APS2 – Vaje 3. teden

Boris Radovič

1. Predstavljajte si, da operacije razširjenega sklada dopolnemo z operacijo *MULTIPUSH*, ki na sklad doda k elementov. Bi v tem primeru imela posamezna operacija razširjenega sklada še vedno amortizirano ceno $O(1)$?

   Ne. Lahko izvršimo zeporedje *MULTIPUSH(n)* kateremu sledi *MULTIPOP(n)* – n je maksimalna velikost sklada. Vsaka operacija ima časovno zahtevnost natanko $\Theta(n)$, saj moramo n krat klicati *PUSH* oz. *POP*. Ker imajo vse operacije kompleksnost $\Theta(n)$, je povprečna amortizirana kompleksnost $\Theta(n)$.

2. Zapišite psevdokodo operacije *ZMANJŠAJ*, ki število zapisano v dvojiškem zapisu zmanjša za 1. Ima binarni števec z operaciji *POVEČAJ* in *ZMANJŠAJ* še vedno amortizirano ceno $O(1)$?

   Prevdokoda

   ```
   i=0
   while i < k  &  A[i] = 0 do
       A[i]=1
       i=i+1
   end while
   if i < k then
       A[i]=0
   end if
   ```

   Časovna analiza: V primeru, da ima števek začetno vrednost 2^k in zaporedno kličemo *ZMANJŠAJ*, *POVEČAJ*, je časovna kompleksnost $\Theta(nk)$

3. Predstavljajte si, da opravimo n operacij na podatkovni strukturi, kjer i-ta operacija stane i natanko tedaj, ko velja $i = 2^k$, za poljuben $k \in N$. V ostalih primerih je cena operacije 1. Z amortizirano analizo natančno določite amortizirano ceno omenjene operacije.

   $$\sum_{i=1}^{n} c(i) = \sum_{i=1}^{\lfloor \log_2 n \rfloor} 2^i + \sum_{i<n,\ \text{i ni potenca stevila 2}} 1 \leq \sum_{i=1}^{\lfloor \log_2 n \rfloor} 2^i + n = 2^{\lfloor \log_2 n \rfloor + 1} - 2 + n$$

   Sledi:

   $$\sum_{i=1}^{n} c(i) < 2n - 2 + n = O(n)$$

Torej amortizirana analize n operacij je $O(1)$
Pri tem smo uporabili formulo za geometricno serijo, $\sum_{i=0}^{n} x^i = \frac{x^{n+1}-1}{x-1}$

4. Z amortizirano analizo po metodi kopičenja rešite nalogo 3.

   Nastavimo, da ima vsaka operacija ceno 3. V primeru, da i ni potenca števila dva ima operacija ceno 1 in se dva kredita shranijo kot kredit. Moramo pokazati, da je kredit vedno ne-negativen. To lahko pokažemo z indukcijo: za $i = 1 = 2^0$ je kredit očitno pozitiven, saj cena operacije je 1 medtem ko amortizirana cena je 3, torej je vrednost kredita $2 \geq 0$. Predpostavimo sedaj, da imamo dovolj kredita, da smo uspeli plačati za operacijo $i = 2^k$ za nek k. Bomo imeli dovolj kredita za placat za operacijo 2^{k+1}? Med 2^k in 2^{k+1} je natanko 2^k operacij. Za vsako to operacijo bomo shranili 2 kredita, torej je skupno število shranjenih kreditov $2 \cdot 2^k = 2^{k+1}$. To je natanko cena 2^{k+1}-te operacije. Sledi, da je kredit vedno ne-negativen.

5. Napišite funkcijo DODAJ, ki dinamični tabeli doda element. Če je tabela polna, se tabela poveča za faktor 2. Tabela T ima atribute `table`, ki vsebuje kazalec na podatke, `size`, ki vsebuje velikost tabele, in `num`, ki vsebuje število elementov trenutno v tabeli.

6. Dinamični tabeli podvojimo velikost vsakič, ko je polna. Zato da ne bi bila preveč prazna, tabeli razpolovimo velikost, če je več kot polovico polna. Je to dobra strategija?
   Ne, amortizirana kompleksnost je lahko $O(n)$ za vsako operacijo.