁLÓZAT-KIÉRTÉKELÉS probléma **P**-teljes.



# Az ELÉRHETŐSÉG probléma NL-teljessége

## Tétel

Az ELÉRHETŐSÉG probléma NL-teljes.

## Ötlet

Azt már láttuk, hogy  $\text{ELÉRHETŐSÉG} \in \text{NL}$ .

Az NL-nehézséghez mint korábban, az **elérhetőségi módszert** alkalmazzuk: egy nemdeterminisztikus,  $\mathcal{O}(\log n)$  tárkorlátos gép konfigurációs grájában megoldva az elérhetőségi problémát a kezdőkonfigurációtól az elfogadóig, készen vagyunk.

## Következmény

Az ELÉRHETETLENSÉG probléma NL-teljes.

Hiszen az Immerman-Szelepcsényi tétel szerint  $\text{NL} = \text{coNL}$ .

## Tétel

A 2SAT probléma **NL**-teljes.

## Ötlet

Azt már tudjuk, hogy  $2SAT \in \mathbf{NL}$ . Tekintsük az **ELÉRHETETLENSÉG** egy példányát. Minden csúcshoz rendeljünk egy  $x$  változót. A 2SAT azon  $\varphi$  példányát készítjük el, mely az összes  $\neg x \vee y$  tagokból áll, ahol  $(x, y)$  él, továbbá az  $s$  és  $\neg t$  tagokból, ahol  $s$  a kezdőcsúcs és  $t$  a cél.

Így  $\varphi$  kielégíthető  $\Leftrightarrow s$ -ből nem érhető el  $t$ .

Emiatt a 2SAT komplementere is **NL**-teljes.

## Összefoglalás

- Megismertük Savitch tételét és következményét.
- Megismertük az Immermann-Szelepcsényi tételt és következményét.
- Megismertük a logaritmikus tárigényű visszavezetést és hogy mi motiválja a bevezetését.
- Láttuk, hogy a logaritmikus tárigényű visszavezetés tranzitív.
- Kimondtuk, hogy a HÁLÓZAT-KIÉRTÉKELÉS probléma **P**-teljes.
- Beláttuk, hogy ELÉRHETŐSÉG, 2SAT és komplementereik **NL**-teljesek.