

AMORTIZIRANA ANALIZA

- koristi se za ojeem prosječnog vremena izvršavanja niza operacija nad nekom strukturom podataka, po operaciji
- u nizu u operacija može se dogoditi da je jedna, ili nekoliko njih, izuzetno "skupa" (skupе), dok su sve ostale jeftinije
- što je s prosjekom?
- amortizirana analiza garantira prosječno vrijeme u najgorém slučaju

$T(n)$ = ukupna cijena niza od n operacija NE!
 $T(n)/n$ = prosječna cijena po operaciji

Agregatna analiza

odnosi se na svaku operaciju, iako se u nizu od n operacija može nalaziti nekoliko tipova operacija

Operacije na stogu

PUSH(S, x) - stavlja x na stog S - $O(1)$

POP(S) - uzima element s vrha stoga S - $O(1)$

MULTIPOP(S, k) - uzima k elemenata s vrha stoga S . Ukoliko S sadrži manje od k elemenata, ova metoda prazni stog

MULTIPOP(S, k)

while not STACK-EMPTY(S) and $k > 0$

 | POP(S)
 | $k \leftarrow k - 1$
end while

- $PUSH(S, x) \quad () \rightarrow 1$
- $POP(S) \quad - 1$
- $MULTIPOP(S, k) \quad - \min(s, k) \quad \text{gdje je } s \text{ broj elemenata na stogu}$
 s

Analiza vremena izvršavanja n operacija u nizu na stogu veličine n

- Vrijeme izvršavanja $MULTIPOP$ operacije je $O(n)$ (u najgorem slučaju)
- n operacija $\Rightarrow T(n) = O(n^2) \Rightarrow T(n)/n = O(n)$
- nije oštra ∇
- Zašto

Agregatna analiza

- broj POP operacija (uključujući i $MULTIPOP$ operacije) je u najgorem slučaju jednak broju $PUSH$ operacija
- broj $PUSH$ operacija na stogu veličine n je najviše n

Zaključek: $T(n) = O(n) \Rightarrow T(n)/n = O(n)/n = O(1)$

$T(n)/n$ je amortizirano vrijeme izvršavanja "cijena"

Inkrementiranje binarnog broja

- broj je spremljen kao binarni broj koji se sastoji od k bitova u polju
 $A[0, \dots, k-1]$

vrijednost brojeva x :

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

Inicijalno $x = 0$

operacije: dodavanje 1 (mod 2^k)

Vrijednost	A[7]	A[6]	A[5]	A[4]	A[3]	A[2]	A[1]	A[0]	Ukup cijena
0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	1	1
2	0	0	0	0	0	0	1	0	3
3	0	0	0	0	0	0	1	1	4
4	0	0	0	0	0	1	0	0	7
5	0	0	0	0	0	1	0	1	8
6	0	0	0	0	0	1	1	0	10
7	0	0	0	0	0	1	1	1	11
8	0	0	0	0	1	0	0	0	15
9	0	0	0	0	1	0	0	1	16
10	0	0	0	0	1	0	1	0	18
11	0	0	0	0	1	0	1	1	19
12	0	0	0	0	1	1	0	0	22
13	0	0	0	0	1	1	0	1	23
14	0	0	0	0	1	1	1	0	25
15	0	0	0	0	1	1	1	1	26
16	0	0	0	1	0	0	0	0	31

INCREMENT (A)

$i \leftarrow 0$

while $i < A.length$ and $A[i] == 1$

$A[i] \leftarrow 0$

$i \leftarrow i + 1$

end while

if $i < A.length$

$A[i] \leftarrow 0$

možne vrijednosti
od 0 do $2^k - 1$

Zadatak: Ocjeniti vrijeme izvršavanja u najgoremu slučaju niza od n INCREMENT operacija.

+ jedna INCREMENT operacija: $O(k)$

$$\Rightarrow T(n) = O(nk) \Rightarrow T(n)/n = O(k)$$

Možemo li bolje? (nismo u svakom pozivu promijenili svaki od k bitova)

$A[k]$	$A[k-1]$...	$A[3]$	$A[2]$	$A[1]$	$A[0]$
			Svaka	Svaka	Svaka	Svaka
			osma	četvrta	druga	operacije
			8	4	2	1
			2^3	2^2	2^1	2^0

kupan broj okretanja bitova u n poziva:

$$T(n) = \sum_{i=0}^{k-1} \left\lfloor \frac{n}{2^i} \right\rfloor \leq n \sum_{i=0}^{k-1} \frac{1}{2^i} < n \sum_{i=0}^{\infty} \left(\frac{1}{2}\right)^i = n \cdot \frac{1}{1 - \frac{1}{2}} = 2n = O(n)$$

$$\Rightarrow T(n)/n \leq O(n)/n = O(1)$$

$$\Rightarrow \boxed{T(n)/n = O(1)}$$

Metoda kreditiranja

- Ideja: u nizu od n operacija svakoj pridružimo amortiziranu cijenu kojom ćemo "kreditirati" vrijeme izvršavanja budućih operacija

- Kako?

Operacije na stogu

Stvarna cijene

PUSH 1

POP 1

MULTIPOP $\min(k, s)$

Amortizirane cijene

PUSH 2

POP 0

MULTIPOP 0

Stvarna cijena: c_i
i-te operacije

Amortizirane
cijena i-te
operacije \hat{c}_i

Mora biti:
$$\sum_{i=1}^n \hat{c}_i \geq \sum_{i=1}^n c_i$$

kredit:
$$\sum_{i=1}^n \hat{c}_i - \sum_{i=1}^n c_i \geq 0$$

Razmotrimo postavljanje tanjura na stog:

- Konkretno se plaća stvaranje tanjura na stog, uzimanje jednog i uzimanje k tanjura prema stvarnim cijenama.

Dovoljno je pletiti konkretno sveko stvaranje tanjura $2\#$. Zato

$\text{Push}, \text{Push}, \text{Push}, \text{Pop}, \text{MULTIPop}(2)$
 $n = 5$
 $\uparrow \quad \uparrow \quad \uparrow \quad \uparrow \quad \uparrow$
 $2\$ \quad 2\$ \quad 2\$ \quad 0\$ \quad 0\$$

$$\sum_{i=1}^n \hat{c}_i = 6$$

OPERACIJA	STVARNA	AMORT.	KREDIT (Σ)
Push	1\$	2\$	1\$
Push	1\$	2\$	2\$
Push	1\$	2\$	3\$
Pop	1\$	0\$	2\$
MULTIPop(2)	2\$	0\$	<u>0\$</u>

kredit uvijek
 manja biti
 nenegativan

Na kraju

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

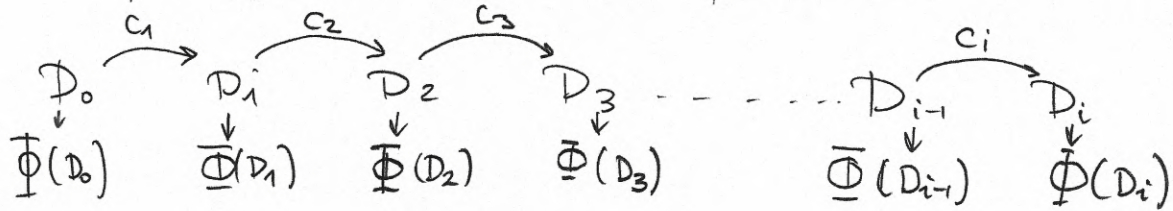
$$\sum_{i=1}^j \hat{c}_i \geq \sum_{i=1}^j c_i \quad \forall j \in \{1, \dots, n\}$$

ne znamo koliko će biti
 operacije una miled \circ

$$\sum \hat{c}_i = O(n)$$

Metoda potencijala

- uvodi se pojam potencijala: koliko plaćamo buduće operacije
n operacija na strukturi podataka \mathcal{D}



amortizirana cijena

$$\hat{c}_i = c_i + \Phi(\mathcal{D}_i) - \Phi(\mathcal{D}_{i-1})$$

$$\begin{aligned} \sum_{i=1}^n \hat{c}_i &= \sum_{i=1}^n (c_i + \Phi(\mathcal{D}_i) - \Phi(\mathcal{D}_{i-1})) \\ &= \sum_{i=1}^n c_i + \sum_{i=1}^n (\Phi(\mathcal{D}_i) - \Phi(\mathcal{D}_{i-1})) \\ &\quad (\cancel{\Phi(\mathcal{D}_1)} - \cancel{\Phi(\mathcal{D}_0)} + \cancel{\Phi(\mathcal{D}_2)} - \cancel{\Phi(\mathcal{D}_1)} + \dots + \Phi(\mathcal{D}_n) - \cancel{\Phi(\mathcal{D}_{n-1})}) \\ &= \sum_{i=1}^n c_i + \Phi(\mathcal{D}_n) - \Phi(\mathcal{D}_0) \end{aligned}$$

Ako je $\Phi(\mathcal{D}_n) \geq \Phi(\mathcal{D}_0)$ onda je $\sum_{i=1}^n \hat{c}_i \geq \sum_{i=1}^n c_i$

Također je potrebno $\Phi(\mathcal{D}_i) \geq \Phi(\mathcal{D}_0) \quad \forall i \in [n]$

jer ne znamo koliko će se operacija izvršiti unaprijed

Operacije na stogu

Φ ← broj elemenata na stogu

$\Phi(\mathcal{D}_0) = 0 \Rightarrow$ odmah sledi $\Phi(\mathcal{D}_i) \geq \Phi(\mathcal{D}_0), \forall i \in [n]$

PUSH

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

$$= 1 + (s+1 - s)$$

$$= 2$$

$$\Phi(D_{i-1}) = s \quad \Phi(D_i) = s+1$$

MULTPOP

$$\Phi(D_{i-1}) = s \ (\geq 0) \quad k' = \min(k, s)$$

$$\Phi(D_i) = s - k' \ (\geq 0)$$

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

$$= \cancel{k'} + s - \cancel{k'} + s$$

$$= 0$$

IP

$$\Phi(D_{i-1}) = s \geq 0$$

$$k' = \min(1, s)$$

$$\Phi(D_i) = s - k'$$

isto

$$\hat{c}_i = 0$$

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

✗ Inkrementiranje binarnog brojača (D.2)

$\Phi(D_i) = b_i \leftarrow$ broj jedinica nakon i -te operacije

$$\Phi(D_0) = 0$$

$$\Phi(D_i) \geq \Phi(D_0) \quad \forall i$$

$$\Phi(D_i) - \Phi(D_{i-1}) \leq (b_{i-1} - 1) + 1$$

Dinamičke tablice (vektori)

- motivacija: dinamički vektor (STL) `vector<...>`

- amortizirano vreme izvršavanja n operacija koje se sastoje od

* INSERT

- ako tablica/vektor ima barem jednu slobodnu lokaciju ($T.\text{num} < T.\text{size}$) onda je $O(1)$

- ako je tablica puna ($T.\text{num} = T.\text{size}$) obujčaj (2 puta) veću, prepisi prvu i ubaci novi element

* DELETE

- obriši element iz tablice stvarajući jednu novu slobodnu memorijsku lokaciju

- ako je broj elemenata u tablici značajno manji od veličine tablice, zamjeniti postojeću tablicu novom.

- faktor ispunjenosti (load factor)

$$\alpha(T) = \begin{cases} 1 & T.\text{num} = 0 \\ \frac{T.\text{num}}{T.\text{size}} & T.\text{num} \neq 0 \end{cases}$$

- želimo postići $\alpha(T) = O(1)$

Proširenje tablice

- potražemo operacijom $\text{TABLE-INSERT}(T, x)$
- Inicijalno, $T.\text{num} = T.\text{size} = 0$

$\text{TABLE-INSERT}(T, x)$

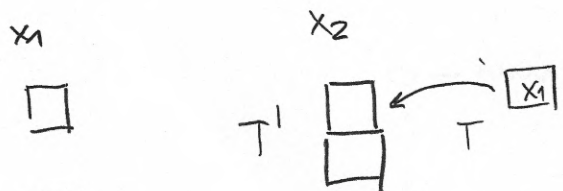
```
if  $T.\text{size} = 0$   
|  
|   alociraj tablicu T veličine 1  
|    $T.\text{size} = 1$   
|  
end if
```

spanzija

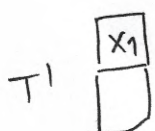
```
if  $T.\text{num} = T.\text{size}$   
|  
|   alociraj tablicu T' veličine  $2 \cdot T.\text{size}$   
|   prepisi sve elemente iz T u T'  
|   dealociraj T  
|    $T \leftarrow T'$  (pointeri)  
|    $T.\text{size} \leftarrow 2 \cdot T.\text{size}$   
|  
end if
```

ubaci x u T
 $T.\text{num} \leftarrow T.\text{num} + 1$

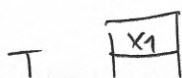
inicijalno



x_1



Hd.



$T.\text{size} = 2$

$T.\text{num} = 1$

Šifarna cijena:

$$c_i = \begin{cases} i, & i-1 = 2^k \text{ za neki } k \\ 1, & \text{inače} \end{cases}$$

Zašto razjasniti na ploči

Agregatna analiza:

$$\sum_{i=1}^n c_i \leq n + \sum_{j=0}^{\lfloor \lg n \rfloor} 2^j < n + \frac{2^{\lg n + 1} - 1}{2 - 1} < n + \frac{2^{n-1} - 1}{1} = 3n - 1 < \underline{\underline{3n}}$$

$$\Rightarrow \sum_{i=1}^n c_i < 3n$$

$$\Rightarrow T(n)/n = O(1)$$

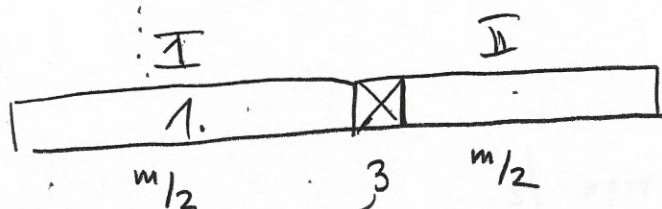
Metoda kreditiranja

ŠIFARNA

TABLE-INSERT	1
TABLE INSERT	1
TABLE INSERT	3 (EXP)
TABLE INSERT	1
TABLE INSERT	5 (EXP)

AMORTIZIRANA

TABLE-INSERT	3
TABLE-INSERT	3
TABLE - - -	3



3 \$

1

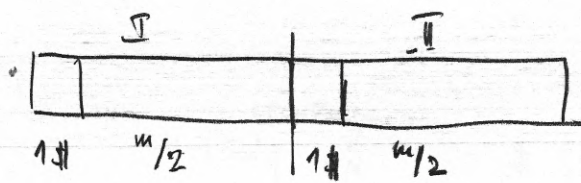
1

1

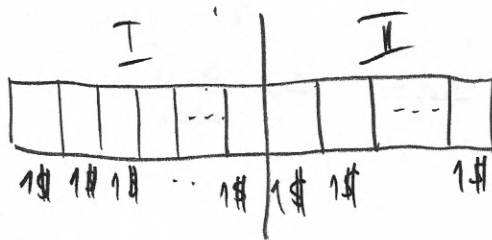
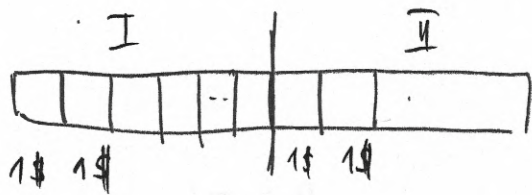
za ubacivanje

kredita za prepisivanje prilikom
slijedeće ekspanzije

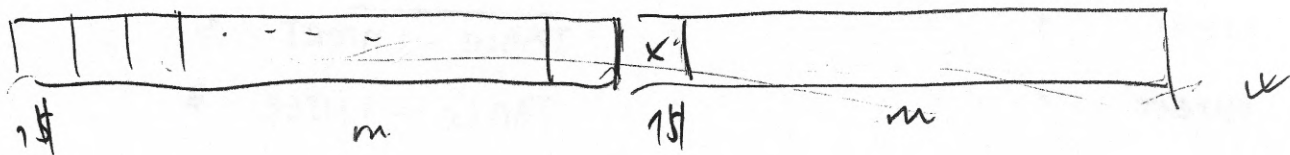
kredita za neki od postojećih
 $m/2$ elemenata u I



1\$ ubacivanju



nekoliko $m/2$ ubacivanja
 Kreditiramo sam cijelu tablicu
 od m elemenata (bloka I + II)
 šta mogu potrošiti na
 prepisivanje



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

$$\sum_{i=1}^n \hat{c}_i = 3n$$

Metoda potencijala

iskazati pomoću T.num i T.size

$$\Phi(T) = 2 \cdot T.\text{num} - T.\text{size}$$

- nakon ekspanzije $T.\text{num} = T.\text{size} / 2$

$$\Phi(T) = 2 \cdot \frac{T.\text{size}}{2} - T.\text{size} = \underline{\underline{0}}$$

- odmah prije ekspanzije $T.\text{num} = T.\text{size}$

$$\Phi(T) = 2 \cdot T.\text{size} - T.\text{size} = \underline{\underline{T.\text{size}}}$$

inicijalno, $\text{num}_0 = 0$ $\text{size}_0 = 0$, $\Phi(T_0) = 0$

$$\hat{c}_i = c_i + \Phi(T_i) - \Phi(T_{i-1})$$

$$\text{num}_i = T_i \cdot \text{num}$$

$$\text{size}_i = T_i \cdot \text{size}$$

- nakon ekspanzije $\text{size}_i = \text{size}_{i-1}$

$$\begin{aligned} \hat{c}_i &= c_i + \Phi(T_i) - \Phi(T_{i-1}) & \text{N} \quad \text{ni} \quad \text{si} \\ &\quad \downarrow \\ &= 1 + (2 \cdot \text{num}_i - \text{size}_i) - (2 \cdot \text{num}_{i-1} - \text{size}_{i-1}) \\ &= 1 + (2 \cdot \text{num}_i - \text{size}_i) - (2 \cdot (\text{num}_i - 1) - \text{size}_i) \\ &= 1 + 2 = 3 // \end{aligned}$$

- neposredno prije ekspanzije

$$\text{size}_i = 2 \text{size}_{i-1}$$

$$\text{size}_{i-1} = \text{num}_{i-1} = \text{num}_i - 1$$

$$\rightarrow \text{size}_i = 2(\text{num}_i - 1)$$

$$\begin{aligned} \hat{c}_i &= c_i + \Phi(T_i) - \Phi(T_{i-1}) \\ &\quad \downarrow \\ &= \text{num}_i + (2 \cdot \text{num}_i - \text{size}_i) - (2 \cdot \text{num}_{i-1} - \text{size}_{i-1}) \\ &= n_i + (2 \cdot n_i - 2(n_i - 1)) - (2(n_i - 1) - (n_i - 1)) \\ &= \cancel{n_i} + (\cancel{2n_i} - \cancel{2n_i} + 2) - (\cancel{2n_i} - 2 - \cancel{n_i} + 1) \\ &= 2 + 2 - 1 = 3 // \end{aligned}$$