1.17

**1.18**

Potencialova funkcia: |S|.log n

Amortizovana cena vzorec: ∑1..nĉi=∑1..nci + φ(Dn) + φ(D0)

Sú 2 prípady: 1) platí podmienka Hodnota(S,X[i]):

ĉi = 1 + |S|.log n - |S-1|.log n = 1 + log n

2) podmienka s Hodnota(S,X[i]) neplati:

ĉi = 1 + |S+1|log n - |S|.log n = 1 + log n

Preto n roznych operacii ma zlozitost n \* (1 + log n) ϵ O(n.log n)

**1.19**

a)

Skupina 1: i-ta operacia ak i nie je mocninou 2.

Zlozitost skupiny je n

Skupina 2: i-ta operacia, ak i je mocnina 2.

Zlozitost skupiny je ∑j=0..log n2j

Spolu preto:∑i=1..nci = n + (2.n – 1) = 3.n

b)

Pocet operacii medzi dvomi mocninami je 2i+1 - 2i – 1 (napr. mezi 64 a 32 to je 31). Tie spotrebuju 1 kredit. Potom operacia 2i spotrebuje i kreditov.

Kazdej operacii priradime 3 kredity:

- 1 kredit zaplati operacia

- 1 kredit ma na ucte dana operacia

- 1 kredit name na ucet operacii, ktora uz nema ziadny kredit

c)

Potencialova funkcia: φ(T) = cena operacie

1) i-ta operacia nie je mocninou 2:

ĉi = 1 +

2) i-ta operacia je mocninou 2:

B – cena pred predvedenim i-tej operacie: 2j-1

A – cena po prevedeni i-tej operacie

ĉi = i + A – B =

**1.20**

Datova struktura: obojsmerny zretazeny zoznam, kde si zaznamenavame pocet poloziek. Tiez mame ukazatel na zaciatok a na koniec pola, preto je pridavanie v O(1) case, pretoze len zmenime ukazatel na novy prvok a z neho dame ukazatel na zoznam.

PUSH – na koniec pola sa pride vkladany prvok a ukazatel na koniec pola sa aktualizuje. Inkrementujeme pocet. Zlozitost O(1).

PULL – Mame ukazatel na zaciatok pola. Prvy prvok odstranime a ukazatel zaciatku pola dame ukazovat na 2. prvok. Dekrementujeme pocet. Zlozitost je O(1)

SIZE – vratime pocet, ktory mame napocitany. Zlozitost je O(1)

DECIMAL – vid implementacia:

1. DECIMAL():
2. s ← SIZE()
3. d ← ⌊s / 10⌋
4. T ← new\_data\_structure
5. for i = 1 to d + 1 do:
6. for j = 1 to 10 do:
7. if i % 10 ≠ 0 then
8. T.PUSH(this.PULL())
9. else
10. this.PULL()
11. fi
12. od
13. od
14. return T

1. Pomocou kreditov:

|  |  |  |
| --- | --- | --- |
| Operacia | Kredity | Cena |
| PUSH | 20 | 1 |
| PULL | 0 | 1 |
| SIZE | 0 | 0 |
| DECIMAL | 0 | 19 |

Vysvetlenie DECIMAL: Ako mozeme vidiet, do premennej s zi ulozim pocet prvkov v structure. Ideou je teraz rozdelit pomyselne prvky v structure na mnoziny o velkosti 10. Preto si do premennej d spocitam, kolko bude takychto mnozin. Ako mozeme vidiet, na riadku c. 5 “prechadzam tie mnoziny”. Ma to zlozitost linearnu, lenze nie je tam ziadna operacia, ktora by nieco stala (tie su az v 2. Cycle for).

Preto si analyzujeme 2 cyklus. Mozeme vidiet, ze po operacii push ma kazdy prvok na svojom ucte 19 kreditov. Druhy cyklus prebehne 10 krat, pricom si mozeme vsimnut, ze operacia PULL (bez PUSH) prehne len raz, tj. nam to vezme na cene 1 kredit. Preto nam na zvysne operacie zostane 18 kreditov. Mozeme povedat, ze operacie na riadku c. 8 prebehnu 9 krat (kedze raz to tam nevojde kvoli tomu, ze sa jedna o 10. prvok). Tych 9 operacii na riadku c.8 nam vezme po dve kredity per riadok (kvoli jednemu PUSH a jednemu PULL), preto nam ostane 0 kreditov. Pritom mame hned vytvorenu novu strukturu T. Vsimnime si, ze s kreditmi ostatnych prvkov sme nerobili nic, preto stale maju na svojom ucte po 19 kreditov.

1. Potencialova funkcia: pocet prvkov \* 10 = 10.n

Vzorec: ∑i=1..nĉi = ∑i=1..nci + φ(Dn) + φ(D0)

Pre PUSH: 1 + 10.(n + 1) – 10.n = 11

Pre PULL: 1 + 10.n – 10.(n – 1) = 1 + 10 = 11

Pre SIZE: 0 + 10.n – 10.n = 0

Pre DECIMAL: 10.(n/10) + 10(n-n/10) – 10.n = n + 10n – n – 10n = 0

Vysvetlenie DECIMAL:

Skutocna cena i–tej operacie decimal je n, pretoze n/10 prvkov zahodime a zvysnych n-n/10 prvkov prekopirujeme do novej dat. struktury (preto pracujeme so vsetkymi n prvkami).

Pocet prvkov pred i-tou operaciou: n preto φ(Di-1)=10.n

Pocet prvkov po itej operacii: vieme, ze odstranime kazdy 10 prvok, preto odstranime n/10 prvkov. Kedze odstranime z n prvkov n-10, ostane name n-n/10 prvkov.

**1.21**

1. Kredity

|  |  |  |
| --- | --- | --- |
| Operacia | Kredity | Cena |
| PUSH | 2 | 1 |
| POP | 0 | 1 |
| MULTIPOP(x) | 0 | x |
| COPY | 0 | k |

Invariant: pocet kreditov nikdy neklesne pod 0. Po kazdom vlozeni prvku do zasobniku mu na ucte ostane 1 kredit. Povlozeni n prvkov mame na uctoch spolu n kreditov. Po volani i operacii POP nam ostane n-i kreditov (a tiez n-i prvkov). Po j krat volani MULTIPOP(x) nam ostane n-i-j.x prvkov, tj aj tolko kreditov. Mozeme vidiet, ze cena operacie COPY je rovna poctu prvkov, preto ak mame n-i-j.x prvkov, ktore maju spolu n-i-j.x kreditov na ucte, tak nas toto zmazanie vyjde na n-i-j.x kreditov, ktore ale mame na ucte, preto nam ostane 0 kreditov. Nemozeme ist so zapornych cisel, pretoze by sme musel zmazat vzdy viac prvkov, ako tam naozaj je.

1. Potencialova funkcia φ(D) = pocet prvkov na zasobniku |S|

PUSH: ĉi = 1 + n + 1 – n = 2

POP: ĉi = 1 + (n - 1) + n = 0

MULTIPOP(x) = x + (n – x) - n = 0 ak x < |S|

MULTIPOP(x) = |S| + 0 - |S| = 0 ak x > |S|

COPY = k - a + 0 – (k – a) = 0, kde k – a je realny pocet prvkov v zasobniku.

**1.22**

Kredity

|  |  |  |
| --- | --- | --- |
| Operacia | Kredity | Cena |
| PUSH-A | 3 | 1 |
| PUSH-B | 2 | 1 |
| MULTI-POP-A(k) | 0 | min(k,|SA|) |
| MULTI-POP-B(k) | 0 | min(k,|SB|) |
| TRANSFER(k) | 0 | min(k,|SA| |

Potencialova funkce: φ(D) = 2\*|SA| + |SB|

PUSH-A: ĉi = 1 + 2\*(|SA| +1) + |SB| - (2\*|SA| + |SB|) = 1 + 2 = 3

PUSH-B: ĉi = 1 + 2\*|SA| + |SB| + 1 - (2\*|SA| + |SB|) = 2

MULTI-POP-A(k): ĉi = k + 2(|SA| - k) + |SB| - (2\*|SA| + |SB|) = -k ak k < |SA|

MULTI-POP-A(k): ĉi = |SA| + 0 + |SB| - (2\*|SA| + |SB|) = -|SA| ak k > |SA|

MULTI-POP-B(k): ĉi = k + 2.|SA| + |SB| - k – (2.|SA| + |SB|) = 0 ak k < |SB|

MULTI-POP-B(k): ĉi = |SB| + 2.|SA| + 0 – (2.|SA| + |SB|) = 0 ak k > |SB|

TRANSFER(k): ĉi = k + 2.(|SA| - k) + |SB| + k – (2.|SA| + |SB|) = 0 ak k < |SA|

TRANSFER(k): ĉi = |SA| + 2.(|SA| - |SA|) + |SB| + |SA| – (2.|SA| + |SB|) = 0 ak k > |SA|

**1.24**

a) nie

Volam n/2 krat INSERT(S,i). Mame zoznam (3,3,3,3,3,…3), kde |S| = n/2 a n/2 krat na tom zavolali MIN-ALL, tak by sme mali zlozitost O((n/2)2) = O(n2/4) ϵ O(n2), kedze pocet prvkov by to nikdy nezmenilo.

b) ano

Ucty:

|  |  |  |
| --- | --- | --- |
| Operacia | Kredity | Cena |
| INSERT(S,i) | 2 | 1 |
| MIN-ONE(S) ak |S| > 1 | 0 | k |
| MIN-ONE(S) ak |S| = 1 | 0 | 0 |

Invariant: Pocet kreditov nikdy neklesne pod 0. Predpokladajme, že sme n-krát zavolali insert (n >1). Každá položka z n má na svojom účte 1 kredit, preto všetky spolu majú n kreditov. Po zavolaní operácie MIN-ONE(S) nám v zozname ostane 1 prvok. Keďže odstránime n-1 prvkov, prídeme o n-1 kreditov, preto po ostránení prvkov ostane len 1 prvok s 1 kreditom na účte. V tomto prípade, ak zavoláme znovu operáciu MIN-ONE(S), tak v konštantnom čase skontrolujeme, či je tam len jeden prvok a to je, takže končíme. Preto m-krát po sebe volaná operácia MIN-ONE má zložitosť O(m).

Potencialova funkcia

Φ(D) = počet prvkov v zozname

INSERT: ĉi = 1 + |S| + 1 - |S| = 2

MIN-ONE(S): ĉi = |S| - 1 + 1 - |S| = 0 (ci = |S| - 1, φ(Di) = 1, φ(Di-1) = |S|)

**1.25**

Kredity

|  |  |  |
| --- | --- | --- |
| Operacia | Kredity | Cena |
| INSERT | 3 | 1 |
| DELETE-EVEN ak n > 1 | 0 | 2 |
| DELETE-EVEN ak n < 2 | 0 | 0 |

Invariant: pocet kreditov nikdy neklesne pod 0. Predpokladajme, ze sme n krat volali operaciu INSERT, co znamena, ze v zozname mame n prvkov (n > 1). Kazdy z nich ma na ucte 1 kredit, spolu preto mame n kreditov na ucte. Po volani operacie DELETE-EVEN sa nam pocet prvkov znizi o n/2 (n/2 je teda zmazanych). Predpokladajme, ze zmazanie ma cenu 2 (1 kredit za zmazanie a 1 kredit za preskocenie 1 prvku za zmazanym – tj. posun o 2 indexy), preto nam ostane n/2 prvkov s n/2 kreditmi na ucte.

V pripade m volani operacie DELETE-EVEN nam ostane v zozname S len jeden prvok. Operacia DELETE-EVEN v konstantnom case zisti pocet prvkov < 2, preto vie, ze nic nesmaze, tak konci. Preto n operacii DELETE-EVEN na 1 alebo 0 prvkovom zozname nas vyjde na zlozitost O(n).

Potencialova funkcia:

Φ(D) – pocet prvkov v zozname \* 2 = 2\*|S|

INSERT: ĉi = 1 + 2(|S| + 1) – 2\*|S| = 3

DELETE-EVEN: ĉi = |S| + 2(|S|/2) – 2\*|S| = 0

Vysvetlenie pre DELETE-EVEN:

Φ(Di) = 2.(|S|/2) – zmensime pocet prvkov na polovicu, preto |S|/2 a kedze potencialova funkcia je 2\*, tak tento zmenseny pocet prvkov vynasobime dvomi

Φ(Di-1) = 2.|S|, pretoze pred operaciou DELETE-EVEN tam mame |S| prvkov a potencialova fce je 2\*|S|.

ci = n, pretoze sice mazeme kazdy druhy prvok, ale v konecnom dosledku prechadzame cez vsetkych n prvkov, co nas stoji n.

**1.26**

Za a) nie

Dokaz: Predpokladajme (n/2) - 1 krat volana operacia INSERT(S,1) a 1x volana operacia INSERT(S,2). Zlozitost to ma O(n/2). Potom predpokladajme volanie DELETE(1), co ma zlozitost O(n/2), pretoze musi prejst n/2 prvkov a zmaze posledny vlozeny prvok 2. Ostane name tam (n/2-1) prvkov. Potom predpokladajme, ze n/2 krat volame DELETE(1) a kedze nam operacia DELETE(1) nijakym sposobom uz nezmensi pocet prvkov v zozname, tak to bude mat zlozitost celkovo (n/2\*(n/2-1)) ≈ O(n2/4) ϵ O(n2).

Za b) ano

Ucty

|  |  |  |
| --- | --- | --- |
| Operacia | Kredity | Cena |
| INSERT | 2 | 1 |
| DELETE | 0 | k |

Invariant: Pocet kreditov neklesne pod 0. Predpokladajme n operacii INSERT. Mame v zozname n prvkov a kazdy ma na ucte 1 kredit. Po volani operacie DELETE(i) sa nam odstrani k prvkov. Ostane nam n-k prvkov. Kedze uz nevolame DELETE s tym istym prvkov, tak vieme, ze v dalsej iteracii nam tiez klesne pocet prvkov o m, preto budeme mat v zozname n – k – m prvkov. Az kym nenarazime na zoznam, ktory ma len 1 prvok. Volanie DELETE(i) ma v tomto pripade zlozitost ale O(1), kedze cena je rovna dlzke zoznamu co je 1. Preto ma v takejto situacii n volani operacie DELETE zlozitost O(n).

Je dobre poznamenat, ze by sa malo DELETE volat len s prvkami, ktore v zozname existuju.

Potencialova funkcia

Φ(D) – pocet prvkov, ktore mame v zasobniku |S|

U operacie DELETE k znamena pocet prvkov, ktore su rovne i => pocet prvkov, ktore nie su zhodne s i je n - k

INSERT: ĉi = 1 + |S| + 1 - |S| = 2

DELETE: ĉi = |S| + |S| - (|S| - k) – |S| = k

Cena operacie DELETE, ktora zmaze k prvkov je k.

**1.27**

Za a)

|  |  |  |
| --- | --- | --- |
| Operacia | Kredity | Cena |
| INSERT | log k |  |
| EXTRACT-MIN | 0 | 1 |

**1.28**

**1.29**

Datova struktura – obojsmerne zretazeny zoznam s ukazatelom na posledny prvok. INSERT operacia bude preto v konstantnom case.

REDUCE(M)

m ← SELECT(M)

c ← M.first

D ← nova\_datova\_struktura

if c ≤ m then:

INSERT(D, c)

if

while c.next ≠ NIL do:

if c ≤ m then:

INSERT(D, c)

If

c ← c.next

od

return D

INSERT(M,i)

If |M| = 0 then

M.first ← i

else

M.last ← i

fi

Ucty

|  |  |  |
| --- | --- | --- |
| Operacia | Kredity | Cena |
| INSERT | 5 | 1 |
| REDUCE | 0 | 4 |

Invariant: pocet kreditov nam nepadne pod 0. Predpokladajme n operacii INSERT. V mnozine mame n prvkov, z ktorych ma na svojom ucte kazdy 4 kredity. Spolu tam teda mame 4.n kreditov. Predpokladame, ze budeme odstranovat n/2 prvkov. Pri hladani median kazdy z n/2 odstranenych prvkov pride o 2 kredity. Preto mazanym n/2 prvkom ostane na ucte len po 2 kredity. Tie dve zvysne kredity posluzia na to, ze linearne prejdemevsetkych n prvkov. To nam zaplatia zvysne 2 kredity, ktore maju n/2 prvkov: 2 \* n/2 = n prechodov. Avsak kreditov nezmazanych prvkov sme sa nedotkli, preto kazdy ma nadalej na svojom konte po 4 kredity. Preto je operaciu mozno opakovat.

Kedze je potrebne na operaciu INSERT 5 kreditov a zaplati nam to aj operaciu REDUCE, staci nam na n operacii 5.n kreditov, preto je aj zlozitost O(5.n).

Potencialova

Φ(D) – 4 \* pocet prvkov v zozname, tj 4 \* |M| = 4 \* n

INSERT: ĉi = 1 + 4(n + 1) – 4.n = 5

REDUCE: ĉi = 2.n + 4(n/2) – 4.n = 2.n + 2.n- 4.n = 0

Zlozitost na n operacii INSERT A REDUCE je O(5n) ϵ O(n)

Vysvetlenie REDUCE:

ci = 2n, pretoze v konecnom dosledku prejdem 2x vsetky prvky (raz pomocou operacie SELECT a 2x linearne pri odstranovani n/2 najvacsich cisel)

φ(Di-1) = 4.(n/2), pretoze vieme, ze po odstraneni tam tam ostane n/2 prvkov. Avsak nasa potencialova funkcia je 4 \* pocet prvkov, preto n/2 vysanobim 4

φ(Di) = 4.n, pretoze pred ostranenim tam mam n prvkov a pomocou potencialovej funkcie to je potencial 4.n.

**1.31**