

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. Dožar: Zbirka nalog

Domače naloge:

- 6 x 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%

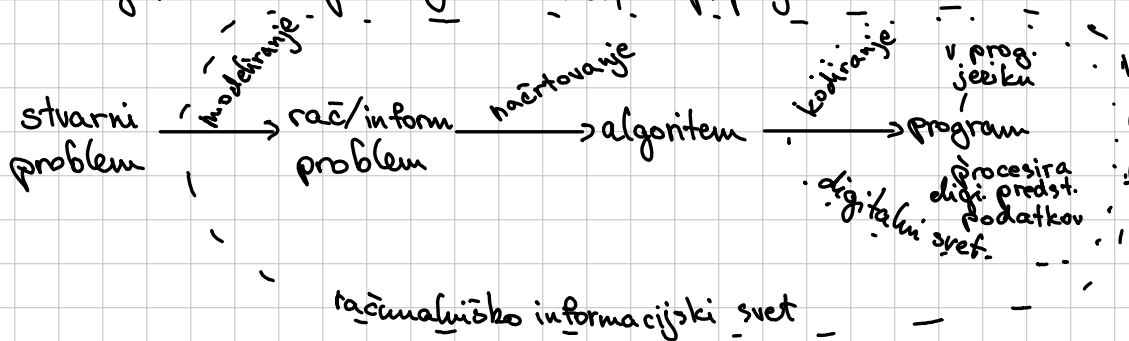
$$\lg = \log_2$$

$$\log = \log_{10}$$

UVOD IN MATEMATIČNE OSNOVE

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

Algoritem - zaporedje korakov, ki pripelje do rešitve



Algoritem

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

Insertion sort:

For $j = 2 \dots n$

poišči največji $i < j$, da $A[i] < A[j]$

premakni $A[k] \rightarrow A[k+1]: k = i+1 \dots j-1$

prestavi kar je bilo v $A[j] \rightarrow A[i+1]$

Orodje za preverjanje pravilnosti delovanja znanca 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 $A \sim T(A)$ - št. korakov (izvedenih osnovnih operacij) alg. A pri vhodnih podatkih velikosti n .

Predp., 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) do bimo lin. $\mathcal{O}(n)$, v najslabšem kvadratno $\mathcal{O}(n^2)$

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. $n! \sim \sqrt{2\pi n} \left(\frac{n}{e}\right)^n C \Rightarrow \lg(n!) \approx n \lg(n) + n C_n$
 - ↳ ni alg. hitrejšega od $n \lg n$

Ocena velikosti funkcij in veliki O

- Za dano fijo $g(n)$ označimo $O(g(n))$ množico f'ij

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

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

- Če je $h(n) = O(g(n))$, potem rečemo da $g(n)$ asimptotično od zgoraj omejuje $h(n)$
 - ↳ nenegativne f'ie, natančneje bi bilo $h(n) \in O(g(n!))$

*

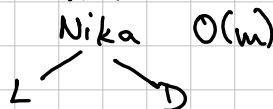
9.10.24

ŠTEVILSKA DREVEŠA

Dvojiško drevo za leksikografsko razporejanje

imen : Nik \$
Nika \$
Nikolaj \$

ustavi : Nikita?



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

Črka $e \in \{A-Z, a-z, \dots\} = \Sigma$ ^{Abeceda}
 $|\Sigma| = O(1)$

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

Trie (iz reTRIEval)

- Rekurzivna podatkovna struktura
 - Ključ tako veliki, da ne moremo naenkrat primerjati
 - Vrednosti so v listih
- *

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

hiter, 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 ključne $\{A, C, E, G, H, I, L, M, R, S\}$

Find(01000)

Iskanje je v 2 fazah:

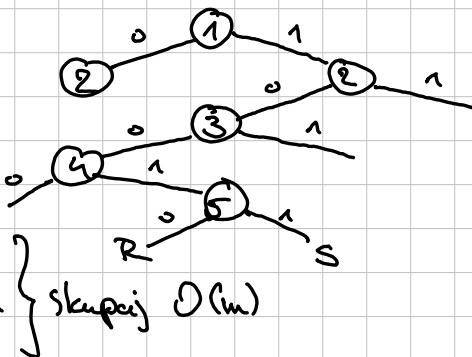
1. Spuščanje do lista

• kandidat

• čas. zaht. $h(\text{trie}) = m$

2. Preverjanje kandidata

• čas. zaht. m



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

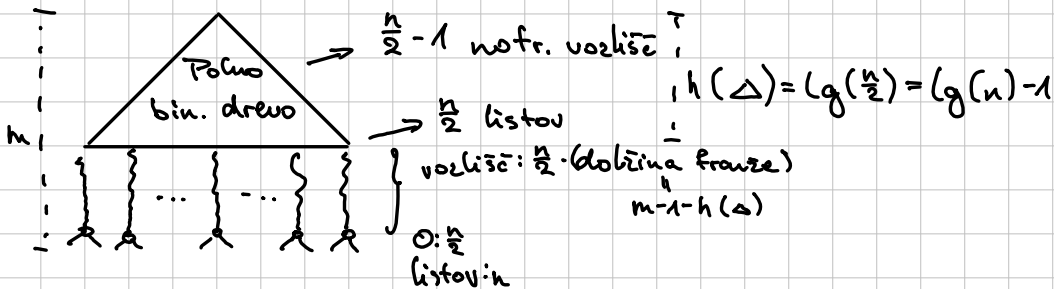
Operacije

• Običajne: Najdi, Dodaj, Izloči

• Dodatne: Najdi Manj, Najdi Več

• Po splošena: Najdi

$n = \# \text{ elementov}$, $m = \text{velikost ključa}$ ($m=5$)



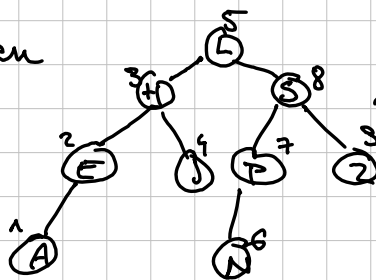
Prostorska zahtevnost:

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

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

Vmesni obhod (in order traversal):

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



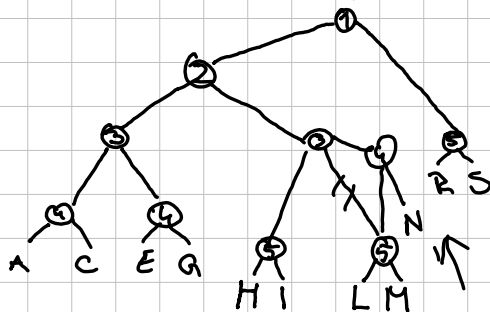
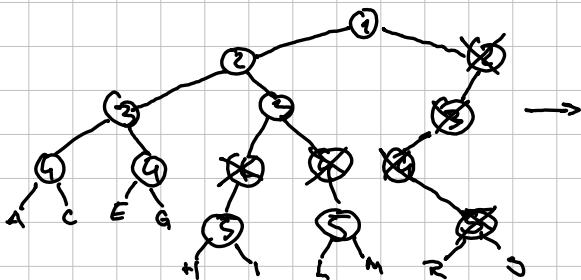
v lin. času
dobimo sortirane
elemente

(isto pri trieju)

Trie s stisnjenimi potmi (path-compressed trie)

- $2n-1$ vozlišče
- boljša prostorska zahtevnost $\rightarrow O(n)$

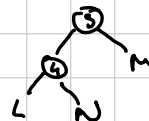
Češ tako da posleda
st. vozlišča in nato prito
mesto v klanu



- Časovna zahtevnost iskanja in vstavljanja: n

Insert (01110) → Pridemo do L-ja (L: 01100)

↓
Novo ustranje vozlišče na četrtem mestu (lahko tudi namesto L-ja)



Potem ni več sortirano.)

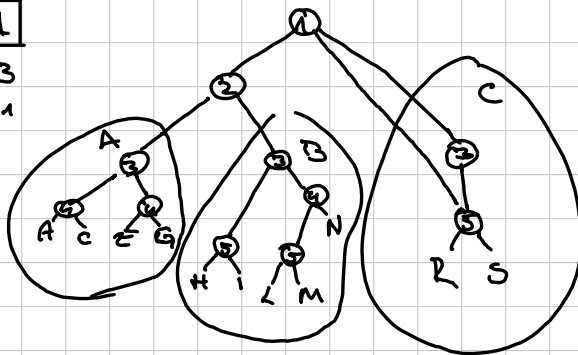
- Izločimo se časovno zahtevnost

⇒ Prazno

A	B	C	L
---	---	---	---

0 1 2 3

00 01 10 11



Find (0100)
↳ Prikranimo čas, ker ne rabimo primerjave na 2

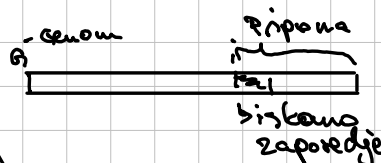
Oz. še več:

000 001 010 011 100 101 110 111

K	B	g	δ	λ	l	l	l
---	---	---	---	---	---	---	---

- ↳ Nivojsko stisnjena drevesa (level-compressed tries - LC trie)
 - Izberemo, koliko praznega prostora smo pripravljени imeti v tabeli

Primer: Iskanje gena v genomu: znamo ngiti na začetku in kjer kolivmes v αm



23.10.24

Isci(s, k) \rightarrow statična PS

Vstavi(s, k) \rightarrow polidinamična PS

Brisi(s, k) \rightarrow dinamična PS

Slovar

*

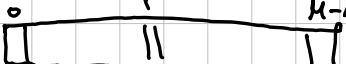
AVL	BBST	BBT	RTPS	UPS	PS	Naiv	
$O(\log n)$	$O(\log n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(M)$	Isci
	$O(n)$	- -	- -	- -	$O(n)$	$O(1)$	Vstavi
	$O(n)$				$O(n)$	$O(1)$	Brisi
	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(M)$	skupaj

Naivna implementacija

kljuc = Integer

Mat: integerl = ∞

Rač: integerl = $2^{32}, 2^{64}, \dots = M$

B: 

recimo $i \in p \Leftrightarrow B[i] = 1$

$j \in p \Leftrightarrow B[j] = \text{podatki}$

Isci(p, k): return B[k]

Vstavi(p, k, a): $1 \neq B[k] \Rightarrow 1 \cdot B[k] = a$

Brisi(p, k): $B[k] = 1$

Časovna zahtevnost: $O(M)$

čas. zahtevnost je
d (ustopne točke, najdalj
oddaljeno vozlišče)

Sernami

- $s = \perp \Rightarrow \text{Prozen}$
- $s = h : t$
 \uparrow elem. \uparrow rep(rudi sernam)

Isi(s, k):

if $s \equiv \perp$: return \perp
if $s.h.k = k \Rightarrow$ return $s.h.d$.
return $Isi(s.t, k)$

Votavi($s, (k, d)$): return $sernam((k, d), s)$

Brisi(s, k)

if $s \equiv \perp$: return \perp
if $s.h.k = k$: return $s.t$
return $sernam(s.h, Brisi(s.t, k))$

Isi(s, k):

if $s \equiv \perp$: return \perp
if $s.h.k \equiv k$: return $s.h.d$
if $s.h.k > k$: return \perp
return $Isi(s, k)$

$s = \perp \mid e \ t_1 t_2 \Rightarrow h = \mathcal{O}(\lg n)$

Isi(s, k):

if $s \equiv \perp$: return \perp
if $s.e.k = k$ return $s.e.d$
 $d = Isi(t_1, k)$: return $Isi(t_2, k)$
return d

$$T(n) = 1 + T(n) + T(n - n - 1)$$

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)

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

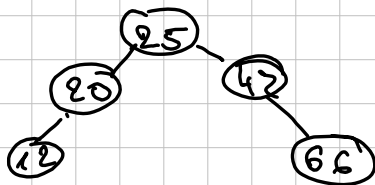
$$n_1 = 0 \Rightarrow a + T(n-1) = O(n)$$

želim: $h(n_1) \approx h(n - n_1 - 1)$

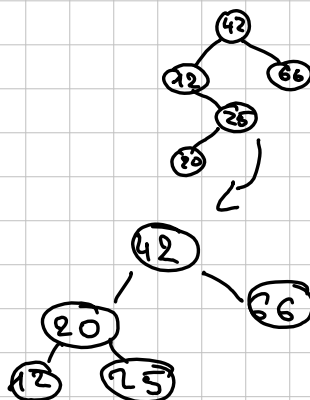
ko res: $n_1 \approx n - n_1 - 1$

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

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



Vstavi(20)



$$UD = \perp \mid r \in D: L < r < D, ||L| - |D|| \leq 1, |h_L - h_D| \leq 1$$

Iterate(s, f()):

if s = ⊥: return ⊥

return seznam(f(s.h), Iterate(s.f, f(i))

$h' = f(s.h)$

$t' = \text{Iterate}(s.t, f())$

return seznam(h', t')

$t' = \text{Iterate}(s.t, f())$ } rakou

ali $h' = f(s.h)$

Iterate(dr, f()):

if $dr \equiv \perp$: return \perp

$r' = f(dr.r)$

$L' = \dots$

$L' = \dots$

$L' = \text{Iterate}(L, f())$

$r' = \dots$

$D' = \dots$

$D' = \text{Iterate}(D, f())$

$D' = \dots$

$r' = \dots$

return Tree(r', L', D')

Vstavi(dr, (k, d)):

if $dr \equiv \perp$: return tree((k, d), \perp, \perp)

if $dr.r.k = k$: return dr

if $dr.r.k < k$:

Return Tree($dr.r, dr.L, \text{Insert}(dr.D, (k, d))$)

Najvecji(dr):

if $dr \equiv \perp$: return \perp

if $dr.D \equiv \perp$: return $dr.r$

else return Najvecji($dr.D$)

Najmanjša možna št. el. v drevesu višine h:

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

slabo: $h_2 = h_0 - 1$

$$\Rightarrow h = h_0 + 1 \Rightarrow$$

$$h_0 = h - 1$$

$$h_2 = h - 2$$

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

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

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

30.10.24

Vstavi (Elt)

Briši (ključ) \rightarrow Elt brišemo

Najdi (ključ) \rightarrow podatek (MAP)

Elt: ključ, podatek

	Arr	LL	OL	BST	AVL	B
V	\uparrow	\uparrow	\uparrow	\uparrow	\uparrow	
B	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(h)$	$O(\log n)$
N	\downarrow	\downarrow	\downarrow	\downarrow	$h = \log n$	$h = \frac{\log n}{\log b}$
S	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$

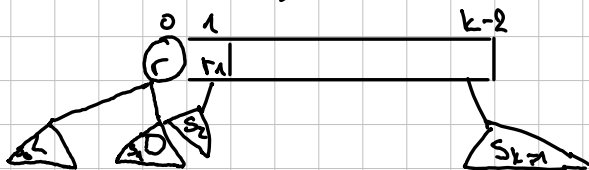
Čas. zahtevnost je odvisna od razdalje med vstopno točko in najbolj oddaljeno točko $PS \equiv h$

Arr: Prostorska zahtevnost prevelika

LL: Časovna zahtevnost lin. sorazmerna s št. el.

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

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



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

$r = r$:

$$L = S_0 < r_0 < S_1 = D$$

$$S_1 < r_1 < S_2$$

$$r_i < r_j \mid i < j$$

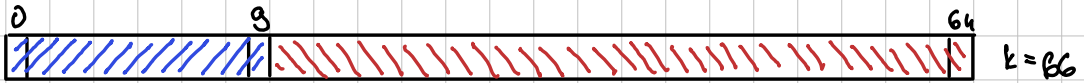
$$h(S_i) = h(S_j)$$

-koliko el. v drevesu h-višine

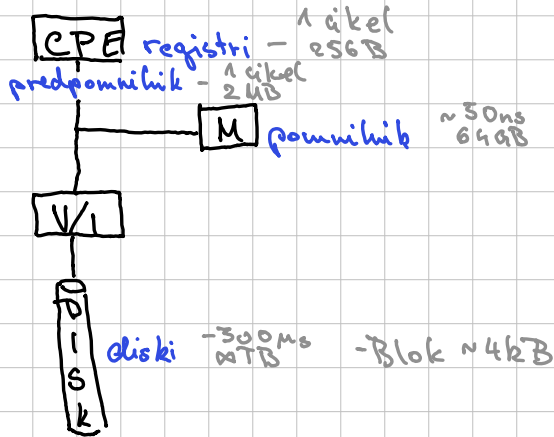
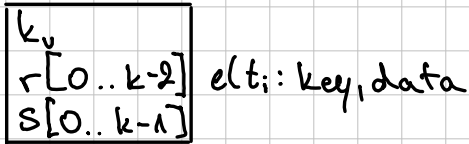
$$(k-1) + k(k-1) + k^2(k-1) + \dots + k^{h-1}(k-1) = (k-1) \cdot \sum_{i=0}^{h-1} k^i = (k-1) \cdot \frac{k^h - 1}{k - 1} = k^h - 1 \leq n_{\max}$$

$$h \log k \geq \log(n+1)$$

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

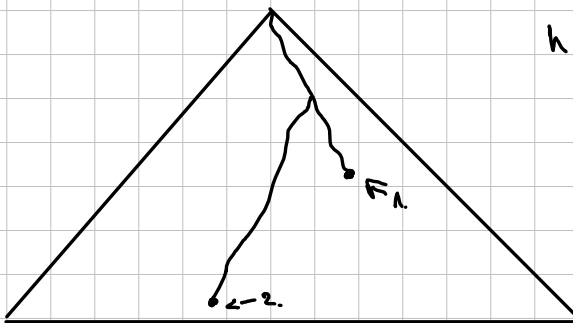


$$k_v = 10$$

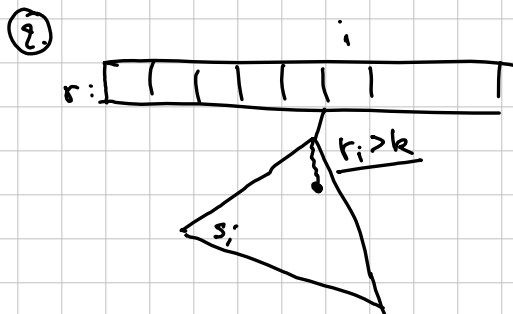
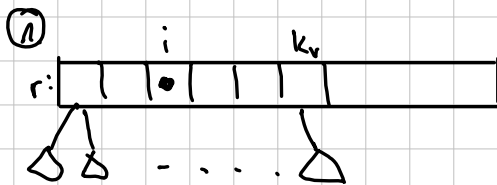


B-drevo (Blokovno drevo)

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



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

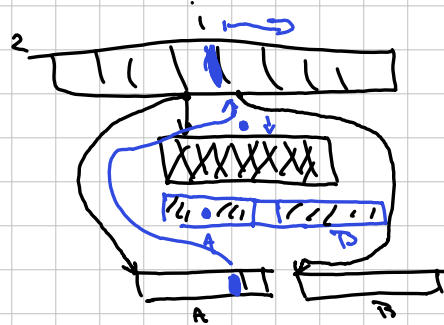
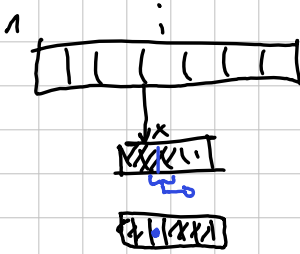
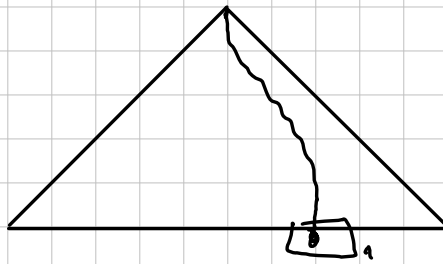


```

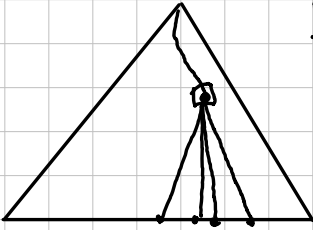
def Najdi(T, k):
    if t = 1 return null
    i = 0
    while (r[i] < k) ^ (i < b)
        i = i + 1
    if r[i] = k:
        return r[i].data
    return Najdi(s[i], k)

```

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



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



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

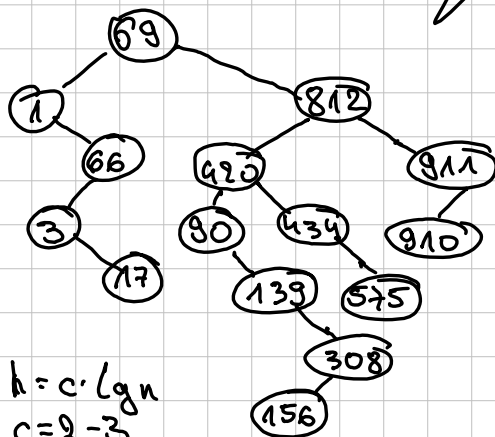
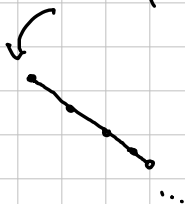
TTF Drevesa (2-3-4 drevesa)

6.11.24

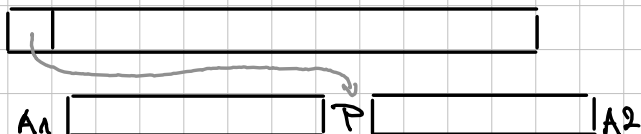
	S	T	E	Opombe
LL	↑	$O(n)$	L	
OLL	$O(n)$	$O(n)$	L	
BST	↓	$O(n)$	S	
BBST		$O(h)$	T	$h = \log_2 n$
SLST	$O(n)$	$O(h)$	S/T	$h = \log_2 n$

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



$A: 1..n$



Preskočni seznam

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

$$\begin{aligned}
 1: & p(d=1) = \frac{1}{n_1} = \frac{1}{2^{\lg n}} \\
 2: & p(d=2) = \frac{n_1}{2^{\lg n - 1}} \\
 & \vdots \\
 d: & p(d) = \frac{1}{2^{\lg n(d-1)}} ; \lg n \geq d
 \end{aligned}$$

RANG IN IZBIRA

$\text{Rang}(S, x)$:

elementov iz S tako, da so manjši ali enaki x .

$$\text{Rang}(S, 337) = 9$$

$$\text{Rang}(S, 308) = 9$$

$$\text{Izbira}(S, i) := \{a, (a \in S) \wedge (\text{Rang}(S, a) = i)\}$$

$$\text{Izbira}(S, 9) = 308$$

$$\text{Izbira}(S, 1) = 1$$

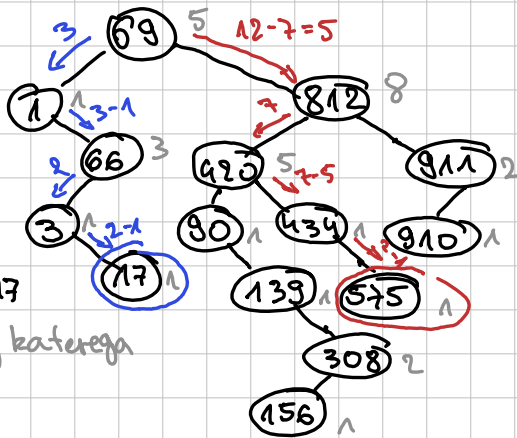
Razširjanje

Drevo: koren, L → Bin. drevo

Izkažo 2-drevo: koren, $L, D \mid L < \text{koren} < D$

AVL drevo: koren, $L, D \mid L < \text{koren} < D \mid h$

RL/AVL drevo: koren, $L, D \mid L < \text{koren} < D \mid h$ & Rang korena



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

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

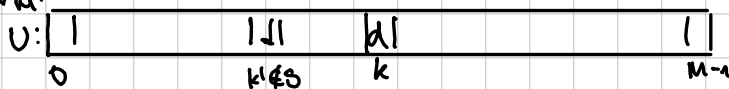
13. 11.24

Uspešna PS za slovar, najslabšem primeru } cilj PSA
 Izboljšanje npr časovne zahtevnosti, prost. zahtevnosti }
 Najboljši alg. za slovar splošno zahtevnost
 Ogle dati različne implementacije za različne primere } side
 Ugotavljanje kolaj je impl. vredn. product

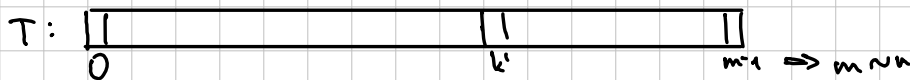
Kaj?	Zakaj? (problem)	Kako? (rešitev)
1. poverani seznam	čas. zahtevnost	urejenost
2. urejeni poverani seznam	čas. zahtevnost	1 rep → poddrevo
3. Izborno binarno drevo	čas. zahtevnost	omejiti k_{max} → uravnotežiti
4. Uravnoteženo bin. drevo / AVL	poenostavitev	žrtvovali prost. za preprostost
5. B in B+ drevo	prost. zahtevnost	uvajemo dvojiska drevesa
6. TTF-drevo, rdeče-črno drevo	$S: O(n)$ $T: O(\log n)$ $Z: 2$	
7. Preskočni seznam $T: O(\log n)$ $S: O(n)$ $Z: 5$	poenostavitev	uvajemo naključnost za poenos. žrtvujemo npr → prižakujemo
Bistvo predavanj		
8. Naivna rešitev $T: O(n)$ $S: O(n)$ $Z: M$	čas. zaht	žrtvujemo prostor za čas
9. razpršilna tabela ? veriziranjem	Prostor	žrtvujemo najsl. prim. za velikost (→ priž. č. a. s.).
10. Razpršena tabela z naslavljanjem		$T: O(1)$ priž. $S: O(n) = m \cdot 161$ bitov

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

Naivai:



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



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

Razpršilna / Hash f'ja

Find(k):
return U[k]

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

$M \Rightarrow m \left\{ \begin{array}{l} \Rightarrow \exists k_1, k_2 (k_1 \neq k_2): \\ h: M \rightarrow m \end{array} \right. \left\{ \begin{array}{l} h(k_1) = h(k_2) \\ \text{Sopadanje} \end{array} \right.$

Find(k₂) → d₁

Find(k):
k' = h(k)
if T[k'].k = k
return T[k'].d
else
return 1

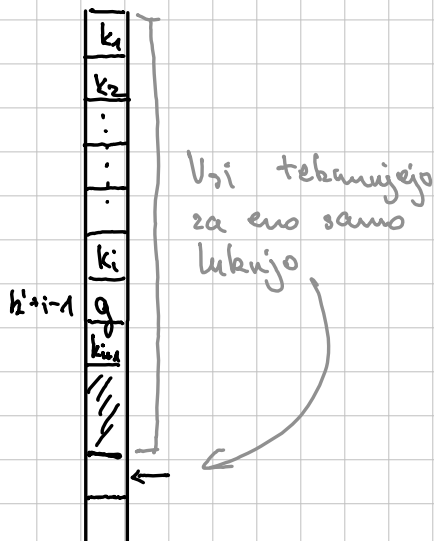
- h: naključno razprši $\forall k: \Pr(h(k) = i) = \frac{1}{m} \quad i = 0, \dots, m-1$
↳ min # sopadanj & razorošimo nasprotitika
- Sopadanja?

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

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

$$k_1, k_2, \dots, k_i : h(k_i) = k'$$

$$g : h(g) = k^{i-1}$$



T: | 1 | 1 | 1 | 1 | 1 | 1 |
 0 k'' k' g' g'' k^{n+1}

$\zeta, g \in S$

$$h(k) = k', h(g) = g'$$

$$h(x) = k'$$

Find(S, x) \rightarrow True

$$h_1(k) = k'', h_1(g) = g'$$

$$h_1(x) = k'''$$