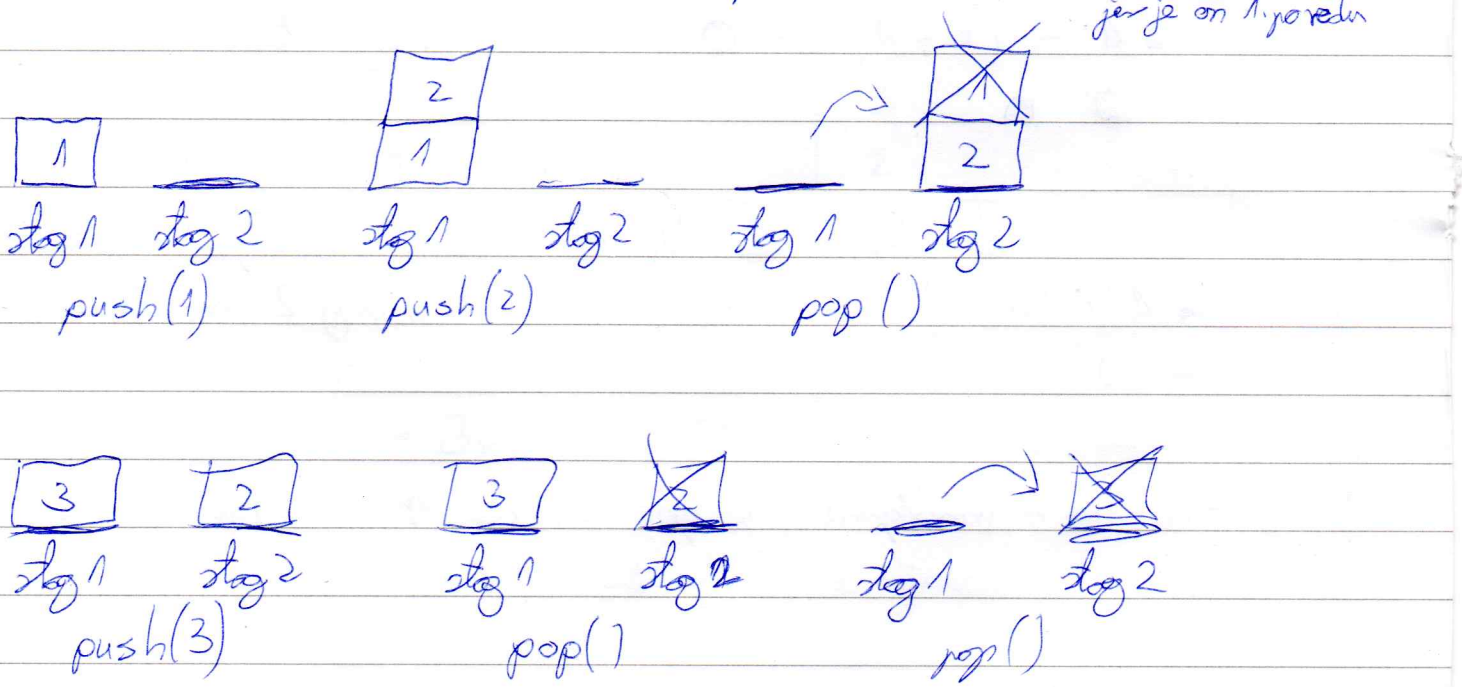
Zoltán Palinkás
Zadaća 5

1)

a); b)

Za rješavanje danog zadatka sam koristio 2 stoga. Moj rad na njegovi način: - elementi se uvijek dodaju u 1. stog (push)
- kada neki element želimo maknuti maknemo sa 2. stoga
- ako je 2. stog prazan elemente iz 1. stoga prebacimo u njega i tek onda izbacimo element (pop)

Slika:



Ka ovaj način kompleksnost od push i pop operacija je O(1) amortizirano

(1) Aggregate method

c)

operacija	stog 1 veličina	stog 2 veličina	cijena
push	1	0	1
push	2	0	1
push	3	0	1
pop	0	2	4
pop	0	1	1
push	1	1	1
pop	1	0	1
pop	0	0	2

push = $O(1)$

c_i = pop < (stog2.size() = 0, stog1.size() + 1 <prebacivanje u stog 2 i pop ; 1, inače)

$pop = \sum_{i=1}^{n} c_i = \underbrace{-1+n}_{\text{broj preostalih elemenata u stogu}} + \underbrace{n+1}_{\text{premještanje i jedan pop}} = 2n$

Jedna pop operacija košta $\frac{O(2n)}{n} = O(1)$

Accounting method

- najlakšimo push amortiziranom cijenom $\hat{c}_i = 3$
  - 1 za push
  - 2 spremamo u banku za potrebe „skupog" pop-a
  - 1 za pop (koda dode do toga)

operacija	stog 1 veličina	stog 2 veličina	banka
push	1	0	2
push	2	0	4
push	3	0	6
pop	0	2	2
pop	0	1	1
push	1	1	3
pop	1	0	2
pop	0	0	0

$$\sum c_i \leq \sum \hat{c}_i = 3n \Rightarrow O(n) \Rightarrow \text{push} = O(1) \; ; \; \text{pop} = O(1)$$

ukupna amortizirana cijena je gornja meda od stvarne cijene

Potential method

$\Phi(D_i)$ - broj elemenata u stogu

• push:

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

$\sum \hat{c}_i = 2n$ za n uzastopnih push op. $\Rightarrow$ push $= O(1)$ amortizirano

• pop:

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

1. slučaj
kada je stog 2 prazan

$$\hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1})$$
$$= 1 + (n-1) - n$$
$$= 0$$

2. slučaj
kada stog 2 nije prazan

$\sum \hat{c}_i = n$ za n uzastopnih pop op. $\Rightarrow$ pop [illegible] $= O(1)$ amortizirano

(2)

b) Korištenje MTF algoritma

- često prilikom pretrage na Googlen pretražujem za osnovu nečega što sam već prije upisivao, pa kada odaberem pojmove, tada se ta preporuka stavi na prvo mjesto prilikom sljedeće pretrage ovo se dogodi kada se koristi i neka druga tražilica
- nisam prorazao kom podatku dati za keširanje koji direktno koriste algoritam, već oni u većini slučajeva koriste „Least Recently Used" (LRU) ili neku varijaciju ovog algoritma kako bi rezultate koji se često koriste držao kako pristupačnima