SELEKCIJA I MEDIJAN

Luka Valenta

Seminar iz kolegija Oblikovanje i analiza algoritama

Prirodoslovno-matematički fakultet

Zagreb, 23. 1. 2018.

Sadržaj

Osnovni pojmovi

- **Donji medijan** niza od n brojeva je broj u nizu koji bi bio na poziciji s indeksom [n/2] da je taj niz sortiran.
- **Gornji medijan** niza od n brojeva je broj u nizu koji bi bio na poziciji s indeksom [n/2] da je taj niz sortiran.
- Neformalno, **medijan** je element u nizu koji je na sredini niza kad je taj niz sortiran.
- U statistici se medijan definira kao aritmetička sredina donjeg i gornjeg medijana, ali mi ćemo u nastavku za medijan koristiti donji medijan.
- **Rang** elementa niza je jedan plus broj elemenata niza koji su manji od njega.

Opis problema

- **Problem selekcije:**

  Neka je A niz od n brojeva i k prirodan broj takav da $1 \leq k \leq n$. Koji je k-ti po veličini član niza A?

- Traženje minimuma, maksimuma i medijana niza su posebni slučajevi problema selekcije, za redom k = 1, k = n, $k = \lfloor n/2 \rfloor$.

- Problem se može lako riješiti u složenosti $\Theta(n \log n)$ sortiranjem niza merge sortom ili heap sortom koji su složenosti $\Theta(n \log n)$ u najgorem slučaju.

- *Kako riješiti problem u boljoj složenosti?*

Rješenje metodom sita

- Metodom sita problem rješavamo rekurzivno u nekoliko koraka.

Rješenje metodom sita

- Metodom sita problem rješavamo rekurzivno u nekoliko koraka.
- 1. Biramo pivotni element x. Postoje dva različita pristupa za biranje pivotnog elementa, ali o tome detaljnije kasnije.

Rješenje metodom sita

- Metodom sita problem rješavamo rekurzivno u nekoliko koraka.
- 1. Biramo pivotni element x. Postoje dva različita pristupa za biranje pivotnog elementa, ali o tome detaljnije kasnije.
- 2. Dijelimo niz za 3 dijela. U prvi dio stavimo sve one koji su manji od x, drugi je sam x, a u treći one koji su veći od x.

Rješenje metodom sita

- Metodom sita problem rješavamo rekurzivno u nekoliko koraka.
- 1. Biramo pivotni element x. Postoje dva različita pristupa za biranje pivotnog elementa, ali o tome detaljnije kasnije.
- 2. Dijelimo niz za 3 dijela. U prvi dio stavimo sve one koji su manji od x, drugi je sam x, a u treći one koji su veći od x.
- 3. Uspoređujemo k i rang pivotnog elementa, xRank = q-p+1

Rješenje metodom sita

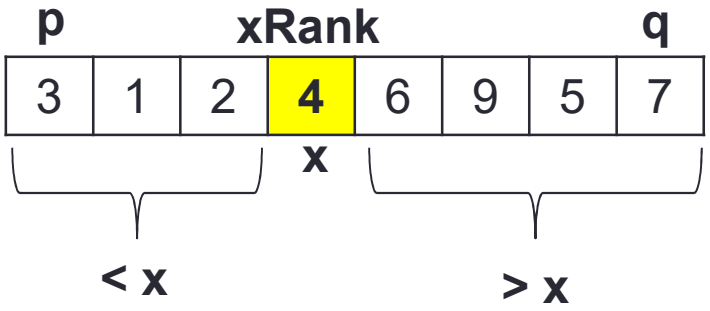
- Metodom sita problem rješavamo rekurzivno u nekoliko koraka.
- 1. Biramo pivotni element x. Postoje dva različita pristupa za biranje pivotnog elementa, ali o tome detaljnije kasnije.
- 2. Dijelimo niz za 3 dijela. U prvi dio stavimo sve one koji su manji od x, drugi je sam x, a u treći one koji su veći od x.
- 3. Uspoređujemo k i rang pivotnog elementa,
  xRank = q-p+1
  - Ako k == xRank gotovi smo,
    ako k < xRank rekurzivno pozivamo funkciju na prvom/lijevom dijelu,
    ako k > xRank rekurzivno pozivamo funkciju na trećem/desnom dijelu.

Pseudokod selekcije metodom sita

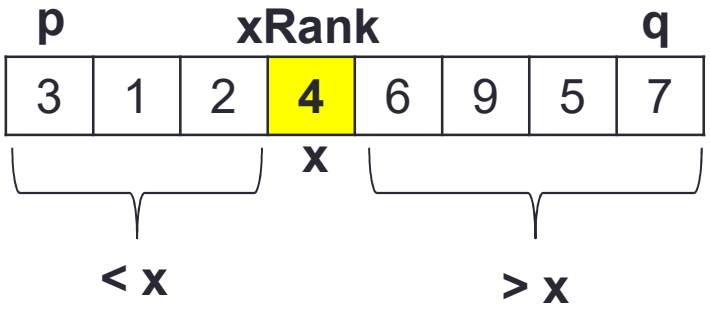
```
Select(array A, int p, int r, int k)
{
        if(p == r) return A[p]
        else
        {
                x = ChoosePivot(A,p,r)
                q = Partition(A,p,r,x)
                xRank = q-p+1
                if(k == xRank) return x
                else if(k < xRank) return Select(A,p,q-1,k)
                else return Select(A,q+1,r,k-xRank)
        }
}
```

Particija

Particija



Particija



Particija je složenosti $\Theta(n)$. To nam daje donju ogradu za bilo koji algoritam za selekciju metodom sita.

Biranje pivotnog elementa

- Pretpostavimo da možemo dizajnirati ChoosePivot td. u svakom koraku eliminiramo pola niza iz razmatranja. Tada:

$$T(n) = \begin{cases} 1 & \text{ako } n = 1, \\ T(n/2) + n & \text{inače.} \end{cases}$$

$$T(n) = n + n/2 + n/4 + \ldots \leq \sum_{i=0}^{\infty} \frac{n}{2^i} = n\sum_{i=0}^{\infty} \frac{1}{2^i} = \frac{1}{1-1/2}\, n = 2n$$

Uočavamo da za O(n) ni ne moramo prepoloviti dio koji razmatramo, samo ga moramo umanjiti za neki postotak strogo veći od 0 jer geometrijski niz konvergira za $|c| < 1$.

Biranje pivotnog elementa (2)

- Prvi način je da za pivotni element uzmemo **nasumičan element** promatranog intervala.
- Algoritam selekcije dobiven takvim biranjem pivotnog elementa ima složenost *$\Theta(n)$ u prosječnom slučaju* i on se koristi u praksi. Međutim, složenost *u najgorem slučaju je $\Theta(n^2)$*.

- Postoji i algoritam za biranje pivotnog elementa kojim možemo postići složenost *$\Theta(n)$ u najgorem slučaju*.
- Zbog svoje kompliciranosti on ima prvenstveno teorijsko značenje.

Randomized selection

- Ilustracija algoritma selekcije uz nasumično biranje pivotnog elementa:

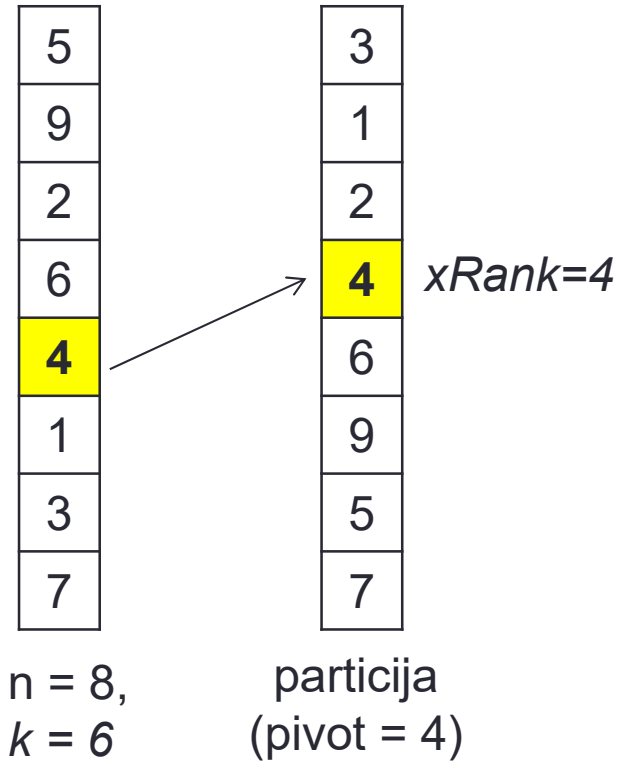
Randomized selection

- Ilustracija algoritma selekcije uz nasumično biranje pivotnog elementa:

n = 8,
k = 6

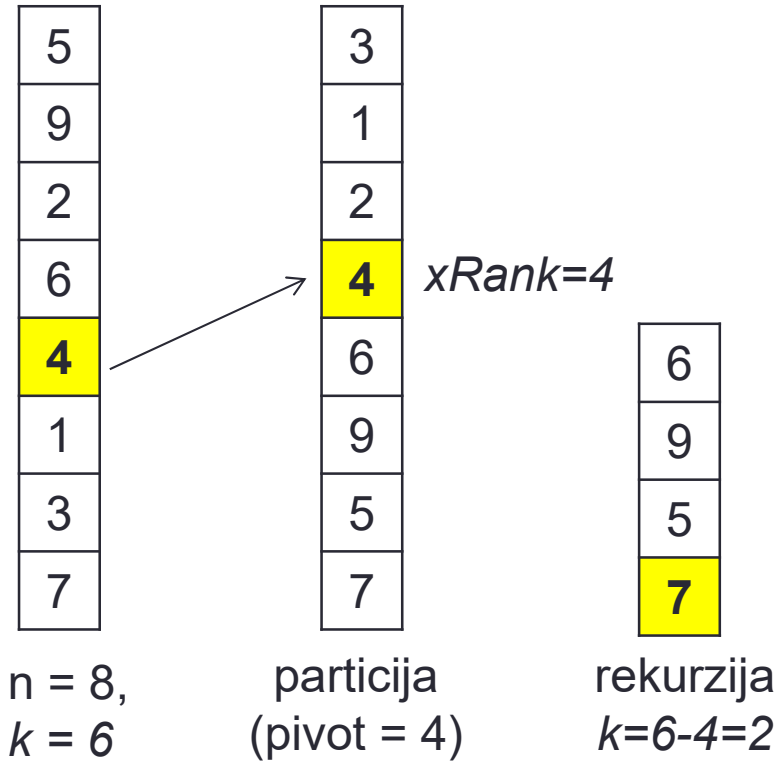
Randomized selection

- Ilustracija algoritma selekcije uz nasumično biranje pivotnog elementa:



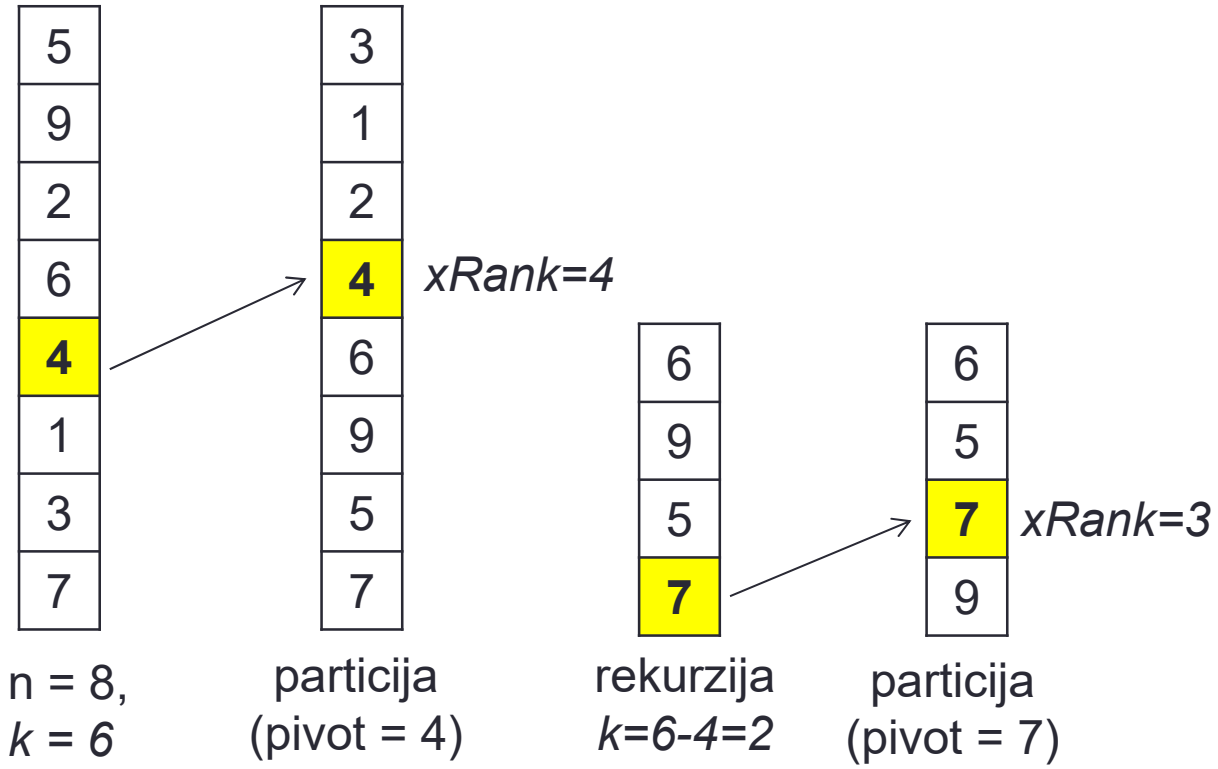
Randomized selection

- Ilustracija algoritma selekcije uz nasumično biranje pivotnog elementa:



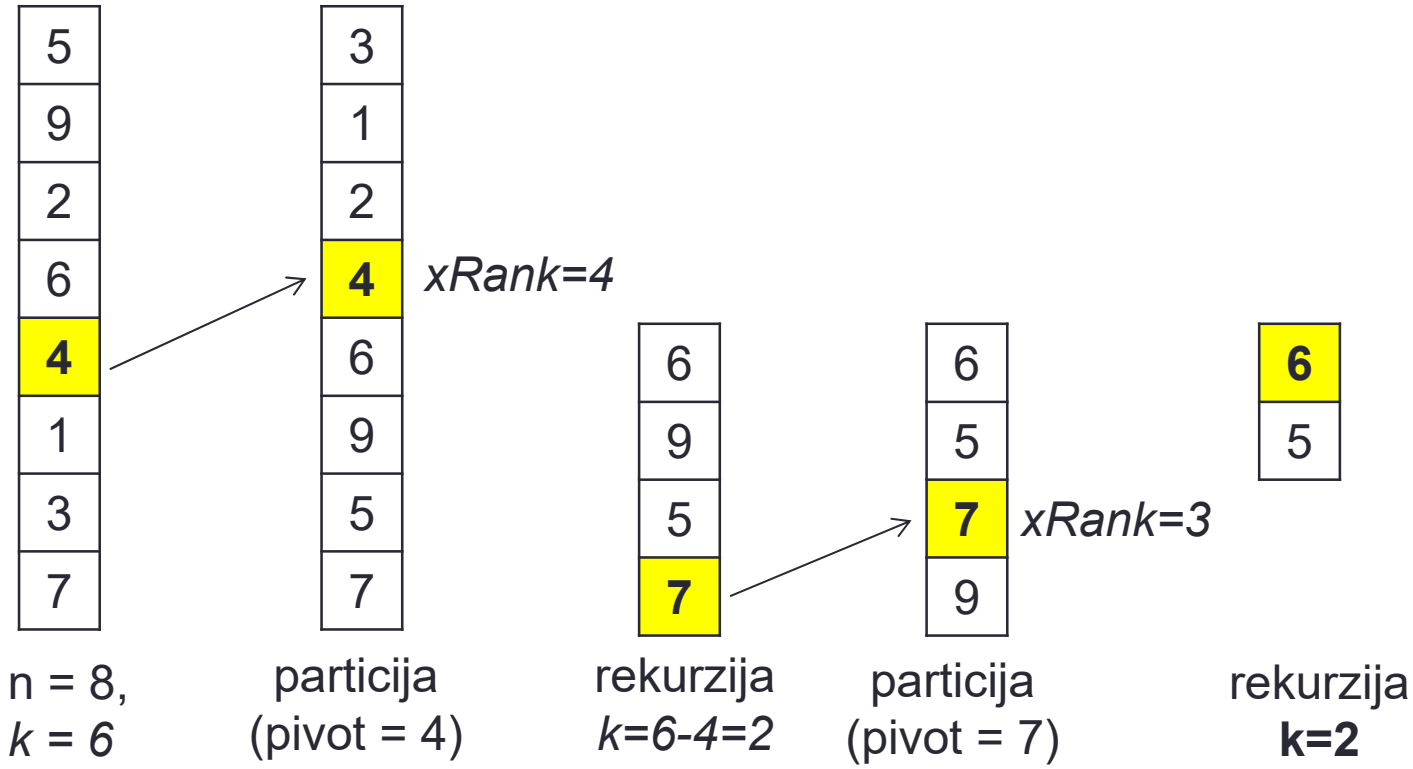
Randomized selection

- Ilustracija algoritma selekcije uz nasumično biranje pivotnog elementa:



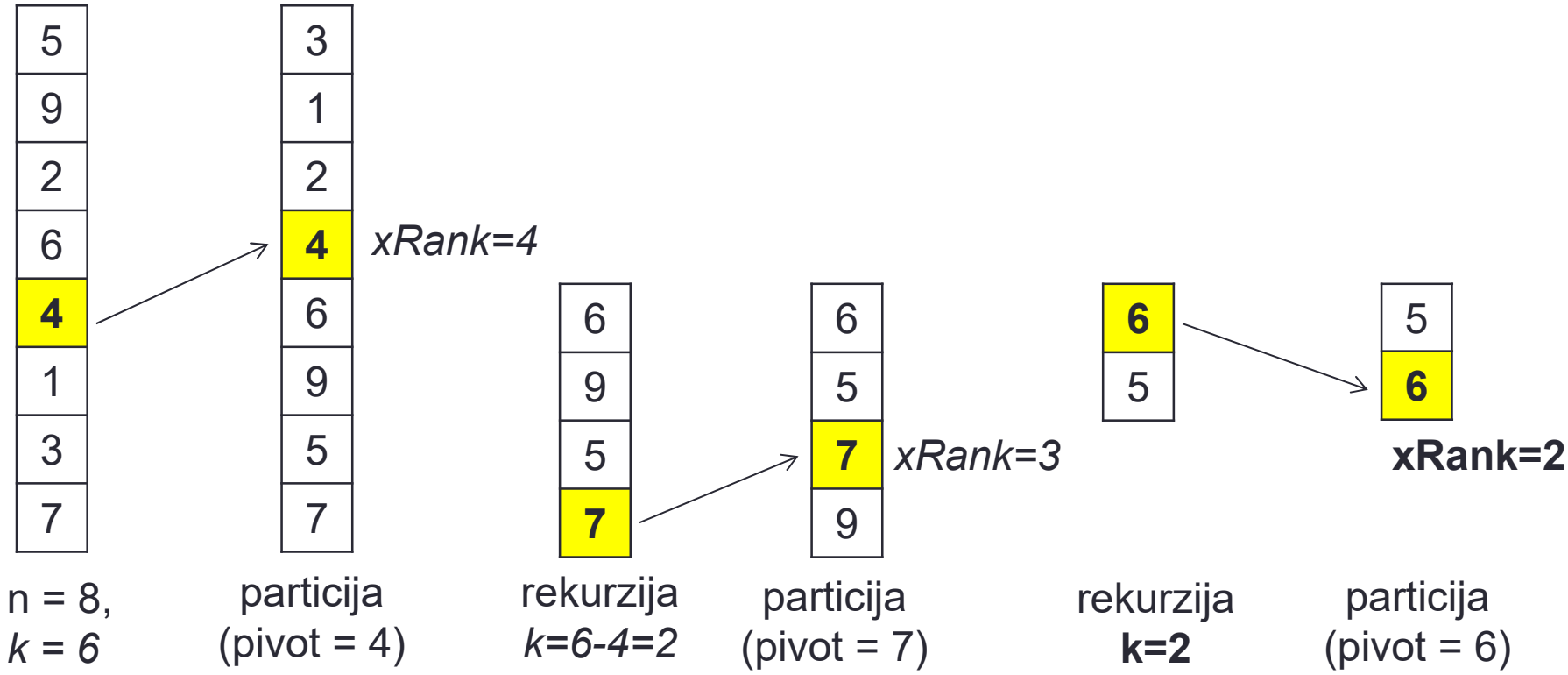
Randomized selection

- Ilustracija algoritma selekcije uz nasumično biranje pivotnog elementa:



Randomized selection

- Ilustracija algoritma selekcije uz nasumično biranje pivotnog elementa:



Randomized selection (2)

- **Složenost randomized selectiona** u najgorem slučaju je $\Theta(n^2)$ jer se može dogoditi da nasumičnim izborom svaki puta za pivotni element izaberemo element najmanjeg ili najvećeg ranga u promatranom intervalu. U tom slučaju će se promatrani interval u sljedećem koraku smanjiti samo za 1. U svakom koraku radimo particiju koja je složenosti $\Theta(n)$ pa tako dolazimo do složenosti $\Theta(n^2)$.

- Ipak, vjerojatnost da se to dogodi su male i u većini slučajeva ćemo smanjiti promatrani interval za neki postotak pa se u prosječnom slučaju dobije složenost $\Theta(n)$.

Algoritam složenosti Θ(n)

- Slijedi opis algoritma za **ChoosePivot** koji omogućuje složenost Θ(n) algoritma za selekciju u najgorem slučaju.

Algoritam složenosti Θ(n)

- Slijedi opis algoritma za **ChoosePivot** koji omogućuje složenost Θ(n) algoritma za selekciju u najgorem slučaju.
- *1. Dijelimo niz A u grupe od 5 elemenata.* Bit će točno [n/5] takvih grupa.

Algoritam složenosti Θ(n)

- Slijedi opis algoritma za **ChoosePivot** koji omogućuje složenost Θ(n) algoritma za selekciju u najgorem slučaju.
- *1. Dijelimo niz A u grupe od 5 elemenata.* Bit će točno [n/5] takvih grupa.
- *2. Nađemo medijan svake od* ⌈*n/5*⌉ *grupa.* To možemo napraviti td. sortiramo petorku i uzmemo treći element. To možemo napraviti u Θ(1) za svaku grupu što nas dovodi do ukupne složenosti Θ(n). Kopiramo medijane grupa u niz B.

Algoritam složenosti Θ(n)

- Slijedi opis algoritma za **ChoosePivot** koji omogućuje složenost Θ(n) algoritma za selekciju u najgorem slučaju.
- *1. Dijelimo niz A u grupe od 5 elemenata.* Bit će točno [n/5] takvih grupa.
- *2. Nađemo medijan svake od [n/5] grupa.* To možemo napraviti td. sortiramo petorku i uzmemo treći element. To možemo napraviti u Θ(1) za svaku grupu što nas dovodi do ukupne složenosti Θ(n). Kopiramo medijane grupa u niz B.
- *3. Tražimo medijan medijana iz 2. koraka tako da rekurzivno pozovemo Selection(B,1,m,k)*, gdje je m = [n/5] i k = ⌊(m+1)/2⌋. Neka je x dobiveni medijan medijana. Vrati x kao željeni pivot.

Algoritam složenosti Θ(n) (2)

- Ilustracija algoritma:

Algoritam složenosti Θ(n) (2)

- Ilustracija algoritma:

1. Podjela na grupe:

14	32	23	5	10	60	29
57	2	52	44	27	21	11
24	43	12	17	48	1	58
6	30	63	34	8	55	39
37	25	3	64	19	41	

Algoritam složenosti Θ(n) (2)

- Ilustracija algoritma:

1. Podjela na grupe:

14	32	23	5	10	60	29
57	2	52	44	27	21	11
24	43	12	17	48	1	58
6	30	63	34	8	55	39
37	25	3	64	19	41	

2. Medijani grupa:

6	2	3	5	8	1	11
14	25	12	17	10	21	29
24	30	23	34	19	41	39
37	32	52	44	27	55	58
57	43	63	64	48	60	

Algoritam složenosti Θ(n) (2)

- Ilustracija algoritma:

1. Podjela na grupe:

14	32	23	5	10	60	29
57	2	52	44	27	21	11
24	43	12	17	48	1	58
6	30	63	34	8	55	39
37	25	3	64	19	41	

2. Medijani grupa:

6	2	3	5	8	1	11
14	25	12	17	10	21	29
24	30	23	34	19	41	39
37	32	52	44	27	55	58
57	43	63	64	48	60	

3. Medijan medijana grupa:

8	3	6	2	5	11	1
10	12	14	25	17	29	21
19	23	24	30	34	39	41
27	52	37	32	44	58	55
48	63	57	43	64		60

Algoritam složenosti Θ(n) (2)

- **Lema:** Element x je ranga barem n/4 i najviše 3n/4 u A.

- Iz gornje leme slijedi da rekurzivni poziv od Selection neće biti poznan na nizu većem od 3n/4. Za ChoosePivot smo morali pozvati Selection na nizu od [n/5] elemenata, sve ostalo je bilo u linearnom vremenu.
- Slijedi:

$$T(n) \leq \begin{cases} 1 & \text{ako } n = 1, \\ T(n/5) + T(3n/4) + n & \text{inače.} \end{cases}$$

Jakom indukcijom se pokaže T(n) = O(n).

Rezultati testiranja

- Promatramo T(n)/n randomized selecta. Budući da je on u prosječnom slučaju linearne složenosti očekujemo da se T(n)/n neće mijenjati kako n raste.

- Uz to T(n)/n randomized selecta uspoređujemo s T(n)/n std::sort algoritma kako bismo i eksperimentalno pokazali da je randomized sort brži od algoritama za sortiranje.

Rezultati testiranja (2)

n	T(n)/n (random. select)	T(n)/n (std::sort)
1.000	3.52118e-008	1.02522e-007
2.000	3.14056e-008	1.13078e-007
4.000	3.17426e-008	1.18411e-007
8.000	3.20666e-008	1.23500e-007
16.000	3.36746e-008	1.38669e-007
32.000	3.63191e-008	1.53725e-007
64.000	2.86998e-008	2.50286e-007
128.000	2.56749e-008	6.21449e-007
256.000	3.13216e-008	5.58036e-007
512.000	2.34509e-008	5.25841e-007
1.240.000	1.95313e-008	2.84831e-007
2.048.000	2.07150e-008	4.88281e-007
4.096.000	1.92021e-008	5.69661e-007
8.192.000	1.89887e-008	8.54492e-007

Rezultati testiranja (3)

n	T(n)/n (random. select)	T(n)/n (std::sort)
20	3.89213e-008	3.59260e-008
40	3.52583e-008	4.13021e-008
60	3.45301e-008	3.94093e-008
80	3.38275e-008	4.93358e-008
100	3.62572e-008	6.03838e-008
200	3.62572e-008	5.36149e-008
300	3.08522e-008	6.10379e-008
400	3.05955e-008	7.44272e-008
500	3.05974e-008	7.63550e-008
600	3.04486e-008	7.92502e-008
700	3.04009e-008	8.61430e-008
800	3.16188e-008	8.69842e-008
900	3.13530e-008	8.74409e-008
1.000	3.10245e-008	1.01048e-007

Rezultati testiranja (4)

- **Zaključak**:

Kod velikih nizova treba koristiti randomized select za traženje elementa ranga k, ali kod relativno malih nizova to nije toliko nužno jer je razlika u brzini optimiziranih algoritama za sortiranje i randomized selecta mala ili zanemariva.

Literatura

- R. Johnsonbaugh and M. Schaefer, *Algorithms*, Pearson Education International, Upper Saddle River, New Jersey, 2004.
- David M. Mount, *Design and Analysis of Computer Algorithms*, University of Maryland, 2015.
- Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, *Introduction to Algorithms*, Massachusetts: The MIT Press, 1990.
- Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, *Introduction to Algorithms, Second Edition*, Massachusetts: The MIT Press, 2000.
- https://github.com/lyonssp/Randomized-Select (31.12.2017.)

Pitanja?