Amortizovaná analýza

- průměrná složitost operace nad datovou strukturou, pokud bereme průměr ze složitostí operací provedených za sebou.

- není to tedy klasická průměrná složitost, pořád jde o složitost v nejhorším případě

Příklady:

1) Zásobník

→ klasické operace POP, PUSH ... $\Theta(1)$

→ přidáme operaci

MULTIPOP(S,k)

while not empty(S) and k>0

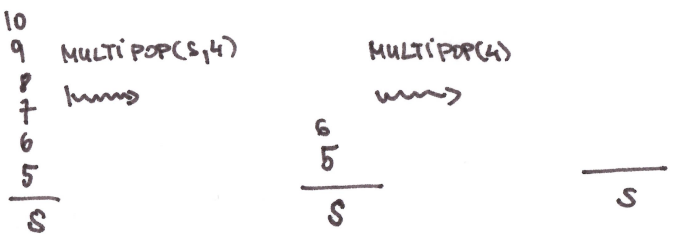
POP(S)

k ← k−1

Složitost: $\min(|S|, k)$, $|S|$ je počet prvků v S

operací POP

Př:



Jaká je složitost posloupnosti n operací? (pop, push, multipop) 1

(když začneme s prázdným zásobníkem)

Hrubá analýza:

nejhorší je MULTIPOP, tady nezmu $O(n)$

máme n operací, celkem je to tedy $O(n^2)$

Přesnější analýza:

uvažujeme celou posloupnost operací

pozorování: každý prvek může být po přidání push tem odebrán max jedenkrát.

Tj. POP může být zavolán pouze tolikrát, kolikrát byl zavolán PUSH. Složitost provedení celé posloupnosti je tedy $O(n)$.

Průměrná složitost jedné operace v posloupnosti je $O(n)/n = O(1)$.

Tj. amortizovaná složitost operací je konstantní.

2) Inkrementování binárního čítače:

- k-bitový binární čítač. Chápeme jako k-prvkové pole bitů A, kde hodnota čítače je

$$x = \sum_{i=0}^{k-1} A[i] \cdot 2^i$$

- na začátku $x = 0$ a pole operací INKREMENT čítač zvyšujeme (ten eventuálně přeteče a začneme znova od 0 → inkrementujeme modulo k).

INKREMENT(A)

```
i ← 0
while length(A) > i and A[i] = 1
    A[i] ← 0
    i ← i+1
if i < length(A)
    A[i] = 1.
```

hodnota \ index	3	2	1	0	počet převrácených bitů AKTUÁLNĚ	počet převrácených bitů CELKEM
0	0	0	0	0		
1	0	0	0	1	1	1
2	0	0	1	0	2	3
3	0	0	1	1	1	4
4	0	1	0	0	3	7
5	0	1	0	1	1	8
6	0	1	1	0	2	10
7	0	1	1	1	1	11

OTÁZKA: Kolik bitů převrátíme během n operací INKREMENT, když začneme z $x = 0$

Hrubá analýza:

jedno zavolání INKREMENT převrátí $O(k)$ bitů,
n zavolání tedy $O(nk)$ bitů.

Jemnější analýza:

bit	jak často je převrácen (po kolika INKREMENTECH)
0	1
1	2
2	4
3	8

Tj. bit i je převrácen každých 2^i INKREMENTů

V posloupnosti n operací je tedy bit i převrácen $\left\lfloor \frac{n}{2^i} \right\rfloor$ krát.

Přitom pro $i > \lfloor \lg n \rfloor$ se bit i nepřevrátí ani jednou. (Pro takové i je zlomek menší než 1)

Celkový počet převrácení je tak

$$\sum_{i=0}^{\lfloor \lg n \rfloor} \left\lfloor \frac{n}{2^i} \right\rfloor < \sum_{i=0}^{\infty} \frac{n}{2^i} \leq n \cdot \sum_{i=0}^{\infty} \frac{1}{2^i} = 2n$$

Amortizovaná složitost operace INKREMENT je tedy $O(n)/n = O(1)$.

U obou příkladů jsme použili agregační metodu

1) spočítáme složitost provedení n operací
2) vydělíme ji n a získáme amortizovanou složitost jednotlivých operací

V důsledku toho mají různé operace stejnou am. složitost. To může být nevýhoda.

V dalších dvou metodách se určují am. složitost jednotlivých operací zvlášť.

Účetní metoda

→ operacím přiřadíme amortizovanou cenu
(= amortizovanou složitost)
→ každý typ operace má jednu cenu.

→ pokud je složitost provedení operace nižší než amortizovaná cena, uložíme si rozdíl jako kredit k nějakému konkrétnímu objektu ve struktuře.

→ později, pokud nějaká operace má amortizovanou cenu menší než skutečnou složitost, lze kredit použít k uhrazení rozdílu (tato operace typicky manipuluje objektem, u kterého je kredit uložen).

skutečné složitosti bereme u konkrétního provedení operace v posloupnosti n za sebou jdoucích operací.

Označme: $\hat{c}_i$ ... amortizovaná cena i-té operace v posloupnosti

c_i .... skutečná složitost i-té operace v posloupnosti

Chceme, aby pro všechny posloupnosti n operací (n je zvoleno libovolně, tedy je to vlastně pro všechna n) platilo

(1) $$\sum_{i=1}^{n} \hat{c}_i \geq \sum_{i=1}^{n} c_i$$

Tj. aby celkový uložený kredit nebyl nikdy záporný.

Uložený kredit se totiž rovná $\sum_{i=1}^{n} \hat{c}_i - \sum_{i=1}^{n} c_i$.

Příklady

1) ZÁSOBNÍK

operace	amortizovaná cena	skutečná složitost
PUSH	2	1
POP	0	1
MULTIPOP	0	min(k, |S|)

Musíme ověřit, že platí (1):

→ každé provedení PUSH uloží do vkládaného prvku kredit velen 1 (viz rozdíl cen v tabulce)

→ POP má rozdíl mezi cenou skutečnou složitostí a cenou rozdíl −1, tento rozdíl je splacen kreditem, který je uložen ve odstraňovaném prvku.

→ MULTIPOP je posloupnost k operací POP, ty už máme podplacený.

Závěr: amortizované složitosti operací jsou $O(1)$.

2) Inkrementování binárního čítače

cena převrácení jídného bitu je 1 (tj. skutečná složitost)
amortizovaná cena operace inkrement je 2

idea: po převrácení bitu $0 \to 1$ vložíme kredit 1 do převráceného bitu.
[tj. jídničkový bit má vždycky kredit 1]

po převrácení bitu $1 \to 0$ uhradíme cenu složitost operace pomocí kreditu uloženého v jídničkových bitech

→ Protože INKREMENT dělá vždy max jídno převrácení $0 \to 1$, má vždy potřebný kredit na vložení do bitu.

→ počet jídničkových bitů je vždy ≥ 0, tj. kredit není nikdy záporný.

Potenciálová metoda

→ velmi podobné účetní metodě / analýza ale začíná jinak
→ kredit se neukládá do objektů ve struktuře, ale do struktury jako celku a říká se mu potenciál

$$D_0 \xrightarrow{OP} D_1 \xrightarrow{OP} D_2 \xrightarrow{OP} D_3 \xrightarrow{OP} D_4 \xrightarrow{OP} \dots$$

D_i ... struktura po provedení i-té operace

$\Phi(D_i)$ ... potenciál struktury po i-té operaci

Pro amortizovanou cenu i-té operace platí

$$\hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1}). \qquad (2)$$

(tj. amortizovaná cena = složitost + změna potenciálu)

Podíváme-li se na celou posloupnost n operací:

$$\sum_{i=1}^{n} \hat{c}_i = \sum_{i=1}^{n} (c_i + \Phi(D_i) + \Phi(D_{i-1}))$$

$$= \left(\sum_{i=1}^{n} c_i\right) + \Phi(D_n) - \Phi(D_0) \qquad (3)$$

Stačí tedy požadovat $\Phi(D_n) \geq \Phi(D_0)$,

ale mezitím by potenciál mohl být záporný.

Obvykle neznáme dopředu n, a specifikují pravidla pro řízení operací.
a pro jistotu chceme $\Phi(D_i) \geq 0$ jako u účetní metody.

Příklady:

1) ZÁSOBNÍK

definujeme $\Phi(D_i)$ = počet prvků v zásobníku po i operacích

Platí $\phi(D_i) \geq \phi(D_0)$ pro všechna i

Dopočítáme amortizované ceny operací:

→ i-tá operace je PUSH, pak

$$\phi(D_i) - \phi(D_{i-1}) = 1$$

a podle rovnice (2) máme

$$\begin{aligned}\hat{c}_i &= c_i + \phi(D_i) - \phi(D_{i-1}) \\ &= 1 + 1 \\ &= 2\end{aligned}$$

→ i-tá operace je MULTIPOP.
Předp. že odstraníme k' prvků ze zásobníku, tj.

$$\phi(D_i) - \phi(D_{i-1}) = -k'$$

Amortizovaná cena je podle (2) rovna

$$\begin{aligned}\hat{c}_i &= c_i + \phi(D_i) - \phi(D_{i-1}) \\ &= k' - k' \\ &= 0\end{aligned}$$

→ i-tá operace je POP

analogicky operaci MULTIPOP, amortizovaná cena je 0.

Amortizované složitosti operací jsou $\Theta(1)$

2) Inkrementování binárního čítače

$\phi(D_i)$ = počet jedniček v counteru
(= počet bitů nastavených na 1)

platí, že

$$\phi(D_i) \geq \phi(D_0)$$

protože tento počet nikdy není záporný a začínáme od 0.

Předp. že i-tá operace INKREMENT přivrátí nejvýše $t_i + 1$ bitů (tj. t_i bitů ve směru $1 \to 0$ a 1 bit ve směru $0 \to 1$)

Pokud je $\phi(D_i) = 0$, pak i-tá operace přivrátila všechny bity, tj. $\phi(D_{i-1}) = k = t_i$. Pokud $\phi(D_i) > 0$, pak $\phi(D_i) = \phi(D_{i-1}) - t_i + 1$.

Tedy:

$$\phi(D_i) - \phi(D_{i-1}) \leq 1 - t_i$$

a amortizovaná cena je

$$\begin{aligned}\hat{c}_i &= c_i + \phi(D_i) - \phi(D_{i-1}) \\ &\leq (t_i + 1) + (1 - t_i) \\ &= 2.\end{aligned}$$

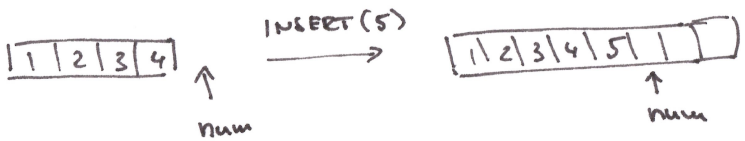
Zvětšující se / zmenšující se tabulka

```
struct table {
    data   // pole
    size   // velikost data
    num    // počet vložených prvků
}
```

INSERT

Složitost:
1 vložení do pole

při vložení do plné tabulky (num = size)
se tabulka zdvojnásobí a původní obsah se zkopíruje



Složitost:
1 vložení do pole
+ kopie původního obsahu

Složitost:
1 vložení do pole

(= mazaný prvek se přepíše posledním prvkem a num se změní o 1)

Složitost:
1 vložení do pole
+ kopie původního obsahu.

Když se po mazání $\frac{num}{size} < \frac{1}{4}$, tabulka se zmenší na polovinu.

Amortizovaná složitost INSERT a DELETE

def: $\alpha(T)$ ... faktor zaplnění tabulky

$$\alpha(T) = \frac{T.num}{T.size}$$

pokud je $T.size = 0$, pak $\alpha(T) = 1$

vždycky tak máme

$$T.num = \alpha(T) * T.size.$$

Použijeme potenciál tabulky T

$$\phi(T) = \begin{cases} 2.T.num - T.size & \alpha(T) \geq 1/2 \\ T.size/2 - T.num & \alpha(T) < 1/2 \end{cases}$$

Pozorování:

1) pokud $T.size = 0$, pak $\phi(T) = 0$.
2) potenciál není nikdy záporný
3) pokud $\alpha(T) = 1/2$, $\phi(T) = 0$
   $\alpha(T) = 1$, $\phi(T) = T.num$
   $\alpha(T) = 1/4$, $\phi(T) = T.num$.

↑ tj. při nutnosti dělat kopii máme dostatek potenciálu.

Analýza sekvence n operací INSERT a DELETE.

(z prázdné tabulky)

c_i ... skutečná složitost i-té operace

$\hat{c}_i$ ... amortizovaná složitost i-té operace

T_i ... tabulka po i-té operaci

značení:

α_i .... $\alpha(T_i)$

num_i ... $(T_i).num$

$size_i$ ... $(T_i).size$

Φ_i .... $\Phi(T_i)$

na začátku máme

$num_0 = size_0 = \Phi_0 = 0$

$\alpha_0 = 1$

↦ INSERT v i-tou operaci: ($num_i = num_{i-1} + 1$)

A) $\alpha_{i-1} \geq 1/2$ $\Rightarrow$ α_i zůst. $> 1/2$

A-1) bez expanze, tj. $size_i = size_{i-1}$

$\hat{c}_i = c_i + \Phi_i - \Phi_{i-1}$

$= 1 + (2 \cdot num_i - size_i) - (2 \cdot num_{i-1} - size_{i-1})$

$= 1 +$ —"— $- 2 \cdot (num_i - 1) - size_{i-1})$

$= 3$

A-2) s expanzí, tj. $size_i = 2 \cdot size_{i-1}$, $size_i = num_{i-1} = num_i - 1$

$\hat{c}_i = c_i + \Phi_i - \Phi_{i-1}$

$= num_i + (2 \cdot num_i - size_i) - 2 \cdot (num_{i-1} - size_i)$

$= 3$

dosadím za $size_i$ zleva $(num_i - 1)$ a upravím

B) $\alpha_{i-1} < 1/2$

B-1) $\alpha_i < 1/2$

$\hat{c}_i = c_i + \Phi_i - \Phi_{i-1}$

$= 1 + (size_i/2 - num_i) - (size_{i-1}/2 - num_{i-1})$

→ dosadím $num_{i-1} = num_i - 1$ a upravím

$= 0$

B-2) $\alpha_i \geq 1/2$

$\hat{c}_i = c_i + \Phi_i - \Phi_{i-1}$

$= 1 + (2 \cdot num_i - size_i) - (size_{i-1}/2 - num_i)$

→ dosadím ~~$num_{i-1} = num_i - 1$~~ $num_i = num_{i-1} + 1$

$size_i = size_{i-1}$

a upravím

$= 3.$

– DELETE i i-tou operaci!

$[num_i = num_{i-1} - 1]$

A) $\alpha_{i-1} < 1/2$

A-1) bez zmenšení: $size_i = size_{i-1}$

$$\hat{c}_i = c_i + \Phi_i - \Phi_{i-1}$$
$$= 1 + (size_i/2 - num_i) - (size_{i-1}/2 - num_{i-1})$$
$$= 2$$

A-2) se zmenšením: $size_i = size_{i-1}/2$

$num_{i-1} = size_{i-1}/4 = num_i + 1$

$$\hat{c}_i = (num_i + 1) + (size_i/2 - num_i) - (size_{i-1}/2 - num_{i-1})$$

dosazení, úprava (a trochu modlení)

$$= 1$$

B) $\alpha_{i-1} \geq 1/2$ ← dva podpřípady $\alpha_i < 1/2$, $\alpha_i \geq 1/2$

pum bez kontraktu, spočítá se nutinně analogicky insertu.