

Algoritmi in podatkovne strukture 2

Algoritmi z rekurzivnim razcepom (»deli in vladaj«)

Luka Fürst

Deli in vladaj

- Metoda reševanja nekaterih problemov
- »Deli«: razdeli problem na medsebojno neodvisne podprobleme, ki imajo enako strukturo kot izhodiščni problem
- »Vladaj«: rekurzivno reši vsak podproblem posebej
- Združi rešitve posameznih podproblemov v rešitev izhodiščnega problema

Primer 1: urejanje z zlivanjem

- **Deli**: razdeli tabelo na dve enako veliki podtabeli
- **Vladaj**: rekurzivno uredi vsako podtabelo posebej
- **Združi**: zlij urejeni podtabeli

Primer 2: hitro urejanje (quicksort)

- **Deli**: izvedi porazdelitev glede na pivot
 - dobimo dve podtabeli, ki pa nista nujno enako veliki
- **Vladaj**: rekurzivno vsako podtabelo posebej
- **Združi**: tega dela nimamo!

Primer 3: dvojiško iskanje (v urejeni tabeli)

- **Deli**: razdeli tabelo na dve enako veliki podtabeli
- **Vladaj**: rekurzivno išči samo v eni podtabeli
- **Združi**: tega dela nimamo!

Časovna zahtevnost

- Recimo, da
 - problem velikosti n razdelimo na podprobleme velikosti n/b_1 , n/b_2 , $\dots$, n/b_k
 - za razdelitev problema na podprobleme in združevanje rešitev podproblemov v rešitev izhodiščnega problema potrebujemo $D(n)$ časa
- Časovna zahtevnost algoritma je potem

$$T(n) = T(n/b_1) + T(n/b_2) + \dots + T(n/b_k) + D(n)$$

Primer 1: Urejanje z zlivanjem

- tabelo razdelimo na 2 podtabeli
- vsaka je velika $n/2$
- $\Theta(n)$ časa za zlivanje

$$T(n) = 2T(n/2) + \Theta(n)$$

Primer 2: Hitro urejanje

- V najboljšem primeru
 - tabelo razdelimo na 2 podtabeli
 - vsaka je velika $n/2$
 - $\Theta(n)$ časa za porazdelitev
 - $T(n) = 2T(n/2) + \Theta(n)$
- V nekoliko slabšem primeru
 - recimo, da vsako (dovolj veliko) (pod)tabelo razdelimo vsaj v razmerju 1:9
 - $T(n) = T(n/10) + T(9n/10) + \Theta(n)$
- V najslabšem primeru
 - ena od podtabel vedno vsebuje samo po en element
 - $T(n) = T(1) + T(n-1) + \Theta(n) = T(n-1) + \Theta(n)$

Primer 3: Dvojiško iskanje

- Tabelo razdelimo na 2 podtabeli velikosti $n/2$
- Rekurzivno iščemo samo po eni podtabeli
- $\Theta(1)$ časa za razdelitev problema na podprobleme in združevanje rešitev podproblema v rešitev izhodiščnega problema

$$T(n) = T(n/2) + \Theta(1)$$

Reševanje rekurenčnih enačb

- Substitucijska metoda

- uganemo rešitev
- z indukcijo dokažemo, da je rešitev pravilna

- Drevesna metoda

- »razvijemo« rekurenco (dobimo drevo)
- ponavadi jo uporabljamo za tvorbo ocen časovne zahtevnosti, ki jih lahko potem preverimo s substitucijsko metodo

- Krovni izrek

- direktna formula, a ima omejeno uporabnost

Substitucijska metoda

- Postavimo **hipotezo**
 - $T(n) \leq cf(n)$ za $n \geq n_0$, če dokazujemo $T(n) = O(f(n))$
 - $T(n) \geq cf(n)$ za $n \geq n_0$, če dokazujemo $T(n) = \Omega(f(n))$
- Hipotezo preverimo z indukcijo
 - predpostavimo, da trditev velja za vse $n' < n$
 - preverimo, ali velja tudi za n
- $\Theta(\cdot)$ dokažemo tako, da dokažemo $O(\cdot)$ in $\Omega(\cdot)$

Substitucijska metoda: primer

- Rešimo enačbo $T(n) = 2T(n/2) + \Theta(n)$
- Zanima nas $f(n)$, za katerega velja $T(n) = O(f(n))$
- Trditev: $T(n) = O(n \log n)$
- Induktivna hipoteza: $T(n) \leq cn \log n$ za vse $n \geq n_0$
- Predpostavimo, da hipoteza velja za vse $n' < n$
 - torej velja tudi za $n' = n/2$
 - $T(n/2) \leq c(n/2) \log(n/2)$
- Preverimo, ali $T(n) \leq cn \log n$ velja tudi za n

Substitucijska metoda: primer

$$\begin{aligned}T(n) &= 2T(n/2) + \Theta(n) \\&\leq 2c(n/2) \log(n/2) + \Theta(n) \\&= cn \log n - cn + \Theta(n)\end{aligned}$$

- $\Theta(n)$ predstavlja funkcijo $kn + \ell$ za poljuben k in ℓ
- c in n_0 lahko poljubno izberemo
- Izberemo ju tako, da je $cn \geq kn + \ell$ za vsak $n \geq n_0$
- Trditev $T(n) = O(n \log n)$ smo dokazali

Substitucijska metoda: past

- $T(n) = T(n - 1) + \Theta(n)$
- »Dokaz«, da je rešitev $T(n) = O(n)$:
 - $T(n) \leq O(n - 1) + \Theta(n) = O(n)$
 - Napaka!
- Poskusiti moramo s $T(n) \leq cn$ (za $n \geq n_0$)
 - $T(n) \leq c(n - 1) + \Theta(n)$
 - $T(n) \leq cn - c + \Theta(n)$
 - Kakorkoli izberemo c , bo $\Theta(n)$ prej ali slej prevladala nad c
 - Torej je $T(n) > cn$ za vse dovolj velike n
 - Trditev $T(n) = O(n)$ je potemtakem napačna
 - (Pravilna trditev je $T(n) = O(n^2)$)

Substitucijska metoda: trik

- $T(n) = 2T(n/2) + \Theta(1)$
- Trditev (pravilna!): $T(n) = O(n)$
- S hipotezo $T(n) \leq cn$ (za vse $n \geq n_0$) se dokaz ne izide

$$\begin{aligned}T(n) &= 2T(n/2) + \Theta(1) \\&\leq 2cn/2 + \Theta(1) \\&= cn + \Theta(1)\end{aligned}$$

- $\Theta(1)$ je nekaj pozitivnega, zato nam ni uspelo dokazati hipoteze $T(n) \leq cn$

Substitucijska metoda: trik

- Uspe pa nam s hipotezo $T(n) \leq cn - d$:

$$\begin{aligned}T(n) &= 2T(n/2) + \Theta(1) \\&\leq 2(cn/2 - d) + \Theta(1) \\&= cn - 2d + \Theta(1) \\&= (cn - d) + (\Theta(1) - d) \\&\leq cn - d \text{ za vse dovolj velike } d\end{aligned}$$

- $T(n) = O(n)$ je torej pravilna rešitev enačbe
 $T(n) = 2T(n/2) + \Theta(1)$

Drevesna metoda

- Včasih je težko uganiti rešitev rekurenčne enačbe
- Drevesna metoda nam pomaga najti kandidatno funkcijo
- Kandidatno funkcijo preverimo s substitucijsko metodo

Drevesna metoda: primer

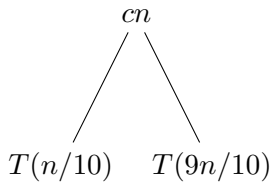
- $T(n) = T(n/10) + T(9n/10) + \Theta(n)$
- Enačbo zapišemo kot

$$T(n) = T(n/10) + T(9n/10) + cn$$

in jo vstavljamo »samo vase«

Drevesna metoda: primer

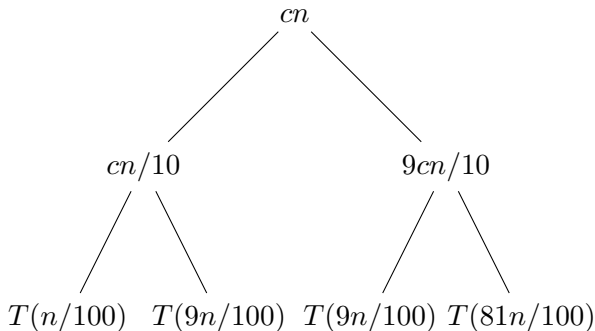
$$T(n) = cn + T(n/10) + T(9n/10)$$



Drevesna metoda: primer

$$T(n/10) = cn/10 + T(n/100) + T(9n/100)$$

$$T(9n/10) = 9cn/10 + T(9n/100) + T(81n/100)$$

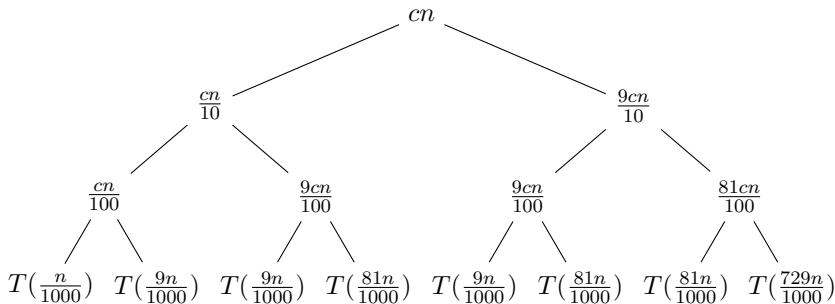


Drevesna metoda: primer

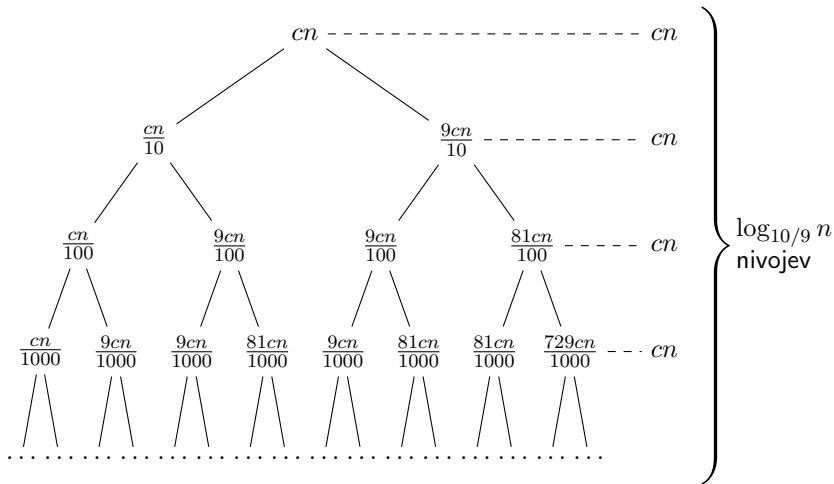
$$T(n/100) = cn/100 + T(n/1000) + T(9n/1000)$$

$$T(9n/100) = 9cn/100 + T(9n/1000) + T(81n/1000)$$

$$T(81n/100) = 81cn/100 + T(81n/1000) + T(729n/1000)$$



Drevesna metoda: primer



Drevesna metoda: primer

- $T(n) = T(n/10) + T(9n/10) + \Theta(n)$
- Hipoteza

$$T(n) = O(cn \log_{10/9} n) = O(n \log n)$$

- Hipotezo preverimo s substitucijsko metodo
- $T(n) \leq cn \log n$ za $n \geq n_0$
- Predpostavimo, da to velja za vse $n' < n$

Drevesna metoda: primer

$$\begin{aligned}T(n) &= T(n/10) + T(9n/10) + \Theta(n) \\&\leq \frac{cn}{10} \log \frac{n}{10} + \frac{9cn}{10} \log \frac{9n}{10} + \Theta(n) \\&= \frac{cn}{10} \log n - \frac{cn}{10} \log 10 + \frac{9cn}{10} \log n - \frac{9cn}{10} \log \frac{10}{9} + \Theta(n) \\&= cn \log n - cn \left(\frac{\log 10}{10} + \frac{\log 10/9}{10/9} \right) + \Theta(n)\end{aligned}$$

- $T(n) \leq cn \log n$, če ustrezno izberemo c

Krovni izrek

Rekurenčna enačba

$$T(n) = aT(n/b) + \Theta(n^d)$$

ima rešitev

$$T(n) = \begin{cases} \Theta(n^d), & \text{če } a < b^d; \\ \Theta(n^d \log n), & \text{če } a = b^d; \\ \Theta(n^{\log_b a}), & \text{če } a > b^d. \end{cases}$$

(Približen) dokaz

Rekurenčno enačbo $T(n) = aT(n/b) + \Theta(n^d)$ zapišimo kot $T(n) = kn^d + aT(n/b)$ in jo vstavljajmo samo vase:

$$\begin{aligned} T(n) &= kn^d + aT(n/b) \\ &= kn^d + ak(n/b)^d + a^2T(n/b^2) \\ &= kn^d + ak(n/b)^d + a^2k(n/b^2)^d + a^3T(n/b^3) \\ &= kn^d + ak(n/b)^d + a^2k(n/b^2)^d + a^3k(n/b^3)^d + a^4T(n/b^4) \\ &= \dots \\ &= \underbrace{kn^d + ak(n/b)^d + a^2k(n/b^2)^d + a^3k(n/b^3)^d + a^4k(n/b^4)^d + \dots}_{(\log_b n + 1) \text{ členov}} \\ &= kn^d \left(\underbrace{1 + \frac{a}{b^d} + \left(\frac{a}{b^d}\right)^2 + \left(\frac{a}{b^d}\right)^3 + \left(\frac{a}{b^d}\right)^4 + \dots}_{(\log_b n + 1) \text{ členov}} \right) \end{aligned}$$

(Približen) dokaz

- Dobimo vsoto

$$T(n) = kn^d \sum_{i=0}^{\log_b n} \left(\frac{a}{b^d}\right)^i$$

- Možnost 1: $a < b^d$

$$\begin{aligned} T(n) &\approx kn^d \sum_{i=0}^{\infty} \left(\frac{a}{b^d}\right)^i \\ &= kn^d \frac{1}{1 - \frac{a}{b^d}} \\ &= Ckn^d \\ &= \Theta(n^d) \end{aligned}$$

(Približen) dokaz

- Možnost 2: $a = b^d$

$$\begin{aligned}T(n) &= kn^d \sum_{i=0}^{\log_b n} \left(\frac{a}{b^d}\right)^i \\&= kn^d (\log_b n + 1) \\&= \Theta(n^d \log n)\end{aligned}$$

(Približen) dokaz

- Možnost 3: $a > b^d$

$$\begin{aligned}T(n) &= kn^d \sum_{i=0}^{\log_b n} \left(\frac{a}{b^d}\right)^i \\&= kn^d \frac{\left(\frac{a}{b^d}\right)^{\log_b n + 1} - 1}{\frac{a}{b^d} - 1} \\&= \Theta \left(n^d \left(\frac{a}{b^d}\right)^{\log_b n} \right) \\&= \Theta \left(n^d \frac{a^{\log_b n}}{(b^{\log_b n})^d} \right) \\&= \Theta \left(n^d \frac{a^{\log_b n}}{n^d} \right) \\&= \Theta(a^{\log_b n}) \\&= \Theta(n^{\log_b a})\end{aligned}$$

Krovni izrek: primeri uporabe

- Urejanje z zlivanjem

- $T(n) = 2T(n/2) + \Theta(n)$
- $a = 2, b = 2, d = 1$
- $a = b^d$
- $T(n) = \Theta(n^d \log n) = \Theta(n \log n)$

- Dvojiško iskanje

- $T(n) = T(n/2) + \Theta(1)$
- $a = 1, b = 2, d = 0$
- $a = b^d$
- $T(n) = \Theta(n^d \log n) = \Theta(\log n)$

Množenje matrik

- Matriki A in B velikosti $n \times n$
- Rezultat: matrika $C = AB$ velikosti $n \times n$
- Brez izgube splošnosti lahko predpostavimo $n = 2^k$
 - če to ne drži, lahko matriko dopolnimo z ničlami
- $\Omega(n^2)$
 - napolniti moramo n^2 elementov
- $O(n^3)$
 - $O(n)$ za vsakega od n^2 elementov
 - $c_{ij} = \sum_{k=1}^n a_{ik} b_{kj}$

Množenje matrik

- Matriki lahko množimo **po blokih**

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \quad B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} \quad C = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$$

$$C_{11} = A_{11}B_{11} + A_{12}B_{21}$$

$$C_{12} = A_{11}B_{12} + A_{12}B_{22}$$

$$C_{21} = A_{21}B_{11} + A_{22}B_{21}$$

$$C_{22} = A_{21}B_{12} + A_{22}B_{22}$$

Množenje matrik

- Namesto enega zmnožka matrik $n \times n$ dobimo 8 zmnožkov matrik $n/2 \times n/2$
- Za razbitje matrike na podmatrike in združevanje zmnožkov podmatrik v zmnožek originalnih matrik potrebujemo $\Theta(n^2)$ časa
 - 4 vsote matrik $n/2 \times n/2$
- $T(n) = 8T(n/2) + \Theta(n^2)$
- $a = 8, b = 2, d = 2$
- $a > b^d$
- $T(n) = \Theta(n^{\log_b a}) = \Theta(n^3)$
- Rekurzivni razcep sam po sebi ni dovolj!
 - moramo zmanjšati število zmnožkov podmatrik

Množenje matrik po Strassenu

$$M_1 = (A_{11} + A_{22})(B_{11} + B_{22})$$

$$M_2 = (A_{21} + A_{22})B_{11}$$

$$M_3 = A_{11}(B_{12} - B_{22})$$

$$M_4 = A_{22}(B_{21} - B_{11})$$

$$M_5 = (A_{11} + A_{12})B_{22}$$

$$M_6 = (A_{21} - A_{11})(B_{11} + B_{12})$$

$$M_7 = (A_{12} - A_{22})(B_{21} + B_{22})$$

$$C_{11} = M_1 + M_4 - M_5 + M_7$$

$$C_{12} = M_3 + M_5$$

$$C_{21} = M_2 + M_4$$

$$C_{22} = M_1 - M_2 + M_3 + M_6$$

Množenje matrik po Strassenu

- 7 množenj namesto 8
- več seštevanj (in odštevanj), a to ne vpliva na asimptotično časovno zahtevnost
 - samo pri členu $\Theta(n^2)$ imamo večjo konstanto
- $T(n) = 7T(n/2) + \Theta(n^2)$
- $a = 7, b = 2, d = 2$
- $a > b^d$
- $T(n) = \Theta(n^{\log_b a}) = \Theta(n^{\log_2 7})$

Množenje matrik po Strassenu

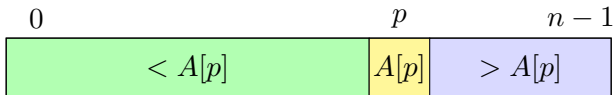
- $\log_2 7 \approx 2,81$ se ne zdi veliko manj kot 3
- Se Strassen sploh splača?
- Da,
 - če sta matriki dovolj veliki (npr. $n \geq 256$)
 - če rekurzijo odrežemo pri, recimo, $n = 64$
 - če drugo matriko transponiramo
 - velja tudi za navadno množenje!

Hitro iskanje (quickselect)

- Problem
 - V podani tabeli z n medsebojno različnimi elementi poišči element na **rangu** k
 - (tj. $(k + 1)$ -vi najmanjši element)
- Rešljiv v času $O(n \log n)$
 - Uredi elemente
 - Iskani element najdemo na indeksu k
- Obstaja rešitev v času $\Theta(n)$

Hitro iskanje (quickselect)

- Izberemo pivot in izvedemo porazdelitev (kot pri hitrem urejanju)
- Po porazdelitvi pivot pristane na indeksu p



- Če je $k = p$, je iskani element kar pivot!
- Če je $k < p$, se iskani element nahaja v levem delu tabele
- Če je $k > p$, se iskani element nahaja v desnem delu tabele

Hitro iskanje

```
function HITRO-ISKANJE( $A, k$ )  
    return HITRO-ISKANJE( $A, k, 0, |A| - 1$ )
```

```
function HITRO-ISKANJE( $A, k, l, d$ )  
     $m \leftarrow$  IZBERI-PIVOT( $A, l, d$ ) ▷ vrne indeks pivota  
    med seboj zamenjaj  $A[m]$  in  $A[d]$   
     $p \leftarrow$  PORAZDELI( $A, l, d$ ) ▷ kot pri hitrem urejanju  
    if  $p = k$  then  
        return  $A[p]$   
    else if  $p < k$  then  
        return HITRO-ISKANJE( $A, k, l, p - 1$ )  
    else  
        return HITRO-ISKANJE( $A, k, p + 1, d$ )
```

Hitro iskanje

```
function PORAZDELI( $A, l, d$ )  
     $pivot \leftarrow A[d]$   
     $i \leftarrow l - 1$   
    for  $j \leftarrow l : d - 1$  do  
        if  $A[j] \leq pivot$  then  
             $i \leftarrow i + 1$   
            med seboj zamenjaj  $A[i]$  in  $A[j]$   
med seboj zamenjaj  $A[i + 1]$  in  $A[d]$   
return  $i + 1$ 
```

Hitro iskanje v »povprečnem« primeru

- Aktivni del tabele vsakokrat zmanjšamo za polovico
- Porazdelitev zahteva $\Theta(n)$ časa
- $T(n) = T(n/2) + \Theta(n)$
- $a = 1, b = 2, d = 1$
- $a < b^d$
- $T(n) = \Theta(n^d) = \Theta(n)$

Hitro iskanje v precej neugodnem primeru

- Aktivni del tabele vsakokrat zmanjšamo za desetino
- $T(n) = T(9n/10) + \Theta(n)$
- $a = 1, b = 10/9, d = 1$
- $a < b^d$
- $T(n) = \Theta(n^d) = \Theta(n)$

Hitro iskanje v najslabšem primeru

- Aktivni del tabele vsakokrat zmanjšamo za en element
- $T(n) = T(n - 1) + \Theta(n)$
- Z drevesno metodo dobimo $T(n) = \Theta(n^2)$
- Preverimo $O(\cdot)$
 - Hipoteza: $T(n) \leq cn^2$ za $n \geq n_0$
 - $T(n) \leq c(n - 1)^2 + \Theta(n) = cn^2 - 2cn + c + \Theta(n)$
 - OK, če je c dovolj velik
- Preverimo $\Omega(\cdot)$
 - Hipoteza: $T(n) \geq cn^2$ za $n \geq n_0$
 - $T(n) \geq c(n - 1)^2 + \Theta(n) = cn^2 - 2cn + c + \Theta(n)$
 - OK, če je c dovolj majhen

Hitro iskanje

- Lahko zagotovimo $O(n)$ tudi v najslabšem primeru?
- V praksi pomaga **randomizacija**
 - naključna izbira pivota
 - še bolje je naključno izbrati npr. tri elemente in za pivot izbrati njihovo mediano
- **Teoretičnega** zagotavlja za $O(n)$ pa še vedno nimamo
 - teoretično bi se nam lahko zgodilo, da bi kljub randomizaciji vedno izbrali ravno tri najmanjše elemente
- Obstaja pa zanimiv (čtetudi v praksi neuporaben) algoritem izbire pivota, ki tudi v praksi zagotavlja $O(n)$

Izbira pivota z mediano median

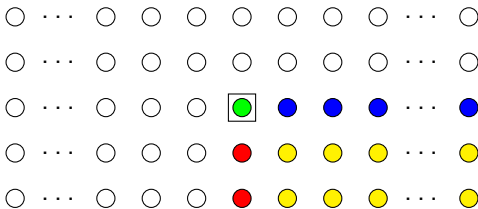
1. Tabelo razdeli na skupine po 5 zaporednih elementov.
2. Za vsako skupino izračunaj mediano.
3. Izračunaj mediano dobljenih median.
4. Dobljena mediana služi kot pivot.
5. Nadaljuj s postopkom hitrega iskanja (izvedi porazdelitev itd.).

Izbira pivota z mediano median

1. Tabelo razdeli na skupine po 5 zaporednih elementov.
 - $\Theta(n)$
2. Za vsako skupino izračunaj mediano.
 - $\Theta(n)$
3. Izračunaj mediano dobljenih median.
 - hitro iskanje na tabeli velikosti $n/5$
 - $T(n/5)$
4. Dobljena mediana služi kot pivot.
5. Nadaljuj s postopkom hitrega iskanja (izvedi porazdelitev itd.).
 - Kako dobra bo porazdelitev s tako izbranim pivotom v najslabšem primeru?

Izbira pivota z mediano median

- Brez vpliva na rezultat lahko predpostavimo, da so
 - elementi v posameznih skupinah urejeni naraščajoče
 - skupine urejene po naraščajočih medianah



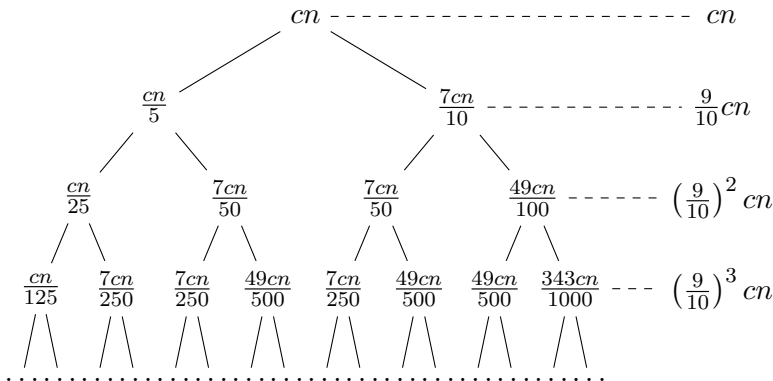
- Pivot je gotovo manjši od modrih, rumenih in rdečih elementov
- Teh elementov je vsaj $\frac{3}{5}$ od $\frac{1}{2}n = 3n/10$
- Ostalih elementov je kvečjemu $7n/10$
- Pivot, izbran kot mediana median, bo tabelo razdelil v razmerju med 3:7 in 7:3.

Izbira pivota z mediano median

1. Tabelo razdeli na skupine po 5 zaporednih elementov.
 - $\Theta(n)$
2. Za vsako skupino izračunaj mediano.
 - $\Theta(n)$
3. Izračunaj mediano dobljenih median.
 - hitro iskanje na tabeli velikosti $n/5$
 - $T(n/5)$
4. Dobljena mediana služi kot pivot.
5. Nadaljuj s postopkom hitrega iskanja (izvedi porazdelitev itd.)
 - $T(7n/10)$ v najslabšem primeru

Izbira pivota z mediano median

- $T(n) = T(n/5) + T(7n/10) + \Theta(n)$
- Poskusimo z drevesno metodo
- Zapišimo $T(n) = cn + T(n/5) + T(7n/10)$



Izbira pivota z mediano median

$$\begin{aligned}T(n) &= cn \left(1 + \frac{9}{10} + \left(\frac{9}{10} \right)^2 + \left(\frac{9}{10} \right)^3 + \dots \right) \\&\leq cn \sum_{i=1}^{\infty} \left(\frac{9}{10} \right)^i \\&= cn \frac{1}{1 - \frac{9}{10}} \\&= 10cn \\&= \Theta(n)\end{aligned}$$

- Trditev $T(n) = O(n)$ preizkusimo s substitucijsko metodo

Izbira pivota z mediano median

- $T(n) = T(n/5) + 7T(n/10) + \Theta(n)$
- Hipoteza: $T(n) \leq cn$

$$\begin{aligned}T(n) &\leq cn/5 + 7cn/10 + \Theta(n) \\&= 9cn/10 + \Theta(n) \\&= cn - cn/10 + \Theta(n)\end{aligned}$$

$T(n) \leq cn$ za vse dovolj velike c

- Trditev $T(n) = O(n)$ je torej resnična
- Če pivot izberemo kot mediano median, bo hitro iskanje tudi v najslabšem možnem primeru teklo v času $O(n)$, a za ceno ogromnega konstantnega faktorja