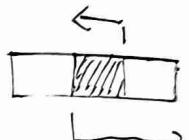


- Hledání podřízení v sekci

Znacení:  $\Sigma$  ... abeceda

- $x, y, z \dots$  znaky abecedy
  - $\alpha, \beta, \gamma \dots$  slova / reťazce  $\rightarrow |\alpha| =$  dĺžka
  - $\varepsilon \dots$  prázdny reťazec,  $|\varepsilon| = 0$
  - $\alpha\beta \dots$  konkatenácia
  - $\alpha[i] \dots$  i-ty znak
  - $\alpha[i:j] \dots \alpha[i]\alpha[i+1]\dots\alpha[j-1] \rightarrow |\alpha[i:j]| = j-i$
  - $\alpha[:j] = \alpha[0:j] \dots$  prefix
  - $\alpha[i:] = \alpha[i:|\alpha|] \dots$  suffix

☞ podzielice von prefixy suffixi resp. suffixy prefixi



## Problem

jebla  $r$ ,  $S := |r|$  ...  $r = \text{ana}$   
 seno  $\sigma$ ,  $S := |\sigma|$   $\sigma = \underline{\text{bananas}}$

- 1, Lineární průchod → když najdu 1. písmenko řeči kontroly  $\Rightarrow \text{H}(\text{d}\cdot\text{s})$   
 → pokud by se jeho bylo 1. písmenko mělo být, tak  $\text{H}(\text{s})$

- 2, Incrementální algoritmus → postupné přidávání znaky & senv

Def: Star algoritmu := nejdelsí prefix pehly, který je suffixem senv.

5  → nový stav je buď prázdný nebo máj žnak  $\alpha'$   
 $\Rightarrow \alpha'x$  je suf.  $\sigma x \Rightarrow \alpha'$  je suf.  $\sigma \quad \left\{ \begin{array}{l} \alpha \text{ je my. prefix } V \\ \alpha' \text{ je suf. } \alpha \end{array} \right.$   
 $\Rightarrow \alpha'x$  je pref.  $V \Rightarrow \alpha'$  je pref.  $V \quad \left\{ \begin{array}{l} \alpha \text{ je my. prefix } V \\ \alpha' \text{ je suf. } \alpha \end{array} \right.$

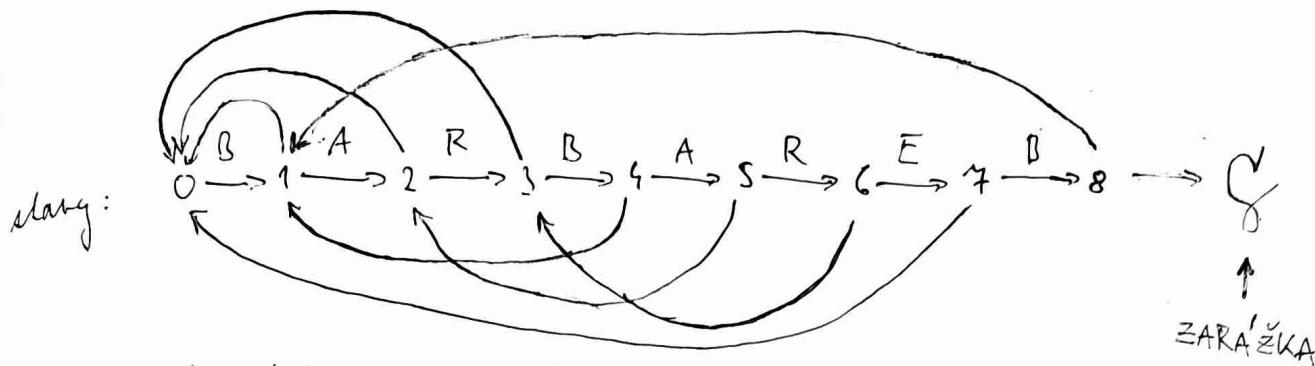
Def: zpečná funkce z(d) přiřadí slavnému jeho nejdelsší vláštní sufix, který je prefitem jehly.

$$\rightarrow y = 20 \cos \Rightarrow z(20 \cos) = 20$$

⇒ Son xpětňov fú budeme zbraňovat & cleva dokud z'x nebude prof. jehly

## KMP (Knuth, Morris, Pratt)

→ vyhledávací automat ... stav = délka prefixu jehly;  $\Sigma = \{A, B, C\}$



- dopředné hrany ~ přidávání písmenek  $\Rightarrow$  pamatujeme v ježku
- zpětné hrany ~ aplikace sítové funkce  $\Rightarrow$  pole  $Z[i:sav] = Z(stav)$

$stav \leftarrow sav$

krok( $s, x$ ):

1. Pokud  $s > 0$  &  $\Sigma[s] \neq x$ :  $\Sigma[0] \rightarrow \Sigma[1] \rightarrow \dots \rightarrow \Sigma[s]$
2.  $s \leftarrow Z[\Delta]$
3. Pokud  $\Sigma[s] = x$ : return  $s+1$
4. Jinak: return 0

## KMP Hledej( $\Sigma$ )

1.  $s \leftarrow 0$
2. Pro  $j$  znaky  $x \in \Sigma$ :
3.  $s \leftarrow krok(s, x)$
4. Pokud  $s = j$ :
5. ohlašme výsledek

poč se zavola krok( $s, x$ )  
 $\Rightarrow$  indexace  $\Sigma[j] \rightarrow$  musí být zazára  
 $\Sigma =$  znak co nukde jinde není  
 $\hookrightarrow$  v C by to fungovalo  $\rightarrow$  string konci \0

Invariáント: Stav je max. délka prefixu jehly, který je suffixem spracovaného sora.

0  $\Rightarrow$  KMP funguje.  $\hookrightarrow$  díky sítové funkci

Složitost:

$$\begin{aligned} & \# \text{dopředních eročí} \leq s \\ & \quad \quad \quad VI \\ & \# \text{zpětných eročí} \end{aligned} \quad \left\{ \begin{array}{l} \# \text{eročí po branách} \leq 2s \\ \Rightarrow \text{KMP Hledej běží v čase } \Theta(s) \\ \Rightarrow \text{KMP celkem v } \Theta(s+j) \end{array} \right.$$

## KMP Konstrukce( $\Sigma$ )

1.  $j \leftarrow |\Sigma|$
2.  $Z[0] \leftarrow \emptyset, Z[1] \leftarrow 0$
3.  $s \leftarrow 0$
4. Pro  $i = 2, 3, \dots, j$ :  $\Theta(j)$
5.  $s \leftarrow krok(s, \Sigma[i-1])$
6.  $Z[i] \leftarrow s$

$\rightarrow$  automat postavením formou automatu  
 $\Sigma A B A \rightarrow$  chci BA  
 $\Rightarrow$  vyhledáme  $\Sigma A B A R E B$  v  $\Sigma A R B A$   
 $\Sigma A B A R \rightarrow$  chci BAR.  
 $\Rightarrow$  vyhledáme  $\Sigma A B A R E B$  v  $\Sigma A R B A R$   $\nearrow$  nový eror  
 $\Rightarrow$  stačí vyhledat  $\Sigma A B A R E B$  v  $\Sigma A R B A R$   $\Rightarrow$  merinopledy

## Aho - Corasick

$\Sigma$  ... senv,  $S := |\Sigma|$

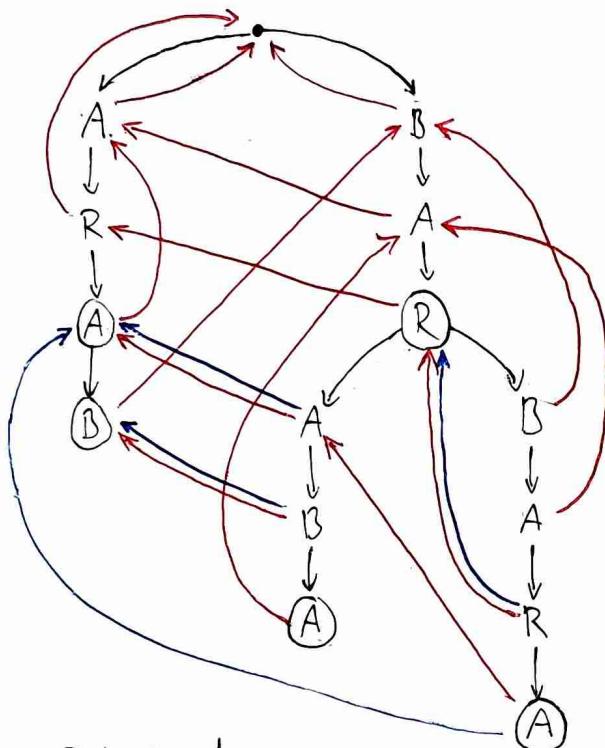
$r_1, \dots, r_m$  ... jehly,  $J := \sum_i |r_i|$

$V$  ... # výsledků jehel v sene

- vyhledávací automat - obdobně jako u KMP

- stav = prefixy jehel
- dopředné hrany:  $d \mapsto d^*$

ARA, ARAB, BAR, BARABA, BARBARA



- Zpětné hrany - stejně jako KMP

$d \mapsto$  nejdálší vlastní se  $d$ , co je slovem

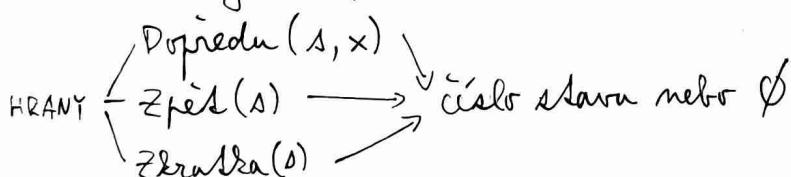
- nové písmeno → je možné ho přidat?
- pokud ne, tak se činí zpětnými hrany, dokud to nepůjde nebo nebrdu v kořeni

- Inv: Aktuální stav je nejdálší se senv, který je prefixem nejdálší jehly

- Zkratkové hrany -  $d \mapsto$  nejbližší stav dosažitelný po z. hranaích, kde končí jehla  
 $\Rightarrow$  za celý běh alg. po nich projde  $\Theta(V)$

### • Reprezentace

- stavov očislujeme  $\Rightarrow$  kořen = 0, rhybky libovolně



- Slovo(s)  $\rightarrow$  slovo co končí v s nebo  $\emptyset$

### • Ackrov(s,x)

1. Pokud  $s \neq$  kořen &  $Dopředu(s,x) = \emptyset$ :

2.  $s \leftarrow Zpět(s)$

3. Pokud  $Dopředu(s,x) = \emptyset$ : return kořen

4. Jinak: return  $Dopředu(s,x)$

### • AchHledaj(s)

1.  $s \leftarrow$  kořen

2. Pro každá  $x \in \Sigma$ :

3.  $s \leftarrow Ackrov(s,x)$

4.  $q \leftarrow s$

5. Pokud  $q \neq \emptyset$ :

6. Pokud  $Slovo(q) \neq \emptyset$ :  $S + \Theta(V)$

- hlásíme Slovo(q)

7.  $q \leftarrow Zkratka(q)$

Složitost - když máme automat

$$\begin{aligned} \textcircled{A}: S &\geq \# \text{dopředních} \geq \# \text{zpětných} \\ V &\geq \# \text{zkratkových} \end{aligned} \quad \left. \begin{array}{l} \text{celkem } \textcircled{H}(S+V) \end{array} \right\}$$

### Konstrukce automatu

→ pro bludinách ⇒ paralelní KMP pro všechny jehly

Ackonstrukce( $v_1, \dots, v_n$ )

1. Založíme tri s kořenem  $r$

2. Vložíme do trije všechny jehly → dopředné h.

3.  $Zpět(r) \leftarrow \emptyset$ ,  $Zkratka(r) \leftarrow \emptyset$  Slovo(-)

4.  $F \leftarrow$  fronta se sygy kořene

5. Sygum kořene nastavíme  $Zpět(-) \leftarrow r$ ,  $Zkratka(-) \leftarrow \emptyset$

$\textcircled{H}(j)$

6. Pokud  $F$  nemá prázdnou:

7.  $i \leftarrow F.\text{Dequeue}()$

8. Pro sygum s vrcholem  $i$ : → Ackrok by řel to dopředné h.

9.  $Z \leftarrow \text{Ackrok}(Zpět(i))$ , znak na hrani  $x$  i do  $s$ )

10.  $Zpět(s) \leftarrow Z$

11. Pokud  $\text{Slovo}(Z) \neq \emptyset$ :  $Zkratka(s) \leftarrow Z$

12. Linak:  $Zkratka(s) \leftarrow Zkratka(Z)$

13.  $F.\text{Enqueue}(s)$

Tady vlastě  
hledám ve  
všech jehlách  
→ lineární s délkon sna  
 $\Rightarrow O(j)$

Výta: Algoritmus A-C najde všechny výsledky jehel v čase  $\textcircled{H}(j+S+V)$ .

### Rabinov-Karpov algoritmus



⇒ porovnáváním hash jehly s hashy obdélníčku

Hash ⇒ chceme h. fci co se dá přepočítat v  $O(1)$  když posuneme okénko.

$$h(x_1, \dots, x_j) := (x_1 p^{j-1} + x_2 p^{j-2} + \dots + x_j p^0) \bmod M$$

$$h(x_2, \dots, x_{j+1}) = p \cdot h(x_1, \dots, x_j) - x_1 p^j + x_{j+1} \Rightarrow \text{přepočítat } p^j \bmod M$$

Alg:

1.  $\sigma \leftarrow h(v)$
2.  $\sigma \leftarrow h(\sigma[i:j])$  ]  $\textcircled{H}(j)$  - hornerovo schéma # false f.
3. Pro  $i = 0, \dots, S-j$ :
4. Pokud  $j = \sigma \wedge \sigma[i:i+j] = v$ :  $j \cdot (V+?)$
5. Ohlášíme výsledek
6.  $\sigma \leftarrow (p \cdot \sigma - \sigma[i] p^j + \sigma[i+j]) \bmod M$  ] S

Casova' složitost → pro dobu následnou  $h(x)$   
 $\textcircled{H}(j+S+j \cdot V + j \cdot \frac{S}{M})$  → # false pos  
 $\Rightarrow$  chceme  $M \in \Omega(j \cdot S)$   
 $\rightarrow p, M$  nesoudebné, M prvočíslo  
 $E[\# \text{false pos. kolizi}] = \frac{S}{M} \rightarrow$  reálné hodiny

## • Tely v síťech

Def: Síť je struktura obsahující

- orientovaný graf  $(V, E)$ , BÚNO symetricky:  $uv \in E \Rightarrow vu \in E$
- zdroj  $z \in V$  a soutěž (stope)  $s \in V$
- Kapacity  $c: E \rightarrow \mathbb{R}^+$

Def Tov je funkce  $f: E \rightarrow \mathbb{R}^+$  splňující

- $\forall e \in E: f(e) \leq c(e)$  → hranou může víc než její kapacita
- $\forall v \in V, v \neq z, s: f^\Delta(v) = 0$  → Kirchhoffův ráčon

Def: Pro  $v \in V$ :  $f^+(v) := \sum_{uv \in E} f(uv)$  ... prítok  $f[\text{In}(v)]$   
 $f^-(v) := \sum_{vw \in E} f(vw)$  ... odtok  $f[\text{Out}(v)]$   
 $f^\Delta(v) := f^+(v) - f^-(v)$  ... příbytek

Def: Velikost toču definujeme jako  $|f| := f^\Delta(s)$ .

Obs:  $f^\Delta(s) = -f^\Delta(z)$

$$\sum_{v \in V} f^\Delta(v) = f^\Delta(s) + f^\Delta(z)$$

$$\forall uv \in E: u \xrightarrow{-1} v \Rightarrow 0$$

$$\sum_{v \in V} f^\Delta(v) = 0 \because \text{Každý } \Sigma \text{ je lin. komb. točů na hranách}$$

■

## • Zvyšování toču

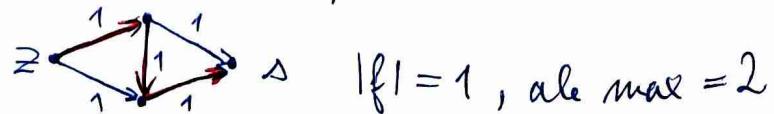
- začneme s nulovým točem a postupně ho budeme zvětšovat
- vybereme náhodnou cestu  $P$  ze  $z$  do  $s$

$z \xrightarrow{P} s \quad f'(e) := \begin{cases} f(e) + \epsilon, & e \in P \\ f(e), & e \notin P \end{cases}$

$$\epsilon := \min_{e \in P} (c(e) - f(e))$$

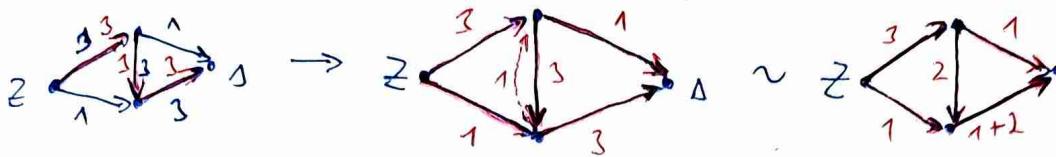
- tohle se nazývá protokl, protože vždy nasylíme alefou 1 hranu

- ale najde to max. toč



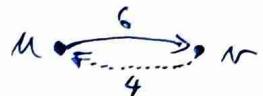
## Fordův - Fullersonův algoritmus

- využívá specifické hrany  ← nevídám jsi, že tohle je ekvivalentní
- když jsem hrany totéž, tak vytvořím odpovídající hrany stejně kapacity



$$c = 10$$

Def: Reserva hrany  $r(uv) := c(uv) - (f(uv) - f(vu))$



Def: Hrana  $e \in E$  je nenasycená  $\equiv r(e) > 0$



Cesta  $P$  je nenasycená  $\equiv \forall e \in P: r(e) > 0$   $r = 10 - (6 - 4)$

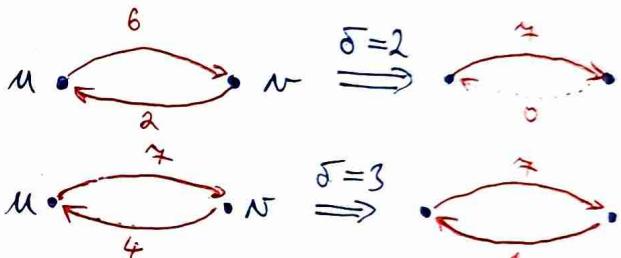
$\Leftrightarrow$ : Hrana  $uv$  je nasyčená  $\Leftrightarrow f(uv) = 10 \wedge f(vu) = 0$ .

## Algoritmus

1.  $f \leftarrow$  všechny nuly

$$c = 10, \quad E = 3$$

2. Dokud  $\exists P$  nenasycená cesta:



3.  $\varepsilon \leftarrow \min_{e \in P} (r(e))$

4. Pro  $\forall uv \in P$ :

5.  $\delta \leftarrow \min(\varepsilon, f(vu))$

$$\begin{cases} r' = c - (f(uv) + \varepsilon - \delta - f(vu) + \delta) \\ = r - \varepsilon \end{cases}$$

6.  $f(vu) \leftarrow f(vu) - \delta$

7.  $f(uv) \leftarrow f(uv) + \varepsilon - \delta$

$\Rightarrow$  reserva všech hrán na  $P$  zlepší se o  $\varepsilon$ , přičemž se vždy nejdříve snížíme odčítství nejvíce od zpětné hrany, než zvyšujeme dopřednou

## Konečnost

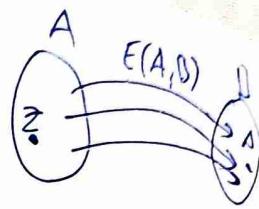
- pro  $c \in \mathbb{N}$  aho - algoritmus rachování celočíselnost - jen sčítá a odčítá  
 $\Rightarrow$   $|f|$  vždy stoupne alespoň o 1  $\Rightarrow$  konečný

- pro  $c \in \mathbb{Q}$  aho - invariance množiny měřitka - jde o definice  
 $\Rightarrow$  vynásobení kapacity NSN jejich jmenovateli  $\Rightarrow \mathbb{N}$

- pro  $c \in \mathbb{R}$  jak kdy, ale max. tot. existuje

Def: Pro  $A, B \subseteq V$ :  $E(A, B) := E \cap (A \times B)$

Def: Elementární řeč je  $E(A, B)$  pro  $A \subseteq V$ ,  $B = V \setminus A$ ,  $z \in A$ ,  $s \in B$

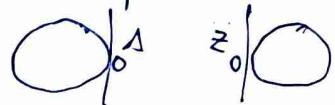


Def:  $f(A, B) := \sum_{e \in E(A, B)} f(e) = f[\text{out}(A)] \dots$  kde  $e \in A$  do  $B$

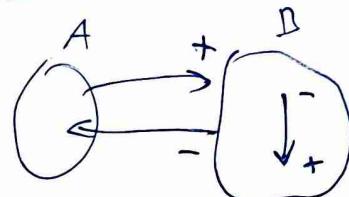
$c(A, B) := \sum_{e \in E(A, B)} c(e) \dots$  zafacita řeč

$f^\Delta(A, B) := f(A, B) - f(B, A) \dots$  když toto reálně řeč je spletitě

Pokud je  $f$  totéž a  $E(A, B)$  ier, pak  $\underline{f^\Delta(A, B) = |f|}$ .



$$f(A, B) - f(B, A) = \sum_{v \in B} f^\Delta(v) = f^\Delta(s) = |f|$$



↳ Ty brány co jsou uvnitř  $B$   
jednon příspěví  $\oplus$ , druhé  $\ominus$

Důsledek:  $|f| = f(A, B) - f(B, A) \leq c(A, B) \dots$  pro  $\nexists$  totéž a řeč

Důsledek: Pokud  $|f| = c(A, B)$  pak  $f$  je max. totéž a  $E(A, B)$  je min. řeč

Lemma: Pokud se F.F. alg. zastaví, pak  $f$  je max. totéž.

Def:  $A := \{v \in V \mid \exists \text{ menajícená cesta } z \rightarrow v\}$

$B := V \setminus A$ ,  $z \in A$ ,  $s \notin A \Rightarrow E(A, B)$  je řeč

brány  $A \rightarrow B$ :  $\left. \begin{array}{l} f = c \\ f = 0 \end{array} \right\}$  jinak  $\text{reserv} > 0 \Rightarrow$  menajícená brana  $\not\in$   
brány  $B \rightarrow A$ :  $\left. \begin{array}{l} f = 0 \\ f = 0 \end{array} \right\}$  jinak  $\text{reserv} > 0 \Rightarrow$  menajícená brana  $\not\in$

$\Rightarrow |f| = f(A, B) - f(B, A) = c(A, B) - 0 \Rightarrow f$  je max. a  $E(A, B)$  je min. ■

Shrnutí

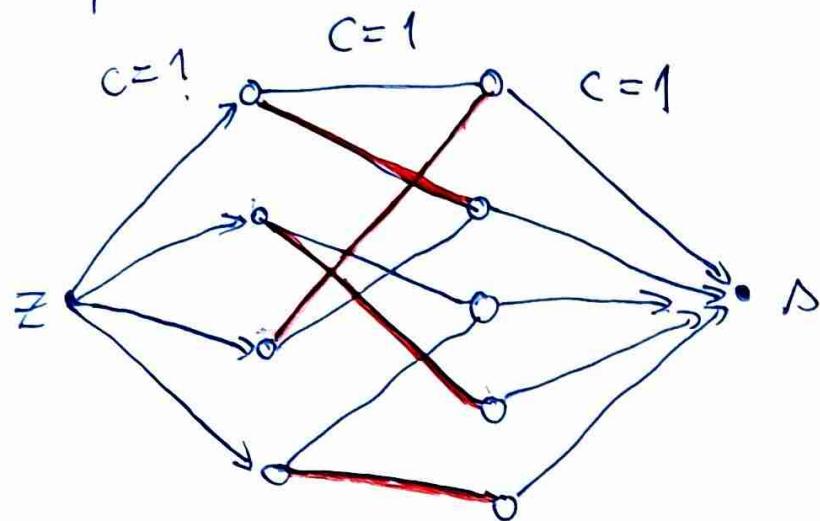
① F.F. alg. pro  $c \in \mathbb{Q}$  skončí a majde max. totéž (a min. řeč)

②  $\nexists$   $f$  max. totéž  $\exists E(A, B)$  min. řeč t.j.  $|f| = c(A, B)$

③ Všichni  $s \in \mathbb{N}$  je alespoň 1 max. totéž celocíselný a F.F. alg. ho majde  
↳ protože zachovává celocíselnost

## Největší párování v bipartitním grafu

↳ párování je  $F \subseteq E$  1.ř.  $\forall e, f \in F: e \cap f = \emptyset$  ... neor. graf



→ najdeme maximální řešení

⇒ bude 0-1 ... pro každou vrcholovou skupinu 0 nebo 1

⊗ R párování umím vytvořit 0-1 řešení,  
jehož velikost = # hran párování

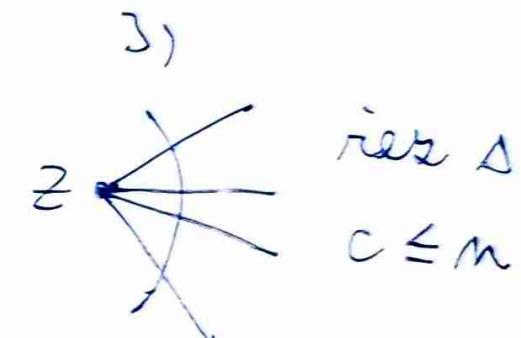
⊗ 0-1 řešení umím vytvořit párování  
→ první venku hranu řešení nazveme řešením

$\Rightarrow \exists$  bijectivní funkce mezi 0-1 řešeními a párováními, navíc  $|f| = \# \text{ hran párování}$

$\Rightarrow$  max 0-1 řešení ~ největší párování

## Složitost F.F. alg. pro $c=1$

- 1) 1 průchod řádky  $O(m)$  ... BFS
  - 2)  $\#$  průchod zvětší  $|f|$  alespoň o 1
  - 3)  $\#$  průchodu  $\leq m$
- $\left. \begin{array}{l} \\ \end{array} \right\} O(n \cdot m)$



## Definice algoritmus

Def: Tzv. f pravidle přílož  $f^*: E \rightarrow \mathbb{R}$ ,  $f^*(uv) := f(uv) - f(vu)$ .

$$\begin{array}{l} 1) f^*(uv) = -f^*(vu) \\ 2) f^*(uv) \leq c(uv) \end{array} \quad \left. \begin{array}{l} c(vu) \leq f^*(vu) \\ \text{Kirchhoffův zákon} \end{array} \right\}$$

$$3, \forall v \neq z, s: \sum_{uv \in E} f^*(uv) = \sum_{uv \in E} (f(uv) - f(vu)) = Im - Out = f^\Delta(v) = 0$$

$$\hookrightarrow \pi(uv) = c(uv) - f(uv) + f(vu) = c(uv) - f^*(uv)$$

Lemma: Pokud fce  $f^*: E \rightarrow \mathbb{R}$  splňuje 1, & 2, & 3, pak  $\exists f$  jehož přílož je  $f^*$ .

Def:  $\forall uv, vu \in E: \text{DÚNO } f^*(uv) \geq 0 \dots \# 1)$

$$\begin{array}{l} f(uv) := f^*(uv) \\ f(vu) := 0 \end{array} \quad \left. \begin{array}{l} \text{takže je 1. a 2. je to nezávisele směr} \\ \text{Kirchhoffův zákon je 3.)} \end{array} \right\}$$

Def: Pro sít  $S = (V, E, Z, A, C)$  a funkci  $f$  na něj je sít s rozdíly  $R(S, f) := (V, E, Z, A, \pi^R = C - f^*)$ .

Lemma: Pro  $\forall f$  na  $S$  a  $\forall g$  na  $R(S, f)$

$\exists$  řešení  $h$  na  $S$  t.č.  $|h| = |f| + |g|$  a navíc lze  $h$  najít v čase  $O(m)$

Def:  $h^* := f^* + g^*$  a  $\# h^*$  umíme sestrojit  $h$  v  $O(m)$

(scheeme učíme, že  $h^*$  je přílož a řešení se velikostí

$\rightarrow h^*$  splňuje ①

$$\rightarrow h^*(uv) = f^*(uv) + g^*(uv) \leq f^*(uv) + \pi(uv) = c(uv) \dots \quad \left. \begin{array}{l} \text{c-f}^* \\ \text{f} \end{array} \right\} \quad \begin{array}{l} h^* \text{ je přílož} \\ ② \end{array}$$

$$\rightarrow ③ \text{ taky protože pro } \forall v \neq z, s \text{ máme } 0+0=0 \checkmark$$

$\rightarrow |h| = |f| + |g|$  protože  $|f| =$  počet spořebnice a  $|g|$  počet spořebnice se sčítají

Def: Tzv. f je blokující  $\equiv \forall P$  cesta  $v \rightarrow s \quad \exists e \in P: f(e) = c(e)$

Def: Sít  $S$  je restrukturační (pruživitelná)

$\equiv \forall$  vrchol i brana lze májet krajně nejméně dvě cesty  $v \rightarrow s$ .

Rimízový alg.  $O(n^2 m)$

1.  $f \leftarrow$  všechny můžoucí řeš (nebo elidně i nějaký lepsi')

2. Operujeme:

3.  $R \leftarrow$  síť rezerv ve hledaném  $\& S \quad O(m)$

4. Smazeme  $\& R$  můžoucí hrany  $O(m)$

5. Pročistíme  $R$  = necháme tam jenom vrcholy a hrany na nejkratších cestách  $O(m)$

6. Pokud je  $R$  prázdná, tak končíme

7.  $g \leftarrow$  blokující řeš  $\& R \quad O(nm)$

8. Vylepšíme  $f$  pomocí  $g \quad O(m)$

# fáci  $\leq n$

Korektnost: Pokud se rozloží řeš  $\&$  cesta  $z \rightarrow s \& R \Rightarrow$  f nebyla řešená  $\& S$   
 $\Rightarrow$   $R$  korektnosti F.F. alg. je současně i řeš

číslení sítě  $O(m)$

1. Pomocí DFS( $z$ ) rozdělíme síť na vrstvy  $O(m)$

2. Smazeme vrstvy  $\& S$

3. Smazeme všechny hrany co nerodí o vrstvu dopředu

4. Počítíme si frontu na slépě uličky a všechny je smazeme do konce  $\& f$  neje

Hledání blokujícího řešení ve vstevnaté síti  $O(m \cdot n)$

1.  $g \leftarrow 0$   $\hookrightarrow$  řeš je i algoritmem 3 indí  $O(n^2)$

2. Do konce  $\exists P$  cesta  $z \rightarrow s \& R$ :

3.  $E \leftarrow \min_{e \in P} (r(e) - g(e))$  ...  $r(e)$  je  $f$ -rezerva =  $g$ -defacita

4.  $\forall e \in P: g(e) \leftarrow g(e) + E$  a pokud  $g(e) = r(e)$ , hrana  $e$  smazeme  $\& R$

5. Dočistíme nově vzniklé slépě uličky - mohou vzniknout při mazání  $e$

• 1 iterace kromě ⑤ trvá  $O(n) = O(\# vrstev)$  & # iterací  $\leq m$   
 $\hookrightarrow$  na cestu hledáme metudem různou způsobem

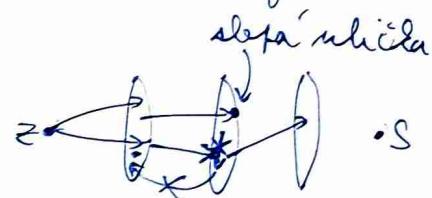
• ⑤ celkově probíhá  $O(m+n) =$  smazání všech vrcholů a hrani  $\} O(mn)$

Lemma: Mezi dvěma sousedními fáci se rozloží # vrstev  $R$  alespoň o 1.

D: Odstraním všechny nejkratší cesty délky  $l$ , takže tam řeší nejkratší bude  $l+1$ . Ale mohla mi tam vzniknout spěšná hrana.  
To ale neradi.  $\because$  neče do zadu  $\Rightarrow$  cesta w jí pouze má délku alespoň  $l+2$   $\blacksquare$

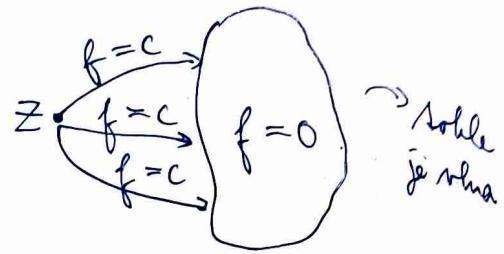
Risulek: # fáci  $\leq n \Rightarrow$  Rimízov algoritmus běží v čase  $O(n^2 m)$

$\hookrightarrow$  reálně to je  $\textcircled{H}(n^2 m)$



## Goldbergův algoritmus

Def:  $f: E \rightarrow \mathbb{R}^+$  je vlna  $\Leftrightarrow \forall e \in E: f(e) \leq c(e)$   
 $\forall v \neq z, s: f^\Delta(v) \geq 0$



## Převedení přebytku

$$\begin{array}{ll} u & f^\Delta(u) > 0, r(uv) > 0 \\ \downarrow & \delta := \min(f^\Delta(u), r(uv)) \\ v & f'(uv) := f(uv) + \delta \end{array} \quad \Rightarrow \quad \begin{array}{l} f' \text{ existuje vlnou} \\ f'(u) - = \delta, r(uv) - = \delta \\ f'(v) += \delta, r(vu) += \delta \end{array}$$

Def: Výška  $h: V \rightarrow \mathbb{N}$  ... a převádíme jen  $\rightarrow$  kopeček

## Algoritmus

1.  $f(*) \leftarrow 0, \forall zv \in E: f(zv) \leftarrow c(zv)$  ]  $O(n)$
2.  $h(*) \leftarrow 0, h(z) \leftarrow n$
3. DoEnd  $\exists u \neq z, s: f^\Delta(u) > 0:$
4. PoEnd  $\exists uv \in E: r(uv) > 0 \& h(u) > h(v):$   
převádíme po  $uv$
5.  $\text{linek } h(u) \leftarrow h(u) + 1$

## Analýza algoritmu

### Inv A (začátku)

- 1)  $f$  je vlna  $\Leftarrow$  4)
- 2)  $\forall v: h(v)$  neleská
- 3)  $h(z) = n, h(s) = 0$  ...  $v$  tom cyklu vglučujeme  $z$  a  $s$
- 4)  $\forall v \neq z: f^\Delta(v) \geq 0$  ... přebytky se mění převedením

### Inv S (o spádu) $\rightarrow$ vždy převádíme jidlo až do kopečku

Def: Spád brany  $uv$  je  $h(u) - h(v)$  ... o kolik dolů ta brana vede

Inv:  $\nexists uv \in E: r(uv) > 0 \& h(u) > h(v) + 1$

Dr: Po inicializaci platí - ty brany už mají + rezervu ale nedou do kopečku.

a, nenasyčená brana se spádem 1 a zvídavou plní ji  $\Rightarrow$  4)

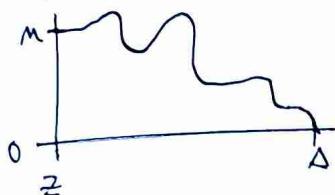
b, nasycená se spádem  $> 1$  a stala se nenasyčenou  $\Rightarrow$  nerostla ji rezerva

Lo rezerva roste při převádění té zpětné brány, ale ta je do kopečku  $\Rightarrow$

Lemma K (Korektnost): Pokud se algoritmus rastaví, tak  $f$  je max. tot.

Dle: ①  $f$  je tot  $\Leftrightarrow$  se rastavil sen cyclus  $\Rightarrow \forall v: f^\Delta(v) = 0$

②  $f$  je max  $\Leftrightarrow$  kdyby ne, tak podle F.F.  $\exists P$  menasycená cesta  $v \rightarrow z$



$\rightarrow$  ta cesta překonává spád  $n$  & má nejméně  $m-1$  hran  
 $\Rightarrow \exists e \in P: \text{spád}(e) \geq 2 \wedge r(e) > 0 \quad \checkmark \text{ InvS}$

InvC (cesta do vrcholu):  $\forall v: f^\Delta(v) > 0 \quad \exists P$  menasycená cesta  $v \rightarrow z$ .

$\Rightarrow$  InvV (o výšce):  $\forall v: h(v) \leq 2m$ .

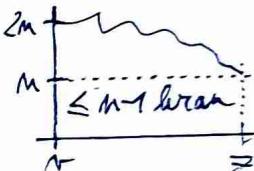
$\Rightarrow$  Lemna Z (o srednici): # srednic  $\leq 2m^2$ .

Dle:  $m$  vrcholů,  $\frac{1}{2}$  srednice max.  $2m$ -krát  $\blacksquare$

Dle: Uvažme první formaci ... to muselo mítat srednici

$\Rightarrow$  zdaříme  $v \in V$  k výšky  $2m$ , tehdy  $f^\Delta(v) > 0$

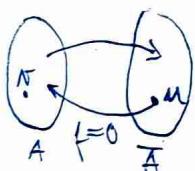
$\Rightarrow$  podle InvC  $\exists P$  menasycená cesta  $v \rightarrow z \quad \checkmark \text{ InvS}$



Dle:  $A := \{u \in V \mid \exists \text{ menasycená cesta } v \rightarrow u\}$ , uvažme  $z \in A$ .

$$\sum_{a \in A} f^\Delta(a) = \sum_{a \in A} (f^{\text{In}}(a) - f^{\text{Out}}(a)) = f[\text{In}(A)] - f[\text{Out}(A)] \leq 0$$

$\underbrace{\text{až hranou mohou být}}_0 \quad \underbrace{\geq 0}_{\text{principiálně } + a \ominus \text{ jinak křížná hraná má rezervu } > 0 \Rightarrow u \in A}$



$\rightarrow$  ta  $\sum$  obsahuje  $f^\Delta(v) > 0$ , ale je nedlouhá

$\Rightarrow$  obsahuje ráformující člen, jediný takový je  $z \Rightarrow z \in A$ .  $\blacksquare$

• Klik bude přivedení?

Dle: Přivedení je nasycené  $\Leftrightarrow$  využívá rezervu hrany.

$\circlearrowleft$  Menasycené přivedení po hrani ut využívá  $f^\Delta(u)$ .

Lemna S (sytá přivedení): # nasycených přivedení  $\leq n \cdot m$ .

Dle: Uvažme hrana  $uv$  ... po systém přivedení  $r(uv) = 0$ . Abychom mohli systém přivedit snor, tak potřebujeme  $r(uv) > 0$ . Na to musíme převést po hrani  $vu$ , ale  $vu$  má myší spád 1, takže  $v$  musí srednot 0.2.  $uv$  má myší spád -1, abych mohl převádět, tak musím  $u$  srednot 0.2.

$\Rightarrow$  podle InvV toto nejvíce  $n$ -krát & mám  $m$  hran  $\Rightarrow n \cdot m \quad \blacksquare$

## Lemma H (bladová převodní) ... nenašly hranu

Def: Potenciál  $\phi := \sum_{v \neq z, A} h(v) f^A(v) > 0$

⊗ ①  $\phi \geq 0$

(2) Na počátku  $\phi = 0$

③ Závěrnutí  $\sim \phi + = 1$

④ SP ... možná  $+ h(v) - h(u) \dots \phi + = \max. 2m$  (může se i smířit)

⑤ HP ... určitě  $- h(u)$ , možná  $+ h(v) \dots \phi - = \text{alespoň } 1$  ( $h(v) - h(u) = 1$ )



výsledek se nemění  
možná  $f^A(u) = 0 \Rightarrow -h(u)$   
možná předkym  $f^A(v) = 0 \Rightarrow +h(v)$

Důsledek:  $\phi \leq 2m^2 \cdot 1 + m \cdot m \cdot 2m = 2m^2(m+1) \Rightarrow \# \text{HP} \leq 2m^2(m+1) \in O(m^3)$   
 $\Rightarrow$  Goldberg je konečný

## Časová složitost Goldberga

→ závěrnutí a převádění sumujeme rychle

→ chceme umět rychle najít řen vrchol s přebytkem

①  $S :=$  seznam vrcholů s přebytkem

↳ údržba při změně  $f^A(v)$  a výběr v Srovnu ③ :  $O(1)$

② Pro  $\forall u \in V : K(u) := \{uv \mid r(uv) > 0 \text{ a } h(u) > h(v)\}$

↳ výběr ve krově ④ :  $O(1)$

→ údržba při převodní: smění  $r(uv)$  a rozšíření  $r(vu)$   $O(1)$

↳ ale v u vede do kopce → nemůžeme řešit  $K(\star)$

→ údržba při závěrnutí: musíme probrat všechny hranы uv a vu --  $O(n)$

↳ ale závěrnutí je jen  $O(n^2)$

$$\Rightarrow \text{celkem } O(2m^2 \cdot m + m \cdot m \cdot 1 + m^2 \cdot m) = \underline{\underline{O(m^3)}}$$

závěrnutí      SP      HP

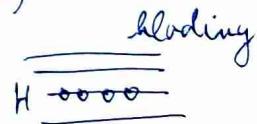
## Vylepšení Goldberga

→ nejvíce nám brzdí HP, chci jich mít méně

Lemma H\*: Když vybíráme nejvyšší vrchol s  $f^A(v) > 0$ :  $\# \text{HP} \in O(n^3)$

Def:  $H := \max \{h(v) \mid f^A(v) > 0, v \neq z\}$

→ během algoritmu rozdělíme na fáze → fáze končí eminenem H



1) konec fáze: rozšíření H ...  $\leq 2m^2$ -krát, vždy právě  $\sigma^1$  }  $\# \text{fází} \in O(n^2)$

2, H ještě aspoň  $\sigma^1$  ... ale  $H \geq 0 \Rightarrow \leq 2m^2$ -krát

⊗ Během 1 fáze děláme  $\leq H$  vrcholů nejvyšší 1 HP



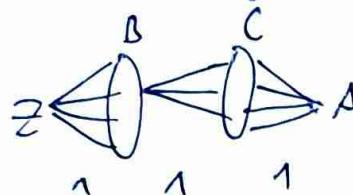
↳ HP  $\Rightarrow f^A(v) = 0$ , aby další HP, Ale jsem do něj museli něco převést, ale H je mal. &

$\Rightarrow \# \text{HP} \text{ ve fázi} \leq m \Rightarrow \# \text{HP celkem} \in O(n^3) \Rightarrow$  Goldberg  $O(n^3)$

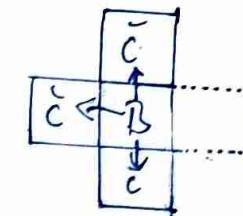
→ Zlepšení potenciál co dává  $\Theta(n^2 \sqrt{m})$

## Aplikace tohoto v silich

1) Déravá šachovnice, chceme ji položit dominem.



→ hrany vedou ze všech bílých políček do jejich černých sousedů

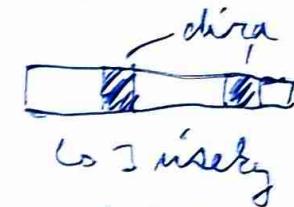


→ najdu nejřetězí párování

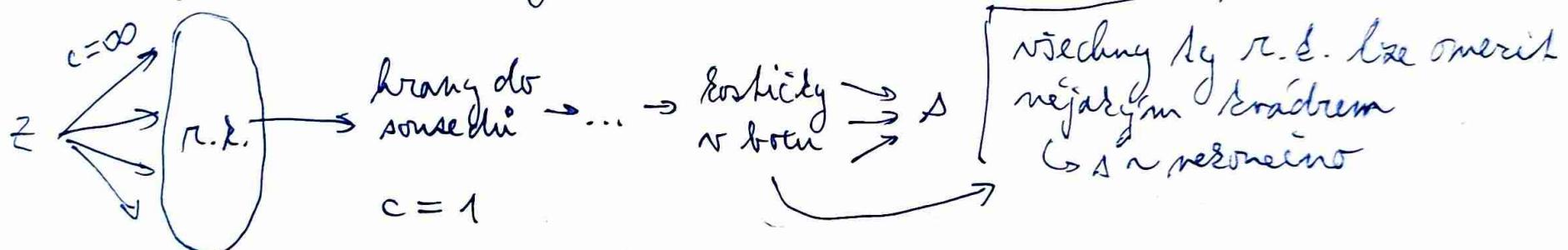
2) Déravá šachovnice, chceme rozumět co nejméně rázů, aby se mohly mezi sebou

↪ ráže nemohou přes díry

⇒ najdeme nejřetězí párování řádku sloupců a rádek



3) Radioaktivní kostičky v minecraftu. Chceme do světa rozumět  
kostičky, co urazí r. k. ⇒  $\exists$  cesta r. k. →  $\infty$  co nepojde kostičkou.



⇒ max.  $\text{LoE} \Rightarrow \min. \text{LoE}$  - hrany toho ráza a umístění kostiček.

## • Fourierova transformace

• Polynomy  $P(x) := \sum_{k=0}^{m-1} h_k x^k$ ,  $\vec{P} = (h_0, \dots, h_{m-1})$ ,  $m = \underline{\text{velikost polynomu}}$

↳ normalizace:  $h_{m-1} \neq 0$  nebo  $m=0$   $(1, 0, 3, 0) \rightarrow (1, 0, 3)$

↑ stupeň polynomu

→ stupeň (1) ≠ 0, stupeň () ≠ -1

↳ stupeň 2

• Násobení polynomů ... BÚNO  $P, Q$  stejné velké

$$R = P \cdot Q$$

$$R(x) = \sum_{j=0}^{m-1} h_j x^j \sum_{k=0}^{m-1} q_k x^k = \sum_{j,k=0}^{m-1} h_j q_k x^{j+k} = \sum_{l=0}^{2m-2} \left( \sum_{j=0}^l h_j q_{l-j} \right) x^l \Rightarrow \text{(H)(M)}$$

• Rovnost polynomů

1, identita  $P \equiv Q$ : stejné vektory pro normalizaci

①

①

↓

↑

2, rovnost fci:  $\#x: P(x) = Q(x)$

②

②

↳ důkazem

Věta: Nechť  $P, Q$  jsou polynomy stupně max. d a  $P(x_j) = Q(x_j)$  pro následující různá čísla  $x_0, \dots, x_d$ . Potom  $P \equiv Q$ .

Lemma: Pro polynom  $P$  stupně  $d \geq 0$ :  $(\#x: P(x) = 0) \leq d$ .

Dz:  $R := P - Q$

$\forall j: R(x_j) = P(x_j) - Q(x_j) = 0 \Rightarrow d+1$  kořenů & stupeň  $R \leq d$

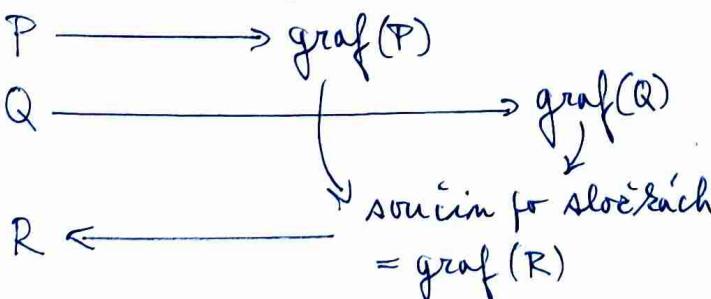
⇒ lemma nepříloží ⇒ nesplnilo jeho předpoklad  $\Rightarrow R \equiv 0 \Rightarrow P \equiv Q$ . ■

Def: Graf polynomu  $P$  velikosti  $n$  pro farní zvolená  $x_0, \dots, x_{n-1}$  je  $(P(x_0), \dots, P(x_{n-1}))$ .

⊗ Polynom je grafem jednoznačné určení.

⊗ Pokud  $R = P \cdot Q$ , potom  $R(x) = P(x) \cdot Q(x)$

Plán: Zvolíme nějak  $x_0, \dots, x_{m-1}$



Problém: Stupeň  $R$  může být  $> m$

⇒ BÚNO horních  $m/2$  koř

$P$  a  $Q$  jsou nuly

Problém: Převod  $P \rightarrow \text{graf}(P)$  a opak neuvede rychle

## Pokus o výhodnoucení polynomu metodou rozděl a sčítaj

Příklad m = 2<sup>2</sup> ⇒ výhodnouceme ν x<sub>0</sub>, ..., x<sub>m-1</sub>, faktorání x<sub>m/2+j</sub> = -x<sub>j</sub>

$$\begin{aligned} Q(x) &= x^{2m} = Q(-x) \\ \textcircled{O} \quad Q(x) &= x^{2m+1} = -Q(-x) \end{aligned}$$

$$\begin{aligned} P(x) &= p_0 x^0 + p_1 x^1 + p_2 x^2 + \dots + p_{m-1} x^{m-1} \\ &= (p_0 x^0 + p_2 x^2 + \dots + p_{m-2} x^{m-2}) = S(x^2) \\ &\quad + (p_1 x + p_3 x^3 + \dots + p_{m-1} x^{m-1}) = x L(x^2) \end{aligned} \quad \left\{ \begin{array}{l} P(x) = S(x^2) + x L(x^2) \\ P(-x) = S(x^2) - x L(x^2) \end{array} \right.$$

Výhodnoucím rozdíl m ν m bodcích → 2x rozdíl m/2 ν m/2 bodcích

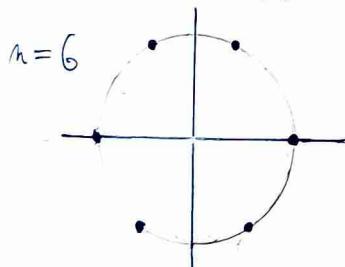
$$T(m) = 2 \cdot T(m/2) + \Theta(m) = \Theta(m \log m)$$

→ ale potřebujeme aby 1x faktorání vydálo i ν 1m podproblemu  
⇒ budť počítáme ν nejakeém chytrém řešením nebo kompletní čísla

## Kompletní odmocniny

$$x = \sqrt[m]{1} \Rightarrow |x| = 1 \Rightarrow x = e^{i\varphi} \text{ pro nějaké } \varphi$$

$$x^m = e^{im\varphi} = (\cos(\varphi_m) + i \sin(\varphi_m)) = 1 \Rightarrow \varphi_m = \frac{\ell \cdot 2\pi}{m}, \ell = 0, \dots, m-1$$



→ m-úhelník s vrcholem ν 1 ⇒ m různých odmocnin

$$\text{pro } |x|=1: \bar{x}^1 = e^{-i\varphi} = \bar{x}$$

Def:  $w \in \mathbb{C}$  je primitivní m-1á odmocnina z 1  $\equiv w^m = 1$  a  $w^1, \dots, w^{m-1} \neq 1$ .

$\textcircled{O} \quad w^j \neq w^k$  pro  $0 \leq j < k < m$ .

Když  $w_j = w_k$ , tak  $\frac{w^k}{w_j} = 1 \Rightarrow w^{k-j} = 1$ , ale  $1 \leq k-j < m$  ↗

$\textcircled{O}$  pro sudé m:  $w^{m/2} = -1$ , protože  $w^{m/2} = \sqrt{w^m} \& w^{m/2} \neq 1$ .

$$\Rightarrow w^{m/2+\ell} = -w^\ell \quad w \boxed{0 \mid 1 \mid \dots \mid m/2-1 \mid m/2 \mid \dots \mid m-1} \quad \begin{matrix} \text{obruba} = - \text{první} \\ \text{nula} = - \text{poslední} \end{matrix}$$

↗ je to sgrafováno

$\textcircled{O}$  pro t m ∃ primitivní m-1á odmocnina 1

$$\rightarrow \text{např. } w = e^{\frac{2\pi}{m}i}, e^{-\frac{2\pi}{m}i}$$

↗ podporovat

$w^2$  je primitivní m/2-1á odmocnina z 1 ...  $\because w^2, w^4, \dots, w^{m-2} \neq 1$

$\Rightarrow i w^0, w^2, w^4, \dots, w^{m-2}$  je dobré sgrafovaná posloupnost

• Algoritmus FFT  $\rightarrow$  převod polynomu na graf

Vstup:  $n = 2^k$ ,  $\omega$  = primitivní  $n$ -ta odmocnina  $\neq 1$

$(p_0, \dots, p_{n-1})$  koeficienty polynomu  $P$ .

Výstup:  $(y_0, \dots, y_{n-1})$  graf( $P$ ) v bodech  $(1, \omega, \omega^2, \dots, \omega^{n-1})$

$\mathcal{O}(n \log n)$

0. Počátek  $n=1$ :  $y_0 = p_0$  a koncime

1.  $(s_0, \dots, s_{n/2-1}) \leftarrow \text{FFT}(n/2, \omega^2, (p_0, p_2, \dots, p_{n-2}))$

2.  $(l_0, \dots, l_{n/2-1}) \leftarrow \text{FFT}(n/2, \omega^2, (p_1, p_3, \dots, p_{n-1}))$

$\left. \begin{array}{l} \\ \end{array} \right\} 2 \times \text{rekurencie}$

3. Pro  $j = 0, \dots, n/2-1$ :

$$y_j \leftarrow s_j + \omega^j l_j \quad = S(x_j^2) + X_j L(x_j^2) = P(x_j) \quad \text{kde } x_j = \omega^j$$

$$y_{n/2+j} \leftarrow s_j - \omega^j l_j \quad = S(x_j^2) - X_j L(x_j^2) = P(-x_j) \quad x_j^2 = (\omega^j)^2$$

$\mathcal{O}(n)$

Def: Diskrétní Fourierova transformace (DFT) je lineární zobrazení

$$\mathcal{F}: \mathbb{C}^n \rightarrow \mathbb{C}^n, \vec{y} = \mathcal{F}(\vec{x}) \equiv Y_j y_j = \sum_{k=0}^{n-1} X_k \omega^{jk} = X(\omega^j) \dots \text{výhodou je polynom } X$$

$\Rightarrow$  FFT počítá DFT ale faš!

$$\omega = \text{pr. } \sqrt[n]{1}$$

$\Leftrightarrow \mathcal{F}$  je l.z.  $\Rightarrow \vec{y} = \Omega \vec{x}$ , kde  $\Omega \in \mathbb{C}^{n \times n}$ ,  $\Omega_{j,k} = \omega^{jk}$ .

$\Leftrightarrow$  Když chci  $x$  zpět z  $\vec{y}$ , tak potřebuju  $\vec{x} = \Omega^{-1} \vec{y}$

Inverzí:  $\Omega \cdot \bar{\Omega} = n I_n \Rightarrow \Omega^{-1} = \frac{1}{n} \bar{\Omega}$ .

Důkaz:  $(\Omega \bar{\Omega})_{jk} = \sum_{\ell} \Omega_{j,\ell} \cdot \bar{\Omega}_{\ell,k} = \sum_{\ell} \omega^{jk} \cdot \bar{\omega}^{k\ell} = \sum_{\ell} \omega^{jk} \cdot \bar{\omega}^{k\ell} = \sum_{\ell=0}^{n-1} (\omega^{j-k})^{\ell}$

$$1, j \neq k: \omega^{j-k} \neq 1 \Rightarrow (\Omega \bar{\Omega})_{jk} = \frac{(\omega^{j-k})^n - 1}{\omega^{j-k} - 1} = \frac{0}{0} = 0.$$

$$2, j = k: \omega^{j-k} = 1 \Rightarrow (\Omega \bar{\Omega})_{jk} = n \cdot 1 = n$$



Důkaz dle: Inverzní DFT můžeme sformulovat pomocí FFT, kde za  $\omega$  zvolíme  $\bar{\omega}$ .

Pot. zdroje ještě musíme vynásobit  $1/n$ .

Důkaz dle: Polynomy lze násobit pomocí FFT v čase  $\mathcal{O}(n \log n)$ .

$\hookrightarrow$  graf  $\rightarrow$  polynom pomocí inverzní FFT

Důkaz dle linearity DFT:

$$\mathcal{F}(\vec{x} + \vec{y}) = \Omega(\vec{x} + \vec{y}) = \mathcal{F}(\vec{x}) + \mathcal{F}(\vec{y})$$

$$\mathcal{F}(\lambda \cdot \vec{x}) = \Omega(\lambda \cdot \vec{x}) = \lambda \cdot \mathcal{F}(\vec{x})$$

## Jiný pohled na DFT

Problém: Máme nějakou funkci  $F$  a chceme ji vyjádřit pomocí sinů a kosínů.  
Stačí nám to v nějakém intervalu  $\Rightarrow$  navozujeme si ji v  $n = 2^k$  bodech.

$\Rightarrow$  chceme majít  $\alpha_1, \dots, \alpha_{n/2-1}, \beta_1, \dots, \beta_{n/2}$  a  $\gamma$  když

$$F(x) = \sum_{\ell=1}^{n/2-1} \alpha_\ell \cdot \sin(2\pi\ell x) + \sum_{\ell=1}^{n/2} \beta_\ell \cdot \cos(2\pi\ell x) + \gamma; \quad x = 0, \frac{1}{n}, \frac{2}{n}, \dots, \frac{n-1}{n}$$

$$\text{pro } n=4: F(x) = \alpha_1 \sin(2\pi x) + \beta_1 \cos(2\pi x) + \beta_2 \cos(4\pi x) + \gamma$$

$$\circlearrowleft \ell = \frac{n}{2}: \sin(2\pi \frac{n}{2} \cdot \frac{\ell}{n}) = \sin(\pi \ell) = 0 \Rightarrow \text{místo } \alpha_{n/2} \cdot \sin(\dots) \text{ máme konstantu } \gamma$$

$\Rightarrow$  Sed' den vektor  $(x_0, \dots, x_{n-1})$  transformuje pomocí DFT  $\rightarrow (y_0, \dots, y_{n-1})$

$\hookrightarrow$  reálné a imaginární složky sice  $y_i$  mi dělají ty koeficienty  $\alpha$  a  $\beta$

$\Rightarrow$  zjistíme DFT navozovaného sinu a kosinu o různých frekvencích

$\Rightarrow$  díky linearity DFT víme jak bude vypadat DFT nějakého složeného souboru sinů a kosínů

$\Rightarrow$  Také víme, že každý reálný vektor lze zapsat jako  $\rightarrow$  v lineární součet sinů a kosínů

$\Rightarrow$  každý reálný vektor může způsobit a konkret se na to,

co mi udělá jaro na nějakon l.k. těch způsobovaných sinů a kosínů

$\Rightarrow$  to mi da', z jakých sinů a kosínů se sčítává ten původní vektor

Věta: Nechť  $\vec{x} \in \mathbb{R}^n$  a  $\vec{y} = \mathcal{F}(\vec{x})$ . Potom  $\forall j: y_j = \overline{y_{n-j}}$ .

$$\text{Dle: } y_j = \sum_{\ell=0}^{n-1} x_\ell w^{\ell j}$$

$$y_{n-j} = \sum_{\ell=0}^{n-1} x_\ell w^{(n-j)\ell} = \sum_{\ell=0}^{n-1} x_\ell \cdot \overline{w}^\ell \cdot \overline{w}^{j\ell} = \sum_{\ell=0}^{n-1} \overline{x_\ell} \cdot \overline{w}^{\ell j} = \overline{y_j} \blacksquare$$

Tworem: Každý reálný vektor lze zapsat jaro l.k. navozovaných sinů a kosínů.

$$\operatorname{Re}(y_j) = \text{kof. } n \cos(jx) \text{ pro } j=1, \dots, n/2$$

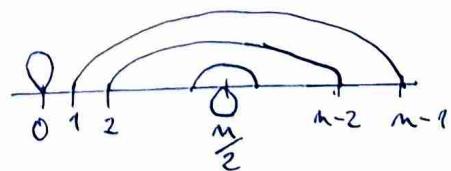
$$\operatorname{Im}(y_j) = \text{kof. } n \sin(jx) \text{ pro } j=1, \dots, n/2-1$$

$$\operatorname{Re}(y_0) = \text{aditivní konstanta } (\omega(0x)=1)$$

$$\operatorname{Im}(y_0) = 0 \quad (\sin(0x)=0)$$

$$\operatorname{Im}(y_{n/2}) = 0$$

Symetrie:



Příklad

$$P(x) = x^4 + x^2 - 3x + 1 \quad \dots \quad \vec{p} = (1, -3, 1, 0, 1, 0, 0, 0)$$

$$Q(x) = x^3 + 2x^2 - 1 \quad \dots \quad \vec{q} = (-1, 0, 2, 1, 0, 0, 0)$$

$$R(x) = P(x)Q(x) = x^7 + 2x^6 + x^5 - 2x^4 - 5x^3 + x^2 + 3x - 1 \quad \dots \quad \vec{r} = (-1, 3, 1, -5, -2, 1, 2, 1)$$

stupeň  $R \neq 7 \Rightarrow$  stáčí mi  $m = 8$  bodů,  $\omega = e^{\frac{2\pi}{8}i} = e^{\frac{\pi}{4}i}$

$$\Omega = (\omega^{ij}) = \begin{matrix} 0 & \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \omega^4 & \omega^5 & \omega^6 & \omega^7 \\ 2 & 1 & \omega^2 & -1 & -\omega^2 & 1 & \omega^2 & -1 \\ 3 & 1 & \omega^3 & -\omega^2 & \omega & -1 & -\omega^3 & \omega^2 \\ 4 & 1 & -1 & 1 & -1 & 1 & -1 & 1 \\ 5 & 1 & -\omega & \omega^2 & -\omega^3 & -1 & \omega & -\omega^2 \\ 6 & 1 & -\omega^2 & -1 & \omega^2 & 1 & -\omega^2 & -1 \\ 7 & 1 & -\omega^3 & -\omega^2 & -\omega & -1 & \omega^3 & \omega^2 \end{bmatrix} & \begin{bmatrix} 1 & -1 \\ -3 & 0 \\ 1 & 2 \\ 0 & 1 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \\ \cdot & \begin{bmatrix} 0 & 2 \\ -3\omega + \omega^2 & -1 + 2\omega^2 + \omega^3 \\ 1 - 3\omega^2 & -3 - \omega^2 \\ -3\omega^2 - \omega^2 & -1 - 2\omega^2 + \omega \\ 6 & 0 \\ 3\omega + \omega^2 & -1 + 2\omega^2 - \omega^3 \\ 1 + 3\omega^2 & -3 + \omega^2 \\ 3\omega^3 - \omega^2 & -1 - 2\omega^2 - \omega \end{bmatrix} \end{matrix}$$

$$\Omega = \begin{bmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \dots & \omega^7 \\ 1 & \omega^2 & \omega^4 & \omega^6 & \dots & \omega^{14} \\ \vdots & & & & & \\ 1 & \omega^7 & \omega^{14} & \dots & \omega^{49} \end{bmatrix} \quad \& \quad \begin{aligned} \omega^{4+8} &= \omega^4 \cdot \omega^8 = -\omega^2 \\ \omega^8 &= 1, \bar{\omega} = \bar{\omega}^{-1} \end{aligned} \quad \left| \quad \Omega \cdot \vec{p} = (P(1), P(\omega), \dots, P(\omega^7)) \right.$$

$\Rightarrow$  když mám graf( $P$ ) a graf( $Q$ )  $\Rightarrow$  vynásobím je pro graf( $R$ )

$\hookrightarrow$  dlelší kohle nad  $\mathbb{C}$  je mechaniky  $\Rightarrow \omega = 2 \bmod 17$  je primitivní  $\sqrt[8]{1}$ !

$$\Omega \cdot \begin{bmatrix} \vec{p} & \vec{q} \end{bmatrix} = \begin{bmatrix} 0 & 2 \\ -2 & -2 \\ 6 & -7 \\ 6 & -7 \\ 6 & 0 \\ 10 & -1 \\ 13 & 1 \\ 1 & 6 \end{bmatrix} \Rightarrow \text{graf}(R) = \Omega \vec{p} \cdot \Omega \vec{q} = \begin{bmatrix} 0 \\ 4 \\ 9 \\ 9 \\ 0 \\ 7 \\ 13 \\ 1 \end{bmatrix} \quad \begin{aligned} \bar{\omega} &= \bar{\omega}^{-1} \\ \bar{\omega}^{-1} &= -\omega^3 \\ \bar{\omega}^{-2} &= -\omega^2 \\ \bar{\omega}^{-3} &= -\omega \end{aligned}$$

$$\bar{\Omega} = \begin{matrix} 0 & \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -\omega^3 & -\omega^2 & -\omega & -1 & \omega^3 & \omega^2 & \omega \\ 2 & 1 & -\omega^2 & -1 & \omega^2 & 1 & -\omega^2 & -1 \\ 3 & 1 & -\omega & \omega^2 & -\omega^3 & -1 & \omega & -\omega^2 \\ 4 & 1 & -1 & 1 & -1 & 1 & -1 & 1 \\ 5 & 1 & \omega^3 & -\omega^2 & \omega & -1 & -\omega^3 & -\omega \\ 6 & 1 & \omega^2 & -1 & -\omega^2 & 1 & \omega^2 & -1 \\ 7 & 1 & \omega & \omega^2 & \omega^3 & -1 & -\omega & -\omega^3 \end{bmatrix} & \begin{bmatrix} 0 \\ 4 \\ 9 \\ 9 \\ 0 \\ 7 \\ -4 \\ -4 \end{bmatrix} \\ \cdot & \begin{bmatrix} 4+1+7-3 \\ 5\omega^3 - 13\omega^2 - 8\omega \\ -\omega^2 - 5 \\ 3\omega + 13\omega^3 - 8\omega^2 \\ -4 - 7 - 5 \\ -3\omega^3 - 13\omega^2 + 8\omega \\ \omega^2 - 5 \\ -3\omega + 13\omega^2 + 8\omega^3 \end{bmatrix} \\ = & \begin{bmatrix} 0 \\ 7+16-16 \\ -4-5 \\ 6-16+4 \\ 1 \\ -7+16+16 \\ 4-5 \\ -6-16-4 \end{bmatrix} = \begin{bmatrix} -8 \\ 7 \\ 8 \\ -6 \\ -16 \\ 8 \\ 16 \\ 8 \end{bmatrix} \end{matrix}$$

$$\bar{r} = \frac{1}{m} \bar{\Omega} = (-1, 3, 1, -5, -2, 1, 2, 1) \quad \checkmark \quad \Rightarrow \text{faktorál jsem nejdou dostatkově všechny bázi koho řešení aby bylo nejpravdivě}$$

$$\hookrightarrow \frac{1}{m} = 8^{-1} = 15 = -2$$

• Nyní pomocí FFT

$$\begin{bmatrix} 1 & 1 \\ 1 & w^2 \\ 1 & -1 \\ 1 & 1 \\ 1 & -1 \\ 1 & -w^2 \end{bmatrix} \begin{bmatrix} 1 & -1 \\ 1 & 2 \\ 1 & 0 \\ 0 & 0 \end{bmatrix}$$

$$S \rightarrow \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 & -1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 2 & -1 \\ 0 & -1 \end{bmatrix}$$

Zase  $w = 2$ , mod 17

*sude*:

$$\begin{array}{c}
 \left[ \begin{array}{ccccc} 1 & -1 & & & \\ 1 & 2 & & & \\ 1 & 0 & & & \\ 0 & 0 & & & \end{array} \right] \\
 \xrightarrow{\text{L}} \left[ \begin{array}{cc} 1 & 1 \\ \omega^2 & -\omega^2 \end{array} \right] \left[ \begin{array}{cc} 1 & 2 \\ 0 & 0 \end{array} \right] = \left[ \begin{array}{cc} 1 & \omega^2 \\ 1 & -1 \end{array} \right] \left[ \begin{array}{cc} 1 & 2 \\ 0 & 0 \end{array} \right] = \left[ \begin{array}{cc} 1 & \omega^2 \\ 1 & 2 \end{array} \right] \left[ \begin{array}{cc} 1 & 2 \\ 1 & 2 \end{array} \right] = \left[ \begin{array}{cc} 1 & 2 \\ 4 & 8 \end{array} \right]
 \end{array}$$

liche:

$$\text{liche: } \begin{array}{c|cc} & 1 & -\omega \\ \hline 1 & 1 & -1 \\ 1 & -1 & -\omega^2 \\ 1 & 1 & 0 \\ 1 & -\omega & 0 \end{array} \begin{array}{c|cc} & -3 & 0 \\ \hline 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{array} \quad \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} -3 & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} -3 & 0 \\ -3 & 0 \end{bmatrix} \Rightarrow \begin{array}{c|cc} & w & w^2 \\ \hline 0 & -3 & 1 \\ 0 & -3 & 4 \\ 0 & -3 & -1 \\ 0 & -3 & -4 \end{array} \begin{bmatrix} -3 & 1 \\ -3 & 4 \\ -3 & -1 \\ -3 & -4 \end{bmatrix} = \begin{bmatrix} -3 & 1 \\ -6 & 8 \\ 5 & -4 \\ -7 & 2 \end{bmatrix}$$

→ 1. pillar

$$\begin{array}{c} \left[ \begin{array}{cc} 3 & -3 \\ 4 & -6 \\ 1 & +5 \\ -4 & -7 \\ 3 & +3 \\ 4 & +6 \\ 1 & -5 \\ -4 & +7 \end{array} \right] \\ + \\ \left[ \begin{array}{cc} 1 & +1 \\ 2 & +8 \\ -3 & -6 \\ 8 & +2 \\ 1 & -1 \\ 7 & -8 \\ -3 & +9 \\ 8 & -2 \end{array} \right] \\ = \\ \left[ \begin{array}{cc} 0 & 2 \\ -2 & -2 \\ 6 & -7 \\ 6 & 10 \\ 6 & 0 \\ 10 & -1 \\ -4 & 1 \\ 3 & 6 \end{array} \right] \end{array}$$

2. Julka

$$\rightarrow \text{graf}(R) = \begin{bmatrix} 0 \\ 4 \\ 9 \\ 9 \\ 0 \\ 4 \\ -4 \\ 1 \end{bmatrix} \quad \checkmark$$

Mötylterý algoritmus = nerekurzivní FFT

→ index se serial podle binärky pořádán

000	1	$\rightarrow$	1	$\omega^0$	2	$\rightarrow$	2	$\omega^0$	3
001	-1	$\nearrow$	1	$\omega^0$	0	$\nearrow$	1	$\omega^0$	4
010	1	$\nearrow$	1	$\omega^0$	1	$\nearrow$	1	$\omega^0$	1
011	0	$\nearrow$	0	$\omega^0$	1	$\nearrow$	1	$\omega^2$	-4
100	1	$\times$	-1	$\omega^0$	f3	$\times$	-1	$\omega^0$	-1
101	0	$\times$	0	$\omega^0$	-1	$\times$	-1	$\omega^0$	-1
110	0	$\searrow$	0	$\omega^0$	0	$\searrow$	0	$\omega^0$	-1
111	0	$\rightarrow$	0	$\omega^0$	0	$\times$	0	$\omega^2$	-1

$$\begin{array}{rcl}
 3 - 3 & = 0 \\
 4 - 3w & = -2 \\
 1 - 3 \cdot w^2 & = 6 \\
 -4 - 3w^3 & = 6 \\
 3 + 3 & = 6 \\
 4 + 3w & = 10 \\
 1 + 3 \cdot w^2 & = 13 \\
 -4 + 3 \cdot w^3 & = 3
 \end{array}$$

→ mögliche Maßnahmen:  $\omega^1 \cdot \omega = 1 \Rightarrow 9 \cdot 2 = 1 \Rightarrow \omega^1 = 9$  ... oder  $\omega = 9, \omega^2 = -4, \omega^3 = -2$

000	0	$\rightarrow$	0	$\rightarrow$	$0+0 = 0$	0
001	4	$\rightarrow$	0	$\rightarrow$	$0-0 = 0$	6
010	9	$\rightarrow$	9	$\rightarrow$	$9-4 = 5$	1
011	9	$\rightarrow$	-4	$\rightarrow$	$9+4 = -4$	w <sup>2</sup>
100	0	$\rightarrow$	4	$\rightarrow$	$4+7 = +6$	
101	4	$\rightarrow$	7	$\rightarrow$	$4-7 = -3$	
110	-4	$\rightarrow$	9	$\rightarrow$	$9+1 = -7$	1
111	1	$\rightarrow$	1	$\rightarrow$	$9-1 = 8$	w <sup>2</sup>

$$\begin{aligned}
 & 0 + 5 = 5 \\
 & 0 - 4w^2 = -1 \\
 & 0 - 5 = -5 \\
 & 0 + 4w^2 = 1 \\
 \\ 
 & -6 - 7 = 4 \\
 & -3 + 8w^2 = -1 \\
 & -6 + 7 = 1 \\
 & -3 - 8w^2 = -5
 \end{aligned}$$

Příklad

$$R(x) = (x^2 - 2x + 1)(x - 2) = x^3 - 4x^2 + 5x - 2 \quad \text{formou } w = \zeta \text{ mod } 12$$

$$\begin{array}{l} \left[ \begin{matrix} 1 & -2 & 1 & 0 \end{matrix} \right] \left[ \begin{matrix} -2 & 1 & 0 & 0 \end{matrix} \right] \begin{array}{l} 1 \\ w \\ w^2 \\ w^3 \end{array} \} \\ \left[ \begin{matrix} 1 & 1 \end{matrix} \right] \left[ \begin{matrix} -2 & 0 \end{matrix} \right] \left[ \begin{matrix} -2 & 0 \end{matrix} \right] \left[ \begin{matrix} 1 & 0 \end{matrix} \right] \begin{array}{l} 1 \\ w^2 \end{array} \} \text{ řešte když chci výhodnějšík} \\ \left[ \begin{matrix} 1 & 1 & -2 & 0 & -2 & 0 & 1 & 0 \end{matrix} \right] \begin{array}{l} 1 \\ 1 \\ 1 \end{array} \\ \Rightarrow \left[ \begin{matrix} 1 & 1 & -2 & 0 & -2 & 0 & 1 & 0 \end{matrix} \right] \begin{array}{l} * \\ 1 \end{array} \end{array}$$

$$\begin{array}{l} \left[ \begin{matrix} 1 & 0 & 0 & 0 & 1 \end{matrix} \right] \rightarrow \text{když } w = w^2 \\ \left[ \begin{matrix} 2 & 0 \end{matrix} \right] \left[ \begin{matrix} -2 & -2 \end{matrix} \right] \left[ \begin{matrix} -2 & -2 \end{matrix} \right] \left[ \begin{matrix} 1 & 1 \end{matrix} \right] \\ \left[ \begin{matrix} -2 & -8 \end{matrix} \right] \left[ \begin{matrix} 1 & 4 \end{matrix} \right] \cdot \left[ \begin{matrix} 1 & w \end{matrix} \right] \rightarrow \text{když } w = w \\ \Rightarrow \left[ \begin{matrix} 0 & -8 & 4 & -8 \end{matrix} \right] \left[ \begin{matrix} -1 & 2 & -3 & -6 \end{matrix} \right] \rightarrow \text{graf P, graf Q} \end{array}$$

$$\Rightarrow \left[ \begin{matrix} 0 & 1 & 5 & 1 \end{matrix} \right] \leftrightarrow \text{výnosobit pro složekách}$$

$$\begin{array}{l} \left[ \begin{matrix} 0 & 5 \end{matrix} \right] \left[ \begin{matrix} 1 & 3 \end{matrix} \right] \begin{array}{l} 1 \\ \frac{-1}{1} \end{array} \dots \bar{\omega} \cdot \omega = 1 \Rightarrow \bar{\omega} = 13, 13 \cdot 4 = 2 \cdot 26 = 2 \cdot 9 = 18 = 1 \\ \left[ \begin{matrix} 5 & -5 \end{matrix} \right] \left[ \begin{matrix} 4 & -2 \end{matrix} \right] \\ \left[ \begin{matrix} 4 & 8 \end{matrix} \right] \cdot \left[ \begin{matrix} 1 & -4 \end{matrix} \right] = \left[ \begin{matrix} 1 & \bar{\omega} \end{matrix} \right] \begin{array}{l} -4 \\ \frac{1}{1} \end{array} \\ \Rightarrow \left[ \begin{matrix} 9 & 5 & 1 & -13 \end{matrix} \right] \longrightarrow \left[ \begin{matrix} -8 & 3 & -16 & 4 \end{matrix} \right] \cdot \frac{1}{4} = \left[ \begin{matrix} -2 & 5 & -4 & 1 \end{matrix} \right] \Rightarrow R(x) = -2 + 5x - 4x^2 + x^3 \end{array}$$

Příklad

Pomocí FFT výnosob  $59 \cdot 24 = 1416$

$$\rightarrow 59 = 9 + 10 \cdot 5 + 0 + 0 \rightarrow (9, 5, 0, 0)$$

$$\rightarrow 24 = 4 + 10 \cdot 2 + 0 + 0 \rightarrow (4, 2, 0, 0)$$

$\rightarrow$  můžu využít velká čísla

$\rightarrow$  až  $2 \cdot 9^2 \Rightarrow$  potřeboval bych větší síleso  $\Rightarrow C$

$$\begin{array}{l} \left[ \begin{matrix} 9 & 5 & 0 & 0 \end{matrix} \right] \left[ \begin{matrix} 4 & 2 & 0 & 0 \end{matrix} \right] \omega = i \\ \left[ \begin{matrix} 9 & 0 \end{matrix} \right] \left[ \begin{matrix} 5 & 0 \end{matrix} \right] \left[ \begin{matrix} 4 & 0 \end{matrix} \right] \left[ \begin{matrix} 2 & 0 \end{matrix} \right] \sim 1, w^2 \Rightarrow 1, -1 \\ \left[ \begin{matrix} 9 & 9 \end{matrix} \right] \left[ \begin{matrix} 5 & 5 \end{matrix} \right] \left[ \begin{matrix} 4 & 4 \end{matrix} \right] \left[ \begin{matrix} 2 & 2 \end{matrix} \right] \\ \left[ \begin{matrix} 5 & 5i \end{matrix} \right] \left[ \begin{matrix} 2 & 2i \end{matrix} \right] \cdot \left[ \begin{matrix} 1 & i \end{matrix} \right] \end{array}$$

$$\left[ \begin{matrix} 14 & 9+5i & 4 & 9-5i \end{matrix} \right] \left[ \begin{matrix} 6 & 4+2i & 2 & 4-2i \end{matrix} \right] \dots y_0 \in \mathbb{R}, y_{\frac{n}{2}} = y_2 \in \mathbb{R}, y_1 = \bar{y}_5$$

$$\Rightarrow \left[ \begin{matrix} 84 & 26+38i & 8 & 26-38i \end{matrix} \right] \bar{\omega} = -i$$

$$\begin{array}{l} \left[ \begin{matrix} 84 & 8 \end{matrix} \right] \left[ \begin{matrix} 26+38i & 26-38i \end{matrix} \right] \sim 1, \bar{\omega}^2 \Rightarrow 1, -1 \\ \left[ \begin{matrix} 92 & 76 \end{matrix} \right] \left[ \begin{matrix} 52 & 76i \end{matrix} \right] \\ \left[ \begin{matrix} 52 & 76 \end{matrix} \right] \cdot \left[ \begin{matrix} 1 & -i \end{matrix} \right] \end{array}$$

$$\Rightarrow \left[ \begin{matrix} 144 & 152 & 40 & 0 \end{matrix} \right] \rightarrow \cdot \frac{1}{4} \rightarrow \left[ \begin{matrix} 36 & 38 & 10 & 0 \end{matrix} \right] \Rightarrow \begin{array}{r} 1000 \\ 380 \\ 36 \\ \hline 1416 \end{array}$$

$$59 \cdot 24 = 1416$$

## Hradlové sítě

Def: hradlo arity je funkce  $f: \Sigma^2 \rightarrow \Sigma$ , kde  $\Sigma$  je konečná abeceda.



## Boolovská hradla ... nad boolovskou abecedou

- binární (arita 2): AND, OR, XOR,  $\leq$  (Implikace)
- unární (arita 1): NOT, Identita
- nulařní (arita 0): 1, 0 ... konstanty

Fakt:  $\exists$  logický výnosek, že AND, OR a NOT jsou univerzální.

Také každé hradlo je možné vyrobit sestavením těchto funkcí.

Lemma: NAND a NOR jsou jediná dvě univerzální binární hradla.

Def: 1, řeďme NAND a NOR mohou být univerzální

$\begin{array}{c cc}  & 0 & 1 \\  \hline  0 & 1 & 0 \\  1 & 0 & 1  \end{array}$	<ul style="list-style-type: none"> <li>abych došlo k negaci, tak potřebujeme <math>(0, 0) \rightarrow 1</math> a <math>(1, 1) \rightarrow 0</math></li> <li>hradlo <math>(0, 1) \rightarrow 0</math> &amp; <math>(1, 0) \rightarrow 0</math> nebo <math>(0, 1) \rightarrow 1</math> &amp; <math>(1, 0) \rightarrow 1</math>,</li> </ul>
	Jinak to je funkce jedné proměnné $\Rightarrow (0, 1) \rightarrow 0 \neq \text{NOR} \quad (0, 1) \rightarrow 1 \neq \text{NAND}$

2, NAND a NOR jsou univerzální

$$\text{NOT}(x) = \text{NOR}(x, x) = \text{NAND}(x, y)$$

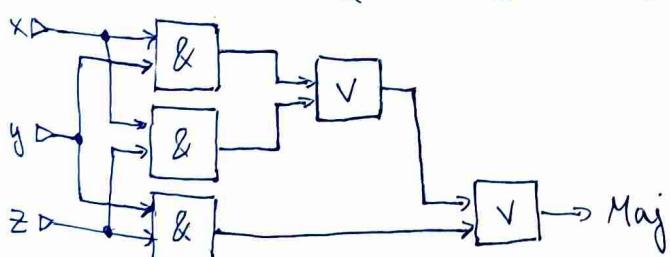
$$\text{OR}(x, y) = \text{NOT}(\text{NOR}(x, y)) = \text{NAND}(\text{NOT}(x), \text{NOT}(y)) \dots x \vee y = \neg(\neg x \wedge \neg y)$$

$$\text{AND}(x, y) = \text{NOT}(\text{NAND}(x, y)) = \text{NOR}(\text{NOT}(x), \text{NOT}(y)) \dots x \wedge y = \neg(\neg x \vee \neg y)$$

■

Majorita ( $x, y, z$ ): počet alebo říkává 2 jedničky 1, jinak 0

0      1      2      3      4       $\leftarrow \text{VRSTVY}$



ekvivalentné

$$\text{Maj}(x, y, z) = (x \wedge y) \vee (x \wedge z) \vee (y \wedge z)$$

💡 Kάždá logická formula lze zapsat jako nějaké hradlo

💡 Ale hradla jsou mnohem mocnější, protože umožňují použití spojitek meruňských vícekrát

Def: Hradlová síť sestává z:

- hradla
- vstupní porty ( $x_1, x_2, \dots, x_m$ )
- výstupní porty ( $y_1, y_2, \dots, y_n$ )
- acyklické projekce vstupů a výstupů
- do & z vstupu & hradla něco nevede
- od & z výstupu & hradla něco nevede
- do vstupních portů síti nic nevede a z výstupních nic nevede

• Výpočet probíhá v tabulech

→ vždy chceme aby výsledky byly věci, co ně mají všechny vstupy

0. tab: ohodnocení vstupní porty síť a konstanty
- i+1. tab: ohodnocení hradla a porty, jejichž vstupy byly ohodnoceny nejdříji v i-tém tabulce.

↳ rozklad síť na vstupy

⇒ v i-tém tabulce paralelně počítá všechno co je v i-tej vrstvě

Čas = # vrstev

maximální arita hradel = 2

prostor = # hradel

⌚ Potřebujeme smerit aritu hradel, jinak je ně sčítatelné v  $O(1)$

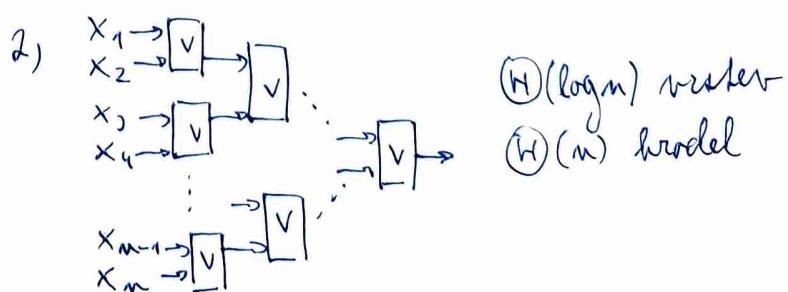
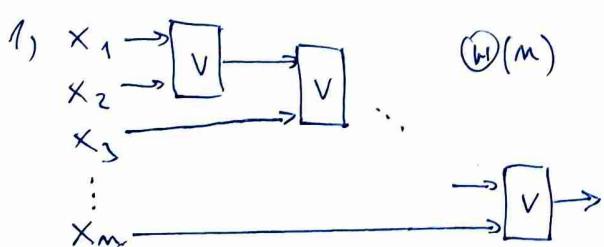
Poznámka: Kombinací obrody - obecná  $\Sigma$

Dvojité obrody -  $\Sigma = \{0, 1\}$

Požadavek: Hradlová síť umí počítat funkce se vstupy dané velikosti  $\Rightarrow$  nemá to alg.

$\Rightarrow$  Budeme chtít, aby  $\exists$  program, který umí efektivně (polynomiálně) generovat hradlové sítě pro danou velikost vstupu.

• m-bit OR( $x_1, \dots, x_m$ )



## Binární scítání n-bit čísel

$\rightarrow \text{XOR} = \text{scissaine' mod } 2$

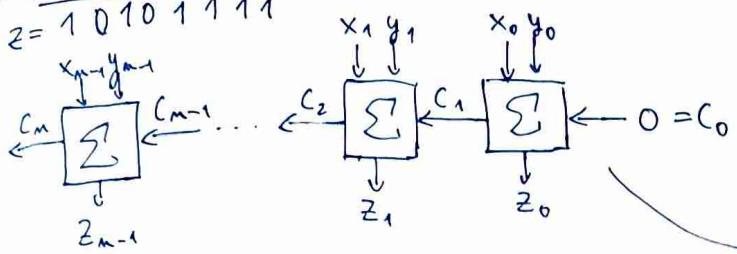
$$\begin{array}{r}
 x = 01110100 \\
 y = 00111011 \\
 \hline
 c_0 = 0 \\
 z = 10101111
 \end{array}$$

$$z_i = x_i \oplus y_i \oplus c_i \quad C_0 = 0$$

$$c_{i+1} = \text{Maj}(x_i, y_i, c_i)$$

$$\zeta_0 = 0$$

$C_m = 1$  period to prefer

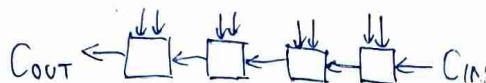


(H)(n) weiter

(1) (n) bradel

1-bit scilacky

## Chorán' Bloch



↳ rafinujeme by rozbíjí  $\mathbb{H} \Rightarrow$  Závislost  $C_{\text{out}}$  na  $C_{\text{IN}}$

- Block 6101:  $x_i$   $y_i$ 

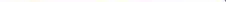
0
0

0
1

1
0

1
1

  
 $C_{out} = 0$      $C_{in}$      $C_{in}$   
 $<$      $<$      $<$

• blok řídký:  bude B rozdělený na podbloky H a D

$$H \left\{ \begin{array}{cccc} * & \overbrace{0 \ 1 \ <} \\ \hline 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ < & 0 & 1 & < \end{array} \right\} \begin{array}{l} \text{horní řada je konstanta} \Rightarrow \text{je jedno co je } D \\ H \text{ reprezentuje} \Rightarrow \text{řídí se podle } D \end{array}$$

⇒ budeme chtít popsat choráin' kanonických bloků = celý vztah, foloring, členění...

0	1	1	1	0	1	0
0	1	1	1	0	1	1
0	1	1	1	0	0	0
0	1	0	1	1	1	1
0	1	0	1	1	1	1

choráni  
kanonicznych (W(log n) razy le-

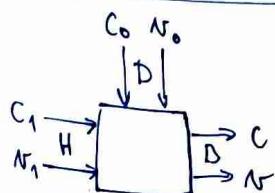
celkem  $\Theta(\log n)$

positive carry  $\Theta(\log n)$  worker

↳ finální XOR je  $O(1)$

Jak založoval svou tabulku do knadla

→ chceme nějaké 'hrátky', co bude  
popisovat chování se 'habulky' \*



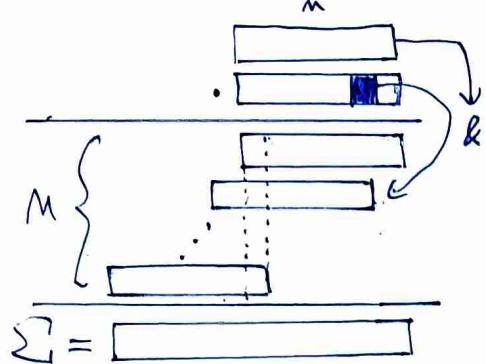
$$C = \text{constant} \quad (1 = \text{true})$$

$\text{N} = \text{value}$  or  $\text{N} = 1$  constant

$\text{const} \Rightarrow \text{konstant}$

$$N = (C_1 = 1) \dot{+} N_1 : N_0 = N_{C_1} = (C_1 \& N_1) \vee (\neg C_1 \& N_0)$$

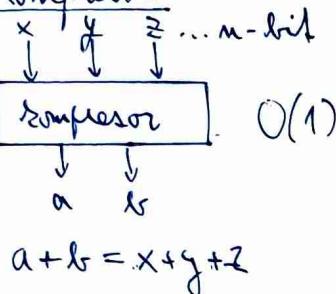
## Násobení čísel



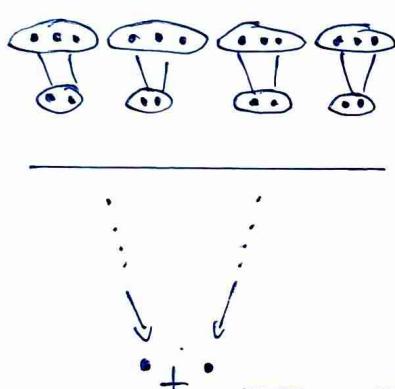
celto by bylo  $\log(n)$  scítání  
 $\Rightarrow$  celkem  $\Theta(\log^2(n))$

paměť:  $\Theta(n^2)$  ... množství jednotek pro scítání

### Kompresor



### Obrázek pro sečení m čísel v log(n)



$$\begin{array}{cccc} x_3 & x_2 & x_1 & x_0 \\ y_3 & y_2 & y_1 & y_0 \\ z_3 & z_2 & z_1 & z_0 \end{array}$$

$$\overline{\quad}$$

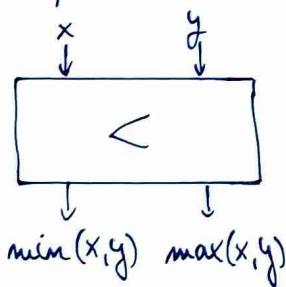
$$\begin{array}{cccc} a_3 & a_2 & a_1 & a_0 \\ b_4 & b_3 & b_2 & b_1 & 0 \end{array}$$

$$\left\{ \begin{array}{l} x_i + y_i + z_i = b_{i+1}, a_i \end{array} \right.$$

→ 2-ciferné čísla

## Násobení $\Theta(\log n)$

### Komparátorová síť



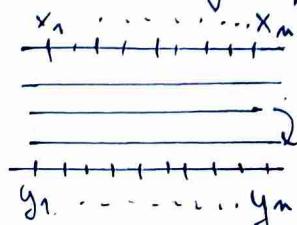
Komparátor ... implementace pomocí

bradla na porovnávání dvou čísel (2-doplňek, odčítače) a potom výsledky zahrádzať pres selektory (if-else)  
 $\hookrightarrow$  hĺbka  $\log(\#bitů)$

→ komparátorová síť se sčítá s komparátorem, nemá nejaká permutační velikost čísel, to je dôležité až pro implementaci

→ cíl je nejaté řazení čísel

⚠️ SÚNO výstupy komparátora se mohou meretivo

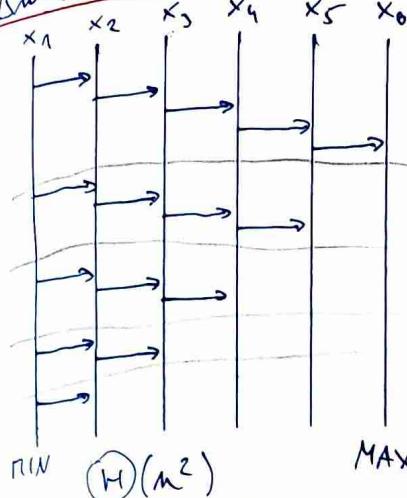


↳ komparátor bere 2 a vráti 2 & číslo na vstupu je stejné ako na výstupu

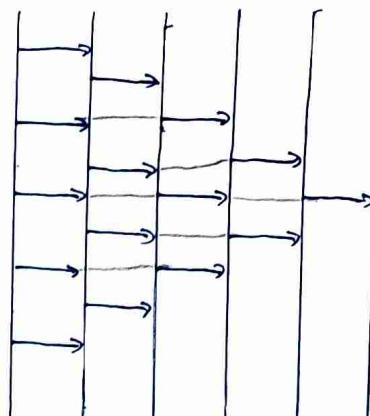
→ to vživem by se stejně robilo

→ na každé vstupní nejaka permutace vstupu

## Bubblesort



příklad řazení 6 čísel  
1 dráha = 1 číslo  
šípka ~ komparátor  
↳ fólie maximum doprava  
← kladická verze  
parallelizace →  
~  $2n$  hladin



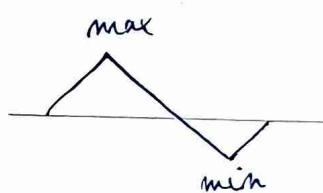
## Mergesort $n=2^k$ (předpokládáme, že čísla jsou mezi sebou rozdílná)

Def: Posloupnost  $x_0, \dots, x_{m-1}$  je

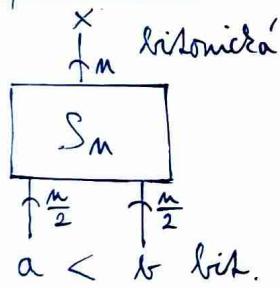
① čistě bitomická  $\equiv \exists \delta : x_0 < x_1 < \dots < x_\delta > x_{\delta+1} > \dots > x_{m-1}$

② bitomická  $\equiv$  má čistě bitomickou rotaci

Když  $\exists l : x_l, x_{l+1}, \dots, x_{l+m-1} \begin{matrix} \text{mod } m \\ \text{mod } m \end{matrix}$  je čistě b.



## Separátor $S_m$



Vstup:  $x_0, \dots, x_{m-1}$  bitomická

Výstup:  $a_0, \dots, a_{m/2-1}$  bitomická

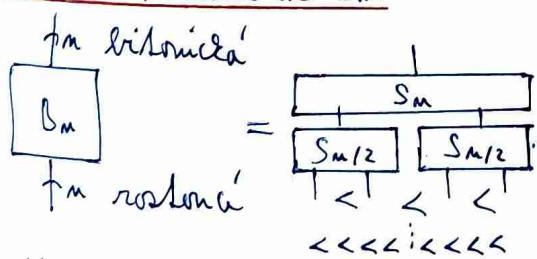
$b_0, \dots, b_{m/2-1}$  bitomická

$\hookrightarrow t_{ij} : a_i < b_j$

fodposloupnosti  $t_{ij} : a_i \in X, b_j \in X$

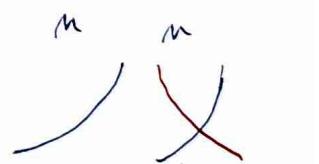
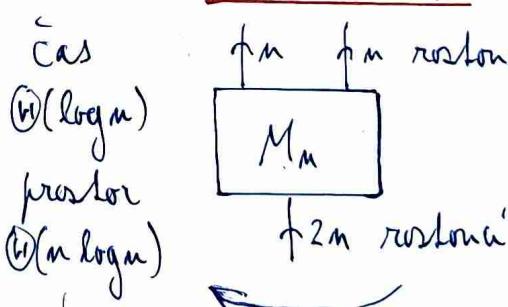
čas  $O(1)$  → ještě  
prostor  $O(n)$  → ještě

## Bitomická řídítka $B_m$



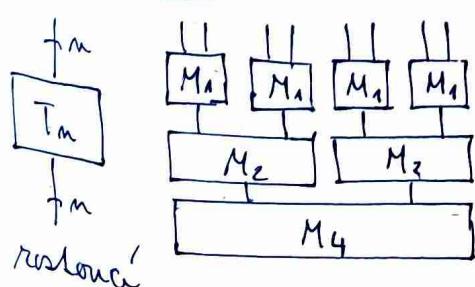
Vstup:  $x_0, \dots, x_{m-1}$  bitomická  
Výstup:  $y_0, \dots, y_{m-1}$  rostoucí

## Slévacka $M_m$



→ složitost  $f(2m)$   
 $\Rightarrow b \cdot f \Rightarrow B_{2m}$

## Řídítka $T_m$



čas  $\Theta(\log^2 m)$   
prostor  $\Theta(m \log^2 m)$  → čas  $\cdot m =$  prostor

↳ na  $k$  hladině (čas) je nejvíce  $m$  hradel  
↳  $m$  drážků  $\Rightarrow$  max.  $m/2$  komparátorů

## • Je mergesort dobrý?

dolní odhad složitosti  
řádový rozdíl výpočtu

$$\Omega(n \log n)$$

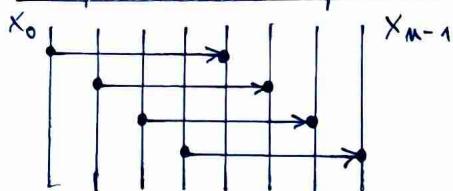
hloubka každé konf. kórové  
následků je  $\Omega(\log n)$

( $\hookrightarrow$  vždy poslední porovnání ... 1 hloubka  $\sim n \log n$ )

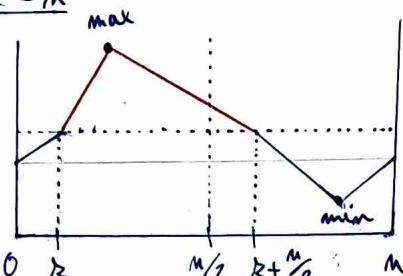
muží se  $\Theta(\log n)$  s obou konstantou

mergesort  $\Theta(\log n)$  je tedy dost dobrý

## • Implementace separátoru $S_m$



Konf.  $X_i \rightarrow X_{m/2+i}$   
pro  $i=0, \dots, m/2-1$



0  $\frac{m}{2}$   $\frac{m}{2} + \frac{m}{2} = m$

DÚNO:  $k < \frac{m}{2}$  ... jinak udelám rotaci o  $\frac{m}{2}$   $\Rightarrow$  konf. kórové fořad na stejné pravé

$\rightarrow$  pro  $i < k$ : výdolí  $\rightarrow$  hora ... neřešací

pro  $i \geq k$ : hora  $\rightarrow$  výdolí ... řešací

HORA:  $m/2$  největších pravé

Souvislá:  $x_k, \dots, x_{k+m/2-1}$

ÚDOLÍ:  $m/2$  nejménších pravé

Souvislá:  $x_{k+m/2}, \dots, x_{k+1}$

$\Rightarrow$  levá pílda: rotace výdolí } a rotace b.f. je b.f. ✓  
pravá pílda: rotace hory }

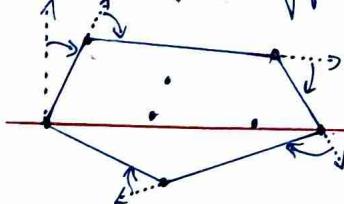
čas  $O(1)$  ... konf. nezávislé  
pravé  $\Theta(n) \sim m/2$  konf.

## • Geometrické algoritmy v rovině

### • Konvexní obal $\Theta(n \log n)$

$\rightarrow$  máme množinu bodů  $X_1, \dots, X_n \in \mathbb{R}^2$

a chceme je co nejefektivněji oploštít



- nejlevější a nejpravější bod tam určitě patří
- inkrementální alg.: má obal pro méně body a objevíme nový bod napravo  $\Rightarrow$  připočítá se do

$n=1$



$n=2$



$n=3$



$n=4$



nebo

$\Theta(n \log n)$

1. Seřídíme body zleva doprava  
a počud  $x_1 = x_2$  zdola nahoru

2.  $H \leftarrow (\text{levý bod})$ ,  $D \leftarrow H$

3. Pro  $b$  zdrobní body  $b$ :

4. Dokud  $|H| \geq 2 \wedge H[-2] H[-1] b$  zdrobní dolera:

Odebereme poslední bod  $\approx H$

Přidáme  $b$  na konec  $H$

5. Podobně pro  $D$  ale zdrobní doprava

6. Vrážíme  $D$  alespoň s  $(H$  pořádkem)

Horní obálka  
Dolní obálka



### Kontrola pravo / levo - hodnoty

$A-C$   $C-B$  determinant

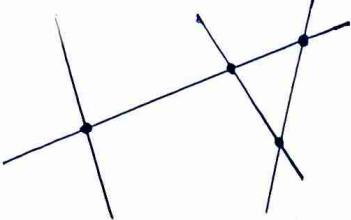
$|B-A| > 0 \Leftrightarrow$

$|D-A| < 0 \Leftrightarrow$

## Obecná poloha bodů

- v všech geometrických úlohách chceme, aby body byly v obecné poloze
  - = aby nenastal nějaký nepříjemný edge-case
- ↳ například u konvexního obalu nechceme aby bylo více bodů nad sebou
  - poté je nemůžeme jednoznačně sestřídit sleva doprava
- ⇒ řešení: otočíme rovinu o nejvíce malé  $\epsilon$ , což rozbité týká obecné polohy
  - ↳ musí být dost malé, aby nevyhnoulo žádnou norme
- $\Rightarrow \vdots \vdots \rightarrow$  něž ji lze sestřídit sleva doprava
  - ~ předním sleva doprava a zadním nahoru
  - ten konvexní obal bude vypadat pořád stejně
- $\Rightarrow$  původní alg funguje i pro neobecnou polohu, ale rádívme  $\rightarrow a \uparrow$
- $\rightarrow$  obecná poloha jsou ty hezké případy
-  Pravděpodobnost o.f. že náhodněmu rozmišlení bodů je 1.

## • Průsečíky úsecík



$\mu := \# \text{průsečíků}$

zkontroloval průsečík řává  $O(1)$  ... linebra

→ cíl je řádovka  $\approx \Theta(m^2)$

→ pro tohle je to optimální

→ chceme něco jako  $O(\log_2 \text{funkce}(m + \mu))$

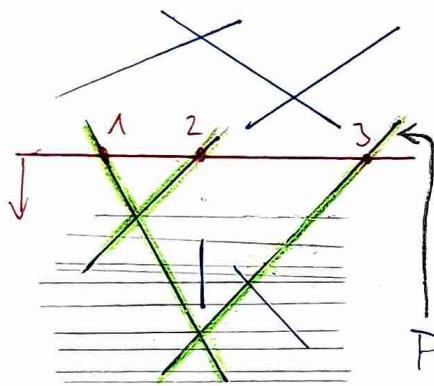
## • obecná poloha → nechceme následující



↳ rodurová  $\Rightarrow ||$  se ramešací p.

→ budeme předpokládat, že to nenastane, v implementaci by se to mělo většinou

## • Zameštání rovin → shora dolů



- děláme diskrétní simulaci spojitého posuvání přímky  
→ konáme se na přímě jen když se deje něco zajímavého

Vzdálosti: Začátek úsecík = horní bod  
Konec úsecík = spodní bod  
Průsečíky úsecík

$P = \text{Průsek} :=$  posloupnost úsecík protínajících zameštání p.  
serazená rova doprava podle  $\times$  průsečíků

$K = \text{Kalendář vzdálostí}$  ... prioritní fronta

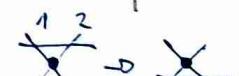
⊗ Těsně před zameštáním průsečíku sousedí protínající se přímky v  $P$

⇒ když spolu úsecík rávou sousedí → naplánujeme průsek \*

↳ když se mezi ně něco vmesává, tak ho odplánujeme,

ale naplánujeme by dva nové → ten starý se pak otevře znova

→ když ohlášíme průsek, tak se pojede úsecík v  $P$  prohodi



→ začátky a konec úsecík můžeme všechny naplánovat na začátku

→ vzdálosti jsou serazeny podle jejich y souřadnice shora dolů

→ odpovidá to pořadí ve kterém je přímka ramešací

(pro 2 stejné y rávou řadíme podle  $\times$  rova doprava ~ otocení roviny s E)

\* začínají spolu sousedit ⇒ podíváme se jestli se protínají

↳ pokud se protínají pod ramešací přímou, tak naplánujeme průsek

## Algoritmus

1.  $P \leftarrow \emptyset \rightarrow$  kancelová fronta  $m \rightarrow \infty$
2.  $K \leftarrow \{\text{začátky a konce úseček}\} \rightarrow$  seřazení podle výšky
3. Dokud není  $K$  prázdný,  
odebiráme nejvyšší událost
4. Záčátek  $\Rightarrow$  přidáme úsečku do  $P$   $\rightarrow$  vložíme nejvýšší 3 sousednosti
5. Konec  $\Rightarrow$  odebereme úsečku z  $P$   $\rightarrow$  1 možnost, 2 přidání
6. Průsečík  $\Rightarrow$  nahlašíme na výšku  $\rightarrow$  3 sousednosti
7. Pro každou sousednost  $v P$   $\rightarrow$  jednu smačka a zase vložení  
procházíme úsečky v  $P$   $\rightarrow$  nejvýšší 6 sousednosti
8.  $\downarrow$  proplácíme dotícné průsečíky v  $K$

## Reprezentace

- Kalendář - prioritní fronta  $\Rightarrow$  haldy nebo BVS  
 ↳ max. 3m událostí najednou (začátky, konce, průsečíky v průřezu)  
 $\Rightarrow$  doba 1 operace je  $O(\log n)$ , paměť  $O(n)$
- Průřez - chci aby můl řešit co je nalevo a napravo od dané úsečky  
 $\Rightarrow$  BVS s úsekami jako klici  $\rightarrow$  máj lineární uspořádání  
 $a < b \equiv (a \wedge \text{ram. p.}) \neq \text{vlevo od } (b \wedge \text{ram. p.})$   $\xrightarrow{x_1 < x_2}$  z. f.  
 $\rightarrow$  y souř. ram. p. = y souř. aktuální události  
 $\Rightarrow$  max. n úseček  $\Rightarrow O(\log n)$  čas/operace, paměť  $O(n)$

## Složitost

$$\# \text{událostí} = 2m + p \rightarrow \text{na 1 událost } O(1) \text{ operací } \circ P \in K$$

$$\Rightarrow \# \text{operací} \in O(m + 2m + p) = O(m + p) \Rightarrow \text{čas } \underbrace{O((m+p) \cdot \log n)}_{\text{prostor } O(n)}$$

initializace  $K$

umí se  $O(n \log m + p)$

pro  $p = m^2$   
 hoří mezi hrubou silou

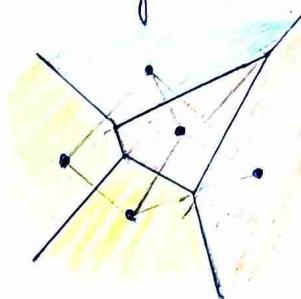
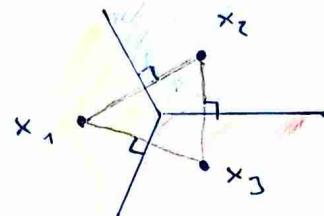
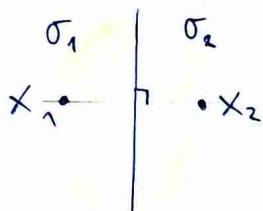
## Voroného diagram

místa  $X = \{x_1, \dots, x_m\} \subseteq \mathbb{R}^2$

→ Eukl. metrika

oblasti  $\Omega_1, \dots, \Omega_m \in \mathbb{R}^2$ ,  $\Omega_i := \{y \in \mathbb{R}^2 \mid \forall j: d(y, x_i) \leq d(y, x_j)\}$

↳  $\Omega_i$  je je to  $x_i$  blíž než  $x_j$  jakémakoli jinému místu



↳ hranice patří do obou oblastí

⊗ oblasti jsou určeny polarorovinami danými přímkami oddělujícími místa

⇒ průnik polarorovin je nějaký souběžný konvexní mnohoúhelník  
↳ mohou být otevřené do nekonečna

⇒ oblasti se umí najít v čase  $O(m \log n)$

## Localizace bodu

→ máme V. d., dostaneme bod → ? do které oblasti patří

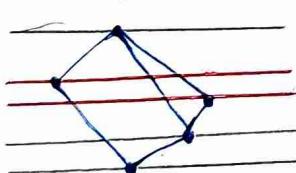
DS na rozdod roviny na mnohoúhelníky (ne nutně konvexní)

→ To řeší mnohoúhelníky mohou být otevřené nevadí

↳ vždy se dají uvrátit do nějaké krabici a mimo k. bin. vzhled moci paprsky

Dotaz: Do jaké oblasti patří zadaný bod?

↳ pro hranici odpoví stereonoklik oblasti



Rovinu rozsekáme na pásky → v pásu



→ pro k úseku si pamatuje jaká oblast je nalevo a napravo \*

⇒ binárně vyhledáme páš a pak oblast v něm ⇒ čas  $O(\log n)$

→ pro k páš si musíme pamatovat úseky v něm ⇒ paměť  $O(n^2)$

⇒ chceme nějak slepit paměť

⊗ pásky jsou vlastně průřez po rozmetení v tom alg. pro průseče úseček

↳ bodu co definuje ten páš

\* je to rovinný graf, řádky # úseček  $\in O(n)$

## semi-persistentní DS

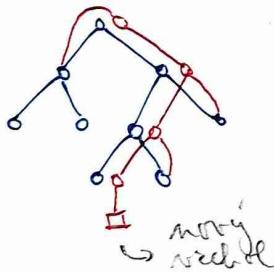
- pamatuje si svou historii ~ Bst
- zápis  $\Rightarrow$  vytvoření nové verze, umíme vyhledat v historii podle ID
- dotaz  $\Rightarrow$  dostaneme ID verze  $\rightarrow$  průřec v pásmu
- budeme si pamatovat všechny průřezy co veníkají při zámelání rovine
- uděláme semi-pers. BVS co má  $O(\log n)$  čas / operace,  $O(\log n)$  paměť / verze
- Konstrukce stromu

- zámeláme rovinu: # průřešek = # bodů =  $n \Rightarrow O(n)$  operací s průřechem
- $\Rightarrow$  čas konstrukce  $O(n \log n)$
- $\Rightarrow O(n)$  verzí  $\Rightarrow$  prostor  $O(n \log n)$   $\rightarrow$  umí se i prostor  $O(n)$
- $\Rightarrow$  dotaz v čase  $O(\log n)$

↳ bin. r. přes pásky  $\rightarrow$  sázám ID průřecem  $\Rightarrow$  konkrétní BVS  $\Rightarrow$  dotaz  $O(\log n)$

## Semipersistentní BVS

- pravidlo: nemí se změnit to, co už je zapsáno



- $\rightarrow$  persistence kopírováním cest
- $\rightarrow$  když operace si kopíruje cestu k dotírenímu vrcholu a vytvoří nový kořen (ID verze)
  - $\hookrightarrow$  pointer na podstromy jiných verzí - ale ty se už nezmění
- $\rightarrow$  pro rozumnení blouben vytvoříme  $O(\log n)$  nových vrcholků

- $\rightarrow$  vytvoření: AVL strom méní pouze vrcholy na cestě a v jejich nejbližším okolí
  - $\rightarrow$  počet pouze  $O(\log n)$  nových vrcholků na verzi

- $\rightarrow$  pro každou verzi si v nejlepším poli pamatuji pointer na kořen jeho stromu

## • Trídy složitosti algoritmů

Def: Rozhodovací problém je funkce  $\alpha: \{0,1\}^* \rightarrow \{0,1\}$ .

→ konečná posloupnost 0 a 1

⊗ Colovit lze rozdělat do posloupnosti bitů.

↪ ANO / NE

Příklad: Bi. f. graf a  $\delta \in \mathbb{N}$

000...0	1	$m$	$\epsilon$	matice sousednosti
$s$	$s$	$s$		$m^2$

→ jednoznačné řešení

→ současní problému je odpovědět NE na vstup ve špatném tvare

Def: Problem A je převoditelný na problem B, psíme  $A \rightarrow B$  nebo  $A \leq_p B$   
 $\equiv \exists f: \{0,1\}^* \rightarrow \{0,1\}^*$  takže

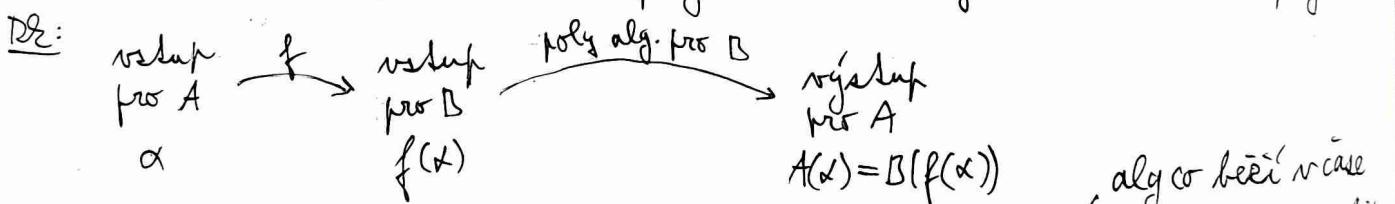
①  $\forall \alpha \in \{0,1\}^*: A(\alpha) = B(f(\alpha))$  ... řešíme A pomocí B

②  $f(\alpha)$  lze spočítat polynomické moci  $|\alpha|$   
 ↳ přenod neboli redukce

B je alespoň tak  
řešitelný A

Příklad:  $\exists$  parování velikosti  
alespoň  $\delta \in \mathbb{N} \wedge G$ ? → Existuje také velikostí alespoň  
 $\delta \in \mathbb{N}$  v síti S?

Lemma: Nechť  $A \rightarrow B$ , B řešitelné v poly. čase. Potom A je také řešitelné v poly. čase.



$\exists c: f(\alpha)$  lze spočítat v  $O(m^c)$  ...  $m=|\alpha|$

$\exists d: B(f(\alpha))$  lze spočítat v  $O(m^d)$  ...  $m=|f(\alpha)| \in O(m^c)$

$\Rightarrow A(\alpha)$  lze spočítat v  $O(m^c + m^d) = O(m^c + m^{c+d}) = O(m^{cd})$  → poly. čas

## • Vlastnosti relace převoditelnosti

①  $A \rightarrow A$  ... reflexivní

②  $A \xrightarrow{f} B \wedge B \xrightarrow{g} C \Rightarrow A \xrightarrow{f \circ g} C$  ... tranzitivní

③  $\exists A, B: A \rightarrow B \wedge B \rightarrow A$  ... např. A: má vstup libovolně dělen } nemá k němu řešení  
 B: má vstup sudov dělen }

④  $\exists A, B: A \rightarrow B \wedge B \rightarrow A$  ... např.  $f: A = \{0,1\} \rightarrow B = \{0,1\}$  } Edgby to bylo např. tak ne lineární

⊗ → je částečně kvazispořádání,

• některé pravidly jsou mezi sebou ekvivalence  $A \leftrightarrow B$ , což mám dává následující  
 ekvivalence → a mezi tímto následuje, že to je částečně uspořádání

## Klída velikosti $k$

Vstup: neorientovaný graf  $G$ ,  $k \in \mathbb{N}$

Výstup:  $\exists A \subseteq V: |A| \geq k \text{ & } \forall u, v \in A: uv \in E$



## Necívila' možností velikosti $k$

Vstup: neorientovaný graf  $G$ ,  $k \in \mathbb{N}$

Výstup:  $\exists A \subseteq V: |A| \geq k \text{ & } \forall u, v \in A: uv \notin E$



## Nz Mma $\Leftrightarrow$ klíča

$\rightarrow$  stačí provést vlastnost být hravou

$\rightarrow$  přirozená funkce je hravový doplněk grafu

} stejná přirozená funkce obíma směry

## SAT splnitelnost

Vstup: booleská formula v CNF = konjunkce klávek

probl. - p

Výstup:  $\exists$  ohodnocení proměnných, co lze formuli splnit? = splnitelné ohodnocení

3-SAT ... navíc  $\forall$  klávek obsahuje nejméně 3 literály

## 3-SAT $\rightarrow$ SAT $\rightarrow$ zjednodušit

$\rightarrow$  jde přirozená funkce nestaci' obyčejná identita, protože

3-SAT ( $p_1 \vee p_2 \vee p_3 \vee p_4$ ) = NE ... nevalidní vstup } musí se provést kontrola  
SAT ( $p_1 \vee p_2 \vee p_3 \vee p_4$ ) = ANO } validity vstupu

Platí: (speciální případ problému A)  $\xrightarrow{\text{"identita"}}$  A

## SAT $\rightarrow$ 3-SAT

$\nearrow$  nová proměnná

$$\overbrace{(\alpha \vee \beta)}^k \Leftrightarrow (\alpha \vee x) \wedge (\beta \vee \neg x) \dots \text{esplnitelné}$$

$\begin{matrix} \uparrow & \uparrow \\ 1 & 2 \end{matrix} \quad \begin{matrix} \uparrow & \uparrow \\ 3 & 2-1 \end{matrix}$

$\Rightarrow \# \text{ literálů} \text{ v klávce klesá o 1} \Rightarrow$  štípeme dál, celkově pol. čas

Takovému kódování se říká gadgets

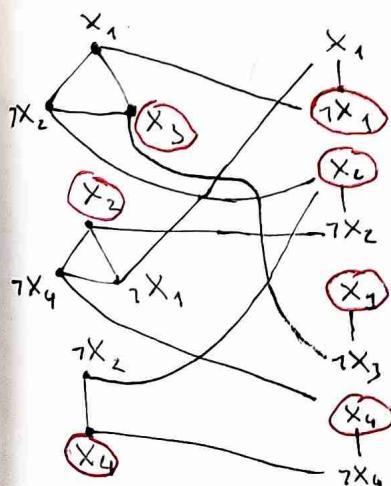
klávek  $\rightarrow$  klíčy o velikosti dané klávek

literály  $\rightarrow$  navíc vedené irany ( $l \in \text{klávek} \rightarrow (\bar{l} \in \text{literál})$ )

$k = \# \text{ klávek}, l = \# \text{ literálů}$

## 3-SAT $\rightarrow$ Nz Mma

klávek literály



platí: formule je splnitelná  $\Leftrightarrow$  v grafu  $\exists$  Nz Mma velikosti  $\geq k+l$

$\Leftarrow:$   $\forall$  klávek je splněna vybraným literálem

$\forall$  literálů jsem zvolil jeho ohodnocení

$\Rightarrow:$  všechnu si nájde splnitelné ohodnocení a ohodnotíme podle něho literály  $\Rightarrow$   $\forall$  klávek  $\exists$  alejší 1 kládající literál, tak jej vyberu (nenastane  $x_1 - \bar{x}_3$ )

Platí: tady sladně nabylo smyslu, aby to byl 3-SAT.

## $N_2 M_{\text{ma}} \rightarrow \text{SAT}$

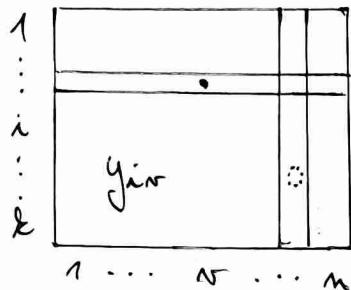
DÚNO  $V(G) = \{1, 2, \dots, n\}$ ,  $N_2 M_{\text{ma}}$  velikostí 2

•  $x_1, \dots, x_m \rightarrow x_m = 1 \Leftrightarrow m \in N_2 M_{\text{ma}}$

•  $y_{i,r} \text{ pro } 1 \leq i \leq 2, 1 \leq r \leq m \rightarrow \text{fornací proměnné}$   
 $\gamma(x_m \wedge x_r)$

1)  $\forall u, v \in E: (\gamma x_m \vee \gamma x_r) \dots \text{nenachádza } x_m = x_r = 1 \Rightarrow \text{jedno nezávisle}$

2) ještě potřebujeme velikost k  $\rightarrow$  vrcholy v něj si seřadíme



$y_{i,r} = 1 \Leftrightarrow \text{vrchol } r \text{ je } i\text{-tý v } N_2 M_{\text{ma}}$

•  $r$  řádku je právě jedna 1

•  $r$  sloupcu je ji nejrytí jedna 1

$$\left\{ \begin{array}{l} \# i, j, r: \gamma(y_{i,r} \wedge y_{j,r}) \sim (\gamma y_{i,r} \vee \gamma y_{j,r}) \\ \# i, m, r: \gamma(y_{i,m} \wedge y_{i,r}) \sim (\gamma y_{i,m} \vee \gamma y_{i,r}) \dots \text{nejrytí 1} \\ \# i: (\gamma y_{i,1} \vee \gamma y_{i,2} \vee \dots \vee \gamma y_{i,m}) \dots \text{alejpoň 1} \end{array} \right\} \text{právě 1}$$

$\rightarrow \exists$  možností k vrcholu

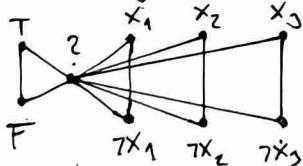
$\Rightarrow \# i, r: y_{i,r} \Rightarrow x_r \sim (\gamma y_{i,r} \vee x_r) \rightarrow \text{fónd je něco v k' možnosti,}\nolimits$   
 $\text{takto je nezávisle}$

## $\exists$ -SAT $\rightarrow \exists$ -Obrazitelnost

barvy: T true, F false, ? filler barva

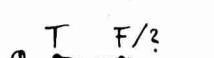
⊗ trojúhelník  $\Delta$  obsahuje všechny 3 barvy

literály



T/F

Elaucele:  $(a_1 \vee a_2 \vee a_3)$  nelze splnit  $\Leftrightarrow a_1 = a_2 = a_3 = \overbrace{\text{F}}$



a1, F  
a2, F

a1, T  
a2, F

a1, F  
a2, T

a1, T  
a2, T

a1, F  
a2, F

a1, T  
a2, F

a1, F  
a2, T

a1, T  
a2, T

a1, F  
a2, F

a1, T  
a2, F

a1, F  
a2, T

a1, T  
a2, T

a1, F  
a2, F

a1, T  
a2, F

a1, F  
a2, T

a1, T  
a2, T

a1, F  
a2, F

a1, T  
a2, F

a1, F  
a2, T

a1, T  
a2, T

a1, F  
a2, F

a1, T  
a2, F

a1, F  
a2, T

a1, T  
a2, T

a1, F  
a2, F

a1, T  
a2, F

a1, F  
a2, T

a1, T  
a2, T

a1, F  
a2, F

a1, T  
a2, F

a1, F  
a2, T

a1, T  
a2, T

a1, F  
a2, F

a1, T  
a2, F

a1, F  
a2, T

a1, T  
a2, T

a1, F  
a2, F

a1, T  
a2, F

a1, F  
a2, T

a1, T  
a2, T

a1, F  
a2, F

a1, T  
a2, F

a1, F  
a2, T

a1, T  
a2, T

a1, F  
a2, F

a1, T  
a2, F

a1, F  
a2, T

a1, T  
a2, T

a1, F  
a2, F

a1, T  
a2, F

a1, F  
a2, T

a1, T  
a2, T

a1, F  
a2, F

a1, T  
a2, F

a1, F  
a2, T

a1, T  
a2, T

a1, F  
a2, F

a1, T  
a2, F

a1, F  
a2, T

a1, T  
a2, T

a1, F  
a2, F

a1, T  
a2, F

a1, F  
a2, T

a1, T  
a2, T

a1, F  
a2, F

a1, T  
a2, F

a1, F  
a2, T

a1, T  
a2, T

a1, F  
a2, F

a1, T  
a2, F

a1, F  
a2, T

a1, T  
a2, T

a1, F  
a2, F

a1, T  
a2, F

a1, F  
a2, T

a1, T  
a2, T

a1, F  
a2, F

a1, T  
a2, F

a1, F  
a2, T

a1, T  
a2, T

a1, F  
a2, F

a1, T  
a2, F

a1, F  
a2, T

a1, T  
a2, T

a1, F  
a2, F

a1, T  
a2, F

a1, F  
a2, T

a1, T  
a2, T

a1, F  
a2, F

a1, T  
a2, F

a1, F  
a2, T

a1, T  
a2, T

a1, F  
a2, F

a1, T  
a2, F

a1, F  
a2, T

a1, T  
a2, T

a1, F  
a2, F

a1, T  
a2, F

a1, F  
a2, T

a1, T  
a2, T

a1, F  
a2, F

a1, T  
a2, F

a1, F  
a2, T

a1, T  
a2, T

a1, F  
a2, F

a1, T  
a2, F

a1, F  
a2, T

a1, T  
a2, T

a1, F  
a2, F

a1, T  
a2, F

a1, F  
a2, T

a1, T  
a2, T

a1, F  
a2, F

a1, T  
a2, F

a1, F  
a2, T

a1, T  
a2, T

a1, F  
a2, F

a1, T  
a2, F

a1, F  
a2, T

a1, T  
a2, T

a1, F  
a2, F

a1, T  
a2, F

a1, F  
a2, T

a1, T  
a2, T

a1, F  
a2, F

a1, T  
a2, F

a1, F  
a2, T

a1, T  
a2, T

a1, F  
a2, F

a1, T  
a2, F

a1, F  
a2, T

a1, T  
a2, T

a1, F  
a2, F

a1, T  
a2, F

a1, F  
a2, T

a1, T  
a2, T

a1, F  
a2, F

a1, T  
a2, F

a1, F  
a2, T

a1, T  
a2, T

a1, F  
a2, F

a1, T  
a2, F

a1, F  
a2, T

a1, T  
a2, T

a1, F  
a2, F

a1, T  
a2, F

a1, F  
a2, T

a1, T  
a2, T

a1, F  
a2, F

a1, T  
a2, F

a1, F  
a2, T

a1, T  
a2, T

a1, F  
a2, F

a1, T  
a2, F

a1, F  
a2, T

a1, T  
a2, T

a1, F  
a2, F

a1, T  
a2, F

a1, F  
a2, T

a1, T  
a2, T

a1, F  
a2, F

a1, T  
a2, F

a1, F  
a2, T

a1, T  
a2, T

a1, F  
a2, F

a1, T  
a2, F

a1, F  
a2, T

a1, T  
a2, T

a1, F  
a2, F

a1, T  
a2, F

a1, F  
a2, T

a1, T  
a2, T

a1, F  
a2, F

a1, T  
a2, F

a1, F  
a2, T

a1, T  
a2, T

a1, F  
a2, F

a1, T  
a2, F

a1, F  
a2, T

a1, T  
a2, T

a1, F  
a2, F

a1, T  
a2, F

a1, F  
a2, T

a1, T  
a2, T

a1, F  
a2, F

a1, T  
a2, F

a1, F  
a2, T

a1, T  
a2, T

a1, F  
a2, F

a1, T  
a2, F

a1, F  
a2, T

a1, T  
a2, T

a1, F  
a2, F

a1, T  
a2, F

a1, F  
a2, T

a1, T  
a2, T

a1, F  
a2, F

a1, T  
a2, F

a1, F  
a2, T

a1, T  
a2, T

a1, F  
a2, F

a1, T  
a2, F

a1, F  
a2, T

a1, T  
a2, T

a1, F  
a2, F

a1, T  
a2, F

a1, F  
a2, T

a1, T  
a2, T

a1, F  
a2, F

a1, T  
a2, F

a1, F  
a2, T

a1, T  
a2, T

a1, F  
a2, F

a1, T  
a2, F

a1, F  
a2, T

a1, T  
a2, T

a1, F  
a2, F

a1, T  
a2, F

a1, F  
a2, T

a1, T  
a2, T

- 3,3-SAT ... navíc  $\#$  proměnná je v nejvýše 3 klauzulech
- 3,3-SAT  $\rightarrow$  3-SAT pomocí identity  $\sim$  kontrolou validity vstupu
- 3-SAT  $\rightarrow$  3,3-SAT
  - $\rightarrow$  nechtě  $x$  je proměnná s  $\ell > 3$  výskyty
  - $\Rightarrow$  pro  $\#$  výskyt nové proměnné  $x_1, \dots, x_s$  a  $i$ -tý výskyt  $x$  nahradime  $x_i$
  - navíc přidáme nové klauzule  $x_1 \Rightarrow x_2, x_2 \Rightarrow x_3, \dots, x_s \Rightarrow x_1$ ,
  - což nám zajiší, že  $v(x_1) = v(x_2) = \dots = v(x_s)$
- $$\begin{array}{l} x_3 \Rightarrow x_4 \sim \neg x_3 \vee x_4 \\ x_4 \Rightarrow x_5 \sim \neg x_4 \vee x_5 \end{array} \left. \begin{array}{l} \# \text{ nové proměnné má 3 výskyty} \\ \text{a alespoň 1 je } \oplus \text{ a jeden } \ominus \end{array} \right\}$$

- 3,3-SAT\* ... navíc  $\#$  literál nejvýše 2-krát

- 3,3-SAT  $\rightarrow$  3,3-SAT\*

$\rightarrow$  pokud mám nějaký literál více než 2-krát, tak pro odformulování proměnnou můžeme stejnou transformaci jako předtím

### 3D-párování

HODINY  
DÍVKY  
VOKY

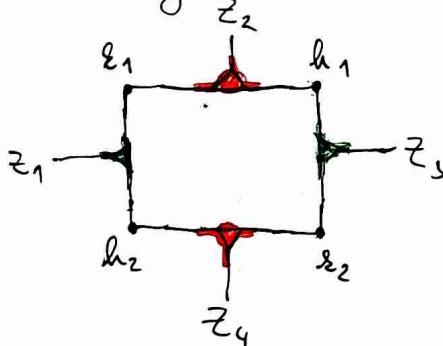
PREFERENCE

Vstup: Konečné množiny  $K, H, Z$  + množina  $T \subseteq K \times H \times Z$

Výstup:  $\exists T' \subseteq T$  t. s.  $\#$  prvků  $K, H, Z$  je v první 1 projekci  $\# T'$ .

míténe želají  
míténe mít?

### literály

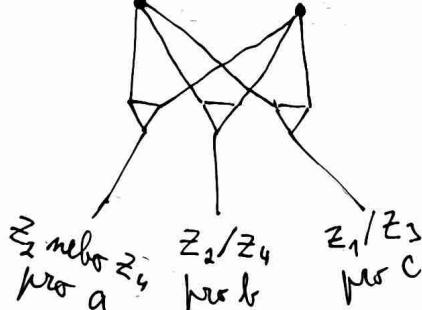


- 0 ... volné  $z_1, z_3$
- 1 ... volné  $z_2, z_4$

### klauzule

$$a \vee b \vee \neg c$$

$z_1$        $h$



- $z_1$  nebo  $z_4$  pro  $a$
- $z_2$  nebo  $z_3$  pro  $b$
- $z_1$  nebo  $z_3$  pro  $c$

3D párování  $\Leftrightarrow$  splnitelné

- $\Leftarrow$ : pro proměnné ohodnotíme podle zadání pro literály
- $\hookrightarrow$  mi dává 2 volná zářítky
- $\hookrightarrow$  každá klauzula je splněna některým z těchto zářítek
- $\hookrightarrow$  zářítko jenom 2  $\Rightarrow$  3,3-SAT\*

\*  $\Leftarrow$ : g. pro literály spárujeme podle ohodnocení proměnných

- $\rightarrow$  klauzule je splněna nějakým literálem  $\Rightarrow$  alespoň 2 volná zářítky
- $\rightarrow$  literál použitý nejvýše 2-krát  $\Rightarrow$  2 zářítky nám stačí

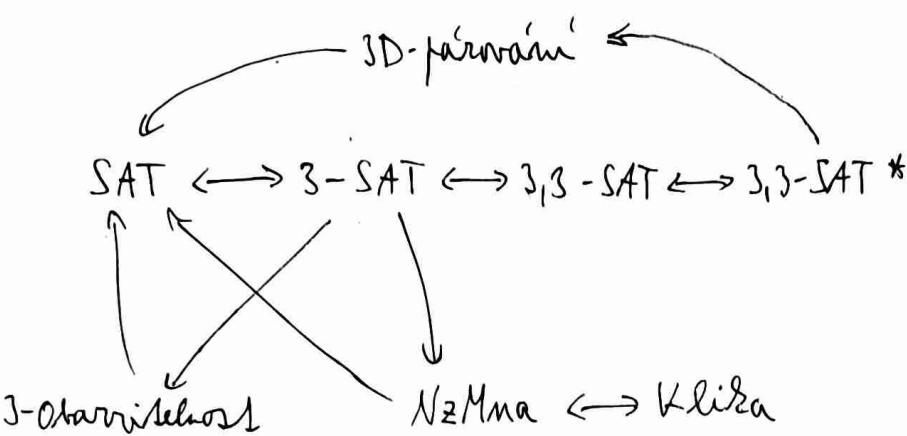
! Dělší vzdálost mezi zářítky ! ...  $\#$  klauzule odpovídají jen 1 zářítku

$\rightarrow (2 \cdot \# \text{ proměnných} - \# \text{ klauzul})$  volných zářítek

$\Rightarrow$  přidáme kolik páru univerzálních mimošlu zářítek

$\hookrightarrow$  každému páru se libí všechna zářítky

## Co ně můžeme převést?



## Třídy problémů

Def: Třída problémů  $P$ . Problem  $L \in P \equiv$

$\exists A$  algoritmus,  $\exists p$  polynom:  $\forall x: A(x) = L(x)$  &  $A(x)$  doběhne do  $p(|x|)$  kroků.

Def: Třída problémů  $NP$ . Problem  $L \in NP \equiv$

$\exists V \in P$ , verifikátor

$\exists g$  polynom  $\leftarrow$  omezení délky certifikátu

$\forall x: L(x) = 1 \Leftrightarrow \exists y$  certifikát

$$|y| \leq g(|x|) \quad \& \quad V(x, y) = 1$$

důkaz

certifikát nemá  
prázdný

$\Rightarrow L \in NP$ , pokud je možné rychle zkontrolovat jeho nádajné řešení

$\rightarrow$  například mají klika je třídy, ale zkontrolovat, jestli je dana' možnost klika je jednoduché. Certifikát by v tomto případě byl ta klika. A verifikátor by byl alg. co s možností rozhodne, zda to je klika.

$P \subseteq NP$ ,  $SAT \in NP$ . ?  $P = NP$ ?

Def: Problem  $L$  je NP-těžký  $\equiv \nexists K \in NP: K \rightarrow L$ .

Def: Problem  $L$  je NP-úplný  $\equiv$  je NP-těžký &  $L \in NP$ .

Lemma: Pokud  $L \in P$  je NP-těžký, potom  $P = NP$ .

Dů: Nechť  $K \in NP$ .  $K \rightarrow L$  &  $L \in P \Rightarrow K \in P$ . ■

Lemma: Nechť  $K, L \in NP$ ,  $K \rightarrow L$ ,  $K$  je NP-úplný, pak  $L$  je NP-úplný.

Dů: Nechť  $M \in NP$ .  $M \rightarrow K \rightarrow L \Rightarrow M \rightarrow L$ . ■

$L \in P \equiv$  3folygonální algoritmus

Věta (Cook, Levin): SAT je NP-úplný.

$B_m$  je vždy poly. velká

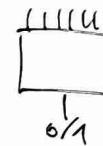
Lemma:  $\forall L \in P \exists A$  polynomiální alg. t.ž.

$A(m)$  je hrodlova síť  $B_m$  s  $m$  vstupy a 1 výstupem řešící  $L$  pro vstupy délky  $m$ .



Důkaz: Neformálně: Pro  $L$  existuje nějaký polynomiální alg. který lze spustit na počítací. Počítací je určitě vlastně velká hrodlova síť, co se méní v čase (jsou tam nějaké obrody co si pomožou) až do doložení toho.  
→ Prvně tuto síť očerpívajeme v každém kroku výstupy záběle měnících se obrody zobražujeme vždy do téže delší síť.  
⇒ Takto získáme polynomiálně mnoho polynomiálně velkých, projevujících síť.  
⇒ To je rada nějaká polynomiálně velká síť. ■

Věta: Ohrodový SAT je NP-úplný.



Důkaz: Ohrodový SAT je problém, kde dosudaneme hrodlova síť →

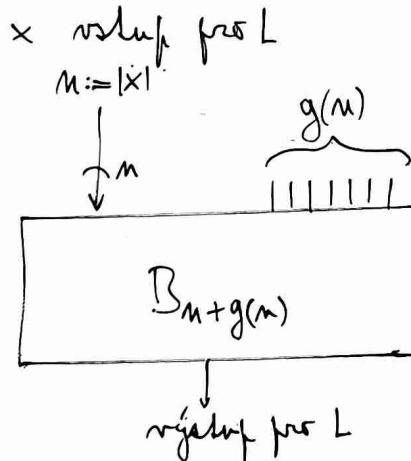
Výstup:  $\exists$  ohodnocení vstupních portů, aby byl výstup 1?

$\Rightarrow$  Nechť  $L \in NP$ , V verifikátor, g ohodnocení délky verifikace dle definice NP.

Díky chceme certifikát délky právě  $g(\text{vstup})$ .

↳ kratejším certifikátu lze dát padding [0000...01 | Actual certifikát]

Plán: chceme ukázat  $L \rightarrow$  Ohrodový SAT. ... pomocí Lemmatu myzdříme obrov



Předchozí lemma nám garantuje polynomiálně velkou síť, co můžeme využít v pol. čase.  
⇒ Ta síť simuluje  $V$  a víme  $V \in P$ .

- zadělávajeme vstup  $x_1, \dots, x_m$   
Ohrodový SAT vydá 1

$\exists$  ohodnocení těch vstupních  $g(m)$  portů t.ž.  
výstup té síťe =  $V(\text{vstup}, \text{ohodnocení}) = 1$

ohodnocení = certifikát

$\exists$  řešení problému  $L$  pro tento vstup ■

Lemma: Ohrodový SAT  $\rightarrow$  SAT v CNF

Důkaz: ① převédeme obvod aby byl celý z NORu... máme jen unární a binární hrodla  
② zavedeme proměnné pro vstupní porty a konstanty → určitě polynomiálně jde  
③ zavedeme proměnné pro výstupy hrodel

$x \rightarrow$  NOR  $\rightarrow z$   
 $y \rightarrow$   
 $00 \rightarrow 1$   
 $11 \rightarrow 0$

$$\begin{aligned} \neg x \wedge \neg y &\Rightarrow z \sim (\neg x \vee \neg y) \\ x &\Rightarrow \neg z \sim (\neg x \vee \neg z) \\ y &\Rightarrow \neg z \sim (\neg y \vee \neg z) \end{aligned}$$

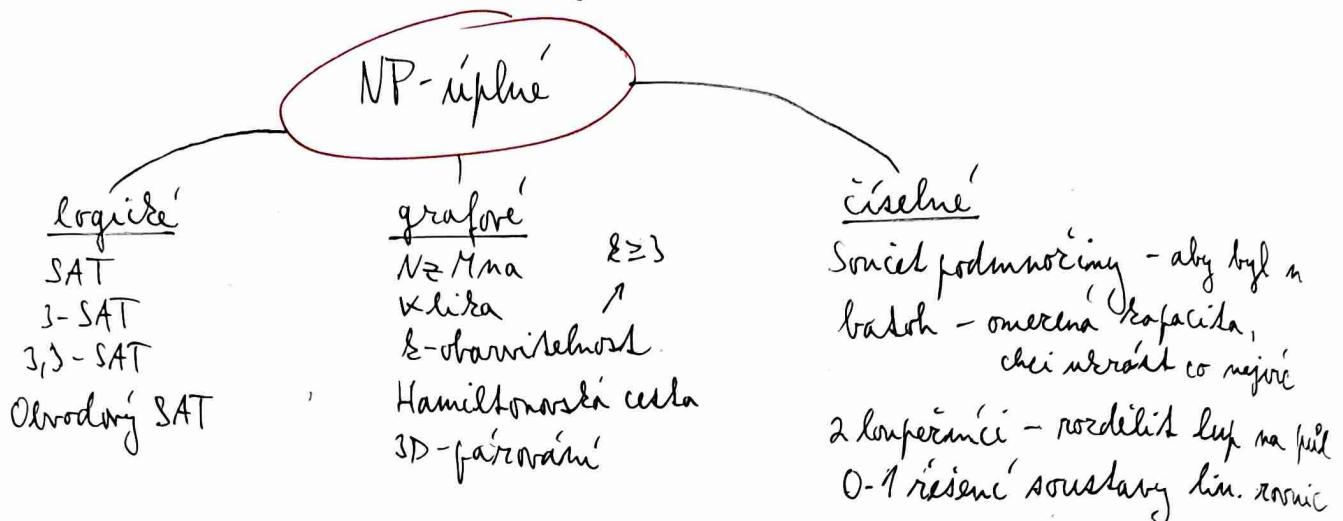
• přidáme tyto slavnule pro 1 hradlo  
• přidáme 1-převod slavnule pro výstupní port síťe  
↳ to je nejdražší proměnná buďže vstup nebo výstup hrodel ■

Důkaz: SAT je NP-úplný. ■

Diskuter: Když jsme SAT omeřili na CNF, tak jsme si to nelehčili.

Dr: Prímo' prenos sice může nabratnou až exponenciálně, ale

SAT co nemá  $\sim$  CNF  $\rightarrow$  Obrotný SAT  $\rightarrow$  SAT  $\sim$  CNF  $\blacksquare$

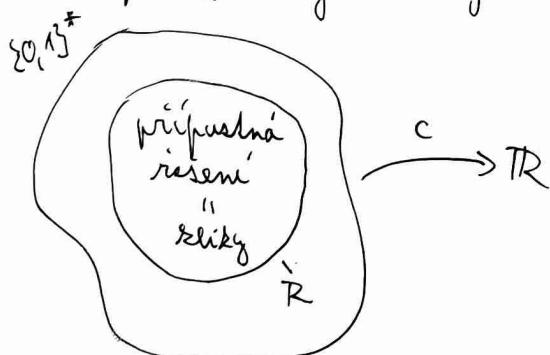


### • Jak řešit NP-úplné problémy?

- ① vyuzít malých vztahů, dělat nějakou lokální optimizaci
- ② speciální případy mohou být polynomialement řešitelné ... 2-SAT je lineární
- ③ užívat nějaké polynomialement aprobací algoritmus
- ④ použít heuristiky nebo evoluční algoritmus

### • Optimalizační problémy

→ Například majit co nejvíce klik.



$$R := \{x \in \{0,1\}^* \mid x \text{ je přípustné řešení}\}$$

$$c: R \rightarrow \mathbb{R} \dots \text{ cena / ohodnocení}$$

$$\text{chci: } x^* \in R \text{ a.s. } c^* = c(x^*) \text{ je min. / max.}$$

$\uparrow$   
optimalní řešení

### • $N \geq M \text{na}$ ve stromech

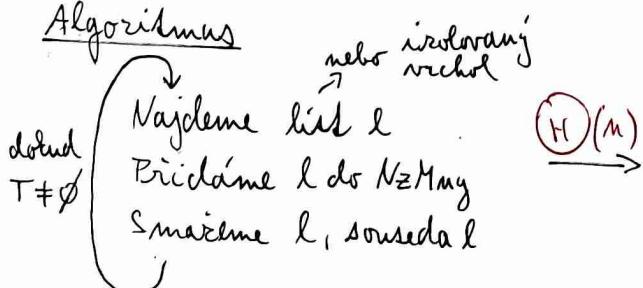
Lemma: Nechť  $T$  je strom a  $l$  jeho list. Pak alespoň 1 max.  $N \geq M \text{na}$  v  $T$  obsahuje  $l$ .

Dr: Nechť  $M$  je nějaká max.  $N \geq M \text{na}$ . Potom budí

1,  $l \in M$  ✓ soused( $l$ )  $\notin M \Rightarrow M$  nebyla max

2,  $l \notin M$  ✓ soused( $l$ )  $\notin M \Rightarrow M' := M - \text{soused} + l$  je pořád max. a máme  $l \in M'$ .  $\blacksquare$

### Algoritmus



### Implementace pomocí DFS

→ z rekurze vrácím, zda jsem byl přidán do  $N \geq M \text{ny}$

→ alespoň 1 syn byl přidán  $\Rightarrow$  já ne

→ žádnej syn nebyl přidán  $\Rightarrow$  já ano

## Rozdělovační přednášel do posluchařů

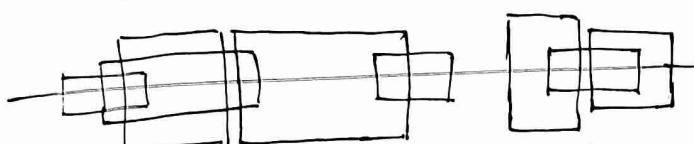
Výstup: uzavřené intervaly  $\langle x_1, y_1 \rangle, \langle x_2, y_2 \rangle, \dots, \langle x_n, y_n \rangle$

mať k nebezpečnému  
průniku

$\Rightarrow$  Intervalový graf:  $\{I_j\} \subseteq E \equiv |I_j \cap I_k| \geq 1$

Rozdělovační přednášel do posluchařů ~ obecnějšího grafu co nejméně barvami

- barvy ~ posluchaři
- 2 přednášky se nemohou překrývat v 1 posluchaři ~ konc kran různé barevné
- $\rightarrow$  obecně je obecně NP-úplné
- $\rightarrow$  ale pro intervalové grafy to jde rychle



$\rightarrow$  uděláme zámečník různy průnik  
 $\Rightarrow$  1D, načež zámečník průniky bodem

- $\rightarrow$  máme seznam volajících posluchařů a  $\#$  vyrobených posluchařů celkem
- $\rightarrow$  Edgě přijde přednáška, tak ji cláme libovolnou volnou posluchaři
- $\hookrightarrow$  pokud nemá, tak ji nejdříve vyrobíme
- $\rightarrow$  na začátku můžeme intervaly seřidit
- $\rightarrow$  na konci  $\#$  vyrobených posluchařů = min  $\#$  potřebných barev  $c(K_m) = m$
- Dle: novou posluch. vyrobíme Edgě po všechny ostatní zábrany
- $\Rightarrow$  ty přednášky se všechny překrývají  $\Rightarrow$  v tom grafu je elita dané velikosti

## Problém balohnu

$n$  předmětů  $\hookrightarrow$  hmotnosti  $h_1, \dots, h_n \geq 0$ ,  $h(P) := \sum_{i \in P} h_i$   $[S] := \{1, 2, \dots, S\}$

$H$  ... možnost balohnu

Výstup:  $P \subseteq [n]$  t.ž.  $h(P) \leq H$  a  $c(P)$  je max. ... chceme ukratit co nejdříve

$\rightarrow$  pro  $c_1, \dots, c_n \in \mathbb{R}$  to je NP-úplné, ale pro  $N$  lze dynamické programování

Dynam. prog.: vyplňujeme tabulku

$A_{\emptyset, C} := \min \{ h(P) \mid P \subseteq [S] \wedge c(P) = C \}$  ... rámeček  $i \sim$  může jít první i předmět

$C = \sum_i c_i$   $\min(\emptyset) = \infty$   $\rightarrow$  tabulce máme min. hmotnosti

$\hookrightarrow$  pokud  $C > C$ :  $A_{\emptyset, C} = A_{\emptyset, C}$

	0	1	2	...	$C = \sum_i c_i$	$\min(\emptyset) = \infty$	$\rightarrow$ tabulce máme min. hmotnosti
0	0	$+\infty$	$+\infty$	...	$+\infty$	$\hookrightarrow$ elichně $> H$	
1	0						
2	0	Vše máme					
$\vdots$							
$\infty$	-	-	-	-	-		
$n$	0						

$\Rightarrow$  Tabulka vyplňujeme za  $\textcircled{A}(n, C)$ , kde  $C = \sum_i c_i$  a  $C^* = \max \{C \mid A_{[n], C} \leq H\}$

$\rightarrow$  chci si optimální množinu  $X_1 = B_2, C^* \dots$  poslední v optimální m.

$B_{\emptyset, C} =$  poslední přidání předmět  $\Rightarrow X_2 = B_{X_1, C^* - c(x_1)} \dots$  předposlední

Je ta složitost  $\Theta(n \cdot C)$  dobrá?

$n \cdot C$  vypadá jako polynom

ale pokud délka vstupu je ten řetězec  $\in \{0,1\}^*$ , tak  
ta čísla jsou zapsaná ve 2- soustavě

$\Rightarrow C$  je exponenciální násobek délky vstupu

$\Rightarrow$  je to pseudopolynomialní algoritmus

$\hookrightarrow$  složitost závisí na interpretaci vstupu

- Aproximaci alg.  $\rightarrow$  relativní chyba  $\frac{C^* - C}{C^*} \leq 1 - \alpha$
  - Maximizační problém: chceme  $C(\text{výsledek}) \geq \alpha \cdot C^*$ ,  $\alpha \in (0, 1]$   $\frac{C}{C^*} \geq \alpha \Rightarrow 1 - \frac{C}{C^*} \leq 1 - \alpha$
  - Minimizační problém: chceme  $C(\text{výsledek}) \leq \alpha \cdot C^*$ ,  $\alpha \geq 1$   $\frac{C^* - C}{C^*} \leq \alpha - 1$
- $\hookrightarrow \alpha\text{-aproximace}$

## Problém obchodního cestujícího

Vstup: hranově ohraničený neorientovaný graf

Výstup: nejkratší hamiltonova kružnice ... obsahuje všechny vrcholy

$\Rightarrow$  Speciální případ: úplný graf s  $\Delta$  nerovností.  $\leftarrow$  "oneing" metody

### 2-aproximace:

①  $T \leftarrow$  min. kostra

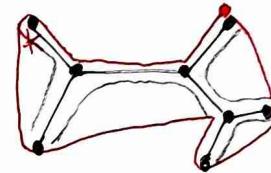
② pomocí DFS obcházejme okolo  $T \Rightarrow$  h. k.

$$ALG := \sum \text{délka hran z této h. kružnice}$$

$$|T| := \text{délka min. kostry}$$

$$\text{ALG} \leq 2|T|$$

$$|T| \leq |\text{optim. h. k. bez nejkratší hrany}| \leq OPT \quad \left. \begin{array}{l} \text{ALG} \leq 2 \cdot OPT \\ \text{(h. k. bez liborolné hrany je kostra \& } T \text{ je min. kostra} \end{array} \right\}$$



$\rightarrow$  zjistěno, že když  $\exists$  nějaká approximace obecného obch. cest., pak  $P = NP$ , protože ta approx. by našla nějakou hamilt. k., což je NP-sídlí.

Věta: Když existuje polynomiální 1-apx. alg. pro p.o.c. v úplných grafech, tedy  $\Delta$  nerovnost nemusí platit, pak  $P = NP$ .

Důkaz: Uvažujme, že pak lze rozhodnout, zda graf má h. k. polynomiálně.

$G \longrightarrow G'$  získaný z  $G$   $\left\langle \begin{array}{l} \text{původní hrany délka } 1 \\ \text{nové hrany délka } L > 1 \end{array} \right.$

$G$  má HK  $\Leftrightarrow$  opt. v  $G'$  má délku  $n$

$G$  nemá HK  $\Leftrightarrow$  opt. v  $G'$   $\geq n-1+L$  ... alespoň 1 nová hrana

$\rightarrow$  pro 1 apx. je jasné když  $G$  má HK

$\rightarrow$  pro 1-apx. chci  $1 \cdot m < n-1+L \Rightarrow$  rovnice  $L > m-n+1$

$$\begin{matrix} \uparrow & \uparrow \\ \text{má} & \text{nemá} \end{matrix}$$

$\Rightarrow$  pokud mi apx. řeší výsledek  $\leq 1 \cdot m \Rightarrow$  má  
jinak nemá



## Aproximace batohu knapsacku v cene

• pokud jsou všechny ceny másového nejake konstanty, tak je tomu konstantou mnoha výdajů, výřešit tento redukovaný problém a výsledek využít tomu knapsacku

⇒ my prvně rozdělíme a nefixovat různoblasné ... to bude fungovat i pro CTR  
 chci  $\langle 0, C_{\max} \rangle \rightarrow \{0, \dots M\}$

$$\hat{c}_i := \left\lfloor \frac{c_i}{C_{\max}/M} \right\rfloor = \left\lfloor \frac{c_i}{C_{\max}} \cdot M \right\rfloor \dots$$

$\frac{C_{\max}}{M}$  je ta škalovací konstanta



rozdělím ho na  $M$  intervalů  
 délky  $C_{\max}/M$   
 ⇒ některému  $i$  je  $\frac{C_{\max}}{M}$

→ výsledek pro cenu  $\hat{c}$

→ vrátíme ty původní předměty

⇒ chyba ceny 1 položky  $< \frac{C_{\max}}{M}$  = délka toho intervalu

⇒ chyba ceny odpovídající množině  $< M \cdot \frac{C_{\max}}{M} \stackrel{?}{\leq} \epsilon \cdot C^*$   $\text{(*)}$

• musíme nejdříve vyplnit předměty  $i$ :  $h(i) > H$  ... mohou mít obecnou  
 ale jsou nejednotelné  
 ⇒ potom  $C_{\max} \leq C^*$

⇒ pro  $M \geq \frac{m}{\epsilon}$   $\Rightarrow \frac{m}{M} \leq \epsilon$  máme chybu  $< \frac{m}{M} C_{\max} \leq \epsilon C_{\max} \leq \epsilon \cdot C^*$

⇒ relativní chyba =  $\frac{\epsilon \cdot C^*}{C^*} = \epsilon = 1 - \alpha \Rightarrow \alpha = 1 - \epsilon$

↑ aprotimaci formule

### Algoritmus

1. Odstraníme položky ležící necelé  $H$

2.  $C_{\max} \leftarrow \max_i \{c_i\}$

3.  $M \leftarrow \left\lceil \frac{m}{\epsilon} \right\rceil$

4.  $\hat{c}_i : \hat{c}_i \leftarrow \left\lfloor \frac{c_i}{C_{\max}} \cdot M \right\rfloor$

5. Vypočítáme sílohu s  $h, \hat{c}$

6. vrátíme ty předměty, které nám to dělají

Složitost:  $T(m) = O(m \cdot \hat{c}) = O\left(\frac{m^3}{\epsilon}\right) \dots \hat{c} \leq M \cdot \hat{c}_{\max} \leq M \cdot M = m \cdot \left\lceil \frac{m}{\epsilon} \right\rceil$

⇒ vytvořili jsme algoritmus, který pro  $\forall \epsilon > 0$  našíme  $(1-\epsilon)$ -aproximaci  
 problému batohu v čase  $O(m^3/\epsilon)$ . → aprotimaciální schéma

Kef: Polyynomialní apx. schéma (PTAS) max. problém je alg., který

pro  $\forall \epsilon > 0$  našíme  $(1-\epsilon)$ -aproximaci v čase  $O(\text{polynom}(m))$

... je plně polyomialní (FPTAS)  $\equiv$  čas  $\in O(\text{polynom}(M, \frac{1}{\epsilon}))$

→ elidní  
 $m^{\frac{1}{\epsilon}}$

nejake rávistlost na  $\epsilon$