Demo-cvičení 8.12 + 12.12 2023/2024

1. Uvažme unární operaci p na jazycích nad abecedou Σ $(p: 2^{\Sigma^*} \to 2^{\Sigma^*})$ t.ž. $\forall L \subseteq \Sigma^*$:

$$p(L) = \{ w \in \Sigma^* \mid \exists w_p \in L \ \exists w_s \in \Sigma^* : w = w_p \cdot w_s \}$$

Dokažte, že $\forall L \in DTime[n^3]$ platí, že

a) $p(L) \in DTime[n^4]$

b) $p(L) \in DSpace[n^3]$

c) $p(L) \in NTime[n^3]$

Ad a) Hlavní myšlenka

- víme, že existuje DTS M_L, který rozhoduje L v čase $O(n^3)$
- sestrojíme DTS M, který rozhoduje $p(L)$: $w \in L(M) \Leftrightarrow w \in p(L)$

1) M systematicky zkouší všechny prefixy w_p slova w

2

2) M w_p překopíruje na novou pásku

3) M odsimuluje M_c na w_p

- pokud M_c akceptuje w_p, tak M akceptuje w
- jinak smaže obsah této pásky a pokračuje s dalším prefixem w_p
- pokud ~~už~~ odsimuloval všechny prefixy a neakceptoval tak M zamítne w

Analýza složitosti:

$$(n+1)\cdot\left(\underset{\substack{\uparrow\\ \text{kopírování}}}{O(n)} + \underset{\substack{\uparrow\\ \text{simulace } M_c + \text{mazání}}}{O(n^3)} \right) = O(n^4)$$

$\uparrow$ počet prefixů

Ad 5) DTS z bodu a) očividně pracuje v prostoru $O(n^7)$

c) NTS M nhádne vhodný prefix w_p a odsimuluje M_c pouze jednou. Tedíž pracuje v čase $O(n^3)$

2 Dokažte že třída $\mathbb{P}$ je uzavřená

a) na operaci $\cdot$

b) na operaci $*$

Ad a) Chceme dokázat, že $\forall L_1, L_2 \in \mathbb{P} \Rightarrow L_1 \cdot L_2 \in \mathbb{P}$

Víme tedy, že existují ~~stroje~~ $\exists^{DTS}\, M_1, M_2$, které rozhodují L_1 a L_2 a oba pracují v polynomiálním čase tj. se složitostí $O(n^k)$ pro nějaké k.

Sestrojíme DTS M, který rozhoduje $L_1 \cdot L_2$:

- M pro daný w postupně zkouší rozdělení $w = w_1 \cdot w_2$
- pro každé rozdělení ověří, že $w_1 \in L_1 \wedge w_2 \in L_2$ pomocí simulace stroje M_1, M_2 (využívá pomocné pásky)
- pokud M_1 i M_2 akceptují w_1 a w_2 pak M akceptuje w

4

- jinak zkouší další rozdělení
- pokud M zkontroloval všechna rozdělení a neakceptoval w, tak w zamítne

Analýza složitosti:

$$\underbrace{(n+1)}_{\substack{\text{počet rozdělení} \\ w = w_1 \cdot w_2}} \cdot \Big(\underbrace{O(n)}_{\text{kopírování}} + \underbrace{O(n^{k_1})}_{\substack{\text{simulace + mazání} \\ M_1}} + \underbrace{O(n^{k_2})}_{\substack{\text{simulace +} \\ \text{mazání } M_2}} \Big) \in \underbrace{O(n^k)}_{\substack{\text{pro nějaké} \\ k \in \mathbb{N}}}$$

Ad 5) Chceme dokázat, že $\forall L \in P \Rightarrow L^* \in P$

- systematická enumerace slova w na všechny možné podřetězce různých délek a začínajících na různých pozicích vede na exponenciální složitost

máme O(n) možných podřetězců

$$\underbrace{O(n)}_{\substack{\text{počet pozic, kde} \\ \text{končí 1. podřetězec}}} \cdot \underbrace{O(n)}_{\text{2. podřetězec}} \cdots \underbrace{O(n)}_{n\text{. podřetězec}} = \underline{O(n^n)} \; !!$$

- Lze řešit dynamickým programováním :
- DTS M pro $\underbrace{\cancel{w}^{*} \text{ nejdříve}}_{\text{vstup } w = a_1 \dots a_n}$ vybuduje matici $\overline{M}$
  - rozměrech $(n+1) \times (n+1)$ t.ž. $\boxed{\overline{M}(i,j) = \text{true} \iff a_i a_{i+1} \dots a_{j-1} \in L}$

    DTS simuluje stroj M_L (rozhoduje jazyk L) na vstup $a_i \dots a_{j-1}$
  - konstrukce matice $\overline{M}$ má časovou složitost

$$\underbrace{(n+1)^2}_{\text{počet políček}} \cdot \Big(\underbrace{O(n^k)}_{\text{simulace + mazání } M_L} + \underbrace{O(n)}_{\text{kopírování}} \Big) \in \underline{\underline{O(n^k)}}$$

- M dále spočítá tranzitivní uzávěr matice $\overline{M}$. Složitost $O(n^3)$.
- M akceptuje $w \iff$ v políčku $\overline{M}(1,n+1)$ je hodnota true

  t.j. umíme rozdělit w na podřetězce

  $a_1 \dots a_{i_1-1} \in L$

  $a_{i_1} \dots a_{i_2-1} \in L$

  $\vdots$

  $a_{i_k} \dots a_n \in L$

3 Uvažme speciální případ SAT problému, kde každá klauzule má dva literály tj. 2SAT problém.

Triviálně víme, že $2SAT \in NP$ a $2SAT \in EXP$

Dokažte, že $2SAT \in P$

Klíčové pozorování: Pokud formule φ ~~obs~~ obsahuje klauzuli ~~...~~ $(a \vee b)$ pak platí, že $\neg a \Rightarrow b \wedge \neg b \Rightarrow a$

tudíž ~~...~~ množina klauzulí generuje "posloupnosti" implikací.
Pokud existuje proměnná a t.ž. existují posloupnosti

$$a \Rightarrow \cdots \Rightarrow \neg a \quad \wedge \quad \neg a \Rightarrow \cdots \Rightarrow a$$

tak víme, že formule je nesplnitelná. Hledání těchto posloupností redukujeme na hledání cest v grafu:

Nechť $Var = \{v_1 \dots v_n\}$ je množina proměnných a C je množina klauzulí.
Sestrojíme graf $G = (V, E)$, kde $|V| = 2 \cdot |Var|$
Pro každou $v \in Var$ ~~máme~~ je $v \in V$ a $\neg v \in V$

$(a,b) \in E \iff (\neg a \vee b) \in C$

t.j. pro každou $(a \vee b) \in C$ přidej hrany $(\neg a, b)$ a $(\neg b, a)$

φ je nesplnitelná $\iff \exists v \in Var$ t.ž. v G existují cesty $v \rightsquigarrow \neg v$ a $\neg v \rightsquigarrow v$

Velikost G je v $O(\varphi)$ a tudíž hledání cest je v $O(n^k)$ kde n je velikost φ a $k \in \mathbb{N}$

$\Leftarrow$ je zřejmý na základě našeho pozorování

$\Rightarrow$ Lze ukázat obměnou:
Na základě neexistence „konfliktních cest v G" sestrojíme splňující přiřazení proměnným z Var

8

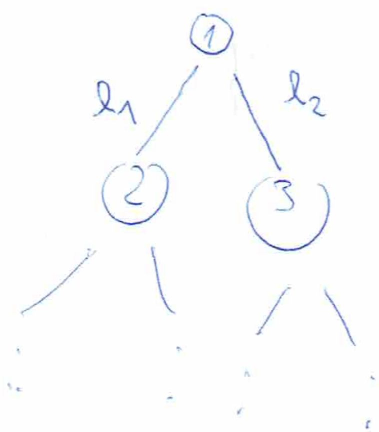
$\boxed{P4}$ Uvažme problém Vertex Cover:

Pro daný neorientovaný graf $G = (V, E)$ najdi nejmenší množinu $V' \subset V$ t.ž. V' ~~pokrývá~~ pokrývá všechny hrany t.j. $\forall (u,v) \in E$: ~~...~~ $u \in V' \vee v \in V'$

Rozhodovací verze tohoto problému je NP-úplná (viz. literatura)

Uvažme problém Vertex Coveru pro stromy
- pro jednoduchost uvažme binární stromy (lze zobecnit)

Klíčové pozorování:



Abychom pokryli hrany e_1, e_2 tak máme dvě možnosti:
1) $v_1 \in V'$ nebo 2) ~~...~~ $v_2, v_3 \in V'$

Tudíž můžeme sestrojit následující Alg.

Nechť T je strom, který je dán

- ~~kořenem~~ root
- v.left a v.right pro $v \in V$ vrací přímé následníky v

```
vCover(root){
  if root = NULL ∨ (root.left = Null ∧ root.right = Null) return 0
  size_inc = 1 + vCover(root.left) + vCover(root.right)   // možnost 1)
  size_excl = 2 + vCover(root.left.left) + vCover(root.left.righ) +
                  vCover(root.right.right) + vCover(root.righ.left)   // možnost 2
  return min(size_inc, size_excl)
}
```

Složitost: každý vrchol ~~na~~ navštívím (analyzuju) tolikrát kolik má předchůdců ~ určitě $O(n)$-krát

Celková složitost je tudíž $O(n^2)$

Lze nějak vylepšit?

Ano, moci jednoduchého "bookkeeping"

Nebudeme pro vrcholy stále dokola počítat vCover ale získanou hodnotu si uložíme do v.vc (inicializujeme na −1)

```
vCover(root) {
  IF root.vc ≠ −1 return root.vc

  | jako předchozí |
  | verze          |

  root.vc = min(size_incl, size_excl)
  return root.vc
```

Složitost: každý vrchol navštívím jednou tudíž celková složitost je $\underline{O(n)}$

5

Navrhněte efektivní implementaci fronty pomocí jednosměrně vázaného seznamu s efektivním přístupem k začátku seznamu (t.j. zásobník či seznamy v logických/funkcionálních jazycích)

Hlavní myšlenka: Budeme používat dva zásobníky a dle potřeby prvky z prvního kopírovat do druhého

```
list head =[]
list tail =[]
enq(x) { tail.push(x)}
deq() { if(head) =[] {
          while (tail != []) {
            x = tail.pop()
            head.push(x)
          }
        }
return head.pop }
```

Složitost (předpokládáme jednotkovou cenu všech operací)

- v nejhorším případě

  enq(x) : $1 \in O(1)$

  x = deq() : $\max(2, 3+3n)$

  $= 3+3n \in O(n)$

Pak sekvence n operací: $n \cdot O(n) = O(n^2)$

→ jde o velice nepřesnou overaproximaci

Amortizovaná složitost - metoda účtů

operace	cena	kredit
enq(x)	1	1 + 3
x = deq()	head = []: 3n + 3 head ≠ []: 2	3

klíčové pozorování

- Cena operace deq() záleží na počtu předchozích volání enq(x)
- Proto si s každou operací enq(x) ~~dáme~~ "předplatíme" 3 kredity (cena jedné iterace while cyklu) na další operaci deq()

- musíme ukázat, že pro libovolnou sekvenci operací neklesne hodnota účtu pod 0
- je snadno vidět, že platí následující invariant $\boxed{\text{účet} \geq 3 \cdot |\text{tail}|}$ }

důkaz např. indukcí

Amortizovaná složitost sekvence $\underline{n}$ operací je pak

$$n \cdot \max(\text{kredit}) = n \cdot \max(4, 3) = 4n \in O(n)$$

bereme operaci, která platí nejvíce kreditů

(6) Uvažme problem 3SAT. Dokažte, že 3SAT je NP-úplný! (13)

(připomeňme si, že $2SAT \in P$)

- Stačí ukázat, že $SAT \leq_p 3SAT$ → existuje polynomiální redukce

- Hlavní myšlenka: potřebujeme transformovat klauzule, které nejsou 3SAT, tak aby se zachovala splnitelnost

  - klauze (x), nebo $(x_1 \vee x_2)$ triviálně doplníme tj.
    $(x \vee x \vee x)$ nebo $(x_1 \vee x_2 \vee x_1)$

  - klauzule $C = (x_1 \vee x_2 \vee x_3 \vee x_4)$ transformujeme na dvě klauzule $\underbrace{(x_1 \vee x_2 \vee y)}_{C_1} \vee \underbrace{(\bar{y} \vee x_3 \vee x_4)}_{C_2}$ $(y \notin Var)$ — nová proměnná

    triviální: C je splnitelná $\Leftrightarrow$ $C_1 \vee C_2$ je splnitelná

- ~~obecně~~

- Obecná transformace (schéma)

$x_1 \vee x_2 \vee \dots \vee x_n$ transformujeme na množinu klauzulí

$(x_1 \vee x_2 \vee y_1) \vee (\neg y_1 \vee x_3 \vee y_2) \vee (\neg y_2 \vee x_4 \vee y_3) \vee \dots$

$(\neg y_{n-3} \vee x_{n-1} \vee x_n)$

každá nová klauzule zmenší původní klauzuli aspoň o 1 literál

vidíme, že došlo pouze k $O(n)$ zvětšení formule

- Zachování příslušnosti je opět přímo vidět
- Proč tato transformace nefunguje pro 2SAT?:

2SAT nám neumožňuje "zkracovat" původní dlouhé klauzule

7 Uvažme problém KLIKA v neorientovaném grafu

- klika: úplný podgraf
- problém KLIKA = $\{ \langle G, k \rangle \mid$ G je neorientovaný graf, který obsahuje kliku o velikosti $k \}$

- KLIKA $\in$ NP

  NTS uhádne k vrcholů a ověří, že jde o kliku

  složitost záleží na kódování grafu ale určitě je v P

- $3SAT \leq_p$ KLIKA

Hlavní myšlenka: Budeme jednoznačně napojit každý literál v každé klauzuli na vrcholy grafu. Pro formuli s k klauzulemi budeme hledat k kliku. Hrany definujeme mezi ~~[illegible]~~ "nekonfliktními" literály z různých klauzulí

Formálně: Mějme 3SAT formuli:

$F = C_1 \wedge C_2 \wedge \dots \wedge C_k$

$\forall\, 1 \leq i \leq k \quad C_i = L_i^1 \vee L_i^2 \vee L_i^3$

$\forall\, 1 \leq j \leq 3 \quad L_i^j$ je buď proměnná nebo její negace

Pro F sestrojíme $G_F = (V_F, E_F)$

$V_F = \{ [i, j] \mid 1 \leq i \leq k \wedge 1 \leq j \leq 3 \}$

index klauzule

index literálu v klauzuli

$E_F = \{ \{[i, j], [i', j']\} \mid i \neq i' \wedge L_i^j \neq \neg L_{i'}^{j'} \}$

a hledáme k-kliku

různé klauzule

nekonfliktní literály

Je lehké ověřit, že tato redukce proběhne v polynomiálním čase a tudíž výsledný graf bude mít max. polynomiální velikost

Ukážeme, že redukce je korektní, tj. zachovává příslušnost

F je splnitelná $\Leftrightarrow$ G_F má k kliku

$\Rightarrow$ Je-li F splnitelná, pak pro každou klauzuli C_i $(1 \le i \le k)$ lze zvolit literál L_i^j $(1 \le j \le 3)$, tak že žádné dva zvolené literály vzájemně nekolidují (nejsou v konfliktu) – tedy $L_i^j \neq \overline{L}_i^j$

Těmto literálům odpovídá k vrcholů, které jsou navzájem propojené hranami a tudíž tvoří k kliku

$\Leftarrow$ Má-li G k kliku, tak ~~existuje k nekonfliktních~~ literálů existuje v každé klauzuli literál, který není v konfliktu s žádným jiným z k literálů. Z těchto literálů lze přímočaře sestrojit splnitelné přiřazení ve formuli F.

[8] Dokažte, že KLIKA $\leq_p$ 3SAT

- Vezměme graf $G=(V,E)$ a velikost kliky k. Nechť $V=\{v_1, v_2, \dots, v_n\}$ a očíslujme vrcholy kliky $1, 2 \dots k$.

- Zavedeme proměnné X_i^j, kde pro $1 \leq i \leq n$, $1 \leq j \leq k$:

$$X_i^j = \begin{cases} \text{true} & \text{jestliže } v_i \text{ je } j\text{-tým uzlem kliky} \\ \text{false} & \text{jestliže } v_i \text{ není } j\text{-tým uzlem kliky} \end{cases}$$

- Aby ohodnocení těchto proměnných kódovalo kliku musí platit

  1. Každý uzel kliky musí odpovídat nějakému uzlu grafu

  $$\bigwedge_{1 \leq j \leq k} \bigvee_{1 \leq i \leq n} X_i^j \equiv \varphi_1$$

  2. Žádné dva různé uzly kliky se nesmí mapovat na stejný uzel graf

  $$\bigwedge_{1 \leq j_1 < j_2 \leq k} \bigwedge_{1 \leq i \leq n} \neg \left(X_i^{j_1} \wedge X_i^{j_2} \right) \equiv \varphi_2$$

3. každé dva uzly kliky X musí být namapované na uzly grafu, které jsou spojeny hranou tj. nesmí existovat dva různé uzly kliky, které nejsou propojené hranou:

$$\bigwedge_{1 \le j_1 < j_2 \le k} \; \bigwedge_{\substack{1 \le i_1 < i_2 \le n \,\wedge \\ \{v_{i_1}, v_{i_2}\} \notin E}} \neg\left(X_{i_1}^{j_1} \wedge X_{i_2}^{j_2}\right) \equiv \varphi_3$$

Celkově pro (G, k) sestrojíme formuli $\varphi_1 \wedge \varphi_2 \wedge \varphi_3$

Opět vidíme, že konstrukce formule pracuje v polynomiálním čase (a prostoru). Rovněž se snadno nahlédne, že konstrukce zachovává příslušnost f_1.

G má k kliku $\Leftrightarrow$ $\varphi_1 \wedge \varphi_2 \wedge \varphi_3$ je splnitelná

$\boxed{19}$ Úloha Subset sum problem (SSP)

- nechť S je konečná množina kladných přirozených čísel, tj. $S \subseteq \mathbb{N}$ a nechť $t \in \mathbb{N}$
- problém $\exists S' \subseteq S : \sum_{s \in S'} s = t$
- $SSP \in NP$ je zřejmé
- ukážeme hlavní myšlenku polynomiální redukce

$$3SAT \leq SSP$$

Mějme 3SAT formuli nad proměnnými... Příklad

$Var = \{w_1 \dots w_n\}$ tvaru

$F = C_1 \dots C_m$

Pro formuli vybuduje následující tabulku která nám koduje přirozená čísla v dosti velké soustavě

(aby nedošlo k přetečení, viz níže)

true/false pro každou proměnnou

	v_1	v_2	…	v_n	c_1	c_2	…	c_m
T_1	1	0	…	0				
F_1	1	0	…	0				
T_2	0	1	0 …	0				
F_2	0	1	0					
⋮								
T_n	0	…	0	1				
F_n	0	…	0	1				
$S1_1$	0	…	…	0	1	0	…	0
$S2_1$	0	…	…	0	2	0	…	0
$S1_2$	0	…	…	…	0	1	…	0
$S2_2$	0	…	…	…	0	2	…	0
⋮								
$S1_m$	0	…	…	…	…	…	0	1
$S2_m$	0	…	…	…	…	…	0	2
t	1	1	…	1	4	4	…	4

Na pozici $T_i c_j$ je 1 $\iff$ v_i je v klauzuli c_j

$F_i c_j$ je 0 $\iff$ $\neg v_i$ je v klauzuli c_j

„slack proměnné" pro každou klauzuli; C_j má proměnnou $S1_j$ a $S2_j$ s hodnotou 1 a 2 ve sloupci C_j

každý řádek je jedno číslo v S

22

Lze nahlédnout, že

3SAT je splnitelná $\Leftrightarrow$ existuje výběr řádků jejichž součet dává hodnotu $\underline{t}$ (poslední řádek v tabulce)

$\Rightarrow$ splnitelné přiřazení definuje pro každou V_i zda použiji řádek T_i či F_i $\rightsquigarrow$ součet všech sloupců pro V_i bude 1

Tento výběr též zařídí že bude aspoň jedna jednička v každém sloupci C_j $\rightsquigarrow$ pak vhodně zvolím slack proměnné, abych získal součet 4 $\rightsquigarrow$ vždy je možné

$\Leftarrow$ výběr řádků který mi dají požadované t definuje splnitelné přiřazení

10 Uvažme problém batohu (knapsack) KP (25)

- konečná množina R
- váhová funkce $w: R \to \mathbb{Z}$
- cenová funkce $v: R \to \mathbb{Z}$
- cílová váha U a cílová cena V
- ptáme se, zda $\exists R' \subseteq R$ t.ž.

$$\left(\sum_{t \in R'} w(t) \leq U\right) \wedge \left(\sum_{t \in R} v(t) \geq V\right)$$

Cena a váha prvku s_i je hodnota s_i

- $KP \in NP$ je opět triviální
- ukážeme, že

$SSP \leq KP$

pouze hlavní myšlenka

Nechť SSP je dáno

- $S = \{s_1, \dots, s_n\}$
- hodnotou t
- Pak $R = S$ a
- $w(s_i) = v(s_i) = s_i \quad \forall 1 \leq i \leq n$
- $U = V = t$

$(x = y \iff x \leq x \wedge y \geq x)$