1) Určete a dokažte, zda platí

a) $3n^2+4n+17 \in O(n^2-n+1)$ ⇝ (intuitivně platí, jelikož máme stejný stupeň polynomu)

$\Leftrightarrow \exists c \in \mathbb{R}^+, n_0 \in \mathbb{N}: \forall n \geq n_0: \quad 3n^2+4n+17 \leq c(n^2-n+1)$

- zvolím $c=4$ a dostávám $3n^2+4n+17 \leq 4n^2-4n+4 \Leftrightarrow$

$8n+13 \leq n^2$ platí pro $n \geq 10$

Našli jsme požadované $c=4$ a $n_0=10$ tudíž $3n^2+4n+17 \in O(n^2-n+1)$

b) $2^n \in O(n^2)$ ⇝ intuitivně neplatí!

- Muselo by platit, že $\exists c \in \mathbb{R}^+, n_0 \in \mathbb{N}: \forall n \geq n_0: 2^n \leq cn^2 \Rightarrow$

$\forall n \geq n_0 \quad \frac{c \cdot n^2}{2^n} \geq 1$ ↝ budeme zkoumat, jak $\frac{c \cdot n^2}{2^n}$ vypadá pro $n \to \infty$

$$\lim_{n\to\infty} \frac{cn^2}{2^n} \overset{L.H.}{=} \lim_{n\to\infty} \frac{2cn}{(\ln 2)\cdot 2^n} \overset{L.H.}{=} \frac{2c}{(\ln 2)^2 \cdot 2^n} = 0$$

($2c$ → konstanta; $n^2 \to \infty \wedge 2^n \to \infty$)

Dostáváme spor ↯

a tudíž $2^n \notin O(n^2)$

(2) Rozhodněte a dokažďte, zda platí

a) $L_1 \in \mathcal{P} \wedge L_2 \in \mathcal{P} \Rightarrow L_1 \cdot L_2 \in \mathcal{P}$ ($\mathcal{P}$ je uzavřena na operaci konkatenace)

Sestrojíme DTS Π t.ž. $w \in L(\Pi) \Leftrightarrow \exists w_1, w_2 : w = w_1 w_2 \wedge w_1 \in L_1 \wedge w_2 \in L_2$

- Kolik je rozdělení w ($|w| = n$) na 2 podslova: umístíme 1 "zatížku" tudíž $O(n)$

$\Leftrightarrow$ umíme v $O(|w_1|^{k_1})$ a $O(|w_2|^{k_2})$

- Π bude systematicky zkoušet všechna rozdělení w na w_1 a w_2 a ověřovat zda $w_1 \in L_1$ a $w_2 \in L_2$. Pokud pro alespoň 1 rozdělení toto platí, tak Π akceptuje w jinak odmítne w.

Složitost: $O(n) \cdot (O(n^{k_1}) + O(n^{k_2}) + O(n)) \in O(n^k)$ pro $k \in \mathbb{N}$

b) $L \in P \Rightarrow L^* \in P$

Opět DTM M: $w \in L(M) \Leftrightarrow \exists w_1 \dots w_m$
$m \leq |w| \wedge w = w_1 \dots w_m \wedge \forall 1 \leq i \leq m \; w_i \in L$

- nelze vyzkoušet všechna rozdělení w na 1, 2, ... n podřetězců
- kolik je takových rozdělení?

  $w = a_1 \dots a_n$: mám $n-1$ pozic, kde w můžu rozdělit
  0 - rozdělím / 1 - nerozdělím $\rightsquigarrow$ celkově 2^{n-1} rozdělení!
  ↯ nevede na polyn. alg.!!
- např. podslovo $a_j \dots a_k$ budu vyhodnocovat ve všech rozděleních, kde nějaké $w_i = a_1 \dots a_k$ $\rightsquigarrow$ vede na dynamické programování
- jednotlivé podúlohy uložím do $(n+1) \times (n+1)$ matice M kde
  $M(i,j) = T \Leftrightarrow a_i \dots a_{j-1} \in L$
  ↳ čísla j

= Složitost konstrukce:

$O(n^2) \cdot (O(n^k) + O(n))$

↳ kopírování

3) volám DTM pro L

Matice M mi definuje graf $G=(V,E)$ kde $|V| = O(n)$ a $|E| = O(n^2)$ přeskoč $n+1$

Pro $w = a_1 \dots a_n$ platí

$w \in L^* \iff$ existuje cesta z 1 do $(n+1)$

$\Downarrow$

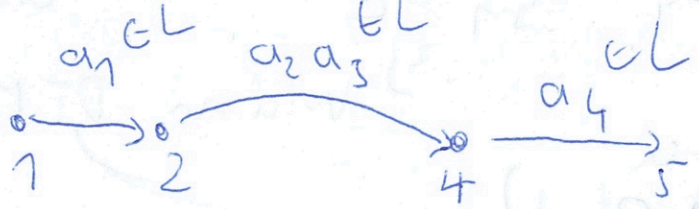
složitost $O(|G|) = O(n^2)$

Celková složitost stroje Π je

$$\underbrace{O(n^2) \cdot \left(O(n^k) + O(n)\right)}_{\text{konstrukce } \Pi} + \underbrace{O(n^2)}_{\text{hledání cesty}} \in O(n^{k'}) \text{ pro } k' \in \mathbb{N}$$

Příklad matice pro $w = a_1 \dots a_4$

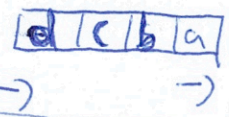
	1	2	3	4	5
1	w	T	F	F	F
2	w	w	F	T	F
3	w	w	w	F	F
4	w	w	w	w	T
5	w	w	w	w	w

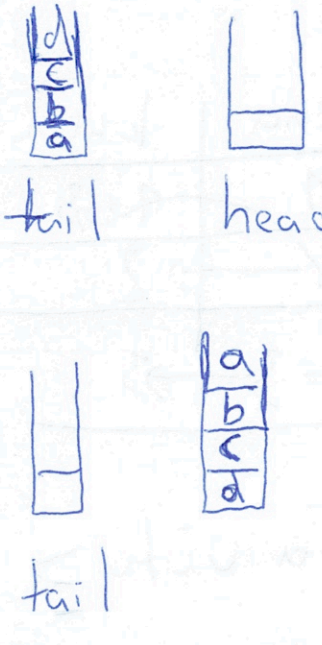


$\Rightarrow w \in L^*$

③ Navrhněte efektivní reprezentaci fronty pomocí 2 zásobníků

Myšlenka: fronta (FIFO)





přesun prvků pokud head = []
a chceme odebrat prvek

kód:

```
stack head ← []
stack tail ← []
enqueue(x) { tail.push(x) }
dequeue()
{  if (head = []) {
      while (tail ≠ []) {
         x ← tail.pop()
         head.push(x)
      }
   }
   return head.pop()
}
```

Worst-case analýza

enqueue: $1 \in O(1)$

dequeue: $\max(2, 3n+3) = 3n+3 \in O(n)$

posloupnost m operací: $m \cdot O(n) = O(n \cdot m)$

možnosti:

$O(m)$, ~~$O(m \cdot \log(n))$~~

velmi nepřesný horní aproximace

cena každého čtení je jedna

Uniformní cenové kritérium ⇝ nefesíme realokaci

Ukážeme, že amortizovaná cena každé operace je $O(1)$

Metoda účtů

		cena	kredity
enqueue		1	1+3
dequeue	head ≠ []	2	2
	head = []	$3n+3$	3

každou operaci enqueue si předplatíme 3 kredity na provedení 1 iterace while cyklu

- počet kreditů na účtu $\geq 3 \cdot |tail|$ (invariant)

↓ tj. vždy můžeme zaplatit drahou operaci dequeue

- složitost m operací:

$$m \cdot \max(\text{kredit}) = m \cdot \max\{4,2,3\} = 4m \in O(m)$$ ✓

4. příklad

Uvažme "balancovanou" frontu, která má kapacitu k (k je sudé číslo):

- *enqueue*(x) vloží prvek x na konec fronty a má konstantní složitost
- *dequque* odstraní prvek z konce fronty a má konstantní složitost
- *size*() vrací velikost fronty (počet prvků) a má konstantní složitost
- fronta je inicializována náhodnými prvky na velikost $k/2$

Uvažme operace *my_enquque*(x) a *my_dequque*(x) implementované níže.

```
1  global(int k)
2  Function my_enqueue(x)
3  |  if size() = k then
4  |  |  while size() ≥ k/2 do
5  |  |  |  dequeue()
6  |  enquque(x)
7  Function my_dequeue()
8  |  if size() = 0 then
9  |  |  while size() < k/2 do
10 |  |  |  enquque(size())
11 |  else dequeue()
```

Uvažte libovolnou posloupnost n operací *my_enqueue*(x) a *my_dequeue*().

1. Jaká je asymptotická složitost této posloupnosti operací při použití analýzy nejhoršího případu?
2. Jaká je amortizovaná složitost této posloupnosti operací?

Poznámka: Předpokládejte uniformní cenové kritérium, kde složitost každého řádku programu je 1.

4) Cena v nejhorším případě

my-enqueue : $O(k)$ ~> pokud size() = k

my-dequeue : $O(k)$ ~> pokud size() = 0

cena v nejhorším případě

cena n operací v nejhorším případě: $O(nk)$

- velice hrubá nad aproximace

Jaké jsou možnosti

$O(n)$, $O(n \cdot \log(k))$, $O(nk)$

↓

amortizovaná cena libovolné operace je $O(1)$

Dolcaželma metodou účtů

		cena	kredity	
enqueue	size() < k	2	2+2	každým vložením si předplácím jednu iteraci while
	size = k	3+2(k/2+1)	3+2	
dequeue	size() ≠ 0	2	2+2	
	size = 0	2+2k/2	2	opět si předplácím

Korektnost: Počet kreditů na účtě nikdy nesmí klesnout pod 0

Invariant: počet kreditů $\geq 2 \cdot |size() - k/2|$

Dává nám korektnost

Cena libovolné posloupnosti n operací $\leq n \cdot \max\{5,4,2\} = 5n \in O(n)$

5. příklad (problém obarvení grafu)

Mějme neorientovaný graf $G = (V, E)$, kde $E \subseteq \{\{u, v\} \mid u, v \in V\}$. Ptáme se, zda existuje obarvení uzlů grafu c barvami tak, aby žádné dva sousedící uzly neměly stejnou barvu.

Uvažme tento "brute-force" algoritmus, který využívá operaci *inc* z minulého příkladu. Předkládáme, že $V = \{0, 1, \ldots, |V| - 1\}$.

```
1  Function is_colorable(V, E, c)
2  |  < 0, ..., c − 1 > color[0, ..., |V| − 1] = [0, ..., 0]
3  |  while true do
4  |  |  res := true
5  |  |  foreach {u, v} ∈ E do
6  |  |  |  if color[u] = color[v] then
7  |  |  |  |  res := false
8  |  |  |  |  break
9  |  |  if res = true then
10 |  |  |  return YES
11 |  |  if color = [c − 1, ..., c − 1] then
12 |  |  |  return NO
13 |  |  else  inc(color, |V|)
```

1. Analyzujte asymptotickou složitost algoritmu *is_colorable* v nelepším případě.
2. Analyzujte asymptotickou složitost algoritmu *is_colorable* v nejhorším případě.

Poznámka: Předpokládejte uniformní cenové kritérium, kde složitost každého řádku programu je 1 (včetně řádku 13).

(5)

složitost v nejlepším případě

- najdu validní obarvení po konstantním počtu while-iterací: $O(1)$
  - ↳ nutné ověřit, že jde o validní obarvení: $O(|E|)$

Celkově $O(|E|)$

složitost v nejhorším případě

- validní obarvení neexistuje ⇒ tudíž musím projít všechny možné obarvení
  - počet barev Γ, $C^{|V|}$ while iterací každá stojí $O(|E|)$

Celkově $O(C^{|V|} \cdot |E|)$ Dá se zdola ohraničit $\Omega(2^{|V|})$ ⇝ exponenciální k $|V|$

6 příklad (2 obarvitelnost souvislých grafů)

Mějme souvislý graf $G = (V, E)$, kde $E \subseteq \{\{u, v\} \mid u, v \in V\}$. Ptáme se, zda existuje obarvení uzlů grafu dvěma barvami tak, aby žádné dva sousedící uzly neměly stejnou barvu.

"Brute-force"algoritmu z minulého příkladu by vedl na exponenciální složitost. Pro dvě obarvitelnost, ale existuje lepší algoritmus.

```
1  Function is_2colorable(V, E, c)
2  |  <−1, 0, 1> color[0, ..., |V| − 1] = [−1, ..., −1] // -1 uncolored
3  |  color[0] = 0
4  |  stack < int > toProcess
5  |  toProcess.push(0)
6  |  while toProcess.nonempty() do
7  |  |  u := toProcess.pop()
8  |  |  foreach v ∈ Adj(u) do
9  |  |  |  if color[v] = −1 then
10 |  |  |  |  color[v] := 1 − color[u]
11 |  |  |  |  toProcess.push(v)
12 |  |  |  else
13 |  |  |  |  if color[v] = color[u] then
14 |  |  |  |  |  return NO
15 |  return YES
```

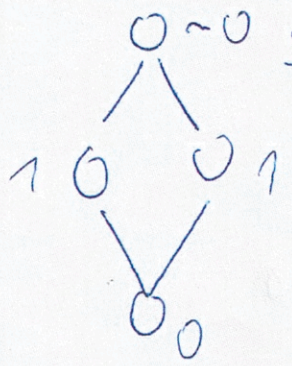
1. Analyzujte asymptotickou složitost algoritmu *is_2colorable* v nelepším případě.

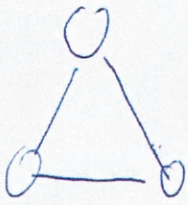
2. Analyzujte asymptotickou složitost algoritmu *is_2colorable* v nejhorším případě.

Poznámka: Předpokládejte uniformní cenové kritérium, kde složitost každého řádku programu je 1.

(6)

Pozorování:

Graf  se dá obarvit 2 barvami,

ale např graf  se nedá obarvit

- Existuje "hladové" kritérium pro obarvování a důkaz, že G obarvit nejde viz alg. is_2colorable() níže.

- nejlepší případ: $O(1)$

stačí projít konstantní část grafu

- nejhorší případ $O(|V|+|E|)$ (pro souvislé G jen $O(|E|)$)
  - musíme jednou projít celý graf

⑦ Rozhodněte a dokažte zda platí: (Předpokládáme, že $P \neq NP$)

a) $\exists$ rekurzivní jazyk L_1 t.ž $L_1 \notin P$

ANO: jakýkoliv NP-těžký jazyk např. SAT → problém splnitelnosti bool. formulí

b) $\exists$ nekonečný jazyk L_2 t.ž. $L_2 \neq \Sigma^* \wedge L_2 \in DTIME[1]$

ANO: Nechť $\Sigma = \{a,b\}$. $L_2 = \{a\}^* \{a,b\}^*$

c) $\exists L_3 \in DSPACE[1] \setminus DTIME[1]$

ANO

1 průchod páskou

d) $\exists L_4 \in DTIME[1] \setminus DSPACE[1]$

NE: $L \in DTIME[1] \Rightarrow L \in DSPACE[1]$

v kterých může zapsat jen na buňky pásky

e) $\exists L_5 \in \mathcal{L}_3 \setminus P$: NE: každý regulární jazyk lze přijmout DTM v lineárním čase