Algoritmi in podatkovne strukture 2

Amortizacijska analiza

Luka Fürst

Vsakdanja motivacija

- S taksijem se peljemo 10 km
- 10 evrov za pričetek vožnje
- 5 evrov za vsak kilometer
- Skupaj 60 evrov
- Amortizirana cena
  - 6 evrov na kilometer
  - začetnih 10 evrov porazdelimo po 10 km vožnje

Programerska motivacija

- Dinamična tabela (`vector` v C++ ali `ArrayList` v javi)
- Dodajanje elementa na konec je večinoma poceni ($\Theta(1)$)
- Včasih pa moramo tabelo »raztegniti«
  - izdelamo novo, večjo tabelo
  - elemente stare tabele premaknemo v novo tabelo
- V takih primerih potrebujemo $\Theta(n)$ časa
- Koliko časa potrebujemo v povprečju?
- Kako načrtovati raztezanje, da bo povprečni čas asimptotično čim manjši?

Amortizacijska analiza

- Agregatna metoda
  - skupni čas, ki ga v najslabšem možnem primeru potrebujemo za izvedbo n operacij, delimo z n
- Računovodska metoda
  - različnim operacijam dodelimo različne računovodske cene
  - če zagotovimo, da je vsota računovodskih cen poljubnega zaporedja n operacij vedno večja ali enaka vsoti dejanskih cen teh operacij, lahko vsoto računovodskih cen uporabimo kot zgornjo mejo porabe časa n operacij
- Metoda potencialov
  - podatkovni strukturi pripišemo potencial, ki predstavlja sposobnost plačevanja za bodoče operacije

Primer 1: sklad z dodano operacijo Odvzemi-več

- Dodaj(x)
  - dodaj element x na vrh sklada
- Odvzemi()
  - odstrani vrhnji element s sklada (če sklad ni prazen)
- Odvzemi-več(k)
  - odstrani vrhnjih k elementov s sklada (oz. izprazni sklad, če ima kvečjemu k elementov)

- Kolikšna je časovna zahtevnost zaporedja poljubnih n operacij v najslabšem primeru?

Naivna metoda

- Dejanske cene posameznih operacij

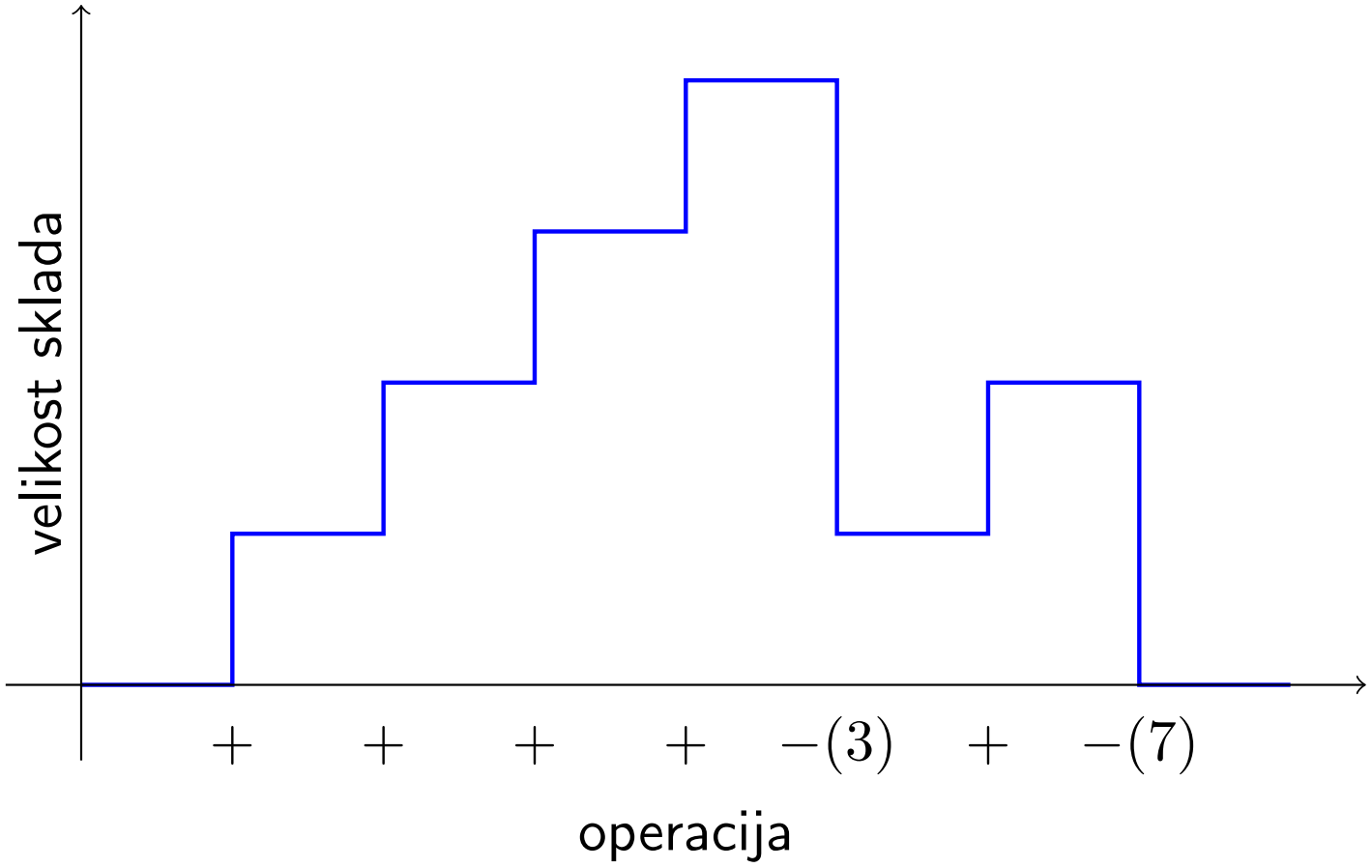
Operacija	Dejanska cena
$\textsc{Dodaj}(x)$	1
$\textsc{Odvzemi}()$	1
$\textsc{Odvzemi-več}(k)$	$\min(\|S\|, k)$

- Zaporedje n operacij lahko vsebuje n operacij $\textsc{Odvzemi-več}$
- Je časovna zahtevnost zaporedja n operacij torej $O(n^2)$?

Agregatna metoda

- n operacij $= O(n^2)$?
- Da, to je zgornja meja, a ni tesna
- Tesna zgornja meja je $O(n)$
  - element lahko odstranimo le, če ga prej dodamo
  - posamezna operacija ODVZEMI-VEČ je lahko časovno zahtevna, a take operacije so redke
  - izvedljive so le v primeru zadostnega števila elementov na skladu

Agregatna metoda



- Skupna cena $= 5 + 5 \leq 2 \cdot$ število operacij

Agregatna metoda

- Poljubno zaporedje n operacij v najslabšem primeru traja $O(n)$ časa
- Amortizirana časovna zahtevnost posamične operacije je potemtakem

$$\frac{O(n)}{n} = O(1)$$

Računovodska metoda

- Namesto dejanskih cen posameznih operacij (c_i) vzamemo njihove »računovodske cene« ($\hat{c}_i$)
- Računovodska cena
  - $\neq$ dejanska cena
  - pripomoček, ki nam lahko poenostavi analizo
  - nekaterim operacijam lahko dodelimo višjo računovodsko ceno od dejanske, drugim pa manjšo
  - razlika med računovodsko in dejansko ceno je »predplačilo« za neko operacijo v prihodnosti
- Terminologija
  - uveljavljen izraz je amortizirana cena (angl. amortized cost), vendar pa ima ta pojem v kontekstu računovodske metode drugačen pomen kot pri agregatni metodi, zato sem se odločil za nov izraz

Računovodska metoda

- Če za poljubno zaporedje n operacij zagotovimo

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

  potem je $\sum_{i=1}^{n} \hat{c}_i$ veljavna zgornja meja časovne zahtevnosti n operacij
- Amortizirana časovna zahtevnost posamezne operacije je potem $\sum_{i=1}^{n} \hat{c}_i \,/\, n$

Računovodska metoda

Operacija	Dejanska cena	Računovodska cena
$\textsc{Dodaj}(x)$	1	2
$\textsc{Odvzemi}()$	1	0
$\textsc{Odvzemi-več}(k)$	$\min(\|S\|, k)$	0

- Ko element postavimo na sklad, plačamo 2 enoti
  - 1 enoto stane postavitev na sklad
  - 1 enoto pustimo na elementu, da lahko kasneje plačamo njegovo odstranitev (predplačilo odstranitve)

Računovodska metoda

- Ker je $\hat{c}(\textsc{Dodaj}) = 2$, $\hat{c}(\textsc{Odvzemi}) = 0$ in $\hat{c}(\textsc{Odvzemi-več}) = 0$, je računovodska cena poljubnega zaporedja n operacij enaka kvečjemu $2n$
- Ker morebitno odstranitev elementa plačamo že ob njegovi postavitvi na sklad, velja $\sum_{i=1}^{n} \hat{c}_i \geq \sum_{i=1}^{n} c_i$
- Za poljubno zaporedje operacij torej velja $\sum_{i=1}^{n} c_i \leq 2n$
- Amortizirana časovna zahtevnost posamezne operacije je potemtakem $\sum_{i=1}^{n} c_i \,/\, n = O(n) \,/\, n = O(1)$

Metoda potencialov

- Podobna računovodski metodi, vendar pa je nakopičeno predplačilo shranjeno na celotni podatkovni strukturi D v obliki potenciala $\Phi(D)$
- Potencial predstavlja sposobnost plačevanja prihodnjih operacij
- Potencial začetne podatkovne strukture (npr. praznega sklada) ponavadi nastavimo na 0
- Zagotoviti moramo, da potencial nikoli ne pade pod 0
  - Na ta način imamo vedno dovolj »denarja« za plačilo operacij

Metoda potencialov

- Operacija, pri kateri je računovodska cena ($\hat{c}_i$) višja od dejanske (c_i), poveča potencial strukture (in obratno)
- S povečanim potencialom bomo kasneje lahko plačali operacijo, pri kateri je računovodska cena manjša od dejanske

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

- Logiko lahko tudi obrnemo in rečemo, da potencialna funkcija določa računovodsko ceno

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

Metoda potencialov

- Potencial sklada = število elementov na njem
- Postavitev na sklad ima računovodsko ceno 2 enoti
  - sama postavitev stane 1 enoto
  - z drugo enoto povečamo potencial za 1
- Odvzem s sklada ima računovodsko ceno 0
  - plačamo ga s potencialom (ki se zato zmanjša za 1)

Metoda potencialov

- Potencial, kot smo ga definirali, je usklajen z računovodskimi cenami, ki smo jih določili pri računovodski metodi

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

- Postavitev
  - $c_i = 1$ (dejanska cena)
  - $\Phi(D_i) - \Phi(D_{i-1}) = 1$
  - $\hat{c}_i = 2$
- Odvzem
  - $c_i = 1$ (dejanska cena)
  - $\Phi(D_i) - \Phi(D_{i-1}) = -1$
  - $\hat{c}_i = 0$

Primer 2: povečevanje dvojiškega števca

- k-mestni dvojiški števec z začetno vrednostjo 0
- n-krat ga povečamo za 1
- Primer za $k = 4$:

$$0000 \to 0001 \to 0010 \to 0011 \to 0100 \to \ldots$$
$$\ldots \to 1110 \to 1111 \to 0000 \to 0001 \to \ldots$$

function POVEČAJ-ŠTEVEC(B)
    $i \leftarrow |B| - 1$
    while $i \geq 0 \wedge B[i] = 1$ **do**
        $B[i] \leftarrow 0$
        $i \leftarrow i - 1$
    if $i \geq 0$ **then**
        $B[i] \leftarrow 1$

Naivna metoda

- Dejanska cena spremembe bita = 1
- Posamezno povečanje števca lahko spremeni vseh k bitov
- Časovna zahtevnost n zaporednih povečanj je potem $O(nk)$
- To je sicer pravilna zgornja meja, a ni tesna
- Včasih bomo res morali spremeniti veliko bitov, a to se ne bo prav pogosto zgodilo

Agregatna metoda

Bit 0	Bit 1	Bit 2	Bit 3
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
...	...	...	...

- Bit $k - i - 1$ se spremeni pri vsakem 2^i-tem povečanju

Agregatna metoda

- Pri n zaporednih povečanjih
  - n sprememb bita $k-1$
  - $n/2$ sprememb bita $k-2$
  - $n/4$ sprememb bita $k-3$
  - $n/8$ sprememb bita $k-4$
  - ...

$$\sum_{i=0}^{k-1} \frac{n}{2^i} = n \sum_{i=0}^{k-1} \left(\frac{1}{2}\right)^i \leq n \sum_{i=0}^{\infty} \left(\frac{1}{2}\right)^i = n \frac{1}{1-1/2} = 2n = O(n)$$

- Amortizirana časovna zahtevnost
  - $O(n) \,/\, n = O(1)$

Računovodska metoda

- Če želimo bit spremeniti na 0, ga moramo najprej spremeniti na 1
- Spremembi $0 \to 1$ damo računovodsko ceno 2
- Spremembi $1 \to 0$ damo računovodsko ceno 0
- Pri spremembi $0 \to 1$
  - 1 enoto stane sama sprememba
  - 1 enoto pustimo na bitu, da bomo kasneje z njo plačali spremembo na 0
- Primer
  - povečanje $00111 \to 01000$ ima dejansko ceno 4, računovodska cena pa je 2
  - spremembe $1 \to 0$ smo že plačali v predhodnjih povečanjih

Računovodska metoda

n	Števec	$\sum_{i=0}^{n} c_i$	$\sum_{i=0}^{n} \hat{c}_i$
0	00000	0	0
1	00001	1	2
2	00010	3	4
3	00011	4	6
4	00100	7	8
5	00101	8	10
6	00110	10	12
7	00111	11	14
8	01000	15	16
...	...	...	...

Računovodska metoda

- Vsako povečanje ima računovodsko ceno 2
- n zaporednih povečanj ima torej skupno računovodsko ceno $2n$
- Čeprav je računovodska cena posameznega povečanja lahko manjša od dejanske, je računovodska cena zaporedja n povečanj (če začnemo z vrednostjo števca 0) vedno najmanj enaka dejanski ceni tega zaporedja
- Za vsak n torej velja

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

- Računovodska cena zaporedja povečanj je potemtakem zgornja meja dejanske cene, zato je amortizirana časovna zahtevnost posameznega povečanja enaka $O(n) \,/\, n = O(1)$

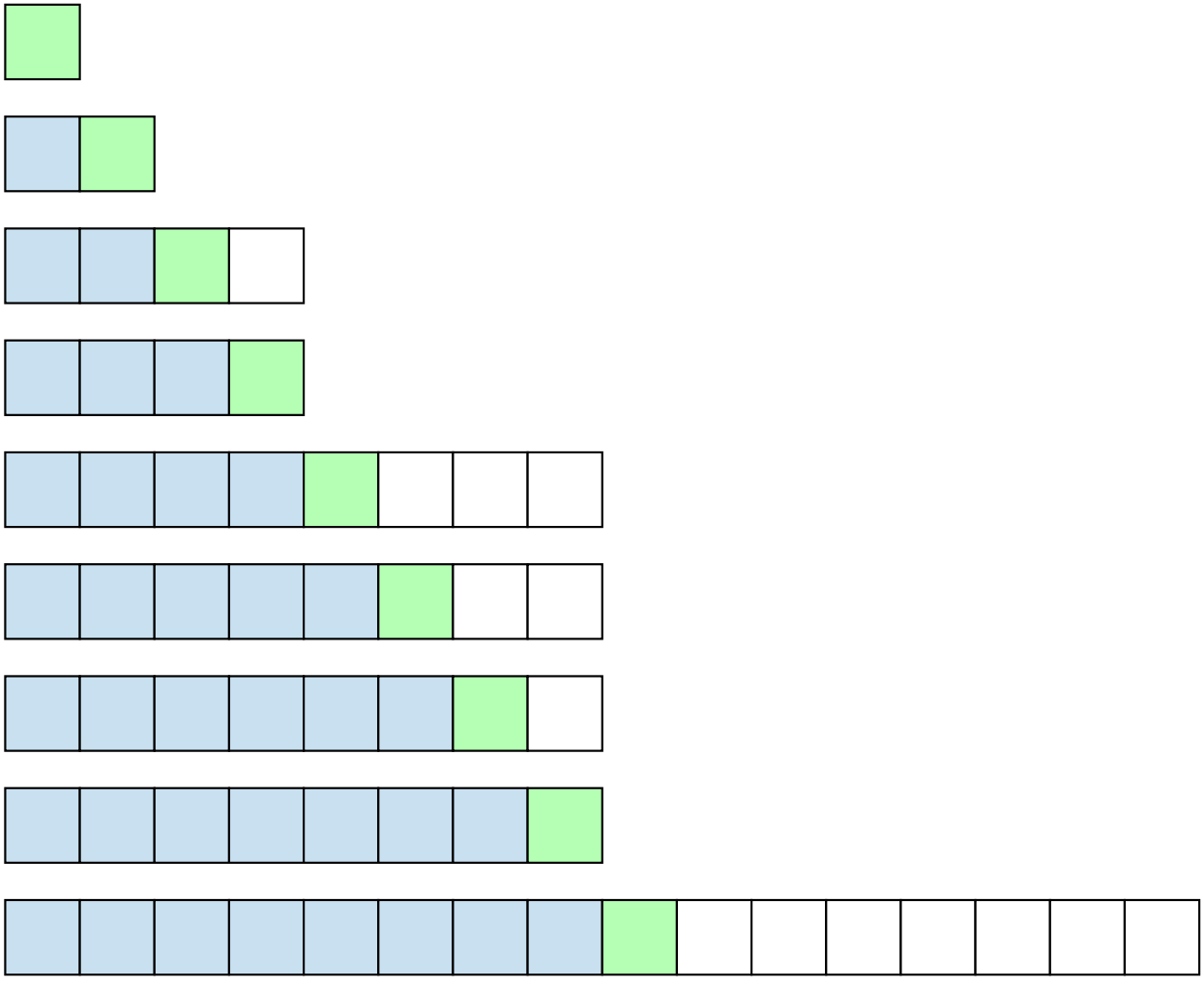
Metoda potencialov

- Potencial = število bitov 1
- Je začetni potencial enak 0?
  - Da!
- Je potencial vedno nenegativen?
  - Da!
- Tudi tokrat je potencial usklajen z računovodsko ceno
  - Recimo, da i-to povečanje izvede t sprememb $1 \to 0$ in eno spremembo $0 \to 1$
  - $c_i = t + 1$
  - $\Phi(D_i) - \Phi(D_{i-1}) = 1 - t$
  - $\hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1}) = (t + 1) + (1 - t) = 2$

Primer 3: »razteglјiva« tabela

- Tabela ima določeno kapaciteto (število celic)
- Ko se zapolni, se ustvari nova, večja tabela, in elementi se vanjo prenesejo
- Recimo, da ima nova tabela vsakokrat 2-krat večjo kapaciteto od obstoječe

Primer 3: »razteglјiva« tabela



Primer 3: »razteglјiva« tabela

```
function DODAJ(R, x)
    ▷ R.n: dejansko število elementov; R.A: dejanska tabela
    if R.n = 0 then
        R.A ← nova tabela dolžine 1
    else if R.n = |R.A| then
        B ← nova tabela dolžine 2|R.A|
        for i ← 0 : R.n − 1 do
            B[i] ← R.A[i]
        R.A ← B
    R.A[R.n] ← x
    R.n ← R.n + 1
```

Primer 3: »razteglјiva« tabela

- Dodajanje elementa je večinoma $O(1)$, včasih pa lahko povzroči raztezanje, ki traja $O(n)$
- Kolikšna je amortizirana časovna zahtevnost?

Agregatna metoda

- Recimo, da smo v tabelo dodali n elementov
- Poleg n nezahtevnih vselitev elementa v tabelo smo tabelo še največ $\lfloor \lg n \rfloor$-krat raztegnili
- V i-tem raztegu smo prestavili 2^i elementov

$$\begin{aligned} T(n) &\leq n + \sum_{i=0}^{\lg n} 2^i \\ &= n + \frac{2^{\lg n+1} - 1}{2 - 1} \\ &= n + 2 \cdot 2^{\lg n} - 1 \\ &= 3n - 1 = O(n) \end{aligned}$$

- Amortizirana časovna zahtevnost dodajanja enega elementa je potemtakem 1

Računovodska metoda

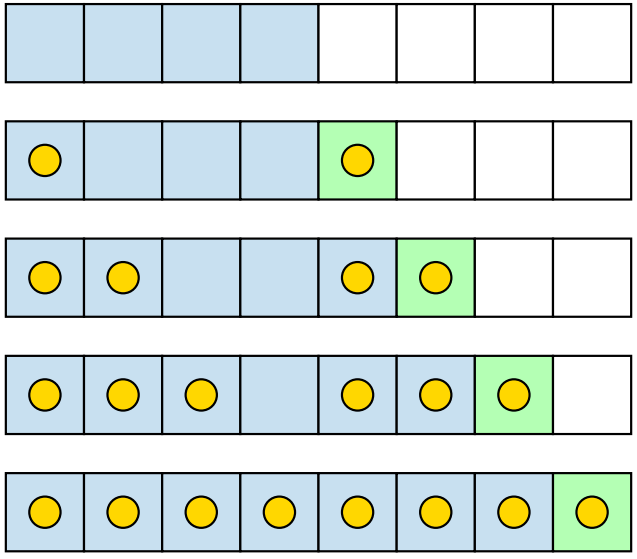
- Določiti moramo računovodsko ceno dodajanja
- Zagotoviti moramo, da je računovodska cena poljubnega zaporedja dodajanj vedno večja ali enaka dejanski ceni
- Računovodsko ceno dodajanja elementa moramo torej nastaviti tako, da bomo z njo pokrili ustrezen delež stroškov prihodnjega raztega

Računovodska metoda

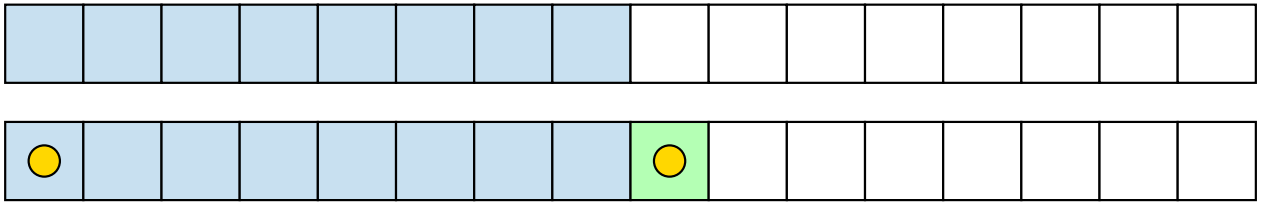
- Računovodsko ceno dodajanja elementa x nastavimo na 3
- Ta cena pokrije
  - strošek dodajanja elementa x
  - strošek prihodnje prestavitve elementa x
  - strošek prihodnje prestavitve enega od elementov v prvi polovici tabele

Računovodska metoda

- Ko dodamo nov element, plačamo njegovo vstavitev, njegov bodoči premik in bodoči premik enega od elementov v levi polovici tabele



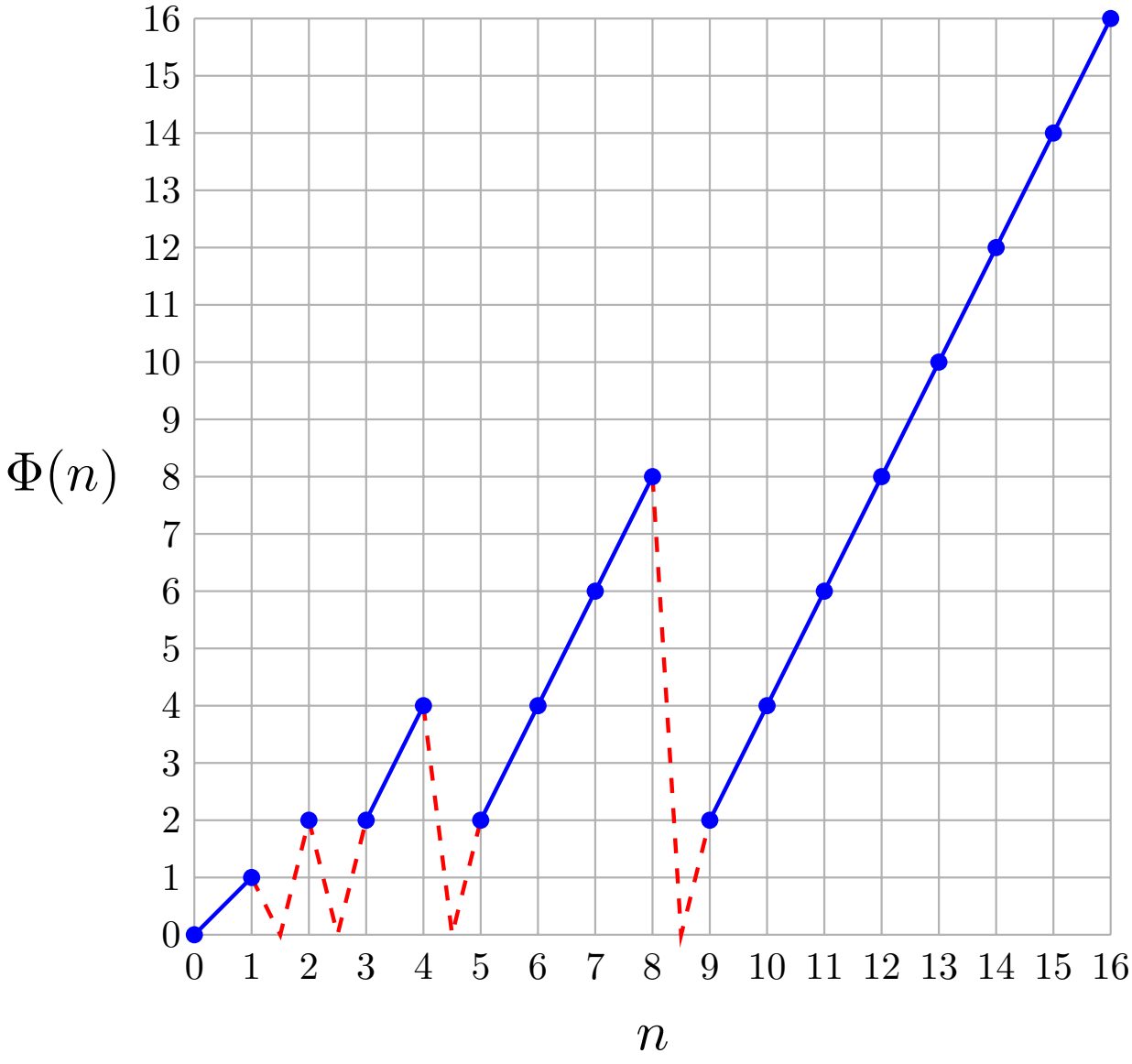
- Celotna operacija raztezanja je sedaj plačana



Metoda potencialov

- Potencial lahko vnovič definiramo tako, da bo usklajen z računovodsko ceno dodajanja
- Potencial je na začetku in tik po raztegu (pred vstavitvijo elementa, ki je povzročil razteg) enak 0
- Nato enakomerno narašča; tik pred raztegom je dovolj velik, da lahko plača celoten razteg

Metoda potencialov



Metoda potencialov

$$\Phi(T_n) = \begin{cases} n, & \text{če } n \leq 1 \\ 2(n - 2^{\lfloor \lg(n-1) \rfloor}) & \text{sicer} \end{cases}$$

- Je potencial usklajen z računovodsko ceno?
- $\hat{c}_i = c_i + \Phi(T_i) - \Phi(T_{i-1})$
- Če vstavitev ne povzroči raztega
  - $c_i = 1$
  - $\Phi(T_i) - \Phi(T_{i-1}) = 2$
  - $\hat{c}_i = 1 + 2 = 3$
- Če vstavitev povzroči razteg (npr. pri $i = 9$ ali $i = 17$)
  - $c_i = 1 + (i - 1)$
  - $\Phi(T_i) - \Phi(T_{i-1}) = 2 - (i - 1)$
  - $\hat{c}_i = 1 + (i - 1) + 2 - (i - 1) = 3$

Odnos med potencialom in računovodsko ceno

- Od treh enot računovodske cene se
  - 1 enota porabi za sámo vstavitev
  - 2 enoti naložita v potencial
- Ko napoči čas za razteg, se nakopičeni potencial v celoti porabi za prestavitev elementov v novo tabelo

Krčenje tabele

- Če zasedenost tabele (α) pade pod določen faktor, lahko tabelo skrčimo
  - ustvarimo novo tabelo, ki je za določen faktor manjša od trenutne, in elemente prenesemo vanjo
- Slaba ideja: tabelo skrčimo, ko α pade pod $1/2$
  - pričnemo s tabelo s polovično zasedenostjo
  - odstranimo en element, tabela se skrči
  - dodamo en element, tabela se raztegne
  - odstranimo en element, tabela se skrči
  - dodamo en element, tabela se raztegne
  - $O(n)$ na element!

Krčenje tabele

- Boljša ideja: tabelo skrčimo, ko α pade pod $1/4$
- Tabele ne skrčimo na $1/4$ prvotne velikosti, ampak na $1/2$ prvotne velikosti
  - α bo tako postal $1/2$
- Potencial lahko naravno razširimo na krčenje
  - pri $\alpha = 1/2$ je potencial enak 0
  - od $\alpha = 1/2$ do $\alpha = 1$ potencial narašča za 2 enoti na element
  - od $\alpha = 1/2$ do $\alpha = 1/4$ potencial narašča za 1 enoto na element
  - na ta način se bo nabralo ravno dovolj potenciala, da bomo lahko elemente prestavili v novo tabelo

Krčenje tabele

Kapaciteta	Št. elementov	Potencial
16	8	0
16	7	1
16	6	2
16	5	3
16	4	4
8	3	1
8	2	2
4	1	1

Računovodska cena pri raztezanju in krčenju

- $\hat{c} = c + \Delta\Phi$
- Analiziramo vse možne situacije
  - dodajanje oz. brisanje pri $\alpha \geq 1/2$ in $\alpha < 1/2$
  - dodajanje oz. brisanje na prehodu od $\alpha < 1/2$ do $\alpha \geq 1/2$ oz. obratno
  - dodajanje oz. brisanje, ki povzroči razteg oz. skrčitev
- V vseh primerih je računovodska cena konstanta ($O(1)$)
- Ker je potencial vedno pozitiven (vedno imamo dovolj »denarja«, da plačamo morebiten razteg oz. skrčitev), je za poljubno zaporedje operacij računovodska cena zgornja meja dejanske cene
- Amortizirana časovna zahtevnost posamične operacije je tako $O(1)$