Podatkovne Strukture in Algoritmi

Predavanja

INFORMACIJE

Literatura:
- T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein: Introductions to algorithms
- open data structures
- A. Brodnik in R. Požar: Zbirka nalog

Domače naloge:
- 6x 4 naloge (1 programersko)
- Oddaja z orodjem Marmoset
- Oddaja preko e-učilnice
- Vsaj 1 zagovor teor. dela DN
- Vsako programersko

Ocena:
- Kolokviji: neobvezni, lahko nadomestijo izpit, vsak 40%
- Končna ocena:
  - 30% domače naloge
  - 60% izpit
  - 10% projekt
  - Skupaj vsaj 50%

$\lg = \log_2$

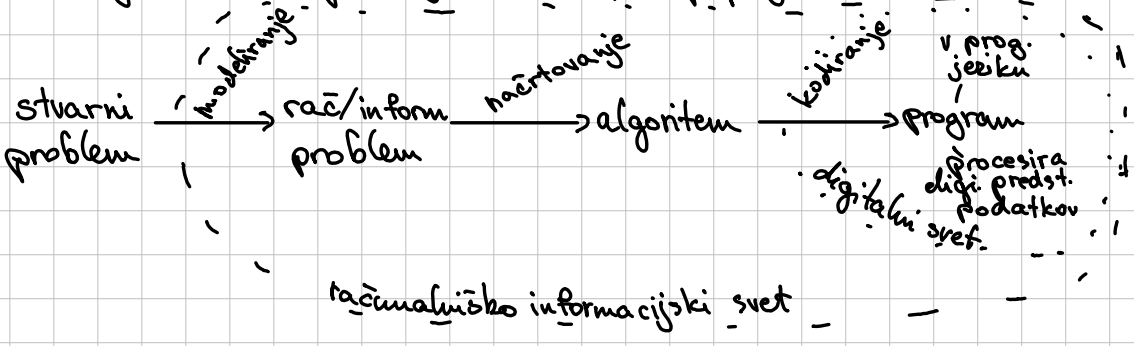
$\log = \log_{10}$

2.10.24

UVOD IN MATEMATIČNE OSNOVE

Pod. strukt. - način organiziranja podatkov, da lahko operacije nad njimi izvedemo čimbolj učinkovito (glede na namen) → definirajo jo operacije, ki so nad njo izvedljive

Algoritem - zaporedje korakov, ki pripelje do rešitve



Algoritem
- Pomembno: pravilnost, časovna zahtevnost (št. korakov), prostorska zahtevnost, ali se da bolje
- Zapisovali bomo v pseudokodi

Insertion sort:

```
For j=2...n
    poišči največji i<j, da A[i] < A[j]
    premakni A[k] → A[k+1]: k = i+1 ... j-1
    prestavi kar je bilo v A[j] → A[i+1]
```

Orodje za preverjanje pravilnosti delovanja zančna invarianca - izjava o stanju spremenljivk v algoritmu, ki velja za vse ponovitve (iteracije)

Model računanja: random-access machine (RAM) oz. primerjalni model (drugi še npr. dostopni, ki šteje samo dostope do pomnilnika)
↳ Ukazi se izvajajo eden za drugim
↳ Predpostavljamo, da so vse operacije enako drage

Časovna zahtevnost / čas izvajanja algoritma A - $T(A)$ - št. korakov (izvedenih osnovnih operacij) alg. A pri vhodnih podatkih velikosti n.
Pred p., da za izvajanje i-te vrstice potrebujemo konst. časa - c_i
eg. čas. zahtevnosti insertion sorta:

$$T(n) = c_1 n + c_2(n-1) + c_3(n-1) + c_4 \sum_{j=2}^{n} t_j + c_5 \sum_{j=2}^{n} (t_j - 1)$$

$$+ c_6 \sum_{j=2}^{n} (t_j - 1) + c_8 (n-1)$$

↳ v najboljšem primeru (podatki že urejeni) dobimo lin. fjo, v najslabšem kvadratno fjo

Ocena zahtevnosti problema
- Če imamo n elementov obstaja $n!$ permutacij
- Alg. mora poiskati vse permutacije in najti pravo
- Vsa možna računanja alg. si lahko predst. kot drevo, listi so permutacije, vozlišča primerjave
- Višina drevesa primerjav je najkrajši možni čas, potreben za urejanje, t.j. $\lg n!$
- Stirlingova aproks. $n! \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n C \Rightarrow \lg(n!) \approx n\lg(n!) + nC_n$
  ↳ ni alg. hitrejšega od $n \log n$

Ocena velikosti funkcij in veliki O

- Za dano fjo $g(n)$ označimo $O(g(n))$ množico fjij

$$O(g(n)) = \{f(n) \mid \text{obstajata poz. konst } c \text{ in } n_0, \text{ tako da } f(n) \leq cg(n) \text{ za vse } n \geq n_0\}$$

$$\Omega(g(n)) = \{f(n) \mid \text{obstajata poz. konst } c \text{ in } n_0, \text{ tako da } f(n) \geq cg(n) \text{ za vse } n \geq n_0\}$$

- Če je $h(n) = O(g(n))$, potem rečemo da $g(n)$ asimptotično od zgoraj omejuje $h(n)$
  ↳ nenegativne fje, natan čneje bi bilo $h(n) \in O(g(n!))$

*

9.10.24

ŠTEVILSKA DREVESA

Dvojiško drevo za leksikografsko razporejanje imen:
Nik $
Nika $
Nikolaj $

vstavi: Nikita?

$$\begin{cases} T(dv.dr.) = O(m \cdot \log m) \\ S(dv.dr.) = O(n) \end{cases}$$

Črka $\in \{A-Z, a-z, \dots\} = \Sigma$ (Abeceda)

$|\Sigma| = O(1)$

Abeceda je lahko tudi: $\{A, C, G, T\}$

Trie (iz reTRIEval())
- Rekurzivna podatkovna struktura
- Ključi tako veliki, da ne moremo naenkrat primerjati
- Vrednosti so v listih

*

$$\begin{cases} T(trie) = O(m) \\ S(trie) = O(m \cdot n) \end{cases}$$

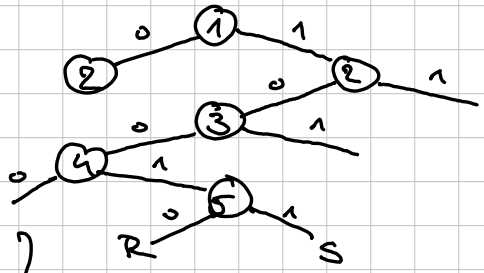
liter, a velik

DN: (Dokaži)
- Binarno drevo, ki ima k listov in vsa notranja voz., ki imajo 2 naslednika, ima $k-1$ takih vozlišč

Primer: Binarne (bitne) črke abecede $\{0,1\}$ in bin. predstavitev naslednjih ključev:

A	00001	C	00011
E	00101	G	00111
H	01000	I	01001
J	01010	K	01011
L	01100	M	01101
N	01110	O	01111
P	10000	R	10010
S	10011	X	11000
Z	11010		

Imejmo ključe $\{A, C, E, G, H, I, L, M, R, S\}$



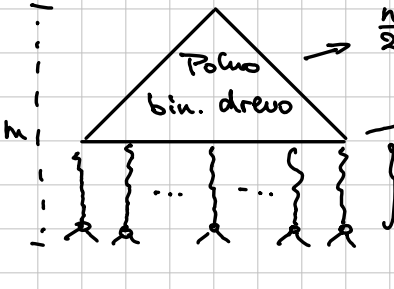
Find(01000)
lokanje je v 2 fazah:
1. Spuščanje do lista
   - kandidat
   - čas. zaht. $h(trie) = m$
2. Preverjanje kandidata
   - čas. zaht. m

} skupaj $O(m)$

Iščemo J = 01010: naletimo na prazno mesto → približek najdemo tako, da gremo eno nazaj, levo in se držimo desne oz. obratno.

Operacije
- Običajne: Najdi, Dodaj, Izloči
- Dodatne: Najdi Manj, Najdi Več
- Posplošena: Najdi

n = # elementov, m = velikost ključa ($m = 5$)



$h(\triangle) = \lg(\frac{n}{2}) = \lg(n) - 1$

Prostorska zahtevnost:

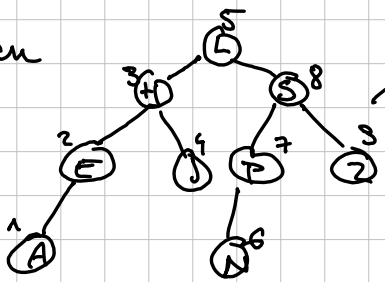
$S(trie) = \frac{n}{2} + \frac{n}{2} - 1 + \frac{n}{2} + n + \frac{n}{2}(m - 1 - (\lg \frac{n}{2}) =$

$= n + n + \frac{n}{2} - 1 + \frac{n}{2}(m - 1 - (\lg n - 1)) = 2{,}5n - 1 + \frac{n}{2}(m - \lg n)$

$= O(m \cdot n)$

Vmesni obhod (in order traversal):

1. Obdelaj L
2. Obdelaj koren
3. Obdelaj D

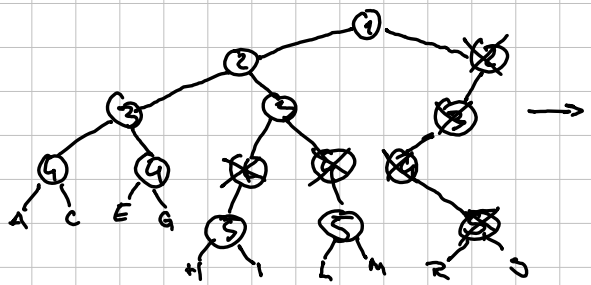


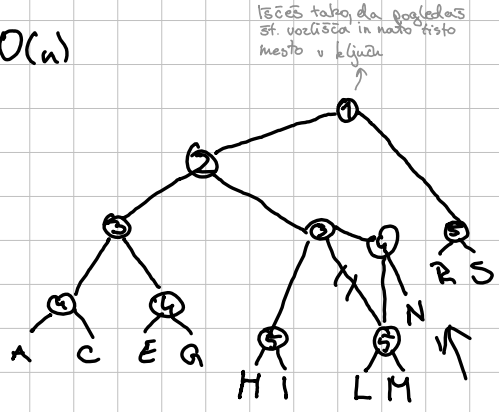
v lin. času dobimo sortirane elemente

(isto pri trie-ju)

Trie s stisnjenimi potmi (path-compressed trie)

- $2n - 1$ vozlišč
- kvalitetnejša prostorska zahtevnost → $O(n)$





- Časovna zahtevnost iskanja in vstavljanja: m

Insert (01110) → Pridemo do L-ja (L: 0111$\hat{0}$0)

↓

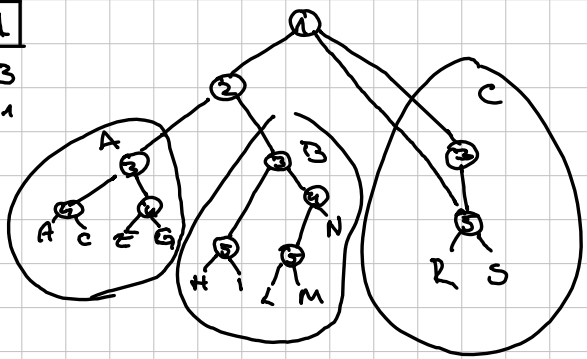
Novo vstavimo vozlišče na četrtem mestu (lahko tudi namesto L-ja

Potem ni več sortirano.)

- Izboljšamo še časovno zahtevnost

→ Prazno

A	B	C	⊥
0	1	2	3
00	01	10	11



B=1

Find (0100)

↳ Prihranimo čas, ker ne rabimo primerjave na 2

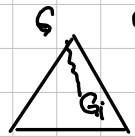
O_2. še več:

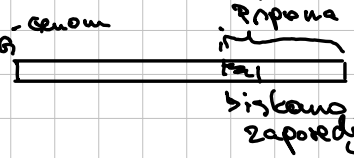
000	001	010	011	100	101	110	111
α	β	γ	δ	λ	⊥	⊥	⊥

↳ Nivojsko stisnjena drevesa (level-compresd tries - LC trie)

- Izberemo, koliko praznega prostora smo pripravljeni imeti v tabeli

Primer: iskanje gena v genomu: znamo najti na začetku in kjer kolivmes v $O(m)$





23.10.24

Isci (s,k) → statična PS
Vstavi (s,(k,d)) → dinamična PS
Brisi (s,k) → dinamična PS

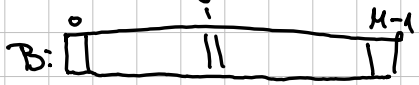
Slovar
*

AVL	BBST	BST	RTPS	UPS	PS	Naiv	
O(log n)	O(log n)	O(n)	O(n)	O(n)	O(n)	O(1)	Isci
	O(n)	-11-	-11-	-11-	O(n)	O(1)	Vstavi
	O(n)				O(n)	O(1)	Brisi
	O(n)	O(n)	O(n)	O(n)	O(n)	O(M)	skupaj

Čas. zahtevnost je
O(ustrezne točke, poiskati
oddaljeno vozlišče)

Naivna implementacija

ključ = Integer
Mat: |integer| = ∞
Rač: |integer| = 2^{32}, 2^{64}, ... = M



recimo $i \notin p \Leftrightarrow B[i] = \perp$
$j \in p \Leftrightarrow B[j] =$ podatki
Isci (p, k): return B[k]
Vstavi (p, (k, d)): if B[k] = ⊥ : B[k] = d
Brisi (p, k): B[k] = ⊥

Časovna zahtevnost: O(M)

Seznami

- $S = \bot$ → Prazen
- $s = h : t$
  elem. ↑ rep (tudi seznam)

```
Isci(s,k):
    if s ≡ ⊥: return ⊥
    if s.h.k = k ⇒ return s.h.d
    return Isci(s.t, k)

Vstavi(s, (k,d)): return seznam((k,d), s)

Brisi(s, k)
    if s ≡ ⊥: return ⊥
    if s.h.k ≡ k: return s.t
    return seznam(s.h, Brisi(s.t, k))

Isci(s,k):
    if s ≡ ⊥: return ⊥
    if s.h.k ≡ k: return s.h.d
    if s.h.k > k: return ⊥
    return Isci(s,k)
```

$S = \bot \mid e\ t_1\ t_2 \Rightarrow h = \mathcal{O}(\log n)$

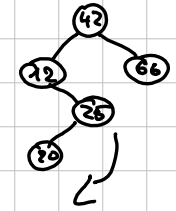
```
Isci(s, k):                          T(n) = 1 + T(n_1) + T(n - n_1 - 1)
    if s ≡ ⊥: return ⊥
    if s.e.k = k return s.e.d
    d = Isci(t_1, k): return Isci(t_2, k)
    return d
```

```
Isci(UD,k):
  if UD=⊥: return ⊥
  if UD.r.k=k: return UD.r.d
  if UD.r.k<k:
    Return Isci(D,k)
  else
    Return Isci(L,k)
```

$T(n) = a + \max(T(n_1), T(n-n_1-1))$

$n_1 = 0 \Rightarrow a + T(n-1) = O(n)$

Ustavi(20)

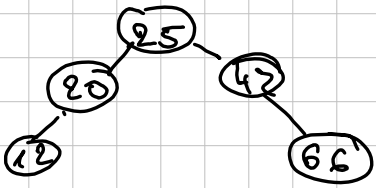


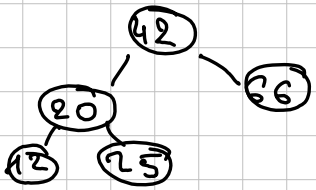
Želimo: $h(n_1) \approx h(n-n_1-1)$

bo res: $n_1 \approx n-n_1-1$

$n_1 \approx \frac{n-1}{2}$

$\equiv |\|L\| - \|D\|| \leq 1 \Rightarrow h = \lg n$





$UD = \bot \mid r \lhd D: \; L < r < D, \; |\|L\| - \|D\|| \leq 1, \; |h_L - h_D| \leq 1$

```
Iterate(s,f()):
  if s=⊥: return ⊥
  return Seznam(f(s.h), Iterate(s.t,f())
    h' = f(s.h)
    t' = Iterate(s.t,f())
      return Seznam(h',t')
```

```
t' = Iterate(s.t,f())
ali h' = f(s.h)
```
] rakov

```
Iterate(dr, f()):
  if dr ≡ ⊥ : return ⊥
     r' = f(dr.r)          L' = ....   L' = ....
     L' = Iterate(L, f())  r' = ....   D' = ....
     D' = Iterate(D, f())  D' = ....   r' = ....
        return Tree(r', L', D')

Vstavi(dr, (k, d)):
   if dr ≡ ⊥ : return tree((k,d), ⊥, ⊥
   if dr.r.k = k : return dr
   if dr.r.k < k:
      Return Tree(dr.r, dr.L, Insert(dr.D, (k,d)))

Največji(dr):
   if dr ≡ ⊥ : return ⊥
   if dr.D ≡ ⊥ : return dr.r
   else return Največji(dr.D)
```

Najmanjše možno št. el. v drevesu višine h:

$$N(h) = 1 + N(h_2) + N(h_0)$$

slabo: $h_2 = h_0 - 1$

$\Rightarrow h = h_0 + 1 \Rightarrow$

$h_0 = h - 1$

$h_2 = h - 2$

$$= 1 + N(h-2) + N(h-1)$$

$$F_n = F_{n-1} + F_{n-2} \leq N(h)$$

$$= \frac{1}{\sqrt{5}} \left(\Phi^n + \hat{\Phi}^n \right) \leq N(h)$$

30.10.24

Vstavi (ELt)
Briši (ključ) → ELt brišemo
Najdi (ključ) → podatek (MAP)
ELt: ključ, podatek

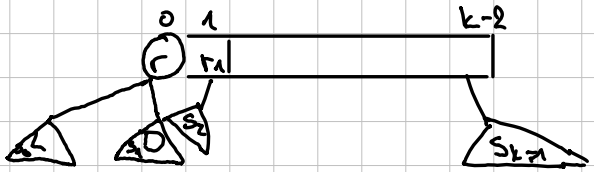
	Arr	LL	OL	BST	AVL	B
V	↑	↑	↑	↑	↑	
B	$O(1)$	$O(n)$	$O(n)$	$O(h)$	$O(h)$	$O(\log n)$
N	↓	↓	↓	↓	$h=\log n$	$h=\frac{\log n}{\log b}$
S	$O(M)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$

Čas. zahtevnost je odvisna od razdalje med ustopno točko in najbolj oddaljeno točko PS. $\equiv h$

Arr: Prostorska zahtevnost prevelika
LL: Časovna zahtevnost lin. sorazmerna s št. el.

$|h(L) - h(D)| \leq 1$

AVL: $L < r < D$; $|h(L) - h(D)| \leq 1$



$\frac{b-1}{2} < k_v \leq b$

$r = r$:
$L = S_0 < r_0 < S_1 = D$
$S_1 < r_1 < S_2$
$r_i < r_j \mid i < j$
$h(S_i) = h(S_j)$

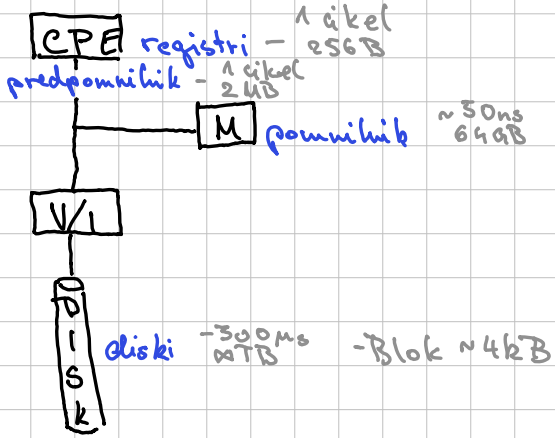
-koliko el. v drevesu h-višine
$(k-1) + k(k-1) + k^2(k-1) + \dots + k^{h-1}(k-1) =$
$= (k-1) \cdot \sum_{i=0}^{h} k_i = (k-1) \cdot \frac{k^h - 1}{k-1} = k^h - 1 < n_{max}$

$h \log k \geq \log(n+1)$
$h \geq \frac{\log(n+1)}{\log k}$

$k_v = 10$

k_v
r[0..k-2]
s[0..k-1]

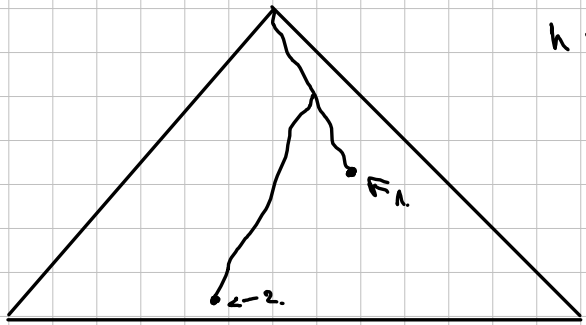
$el(t_i$: key, data



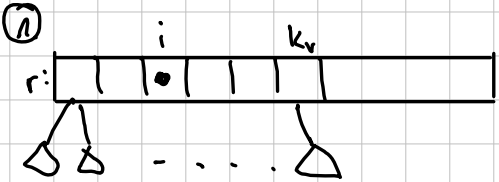
B-drevo (Blokovno drevo)

Velikost B: $32b + 8(b+1) + 2 \leq 4096$

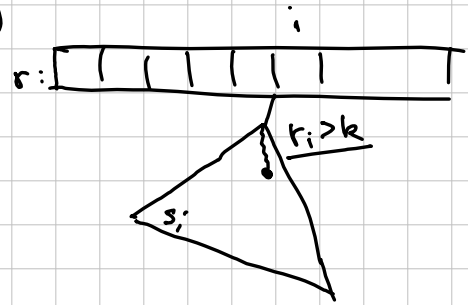
$$h = \frac{\lg(n)}{\lg(b)}$$



①



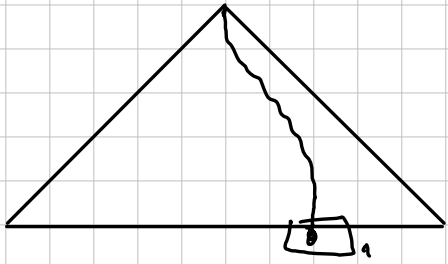
②

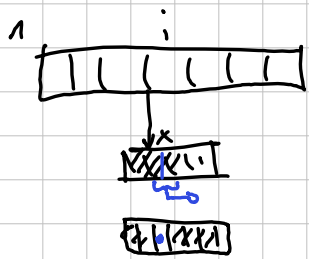


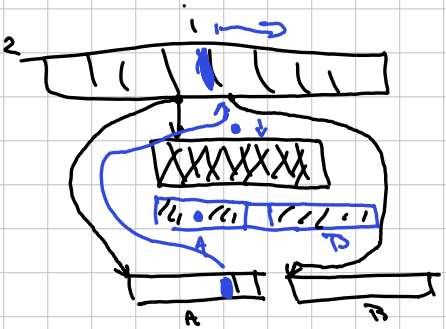

```
def Najdi(T, k):
    if t=⊥ return null
    i = 0
    while (r[i]<k) ∧ (i<b)
        i = i+1
    if r[i] = k:
        return r[i].data
    return Najdi(S[i], k)
```

$T_{Najdi}(n) = O(h \cdot b) = O(\log n \cdot \frac{b}{\lg b})$ → če uporabimo bisekcijo namesto while loopa $= O(\lg n \cdot \frac{\lg b}{\lg b}) = O(\lg n)$

$T_{Najdi}(n) = O(\lg n)$

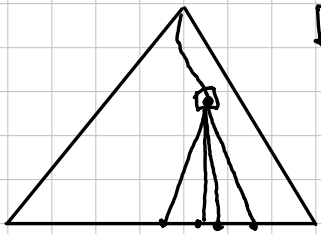






$T_{ustavi}(n) = O(h \cdot b) = O(\lg n \frac{b}{\lg b})$

$k_v = \frac{b-1}{2} - 1$



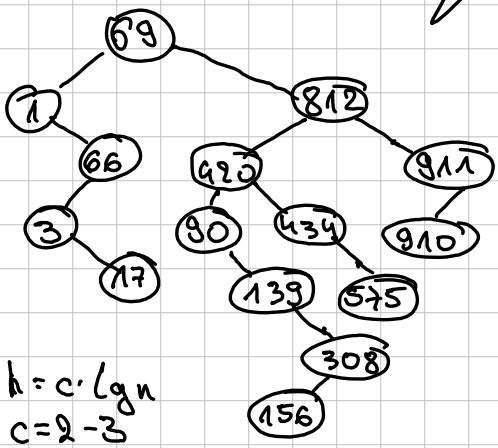
TIF Preveza (2-3-4 preveza)

6.11.24

	S	T	E	Opombe
LL	↑	O(n)	L	
OLL	O(n)	O(n)	L	
BST	↓	O(n)	S	
BBST		O(h)	T	$h = \log n$
SLST	O(n)	O(h)	S/T	$h = \log n$

S_1: 69, 1, 812, 420, 911, 66, 3, 17, 910, 90, 139, 434, 308, 575, 156

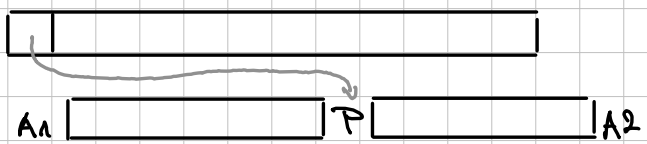
S_2: 1, 3, 17, 66, 68, 90, 139, 156, 308, 420, 434, 575, 812, 910, 911



$h = c \cdot \log n$

$c = 2 - 3$

A: 1..n



Preskočni seznam

- Def: Element nivoja i vsebuje i referenc, ki jih označimo $i \leq k$
- Pogostost: imeti približno $n/2^i$ elementov nivoja i, kjer $i \leq \lfloor \log n \rfloor$
- Porazdeljenost: da so elementi nivoja i enakomerno porazdeljeni po seznamu

1: $p(d=1) = \frac{1}{n_1} = \frac{1}{2^{\lg n}}$

2: $p(d=2) = \frac{1}{2^{\lg n - 1}}$

⋮ ⋮

d: $p(d) = \frac{1}{2^{\lg n - (d-1)}}$; $\lg n \geq d$

RANG IN IZBIRA

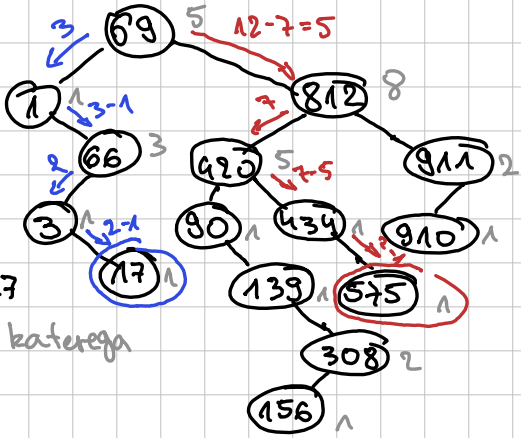
Rang(S, x):
elementov iz S tako, da so manjši ali enaki x.

Rang(S_1, 337) = 9
Rang(S_1, 308) = 9

Izbira(S_1, i) := {a, (a ∈ S) ∧ (Rang(S, a) = i)}
Izbira(S_1, 9) = 308
Izbira(S_1, 1) = 1

Razširjanje

Drevo: koren, L, D → Bin. drevo
Iskalno 2-drevo: koren, L, D | L < koren < D
AVL drevo: koren, L, D | L < koren < D | h
RI/AVL drevo: koren, L, D | L < koren < D | h | & Rang korena



Izbira(S, 3) = 17
iščemo element, katerega rang je 3

Izbira(S, 12) =
iščemo element, katerega rang je 12

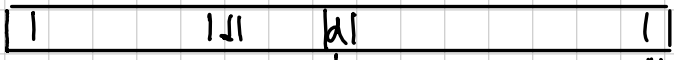
13.11.24

Uspešna PS za slovar — najslabšem primeru
Izboljšanje (npr. časovne zahtevnosti, prost. zahtevnosti splošno zahtevnost) } Cilj PSA
Najboljši alg. za slovar
Ogledati različne implementacije za različne primere } side product
Ugotavljanje kdaj je impl. vredu

Kaj?	Zakaj? (problem)	Kako? (rešitev)
1. poveznni seznam		
2. urejeni poveznni seznam	čas. zahtevnost	urejenost
3. iskalno binarno drevo	čas. zahtevnost	1 rep → poddrevo
4. Uravnoteženo bin. drevo / AVL	čas. zahtevnost	omejiti h_{max} s uravnoteževati
5. B in B+ drevo	poenostavitev	žrtvovali prost. za preprostost
6. TTF-drevo, rdeče-črno drevo	prost. zahtevnost	uvedemo dvojiška drevesa
	S: O(n) T: O(log n) Z: 2	
7. Preskočni seznam T: O(log n) S: O(n) Z: 5	poenostavitev	uvedemo naključnost za poenos. žrtvujemo npr → pricakovano
	Bistvo predavanj	
	čas. zaht	žrtvujemo prostor za čas
	Prostor	žrtvujemo čas. (→ prit. čas).
8. Naivna rešitev T: O(1) S: O(M) Z: M		
9. razpršilna tabela z verižnjem		
10. Razpršena tabela z naslavljanjem		T: O(1) prič. S: O(n) = m·|k| bitov

$T(n) = O(1)$; $e(= \{k, d\}; k \in \mathbb{Z} : 0, \dots, 2^{32} \overset{M}{=}; k \in S$

Naivni:

U: 

Prostor: $O(M) \rightsquigarrow O(n)$

T: 

$\Rightarrow m \sim n$

$h()$: $h(n) = k'$

Razprsilna / Hash fja

```
Find(k):
    return U[k]
```

```
Find(k)
    k' = h(k)
    return T[k']
```

$M \gg m \Rightarrow \exists k_1, k_2 \ (k_1 \neq k_2)$: $h(k_1) = h(k_2)$ — Soupadanje

$h: M \to m$

$Find(k_2) \to d_1$

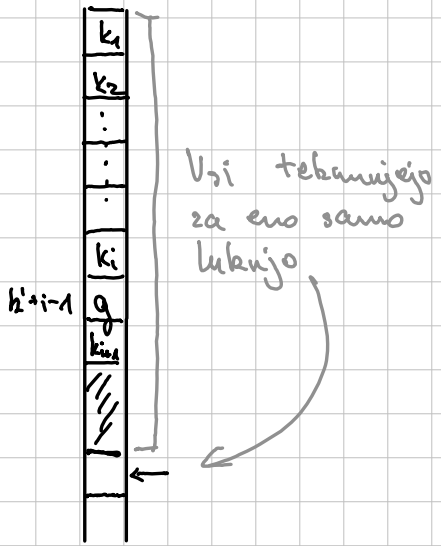
```
Find(k):
    k' = h(k)
    if T[k'].k = k
        return T[k'].d
    else
        return ⊥
```

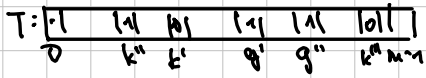
- h: nakljucno razprsi $\forall k: Pr(h(k) = i) = \frac{1}{m}$, $i = 0, \dots, m-1$
  ↳ min # soupadanj & razporeditev nasprotnika
- Soupadanja?

$Hash: S(n) = O(n)$, $T(n): O(1) + O(h) \Rightarrow E(T(n)) = O(1)$

$E(n_k) = ? \Rightarrow \frac{n}{m} < c = O(1)$

$k_n, k_2, \dots, k_i : h(k_i) = k'$ $\quad$ $q : h(q) = k'_{i-1}$





$k, q \in S$ $\quad$ $h(k) = k'$, $h(q) = q'$
$h(x) = k'$
Find(S, x) -> True
$h_n(k) = k''$, $h_n(q) = q'$
$h_n(x) = k'''$

Vrste s prednostjo $\quad$ 21.11.24

Posplošeni slovar
- Operacije:
  - Insert, search, delete
  - Rang, izbira, levi-, desni sosed, sosed

Vrsta s prednostjo
- Abstraktna podatkovna struktura
- Operacije: Insert(S, x), Min(S), DelMin(S)
- Posplošena ima še op: izločanje, spreminjanje, zlij, levi-, desni sosed

Povezani seznam

$T_{ins}(n) = O(1)$, $T_{min}(n) = O(n) \mid O(1)$, $T_{delmin}(n) = O(n)$

↳ izloči in najde nov min

Urejen P.S.

$T_{ins}(n) = O(n)$, $T_{amin}(n) = O(1)$, $T_{delmin}(n) = O(1)$

Implementacija z uravnoteženim iskalnim drevesom

Ins: Vstavljanje v drevo

Min: Si zapomnimo

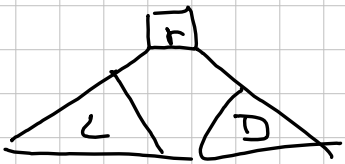
DelMin: V D|D

$S(n) = O(n)$

$T(n) = T_{MIN}(n) = O(1)$

$T_{DELMIN}(n) = O(\log n)$?

$T_{INS}(n) = O(\log n)$



$L < r < D$, $\#(L) \approx \#(D)$

$r < L, D$

$||L| - |D|| \leq 1 \Rightarrow h(n) = \lceil \log_2 n \rceil$

Kopica (hip)

- Rekurzivna def. z lastnostmi:
  - kopica sestoji iz korena in dveh podkopic, ki sta lahko prazni
  - najmanjši el. je v korenu
  - v vsaki podkupici je približno enako št. el.

$T_{min}(n) = O(1)$, $T_{ins}(n) = O(\log n) = \lceil \lg n \rceil$, $T_{delmin}(n) =$

Kanonično drevo

- dve spodnji plasti $\Rightarrow h = \lceil \lg n \rceil$

$S(n) = O(n)$

$T(n) = T_{min}(h) = O(1)$

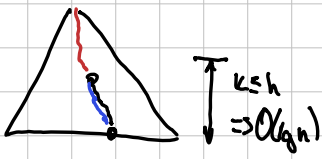
$T_{ins}(n) = O(\lg(\lg n)) / O(\log n)$

1) $T_{delmin} = O(h) = O(\log n)$

2) $T_{delmin} = O(h+h) = O(\log n)$

Insert: ustavimo kot mankajoči list

El. splava navzgor do ustrezne višine



Del Min:

1) izbriši koren
2) nadomesti koren
3) potopi novi koren

1) nadomestni koren

primerjav: $2(\lg n - k)$

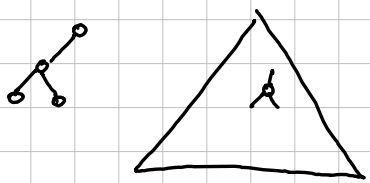
$k \approx \lg(\lg n)$

2) potopi verzel

primerjav: $k = \lg n * k$

$k = \frac{\lg n}{3}$

Dvojiško drevo



Marshaling / Pootrajevanje

koren, |L|, α_1 _____, α_2 _____

predstavitev drevesa

xml zapis drevesa (brez referenc):

<drevo> <koren> koren </koren> <levo> _____ </levo>

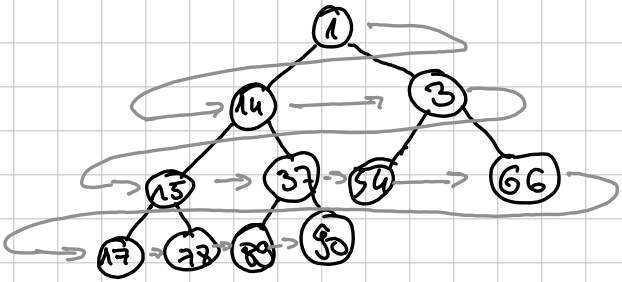
<desno> _____ </desno>

</drevo>

json tudi brez referenc

Eksplicitne PS - reference vijne
Implicitne PS - brez referenc

levo kanonično drevo



1	14	3	15	37	54	66	17	78	89	90
0	1	2	3	4	.	.	.			
1	2	3	4	5	6	7	8	9	10	11

koren: $ind_o = 2^k + g$
otrok: $ind_L = 2^{k+1} + 2g = 2\,ind_o$
$ind_D = ind_L + 1 = 2\,ind_o + 1$
starš: $ind_s = \lfloor \frac{ind_o}{2} \rfloor$

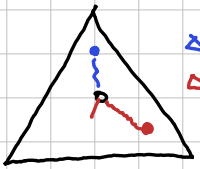
27.11. 2h

Vrste s prednostjo

operacije (OS):	B.Heap
Insert(x)	O(h)
Min(S)	O(1)
DeleteMin(S)	Oh)
ChangePrio((x, Δ)	O(h)
Delete(x)	O(h)
Merge(H_1, H_2)	

Dvojiška kopica
1. Koren<LD ⇒ 2 podkopici
2. Kanonično drevo ⇒ $h = \log n$

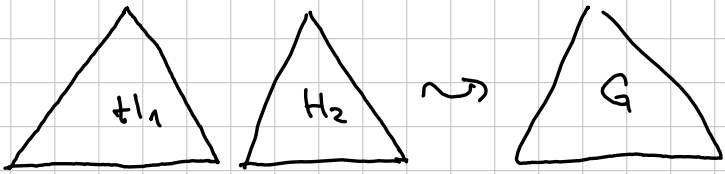
ChangePriority (x, Δ)



$\Delta < 0$: $O(h)$ - ena primerjava
$\Delta > 0$: $O(h)$ - [št. otrok] primerjav

Delete(x):	
ChangePriority $(x, -\infty)$	$O(h)$
DeleteMin()	$O(h)$

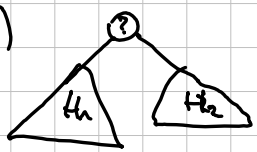
Merge:



$|H_1| \geq |H_2| = \mu$

1) Merge:
for x in H_2:
Insert(H_1, x) $O(\mu \log_2 n)$

2)

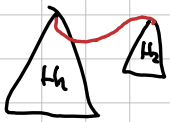


$? = \min(H_{1\min}, H_{2\min})$

3) $\forall e \in H_1, H_2$: damo v seznam, zgradimo kopico
$O(n)$ $O(n)$

1. a) Koren $\in S_1, S_2, \dots S_n$

Merge $(H_1, H_2) \to G$
recimo: $k_1 > k_2$
$T(n) = O(1)$



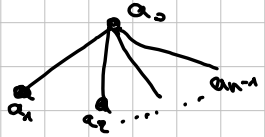
(H_2 postane podkopica)

Primer:
Najdobo $a_0, a_1, \dots, a_{n-1} : a_i > a_{i+1}$

```
pq = MakePQ(a_0)
for i = 1 .. n-1
    pq = Merge(pq, MakePQ(a_i))
```

$O(1)$

Primer:
Imamo $a_0, \dots, a_{n-1} : a_0 < a_i, i > 0$



Recimo naslednje zap.:

```
Najdi a_i v pq
Min(pq)
DeleteMin(pq) -> pq
```

$n \cdot O(1) = O(n)$
$\Downarrow$
$O(m)$ m = št. otrok
$1 \le m < n-1$

$m \approx h = \log n$

Bitno seštevanje

$X_1 = 86_{(10)} = 1010110_{(2)}$

$X_2 = 48_{(10)} = 110000_{(2)}$

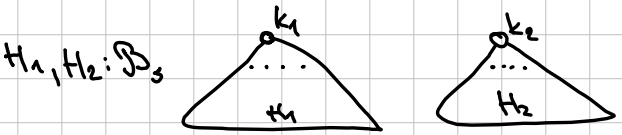
$100001 10_{(2)}$

Velikost bitnega zapisa št. x: $\lceil \log_2 x \rceil$

$T(x) = O(\#bita) = O(\log x)$

Imamo strukturo

1) Ima koren < vsi elementov
2) Ima 2^s elementov
3) Koren ima s otrok
4) Podstrukture so $B_0, B_1, \dots, B_{s-1}$

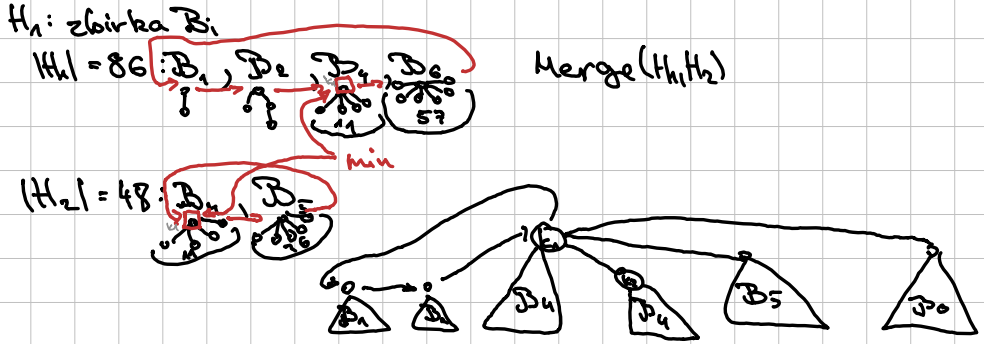


$G \leftarrow$ Merge (H_1, H_2):

Recimo: $k_1 > k_2$ $O(1)$

$|G| = 2^{s+1}$; otrok ima $s+1$ in min je koren $\Rightarrow B_{s+1}$

H_1: zbirka B_i



$T_{MERGE}(n) = O(\log n)$

	Dvoj. K.	Bin. K.	Lena bin. K	c	$\Delta\Phi(H)$
Min	$O(1)$	$O(1)$	$O(1)$	$O(1)$	$\emptyset$
Insert	$O(h)$	$O(\log n)$	$O(h)$	$O(1)$	$\emptyset$
DeleteMin	$O(h)$	$O(\log n)$	$O(n)$	$O(\log n)$	
ChangePrio	$O(h)$	$\Delta<0: O(\log n)$	$\Delta<0: O(\log n)$	$O(\log n)$	$\Delta<0:$
Delete	$O(h)$	$O(\log n)$	$O(n)$	$O(\log n)$	
Merge	$O(n)$	$O(\log n)$	$O(1)$	$O(1)$	$\emptyset$

$h=\log n$

Insert(pq,x): Merge(pq, CratePQ(x)) $O(\log n)$

Delete(pq,x): ChangePrio(x, $-\infty$); DeleteMin(x) $O(\log n)$

DeleteMin(pq): 1. iz pq # izločimo B_k, kjer je koren min $\to H'$ $O(1)$

2. iz B_k naredimo novo pq G, ki sestoji iz otrok $O(\log n)$ korena B_k

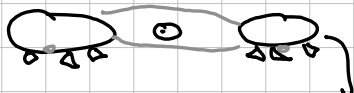
3. Merge(H', G) $\to$ Rezultat $O(\log n)$

$O(\log n)$

Lena binomska kupica

- sestoji iz binom. dreves B_i
- vemo kaj je min in kje je
- Največji $B_i = B_{\log n}$

Merge (H_1, H_2):



DeleteMin: -Iskanje Min - $O(n)$

-CleanUp dobimo bin K. $O(n)$

Kij

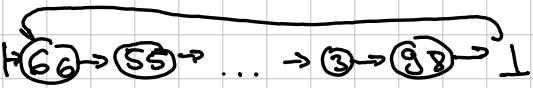
Kakršno koli zap. op. imamo, potem je okp. cena op. kot je bi pomnožili vse op skupaj

4.12.24

$U = \{a_1, a_2, \ldots, a_n\}$

- $S_i \subseteq U$, $i = 1, \ldots k$
- $S_i \cap S_j = \emptyset$, če $i \neq j$
- Make Set $(a) \rightarrow S_n$
- Union $(S_i, S_j) \rightarrow S_{ij}$
- Find $(a) \rightarrow @S_i$ - najde množico v kateri je element

$S_1 = \{66, 55, 85, 3, 98\}$
$S_2 = \{77, 96, 24, 8, 12, 64\}$
Find $(66) \rightarrow 66$, Find $(3) \rightarrow 66 \Rightarrow a_1, a_2$ v isti mn. s predstavnikom
Find (85) ? = Find (12)

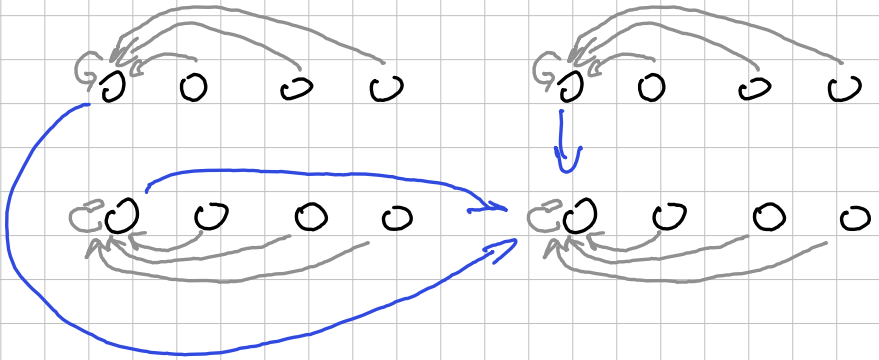
1. 

$T_F(n) = O(n)$

2. 

$T_U(n) = O(1)$

3. 
$T_U(n) = O(1)$



$T_M(n) = O(1)$
$T_U(n) = O(1)$
$T_F(n) = O(n) \Rightarrow O(\log n)$

Urejanje

- Z ustavljanjem $O(n^2)$

11.12.24

Urejanje
- Zlivanje } Deli &
- Hitro } Vladaj

Urejanje z VP
- $O(n \log n)$ primerjav ključev
- Zahtevnost problema: $\Omega(n \log n)$
- Model: ključe primerjamo $(>, <), (\leq, >), (\geq, <)$
- Model: Primerjalni

RAM model:
- Štejemo operacije
- Operacije nad ključi
- Znamo urejati v $O(n \log n)$

Cela št. $Int = [0, 2^{32}-1]$, $2^b = M$, $b = \{1, 2, 4, 8, 16, \dots\}$
$|Int| = M$

Mn. $A \subseteq Int$:
$Rang_A(x) = |\{y \in A \mid y \leq x\}|$

$A = \{70, 23, 36, 42, 15, 17\}$ → 15, 17, 23, 36, 42, 70

π_1 | 6 | 3 | 4 | 5 | 1 | 2 | → permutacijski vek. } Permute: v času $O(n)$

→ 15, 17, 23, 23, 36, 42, 70

$A = \{70, 23, 36, 42, 15, 17, 23\}$

π_2 | 7 | 4 | 5 | 6 | 1 | 2 | 3 | Stabilno → 15, 17, 23, 23, 36, 42, 70

Nestabilno → 15, 17, 23, 23, 36, 42, 70

```
MergeSort(A):
    A => A1, A2
    A1' <- Sort(A1)
    A2' <- Sort(A2)
    i1 = i2 = 1
    For i = 1...n:
        If A1'[i1] ≤ A2'[i2]
            A[i] = A1'[i1]; i1++
        Else
            A[i] = A2'[i2]; i2++
```

A: | 70 | 23 | 36 | 42 | 15 | 17 | 23 |

$R[i] = Rang_A(A[i])$

R: | 7 | 4 | 5 | 6 | 1 | 2 | 4 |

```
Vector R(A):
    n = |A|
    For i = 1...n:                T = O(n^2)
        R[i] = 0
        For j = 1...n:
            If A[i] ≥ A[j]:
                R[i] = R[i] + 1
    Return R
```

B: | 15 | 7 | 12 | 14 | 2 | 5 | 7 |

$B[i] \in \{0, \dots, 2^4 - 1\}$

$C[0, 2^6 - 1]$: $C[i] = \#$ pojawiania i w A

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
C:	0	0	1	0	0	1	0	2	0	0	0	0	1	0	1	1

Predponska vsota : $P[i] = \sum_{j=0}^{i} C[j]$

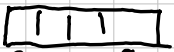
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
P:	0	0	1	1	1	2	2	4	4	4	4	4	5	5	6	7

```
PrefixSum(C):
    P[0] = C[0]                          T = O(M)
    For i = 1,..,M-1
        P[i] = P[i-1] + C[i]
    Return P

Count(A):
    C[*] = ∅          /*|C| = M*/
    n = |A|                              T = O(n)
    For i = 1,...,n
        C[i] = C[i] + 1
    Return C

Uredi(A):
    C = Count(A)          O(n+M)   }
    P = PrefixSum(C)      O(M)     }  ⟶  T = O(n+M)
    n = |A|                              ↑
    For i = 1,...,n:                     ] O(n)
        R[P[A[i]]] = A[i]   /* R[π[i]] = A[i] */
    Return R                             ↓
                                         če n = Θ(M) ⇒ lin. urej.
```

Nadgrajen:

```
Uredi(A):
    C = Count(A)
    P = PrefixSum(C)
    n = |A|
    For i = n...1:
        R[P[A[i]]] = A[i]
        P[A[i]] = P[A[i]] - 1
    Return R
```

$a \in \{0, \dots, 2^{32}-1\} = a_3\ a_2\ a_1\ a_0 : a \quad a_i = 0, \dots, 255 \quad a > b$
$b_3\ b_2\ b_1\ b_0 : b$

a: 

Razdelitev na koše / vedra

For $a \in A$:
    Vržemo v koš $k(a)$] $T = O(n)$

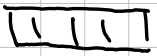
a: $a_3, a_2\ a_1\ a_0$

$M = 2^{32}$: $0, \dots, 2^{32}-1$ → 8 enako velikih košev

$b(0) = 0, \dots, 2^{29}-1$ $\qquad k[i] < k[i+1]$
$k(1) = 2^{29}, \dots, 2^{29}\cdot 2 - 1$
$k(2) = 2^{29}\cdot 2, \dots, 2^{29}\cdot 3 - 1$
$k(3) = 2^{29}\cdot 3, \dots, 2^{29}\cdot 4 - 1$
$\vdots \qquad\qquad \vdots$
$k(7) = 2^{29}\cdot 7, \dots, 2^{29}\cdot 8 - 1$

Urejanje vedra
    #el. v vedru $\nu = O(1) : T = O(1)$
    # el. v vedru $\nu = \mu : T = O(\nu)$

Korensko urejanje
$A : \{a_0, a_1, \dots, a_n\}$
$a_i : a_{i,\ell-1}$

$36217 \in \{2^{16}\}$
$b = 10 \quad \ell = 5$
$b = 16 \quad \ell = 4$ 

Korak s $a_i < a_j \Leftrightarrow \alpha < \beta$

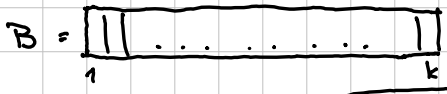
$a_i : a_{i,\ell-1}\ a_{i,\ell-2} \dots a_{i,0}$ ($a_{i,s-1} \dots a_{i,0}$: α)
$a_j : a_{j,\ell-1}\ a_{j,\ell-2} \dots a_{i,0}$ ($a_{j,s-1} \dots a_{j,0}$: β)

1) $a_{i,s} < a_{j,s} \Rightarrow a_i$ pred a_j
2) $a_{i,s} > a_{j,s} \Rightarrow a_j$ pred a_i
3) $a_{i,s} = a_{j,s} \Rightarrow a_i$ pred a_j, ker $\text{bo } \alpha < \beta \Rightarrow$ Stabilno urejanje

18.12. 2024

$A = \{a_1, a_2, \dots, a_n\}$
Select(A, i) →

i = 1: Min
i = 2:
i = 3:
⋮
i = n: Max
i = k ($k = O(1)$): $O(n \cdot c) \leftarrow O(n)$
($k = \omega(1)$):



→ · Sort(A); A[k] ⇒ $O(n \log n)$
· ∀i: Insert(T, a_i); Select(t, k)
Insert(T, a_i): $n \log n$
Select(t, k): $\log n$

Mediana (k-ti el po velikosti)

```
QuickSort(A)
  p = Pivot(A)
  (A1, A2) ← Partition(A, p)   // A1 < p, A2 > p   O(n)
  QuickSort(A1), QuickSort(A2)
  if |A1| k:                    // |A1| < k ⇒ k ∈ A2 ∧ |A1| ≥ k ⇒ k ∈ A1
    QS(A1, k)
  else:
    QS(A2, k - |A1|)
```

$T_{QS}(n) = c \cdot n + T(n - \min(|A_1|, |A_2|))$
WC: $T_{QS}(n) = O(n^2)$
BC: $T_{QS}(n) =$

$$n + \frac{n}{2} + \frac{n}{4} + \dots + \frac{n}{2^i} + \dots + 1 = 2n - 1$$

$$\Rightarrow n + nk + nk^2 + \dots + nk^i \dots = n \cdot \left(\sum_{i=0}^{\infty} k^i\right) = O(n)$$

Quick Select

- Razlika od QuickSorta je da ne potrebujemo iskati po obeh podmnožica

Št. el. zagotovo večjih od pivota vsaj:
$3(\lceil \frac{1}{2}\lceil \frac{n}{5} \rceil \rceil - 2) \geq \frac{3n}{10} - 6$

Št. el. zagotovo manjših od pivota:
$n - (\frac{3n}{10} - 6) = \frac{7n}{10} + 6$

- $T_{QSel.}(n) \leq T(\lceil \frac{n}{5} \rceil) + T(\frac{7n}{10} + 6) + O(n) = O(n)$

Dinamično programiranje

- Rekurzija
- Pomnenje (memorizacija)
- Deli in vladaj
- Grobа sila - sistematični pregled

- Rekurzivna def.: preprosto do programa
- Pomnenje: ni več ponavnega računanja
- Tabela pomnenja vrednosti: lahko naračunamo drugače

- Levenshteinova razdalja
  - Razdalja med zapisoma $\leftarrow d(n-1, n-1)$
  - 1) Zamenjaj: pretvori $(u_A \to u_B) \wedge$ zamenjaj $(c_A \to c_B) - c_z$
    2) Vstavljanje: pretvori $(A \to u_B) \wedge$ ustavi $(c_B) \leftarrow c_v$ $\quad d(n, n-1)$
    3) Brisanje: briši $(c_A) \wedge$ pretvori $(u_A \to B)$ $\quad d(n-1, n)$
- Izrazim Levenshteinove razdalje

$$d(A,B) = \begin{cases} m \cdot c_1 & n = 0 \\ n \cdot c_0 & m = 0 \\ \min(c_0 + d(u_0, B), d(A, u_B) + c_1, d(u_A, u_B) + c_r) & \text{sicer} \end{cases}$$