Tretje vaje APS2: amortizirana časovna zahtevnost

(1) Na predavanjih smo elemente v dinamično tabelo dodajali tako, da smo ob vsaki napolnitvi trenutne tabele izdelali novo tabelo, ki je bila dvakrat večja od trenutne, in elemente iz trenutne tabele prestavili v novo tabelo. Tabelo smo tako efektivno »povečali« za faktor 2.

Recimo, da tabelo namesto tega vsakokrat povečamo za natanko 10 elementov. Kolikšna bo po novem amortizirana časovna zahtevnost dodajanja posameznega elementa? Na vprašanje odgovorite z uporabo vseh treh metod amortizirane časovne zahtevnosti. □

- Po *agregatni metodi* izračunamo amortizirano časovno zahtevnost tako, da skupno časovno zahtevnost izvedbe n operacij (za dovolj velik n) delimo z n. Recimo torej, da v tabelo zaporedoma dodamo n elementov. Zadnjih $(n-1) \bmod 10 + 1$ elementov bomo v tabelo dodali le enkrat. Predzadnjih 10 elementov bomo dodali oz. prestavili dvakrat, saj jih bomo morali po raztegu tabele prestaviti v novo tabelo. Predpredzadnjih 10 elementov bomo dodali oz. prestavili trikrat. Skupno število dodajanj oz. premikov elementov bo torej enako

$$\begin{aligned}
&((n-1) \bmod 10 + 1) + 10 \cdot 2 + 10 \cdot 3 + \ldots + 10 \cdot \lceil n/10 \rceil \\
&\leqslant 10 \cdot (1 + 2 + \ldots + \lceil n/10 \rceil) \\
&= 10 \cdot \frac{1}{2} \lceil n/10 \rceil (\lceil n/10 \rceil + 1) \\
&\leqslant 5(n/10 + 1)(n/10 + 2) \\
&= O(n^2),
\end{aligned}$$

  amortizirana časovna zahtevnost za premik enega elementa pa je potemtakem $O(n)$. Zlahka vidimo, da velja tudi $\Omega(n^2)$ za n operacij in $\Omega(n)$ za eno samo, zato je dobljena zgornja meja tesna.

- Po *računovodski metodi* vsaki operaciji (v tem primeru je ena sama) dodelimo računovodsko ceno, in to tako, da je skupna računovodska cena poljubnega zaporedja operacij vedno večja ali enaka dejanski ceni tega zaporedja. Če nam to uspe, lahko računovodsko ceno zaporedja (ki jo je običajno enostavno izračunati) uporabimo kot zgornjo mejo dejanske cene zaporedja (ki jo je marsikdaj bistveno težje izračunati).

  Kako bi določili računovodsko ceno dodajanja elementa v tabelo? Recimo, da tabela vsebuje 50 elementov in da smo jo ravnokar (tik pred vselitvijo 51. elementa) povečali na kapaciteto 60. To pomeni, da se bo znova povečala po 10 dodanih elementih. Naslednjih 10 dodanih elementov mora torej s skupnimi močmi plačati

  - lastno vselitev v tabelo;
  - lastno prestavitev v novo, večjo tabelo;
  - prestavitev obstoječih 50 elementov v večjo tabelo.

  Če se osredotočimo na en sam element, ugotovimo, da mora plačati lastno vstavitev (1 enota), lastno prestavitev (1 enota) in prestavitev pripadajočega deleža (torej $50/10 = 5$) obstoječih elementov. Če razmišljanje posplošimo na tabelo z n elementi, lahko računovodsko ceno vstavitve posamičnega elementa določimo kot

$$\hat{c} = 1 + 1 + \lfloor n/10 \rfloor = 2 + \lfloor n/10 \rfloor$$

Pri takšni določitvi računovodske cene zagotovimo, da so vsi raztegi tabele plačani vnaprej. To pomeni, da bo za poljubno zaporedje operacij veljalo, da je njegova dejanska cena kvečjemu enaka računovodski, zato lahko računovodsko ceno vzamemo kot zgornjo mejo dejanske cene. Časovna zahtevnost dodajanja n elementov je potemtakem $O(n^2)$, amortizirana časovna zahtevnost posameznega dodajanja pa je $O(n)$ (v resnici $\Theta(n)$, a glede na to, da je računovodska cena le zgornja meja dejanske, je bolj varno pisati $O(n)$).

- *Potencial* tabele določimo na podoben način kot na predavanjih. Ko je tabela prazna in takoj zatem, ko se poveča (tik preden se v tabelo doda element, ki je povzročil razteg), je potencial enak 0. Ko elemente v tabelo dodajamo, se potencial postopoma povečuje, tako da bo tik pred raztegom tabele dovolj velik, da bo lahko plačal celotno ceno prestavitve obstoječih elementov v novo tabelo. Potencial se mora torej spreminjati po vzorcu 1, 2, 3, ..., 9, 10, 2, 4, 6, ..., 18, 20, 3, 6, 9, ..., 27, 30 itd. Potencial tabele z n elementi lahko torej zapišemo takole:
$$\Phi(n) = \lceil n/10 \rceil ((n-1) \bmod 10 + 1).$$

  Kakšno računovodsko ceno določa tako definiran potencial? Poglejmo:

  - Ena možnost je, da dodajanje $(n+1)$-ega elementa ne povzroči raztega. To se zgodi, ko n ni večkratnik števila 10. V tem primeru je dejanska cena enaka 1, saj je treba element zgolj vstaviti, računovodsko ceno pa izračunamo takole:
$$\hat{c} = c + \Phi(n+1) - \Phi(n) = 1 + \lceil n/10 \rceil \cdot 1 = 2 + \lfloor n/10 \rfloor.$$

  - Če dodajanje $(n+1)$-ega elementa povzroči razteg, torej če je n večkratnik števila 10, je dejanska cena enaka $1+n$, saj je treba pred vstavitvijo elementa vse obstoječe elemente prestaviti v novo tabelo. V tem primeru dobimo takšno računovodsko ceno:
$$\hat{c} = c + \Phi(n+1) - \Phi(n) = 1 + n + ((n/10) + 1) - n = 2 + n/10 = 2 + \lfloor n/10 \rfloor.$$

  Vidimo, da v obeh primerih dobimo natanko takšno računovodsko ceno, kot smo jo definirali v prejšnji alineji.

(2) Dokažite, da za vse $x \in (0,1)$ velja
$$\sum_{i=1}^{\infty} i x^{i-1} = \frac{1}{(1-x)^2}.$$

Pomagajte si z dobro znanim dejstvom, da je odvod funkcije x^i po spremenljivki x enak ix^{i-1}. □

$$\sum_{i=1}^{\infty} i x^{i-1} = \sum_{i=1}^{\infty} \frac{d}{dx} x^i = \frac{d}{dx} \sum_{i=1}^{\infty} x^i = \frac{d}{dx} \frac{1}{1-x} = \frac{1}{(1-x)^2}.$$

(3) Tabelo z $n = 2^k - 1$ elementi za nek $k \geqslant 1$ bi radi preoblikovali v kopico. Kolikšna je amortizirana časovna zahtevnost obravnave enega elementa? □

Tabelo preoblikujemo v kopico tako, da pričnemo s predzadnjim nivojem, ki vsebuje $n/4$ elementov (v resnici $\lceil n/4 \rceil$, a tokrat si brez škode lahko malenkost poenostavimo

življenje[1]), in na vsakem od njih izvedemo operacijo pogrezanja. V najslabšem primeru se bo vsak od teh elementov zamenjal z enim od svojih otrok (torej se bo enkrat pogreznil), kar pomeni, da bomo imeli $n/4$ zamenjav. Na predpredzadnjem nivoju imamo $n/8$ elementov, vsak od njih pa se bo v najslabšem primeru pogreznil dvakrat. Na predpredpredzadnjem nivoju se bo vsak od $n/16$ elementov pogreznil kvečjemu trikrat. Na prvem nivoju se bo edini element pogreznil kvečjemu $(k = (\lg n) - 1)$-krat (vnovič bi morali pisati $\lceil \lg n \rceil$). Skupno število zamenjav bo potemtakem kvečjemu

$$\begin{aligned}
&\frac{n}{4} \cdot 1 + \frac{n}{8} \cdot 2 + \frac{n}{16} \cdot 3 + \frac{n}{32} \cdot 4 + \ldots + 1 \cdot (\lg n - 1) \\
&= \sum_{i=1}^{\lg n - 1} \frac{n}{2^{i+1}} i \\
&= n \sum_{i=1}^{\lg n - 1} \frac{i}{2^{i+1}} \\
&= \frac{n}{4} \sum_{i=1}^{\lg n - 1} \frac{i}{2^{i-1}} \\
&= \frac{n}{4} \sum_{i=1}^{\lg n - 1} i \left(\frac{1}{2}\right)^{i-1} \\
&\leqslant \frac{n}{4} \sum_{i=1}^{\infty} i \left(\frac{1}{2}\right)^{i-1} \\
&= \frac{n}{4} \frac{1}{(1 - \frac{1}{2})^2} \quad \text{(uporabimo formulo iz prejšnje naloge za } x = 1/2) \\
&= n,
\end{aligned}$$

amortizirana časovna zahtevnost obravnave enega elementa pa znaša $O(1)$.

(4) Števila $1, 2, \ldots, n = 2^k - 1$ uredimo naraščajoče po zrcalni sliki njihovih dvojiških predstavitev (npr. 4, 2, 6, 1, 5, 3, 7 za $k = 3$) in jih zaporedoma dodamo v navadno dvojiško iskalno drevo. Če malce pomislimo, bomo na ta način dobili drevo s k polno zapolnjenimi nivoji: v korenu drevesa bo pristalo število z dvojiškim zapisom 10^{k-1}, ki je ravno na sredini originalnega zaporedja, njegova otroka bosta števili z dvojiškim zapisom 010^{k-2} oz. 110^{k-2}, ki sta spet na sredini pripadajočih podzaporedij, itd. Kolikšna je amortizirana časovna zahtevnost dodajanja posameznega elementa? □

Razmišljamo podobno kot pri prejšnji nalogi, rezultat pa bo drugačen. Štejmo primerjave elementov. Pri dodajanju korena drevesa ne potrebujemo nobene primerjave. Za drugi nivo potrebujemo $2 \cdot 1 = 2$ primerjavi. Da zgradimo tretji nivo, potrebujemo $4 \cdot 2 = 8$ primerjav. Skupno število primerjav bo potemtakem

$$T(n) = 2 \cdot 1 + 4 \cdot 2 + 8 \cdot 3 + \ldots + \left\lceil \frac{n}{2} \right\rceil (k - 1).$$

Enostavneje bo, če vsoto beremo od zadaj naprej (prvi faktorji posameznih členov bodo tako $\lceil n/2 \rceil$, $\lceil n/4 \rceil$, $\ldots$), prav tako pa si bomo poenostavili življenje in se znebili operatorja $\lceil . \rceil$. Torej:

$$T(n) = \frac{n}{2}(k - 1) + \frac{n}{4}(k - 2) + \ldots + 2 \cdot 1$$

[1]Za matematično sitne: $\lceil x \rceil \leqslant x + 1$, to pa je manjše ali enako od $2x$, če je x vsaj 1, zato takšna poenostavitev ne bo vplivala na asimptotično časovno zahtevnost.

$$
\begin{aligned}
&= \sum_{i=1}^{k} \frac{n}{2^i}(k-i) \\
&= n \sum_{i=1}^{k} \frac{k-i}{2^i} \\
&= n \left(\sum_{i=1}^{k} \frac{k}{2^i} - \sum_{i=1}^{k} \frac{i}{2^i} \right) \\
&= nk \sum_{i=1}^{k} (1/2)^i - \frac{n}{2} \sum_{i=1}^{k} i(1/2)^{i-1}.
\end{aligned}
$$

Prva vsota je manjša od $\sum_{i=0}^{\infty}(1/2)^i = \frac{1}{1-1/2} = 2$, druga pa od $\sum_{i=1}^{\infty} i(1/2)^{i-1} = \frac{1}{(1-1/2)^2} = 4$ (gl. nalogo 2). Upoštevamo še $k = \lceil \lg n \rceil$ in dobimo

$$T(n) \leqslant n(\lg n)\Theta(1) - n\Theta(1) = O(n \log n)$$

Amortizirana časovna zahtevnost dodajanja posamičnega elementa je potemtakem $O(\log n)$.