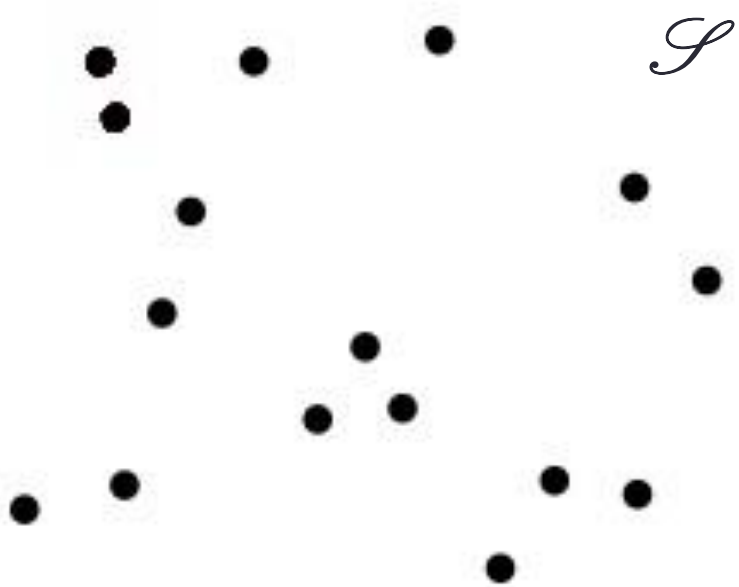
NAJBLIŽI PAR TOČAKA

Sebastijan Horvat

Seminar iz kolegija Oblikovanje i analiza algoritama
Prirodoslovno-matematički fakultet
Zagreb, 24. siječnja 2017. godine

Sadržaj

$\mathscr{S}$ = skup n točaka u ravnini, n ⩾ 2

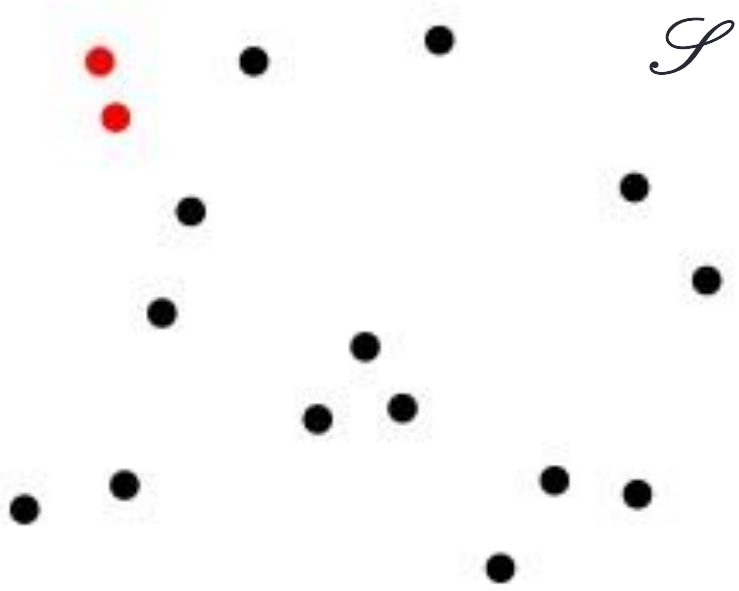


Primjer 1.
$|\mathscr{S}| = n = 16$

Problem: Naći par točaka

$$p_1 = (x_1,y_1) \in \mathcal{S},\ p_2 = (x_2,y_2) \in \mathcal{S}$$

tako da udaljenost $d(p_1,p_2)$ bude minimalna.

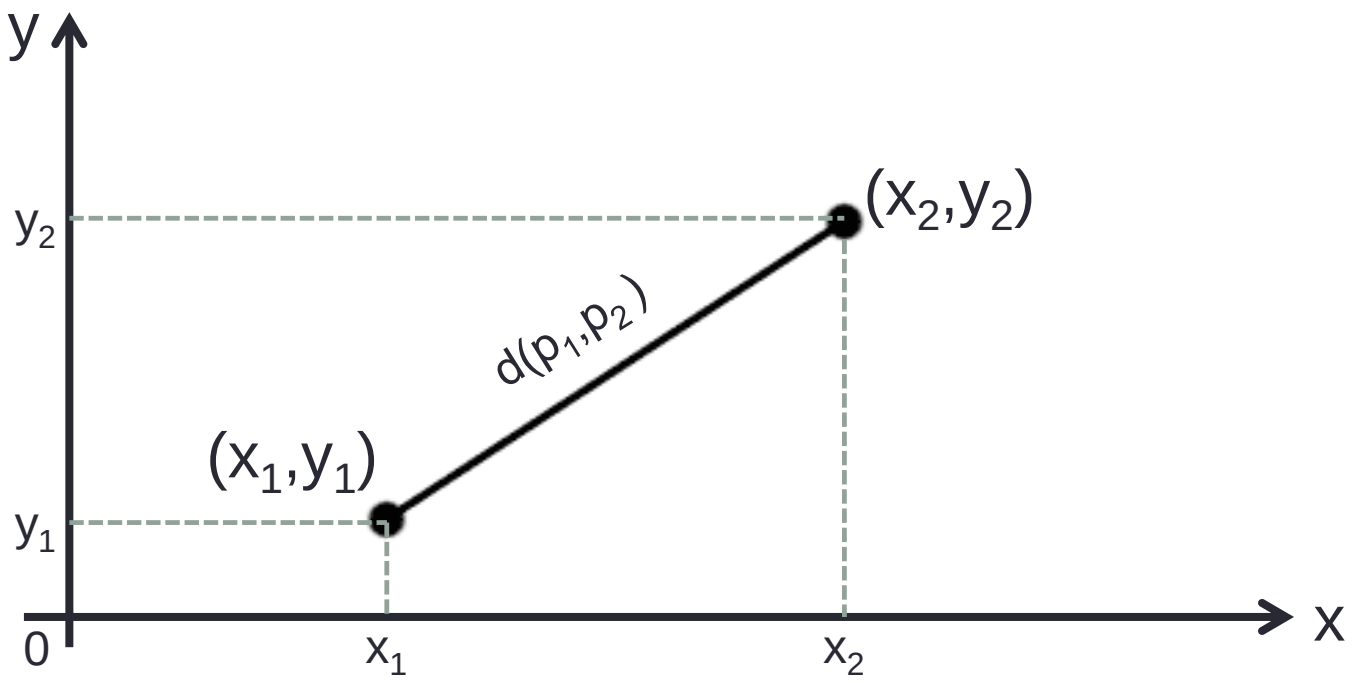


Primjer 1.
$|\mathcal{S}| = n = 16$

Napomena Točke se mogu podudarati (tada min. udalj. 0)

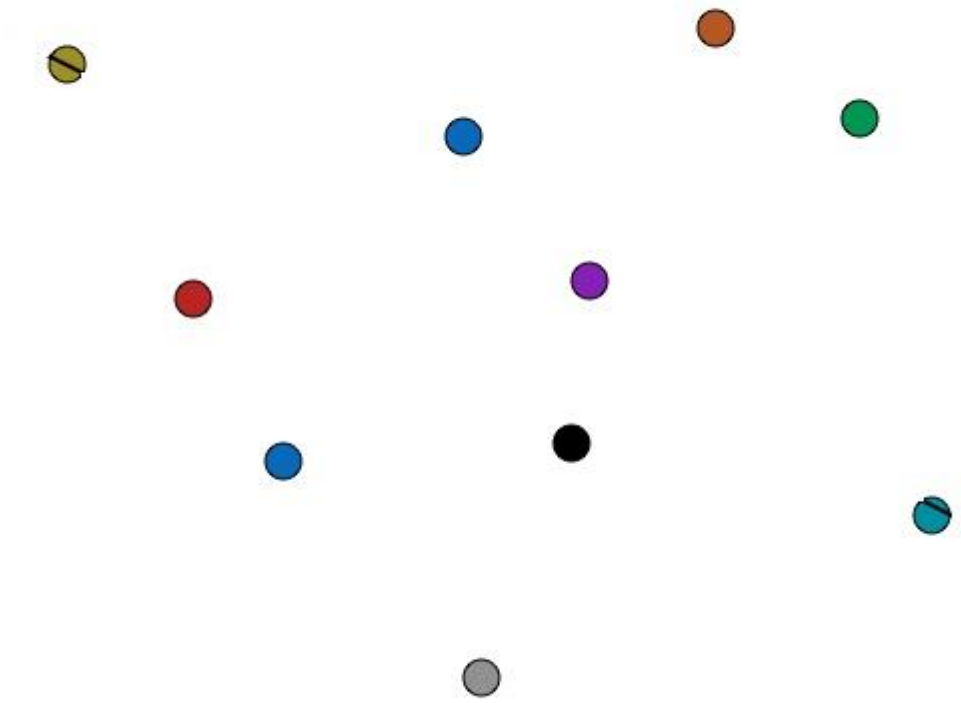
- $p_1 = (x_1,y_1)$, $p_2 = (x_2,y_2)$
- $d(p_1,p_2)$ - **Euklidska udaljenost** točaka p_1 i p_2

$$d(p_1,p_2) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

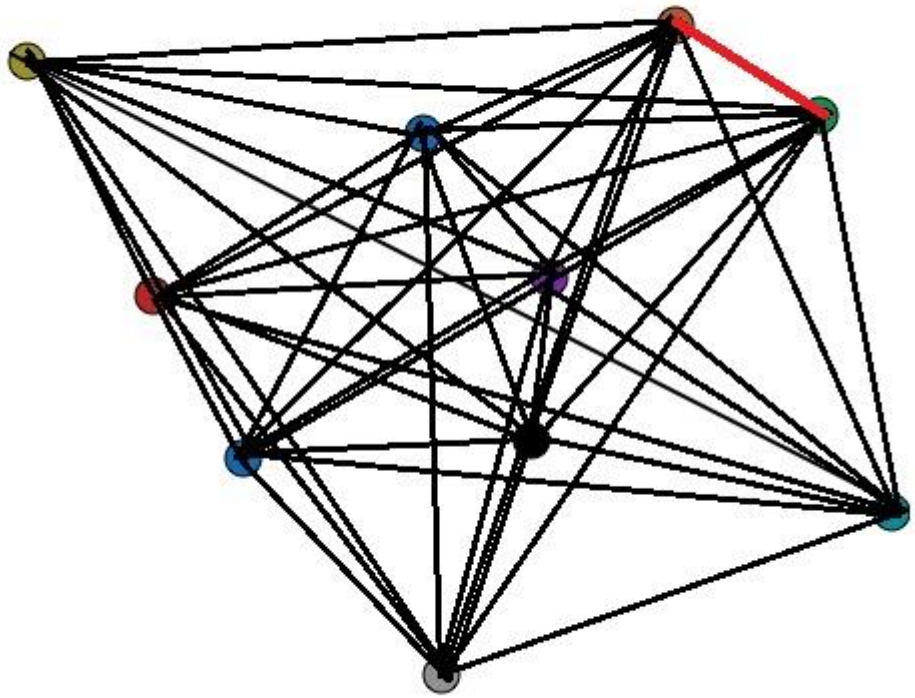


Iscrpan (brute-force) algoritam

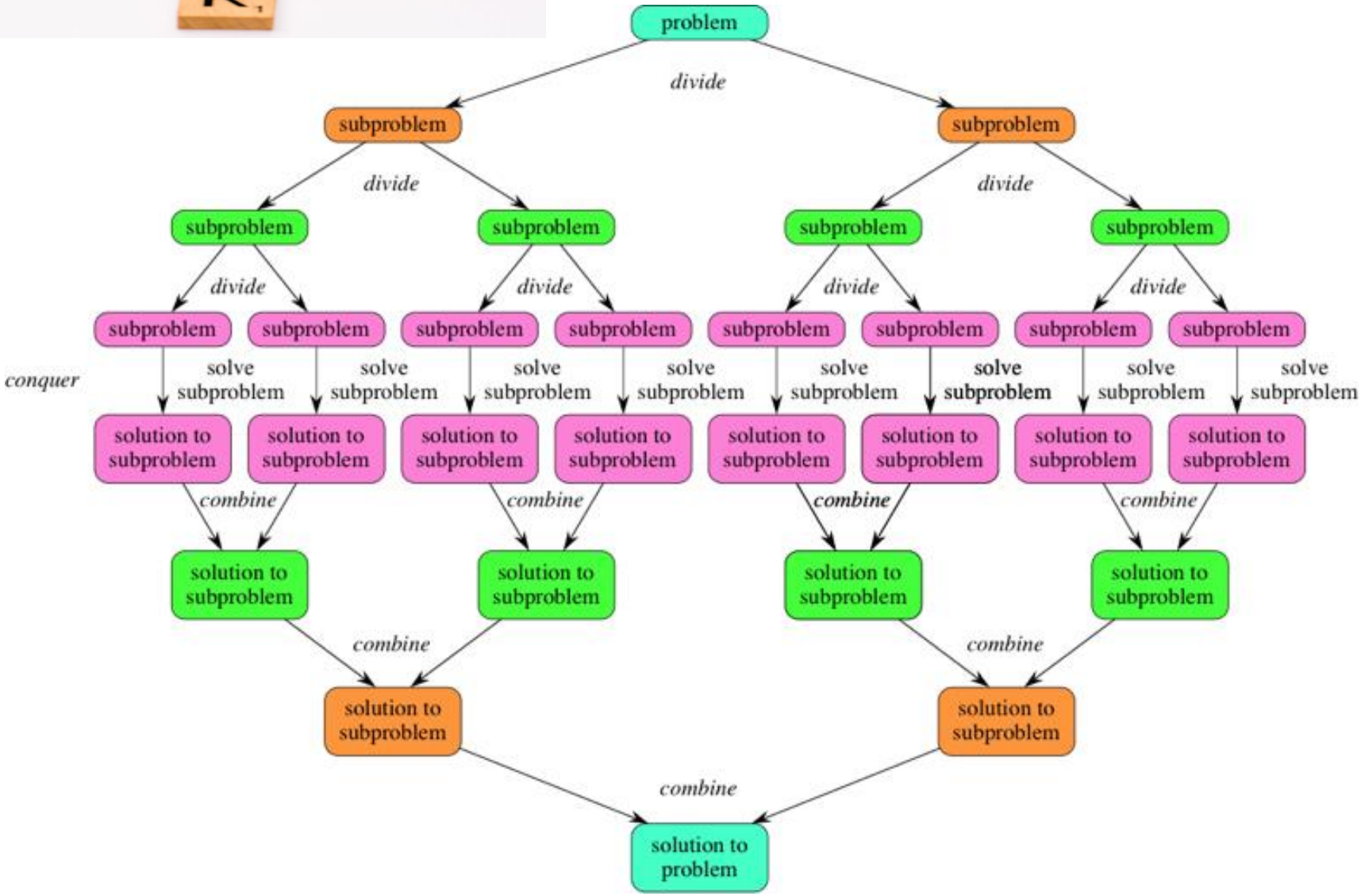
- provjeri svih $\binom{n}{2} = \frac{n(n-1)}{2}$ udaljenosti i vrati par točaka s najmanjom udaljenošću



- ovo daje $\Theta(n^2)$
- može bolje: $\Theta(n \log n)$
    - pristupom „podijeli pa vladaj”



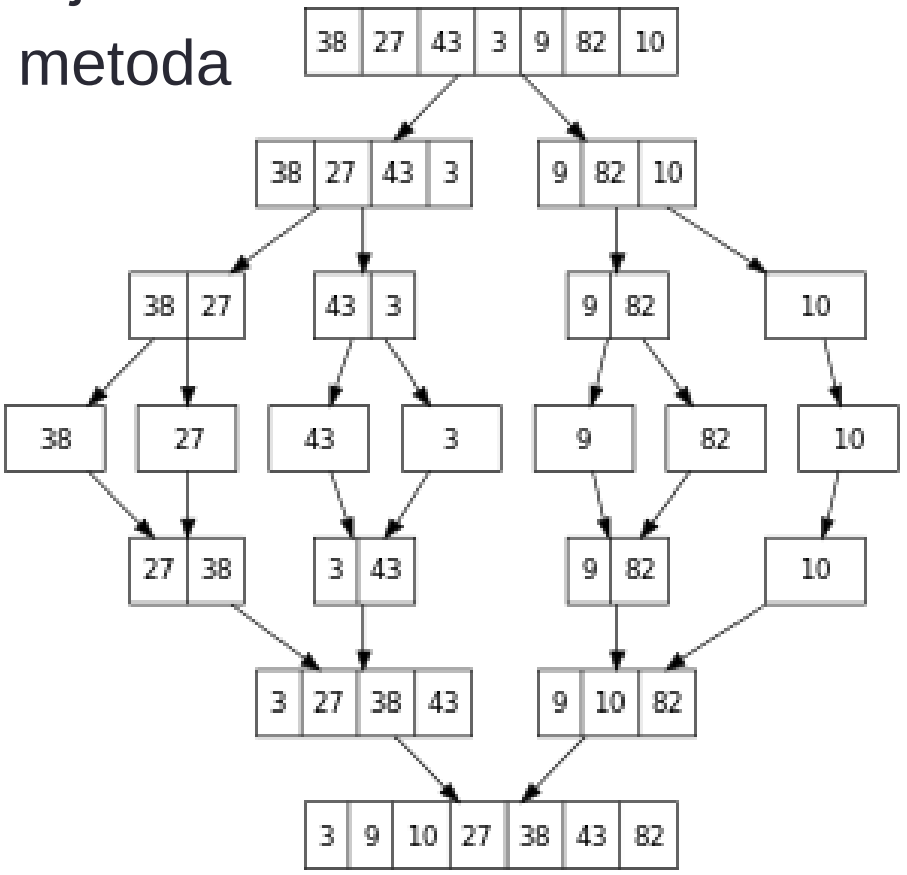

DIVIDE
CONQUER


problem
divide
subproblem
subproblem
divide
divide
subproblem
subproblem
subproblem
subproblem
divide
divide
divide
divide
subproblem
subproblem
subproblem
subproblem
subproblem
subproblem
subproblem
subproblem
conquer
solve subproblem
solve subproblem
solve subproblem
solve subproblem
solve subproblem
solve subproblem
solve subproblem
solve subproblem
solve subproblem
solve subproblem
solution to subproblem
solution to subproblem
solution to subproblem
solution to subproblem
solution to subproblem
solution to subproblem
solution to subproblem
solution to subproblem
combine
combine
combine
combine
solution to subproblem
solution to subproblem
solution to subproblem
solution to subproblem
combine
combine
solution to subproblem
solution to subproblem
combine
solution to problem

Divide and Conquer

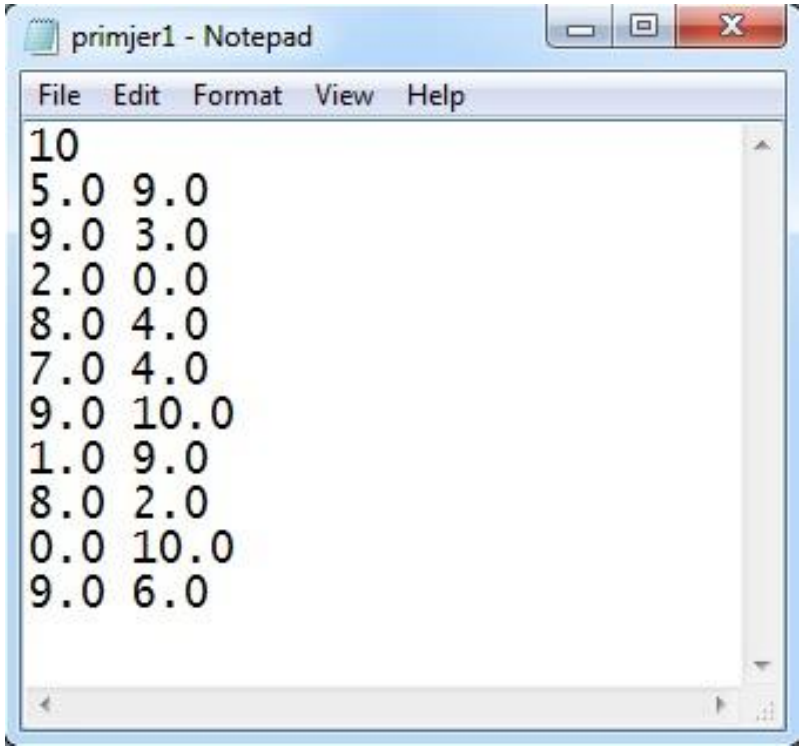
- klasa algoritamskih tehnika u kojoj ulaz razbijemo na nekoliko dijelova, rekurzivno riješimo svaki, te kombiniramo rješenja tih podproblema u sveukupno rješenje
- može biti jednostavna i snažna metoda

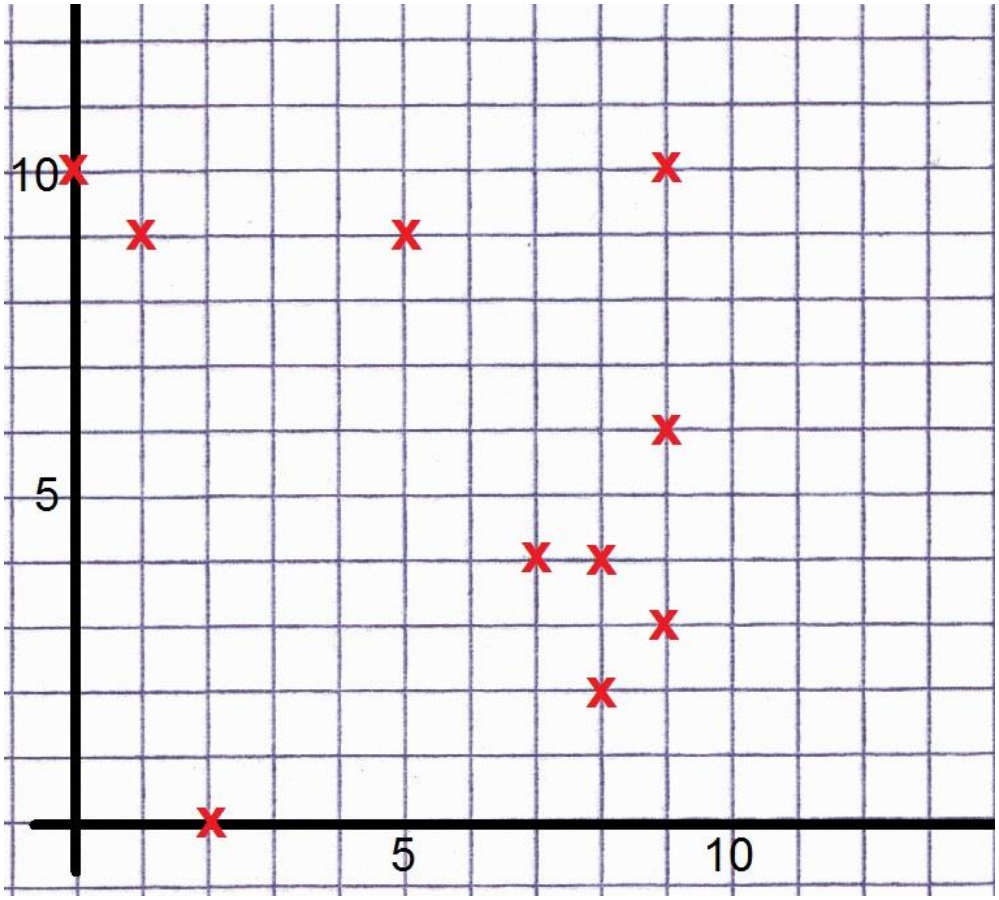
"Really? — my people always say *multiply* and conquer."



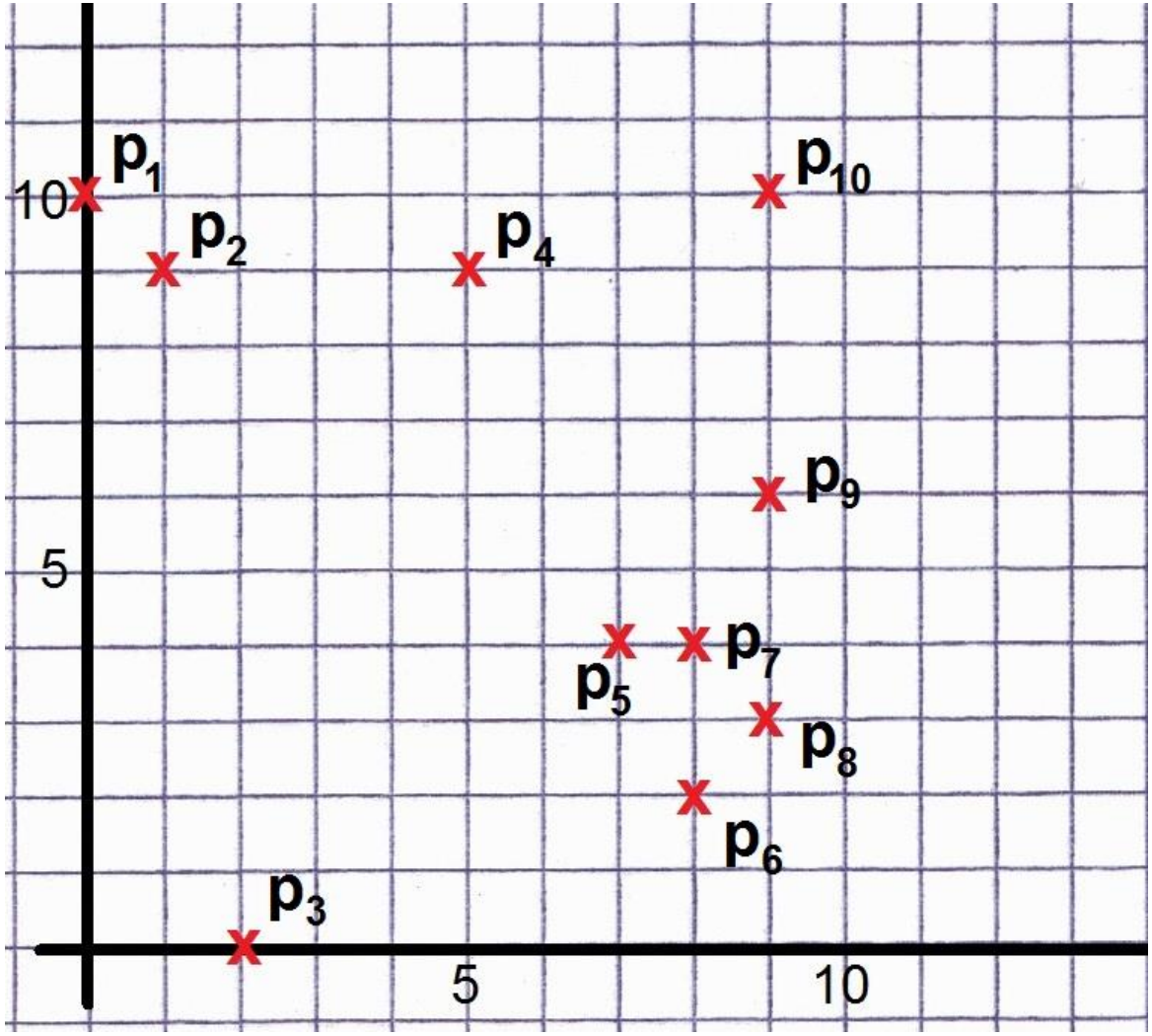
Algoritam

- Ulaz:





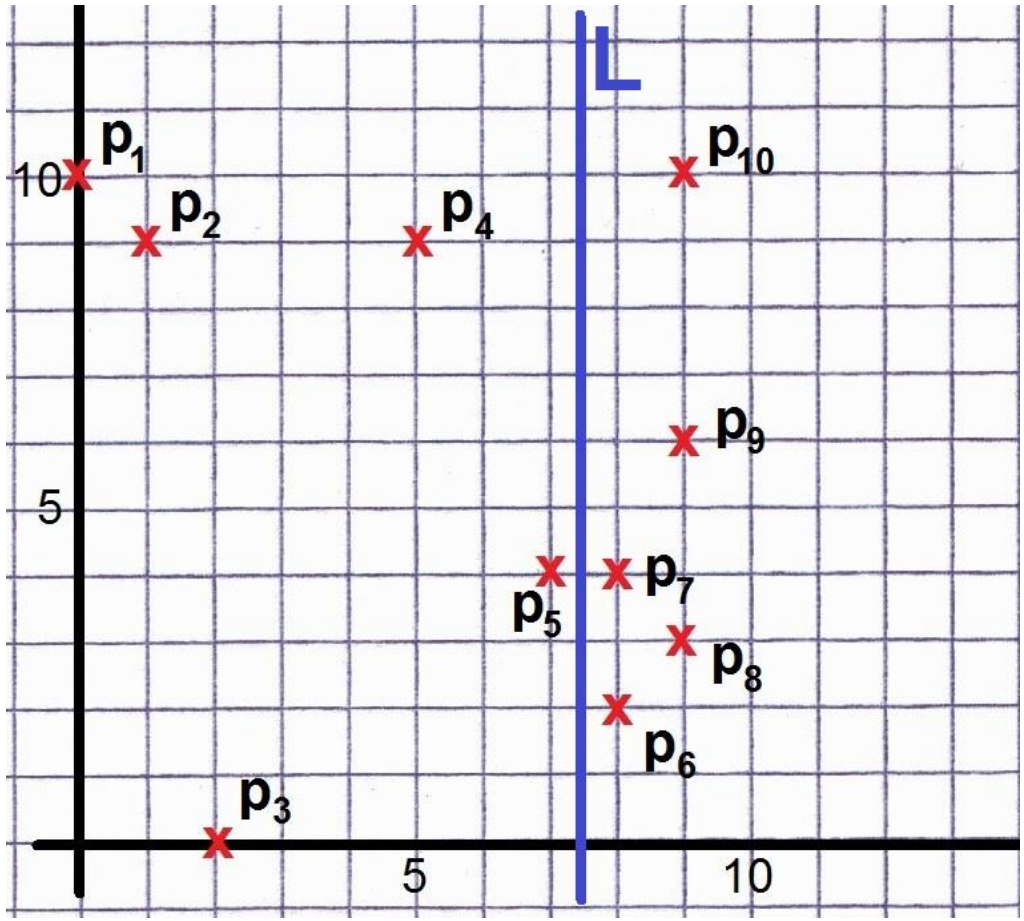
1. Sortiraj točke iz $\mathscr{S}$ rastuće po x-koordinati

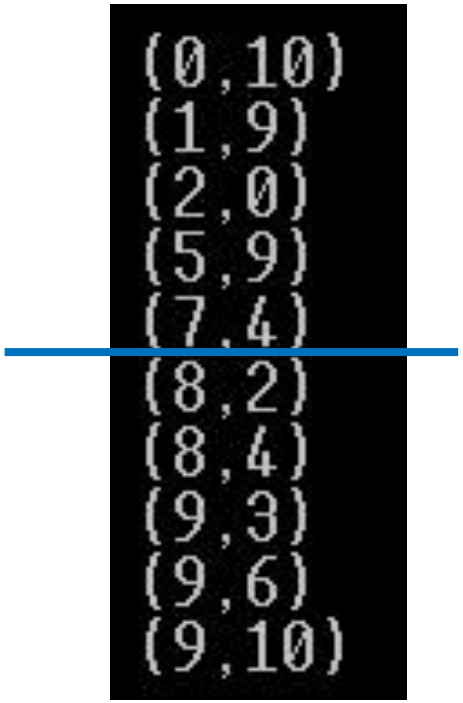



```
(0,10)
(1,9)
(2,0)
(5,9)
(7,4)
(8,2)
(8,4)
(9,3)
(9,6)
(9,10)
```

2. Podijeli $\mathcal{S}$ pravcem L (paralelnim y-osi) u $\mathcal{S}_l$ i $\mathcal{S}_r$ tako da

$$|\mathcal{S}_l| = \left\lfloor \frac{|\mathcal{S}|}{2} \right\rfloor \qquad |\mathcal{S}_r| = \left\lceil \frac{|\mathcal{S}|}{2} \right\rceil$$

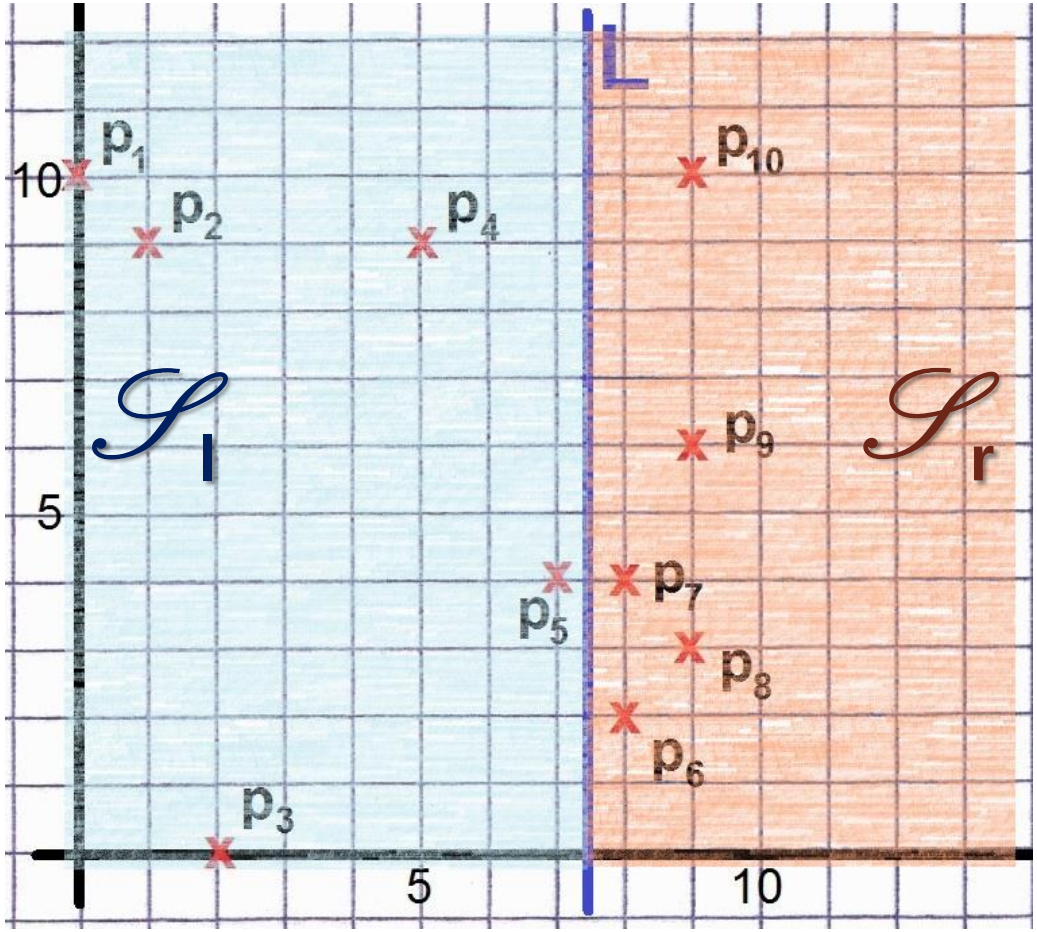




Neka L prolazi x-koordinatom (*pored)* točke $S[\lfloor \frac{n}{2} \rfloor]$

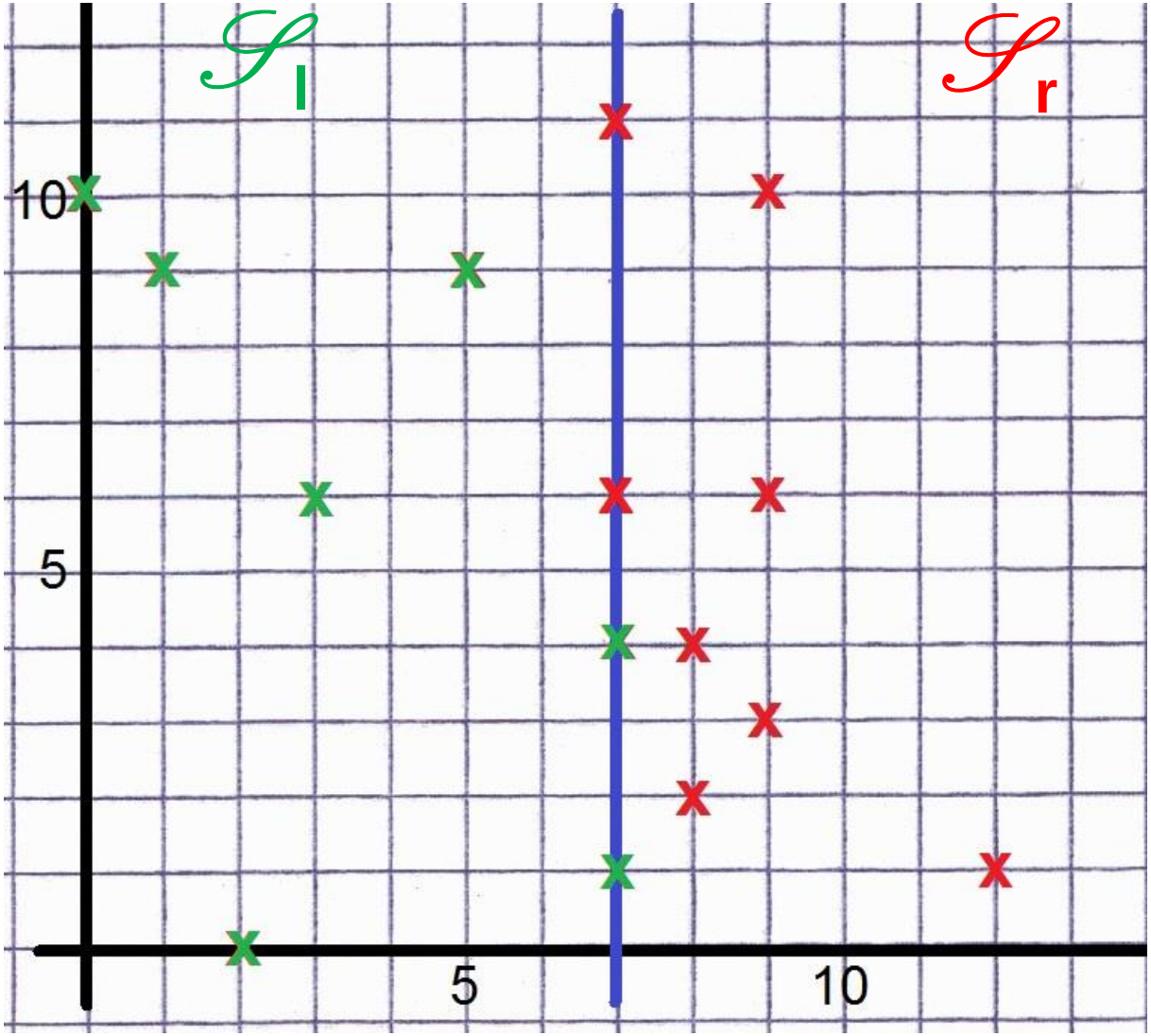
→ sve točke iz $\mathscr{S}_l$ su s lijeve strane od L (ili na L)

→ sve točke iz $\mathscr{S}_r$ su s desne strane od L (ili na L)



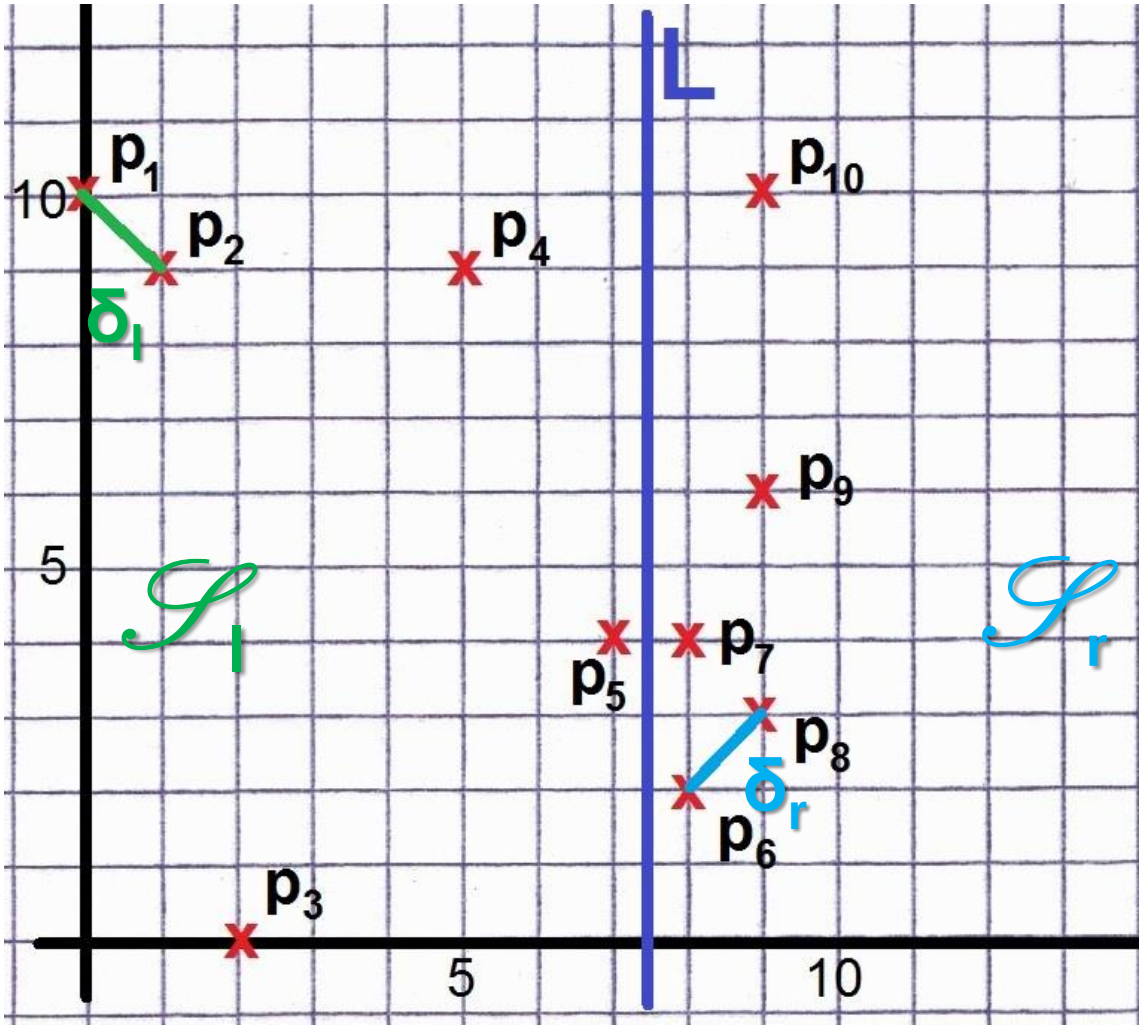
Napomena: I točke iz $\mathscr{S}_l$ i točke iz $\mathscr{S}_r$ mogu biti na L

Primjer:



3. rekurzija - izračunaj minimalnu udaljenost

- ► δ_l za $\mathscr{S}_l$
- ► δ_r za $\mathscr{S}_r$



Na ovom primjeru:

$\delta_l = \sqrt{2} \approx$ 1.41421

(za (0,10) i (1,9))

$\delta_r = \sqrt{2} \approx$ 1.41421

(za (8,2) i (9,3))

4. combine step

Pitanje: Što ako je najmanja udaljenost između jedne točke iz $\mathscr{S}_l$ i jedne točke iz $\mathscr{S}_r$?

→ Izračunaj najmanju udaljenost δ’ između točke iz $\mathscr{S}_l$ i točke iz $\mathscr{S}_r$

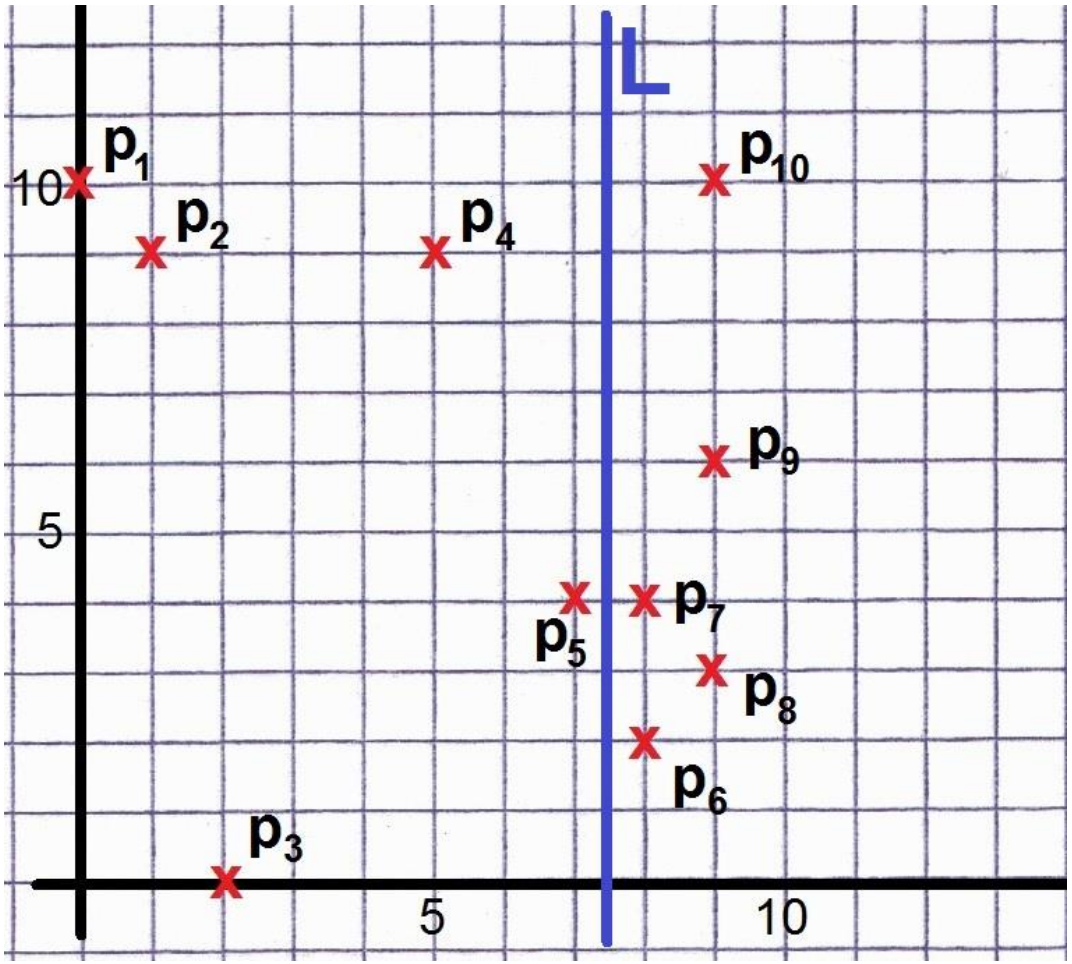
→ Rezultat = min $\{\delta_l, \delta_r, \delta'\}$

→ Izračunaj najmanju udaljenost δ’ između točke iz $\mathscr{S}_l$ i točke iz $\mathscr{S}_r$

- najviše posla

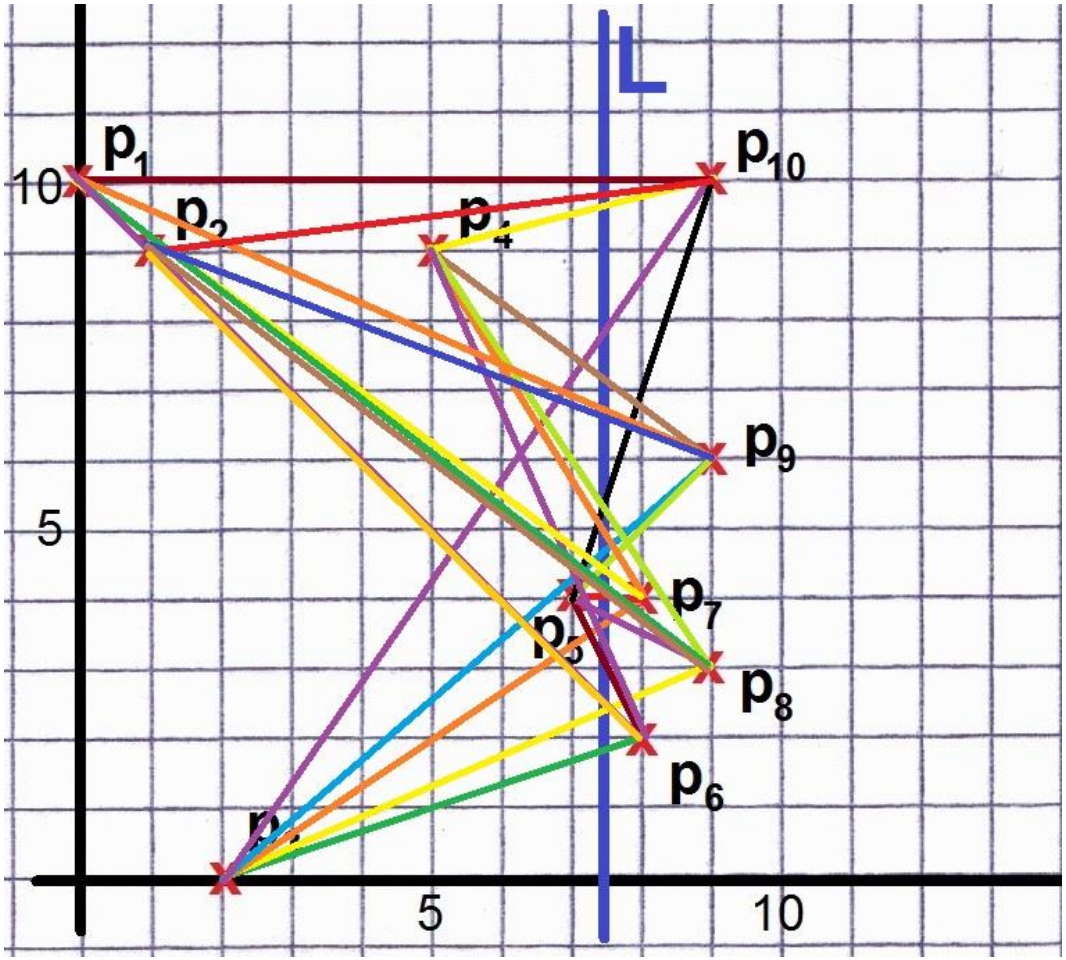
Naivna metoda

- Izračunaj udaljenost između svake točke iz $\mathscr{S}_l$ i svake točke iz $\mathscr{S}_r$

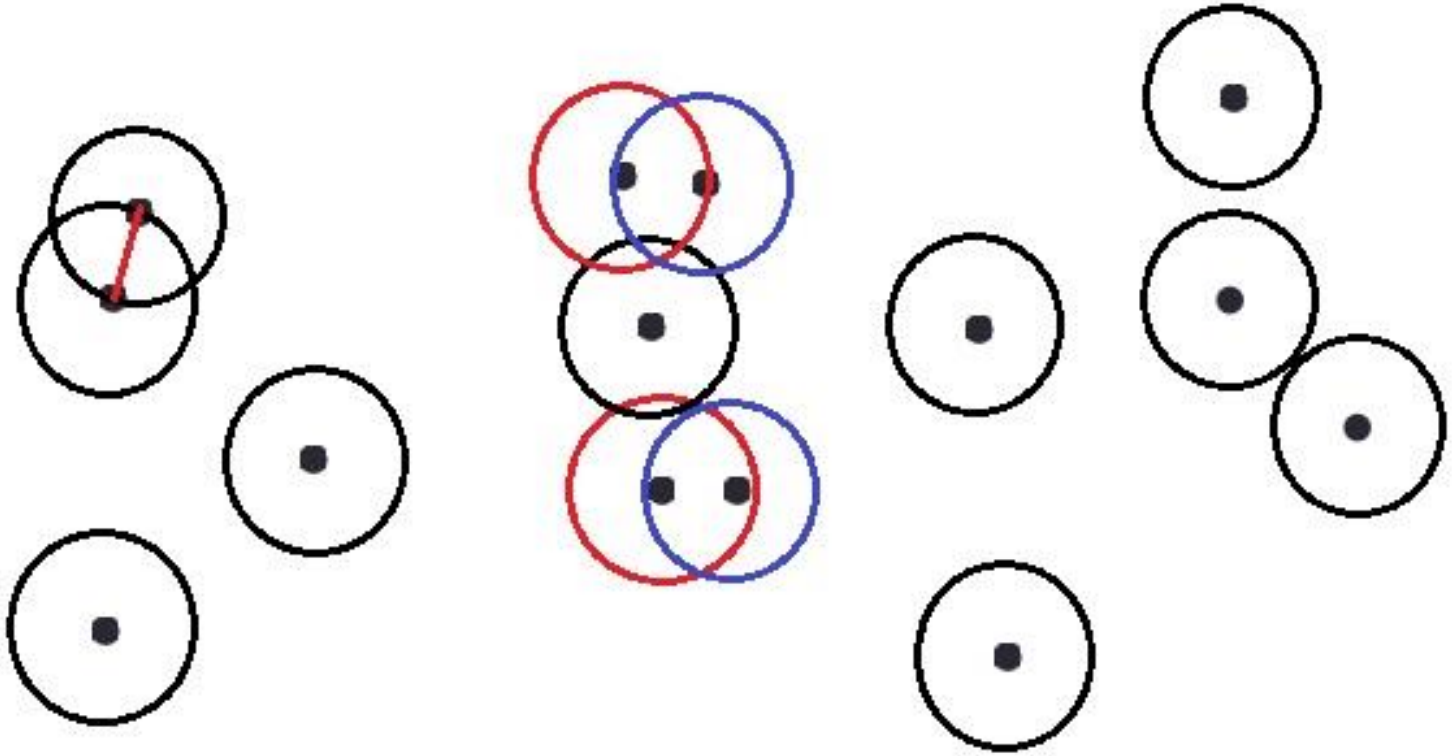


Naivna metoda

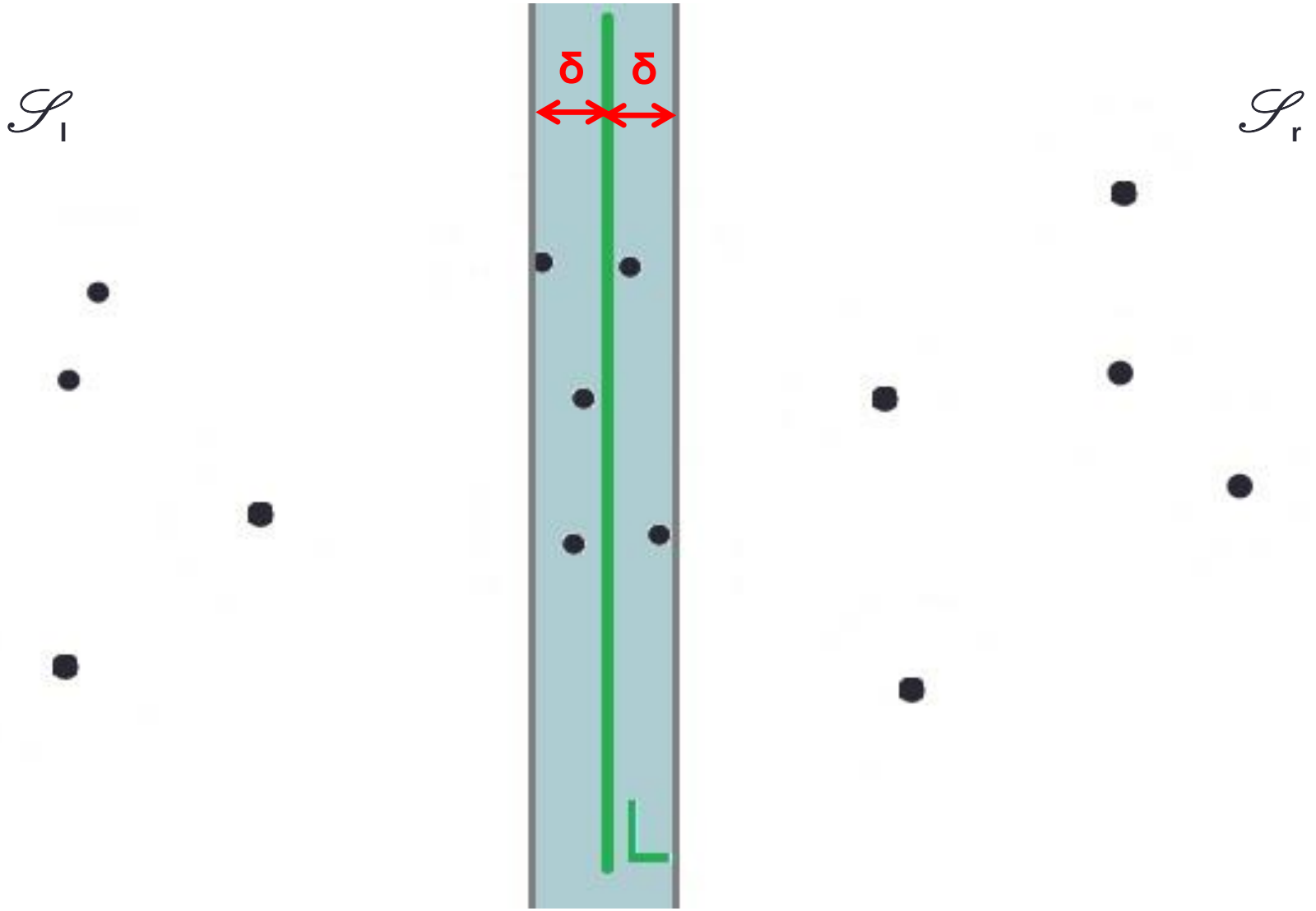
- ne tako - želimo bolje od $\Omega(n^2)$



Označimo $\delta := \min\{\delta_l, \delta_r\}$.
Ako točke p i q daju bolje rješenje, tada je q sadržan u zatvorenoj kugli oko p polumjera δ.



Ako se najbliži par sastoji od neke točke $p_l \in \mathcal{S}_l$ i neke točke $p_r \in \mathcal{S}_r$, tada p_l i p_r moraju biti unutar udaljenosti δ od pravca L.



- Formalnije,
- **Tvrdnja 1**

  Ako postoje $q \in \mathscr{S}_l$ i $r \in \mathscr{S}_r$ takvi da je $d(p,L) \leqslant \delta$,

  tada i q i r leže unutar udaljenosti δ od L.
- Dokaz

  Pretpostavimo da takvi q i r postoje. Označimo

  $q = (q_x, q_y)$

  $r = (r_x, r_y)$

  x^* = x-koordinata točke kojom prolazi pravac L

  Znamo (definicija od L) $q_x \leqslant x^* \leqslant r_x$.

  Sada $x^* - q_x \leqslant r_x - q_x \leqslant d(q,r) \leqslant \delta$

  te $r_x - x^* \leqslant r_x - q_x \leqslant d(q,r) \leqslant \delta$,

  pa i q i r imaju x-koordinatu unutar udaljenosti δ od x^*.

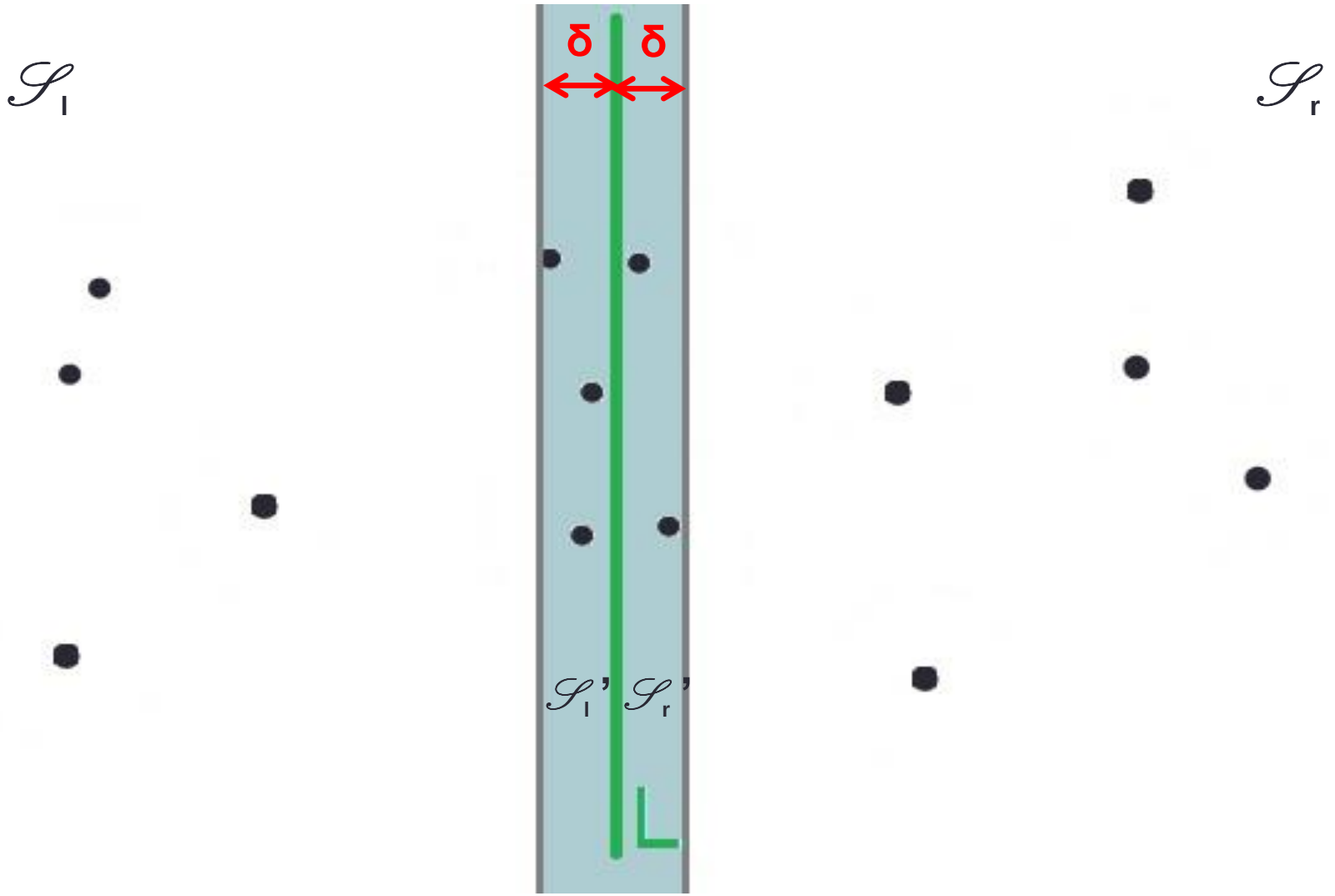
  Stoga obje točke leže unutar udaljenosti δ od pravca L.

■

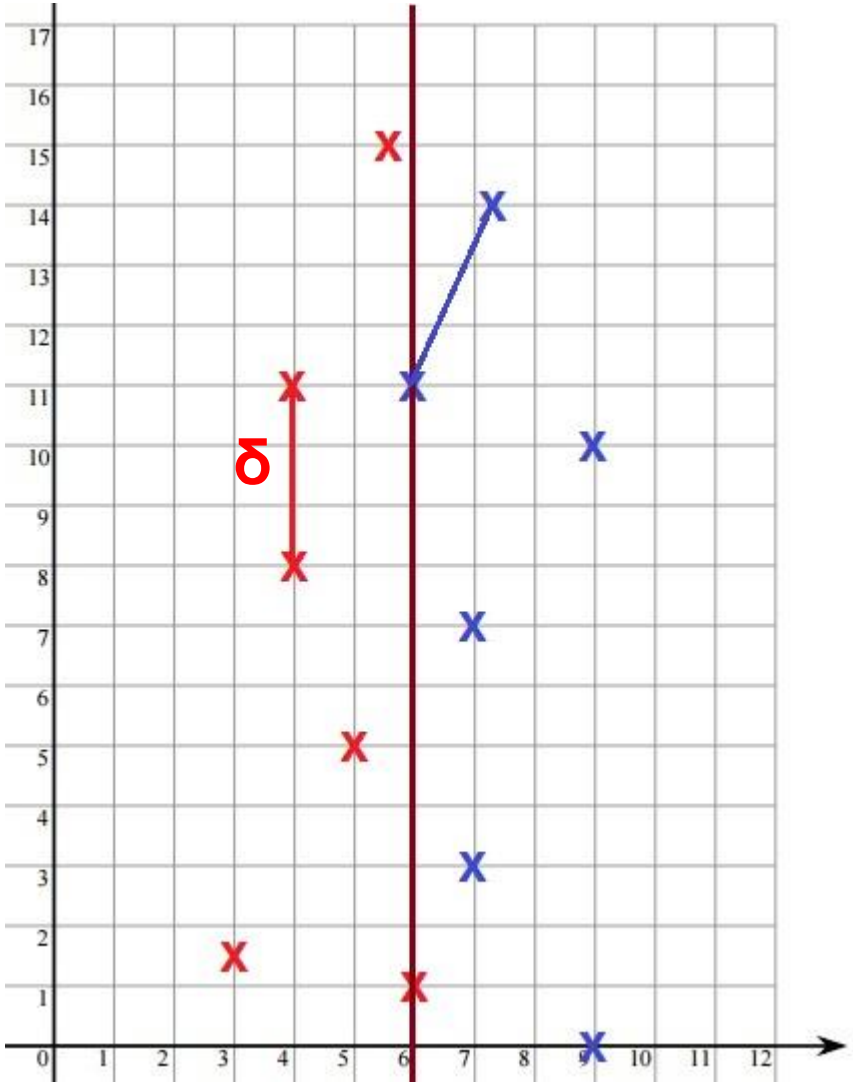
$\mathscr{S}_l' \coloneqq \{ p \in \mathscr{S}_l \mid d(p,L) \leqslant \delta \}$

$\mathscr{S}_r' \coloneqq \{ p \in \mathscr{S}_r \mid d(p,L) \leqslant \delta \}$

$\rightarrow p_l \in \mathscr{S}_l' \;\&\; p_r \in \mathscr{S}_r'$

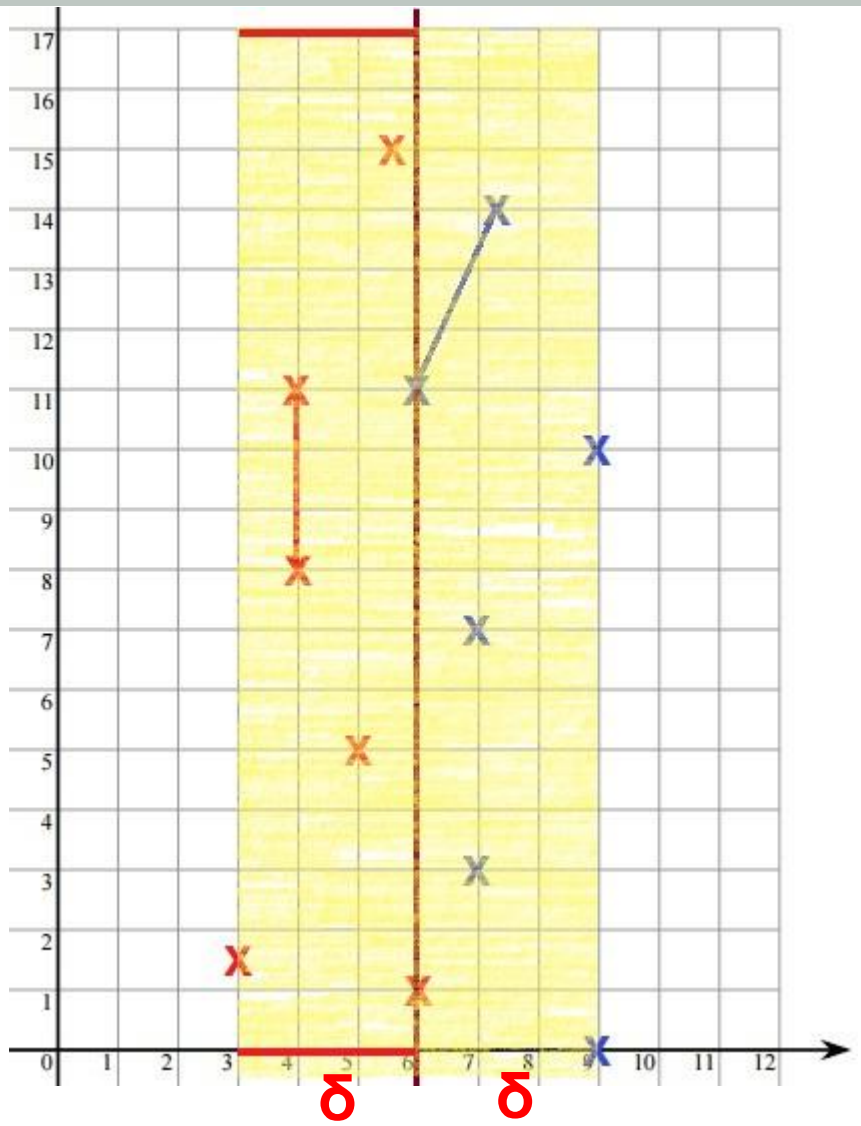


Primjer



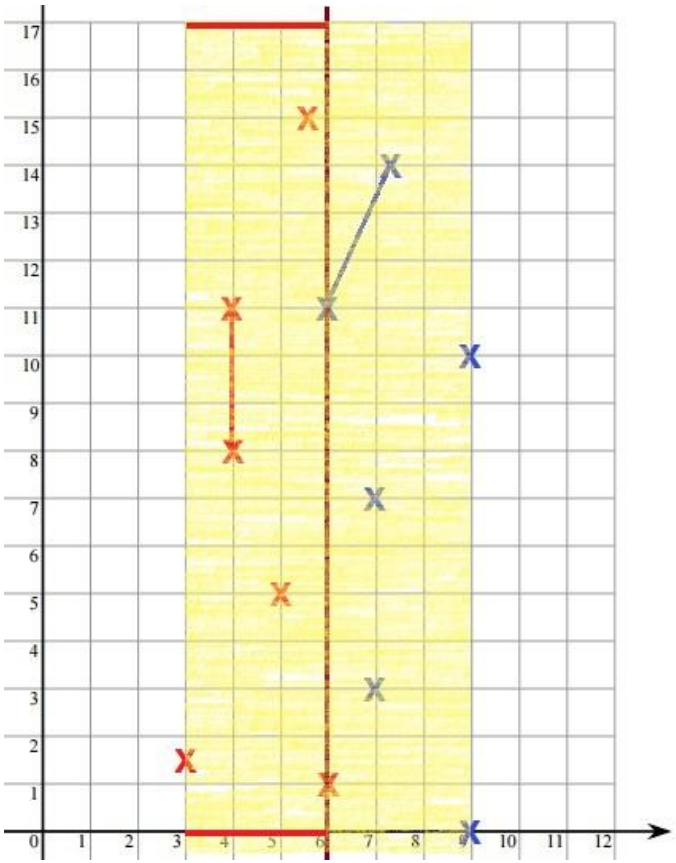
Primjer

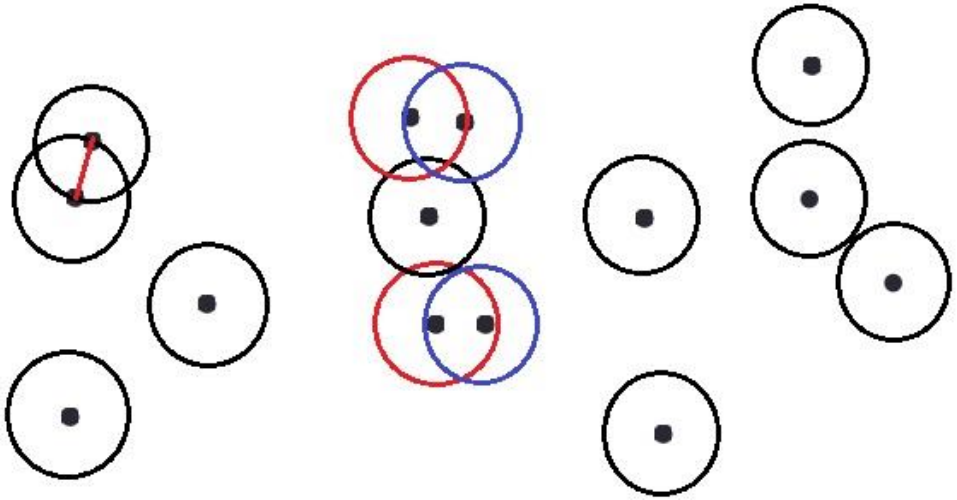
U najgorem slučaju, $\mathscr{S}_l' = \mathscr{S}_l$ i $\mathscr{S}_r' = \mathscr{S}_r$ pa svaku točku iz $\mathscr{S}_l$ treba usporediti sa svakom iz $\mathscr{S}_r$



Što sad?

- Uočimo, nije nužno svih $\mathcal{O}(n^2)$ usporedbi
- Svaku točku p ϵ $\mathcal{S}_i$' treba usporediti samo sa onima unutar udaljenosti δ





Pretpostavimo $\delta' \leqslant \delta$.

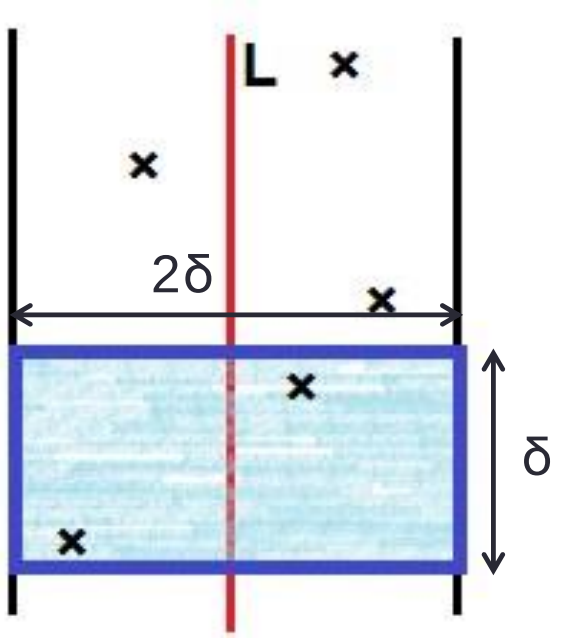
Tada postoje $p_l \in \mathscr{S}_l'$ i $p_r \in \mathscr{S}_r'$ tako da vrijedi

$$d(p_l, p_r) = \delta'.$$

→ Vertikalna udaljenost (razlika y-koordinata)

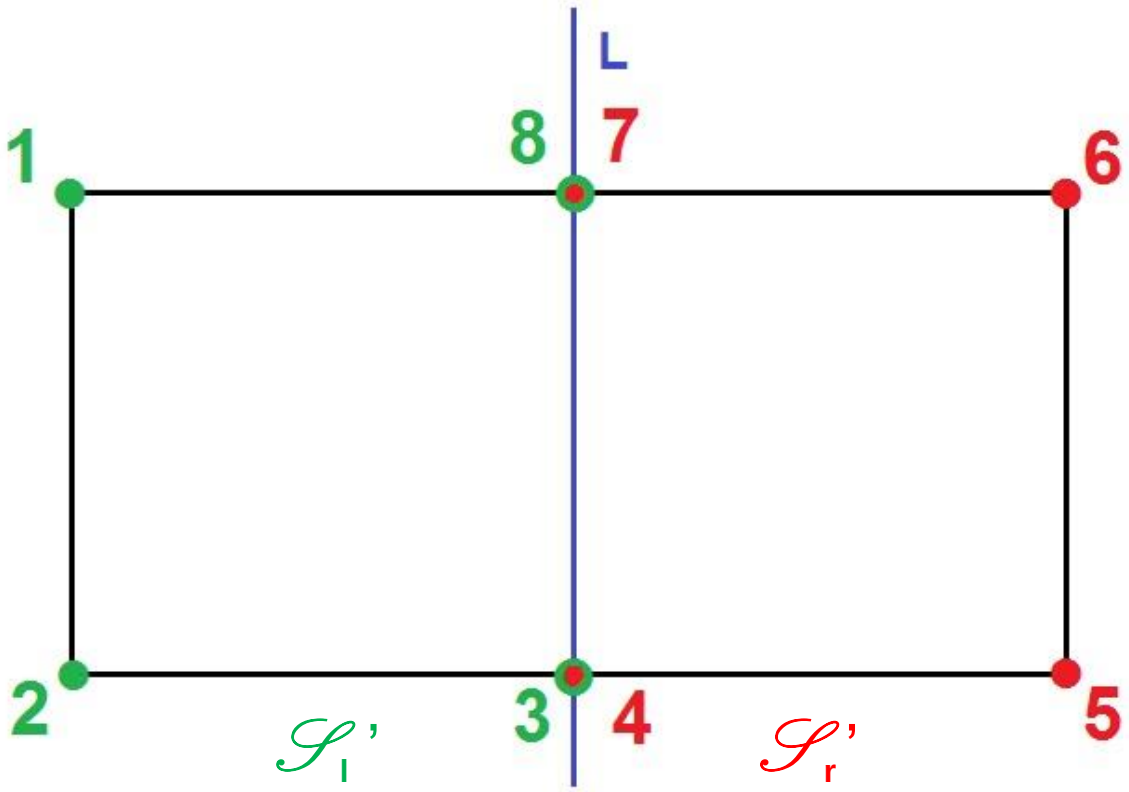
p_l i p_r iznosi najviše δ.

→ Te dvije točke leže unutar $\delta \times 2\delta$ pravokutnika centriranog oko L.



$\mathcal{T} := \mathcal{S}_l' \cup \mathcal{S}_r'$

Ako je udaljenost između bilo koje dvije točke u δ×2δ pravokutniku najviše δ, tada on može sadržavati najviše osam točaka: najviše 4 točke iz $\mathcal{S}_l$ i najviše 4 točke iz $\mathcal{S}_r$.



Svaka točka iz $\mathscr{T}$ treba se usporediti sa najviše sedam točaka iz $\mathscr{T}$.

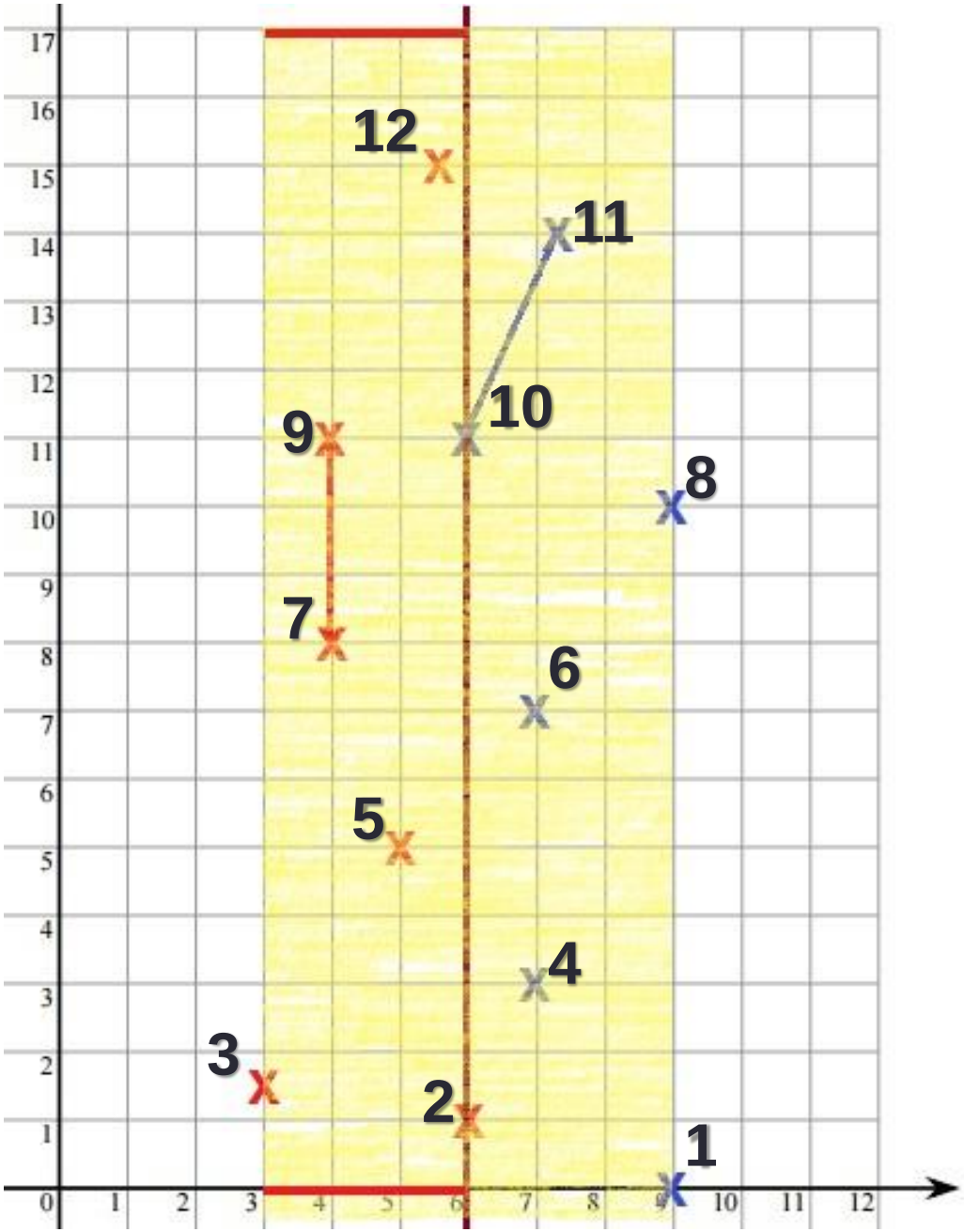
→ Dobili gornju ogradu na broj točaka sa kojima uspoređujem točku $p \in \mathscr{T}$.

? Koje su to točke (za neku točku $p \in \mathscr{T}$) ?

Točku p treba usporediti sa „susjedima” u $\mathscr{T}$.

→ sortiramo točke iz $\mathscr{T}$ rastuće po y-koordinati

→ svaku točku uspoređujemo sa 7 točaka nakon p ϵ $\mathscr{T}$ u rastućem poretku njihovih y-koordinata



- Formalno, za ocjenu složenosti dovoljno:
- **Tvrdnja**

  Ako su p,q ϵ $\mathscr{S}$ takvi da vrijedi d(p,q)<δ, tada su p i q unutar 15 pozicija razmaka u sortiranoj listi S_y po y-koordinatama.
- Dokaz

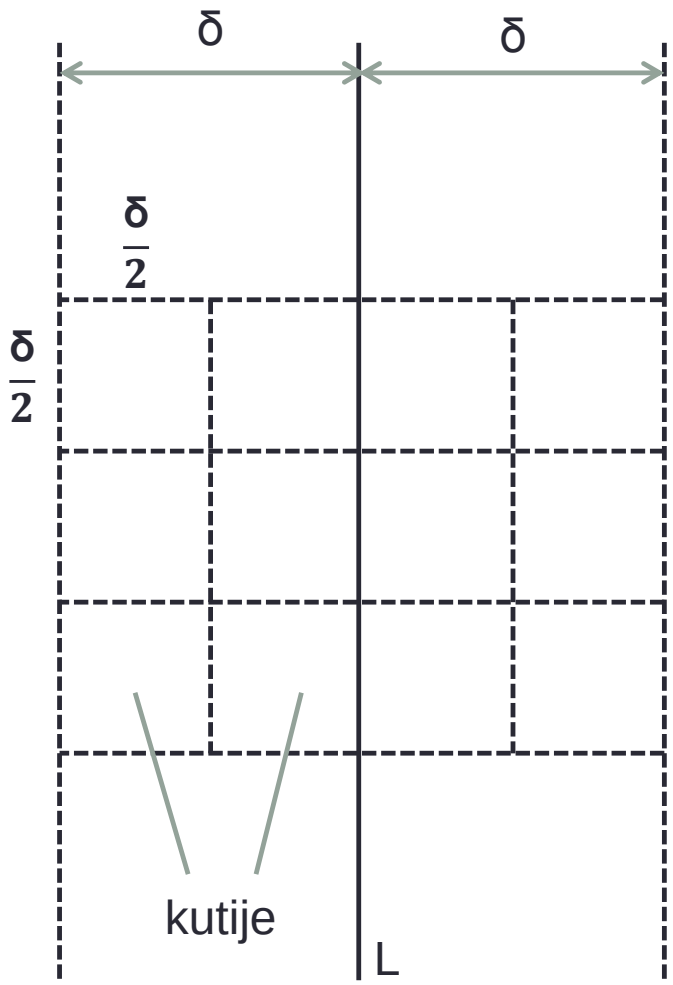
  Neka je Z podskup ravnine koji se sastoji od svih točaka unutar udaljenosti δ od L. Dijelimo Z u kutije (engl. boxes): kvadrate sa stranicom duljine δ/2. Jedan redak od Z se sastoji od četiri kutije čije horizontalne strane imaju istu y-koordinatu. (vidi sliku na slijedećem slajdu)

  Pretpostavimo da dvije točke iz $\mathscr{S}$ leže u istoj kutiji. S obzirom da sve točke u istoj kutiji leže s iste strane od L, te obje točke ili pripadaju u Q ili pripadaju u R. Ali bilo koje dvije točke u istoj kutiji su unutar udaljenosti manjoj od $\frac{\delta}{2}\sqrt{2}$< δ, što je u kontradikciji sa minimalnom udaljenošću δ između bilo kojeg para točaka u Q ili u R. Dakle, svaka kutija sadrži najviše jednu točku iz $\mathscr{S}$.

- (nastavak dokaza sa prethodnog slajda)

Pretpostavimo sada da p,q ϵ $\mathscr{S}$ imaju svojstvo d(p,q) < δ, te da su barem 16 pozicija razmaknuti u S_y. BSOMP da p ima manju y-koordinatu. S obzirom da u svakoj kutiji može biti najviše jedna točka, najmanje tri retka od Z leže između p i q. Ali bilo koje dvije točke od Z odvojene s najmanje tri retka moraju biti udaljene za najmanje 3δ/2 - kontradikcija.

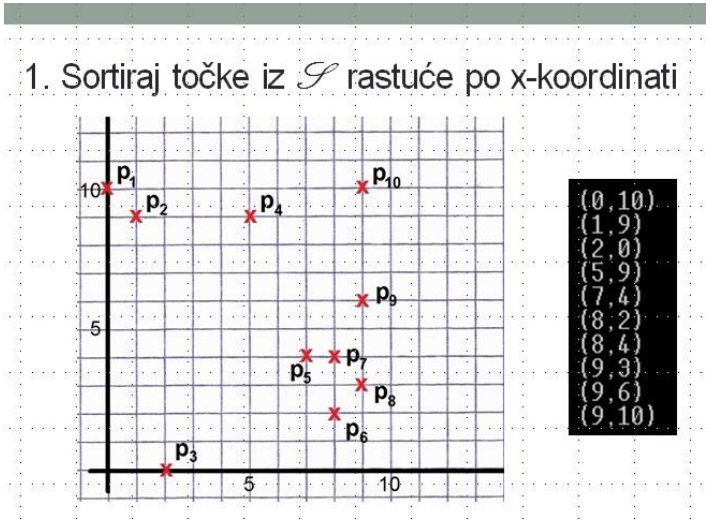
Q.E.D.



Svaka kutija može sadržavati najviše jednu točku ulaza.

Vremenska složenost

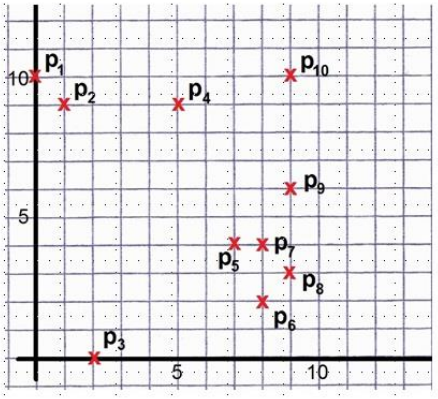
- sortiranje točaka u $\mathcal{S}$
- $\mathcal{O}$(nlogn)



Vremenska složenost

- sortiranje točaka u $\mathcal{S}$
- $\mathcal{O}$(nlogn)

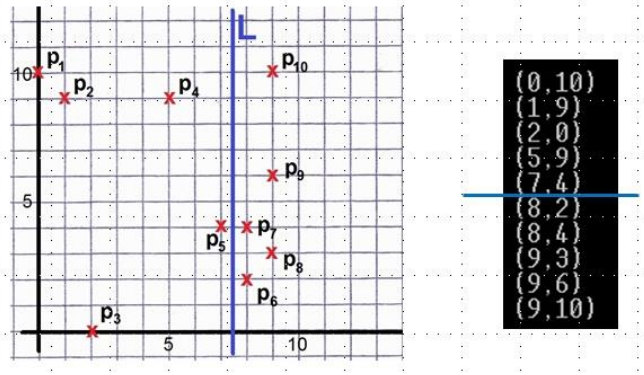
1. Sortiraj točke iz $\mathcal{S}$ rastuće po x-koordinati



- podjela točaka u $\mathcal{S}_l$ i $\mathcal{S}_r$
- Θ(1) (točke su sortirane)

2. Podijeli $\mathcal{S}$ pravcem L (paralelnim y-osi) u $\mathcal{S}_l$ i $\mathcal{S}_r$ tako da

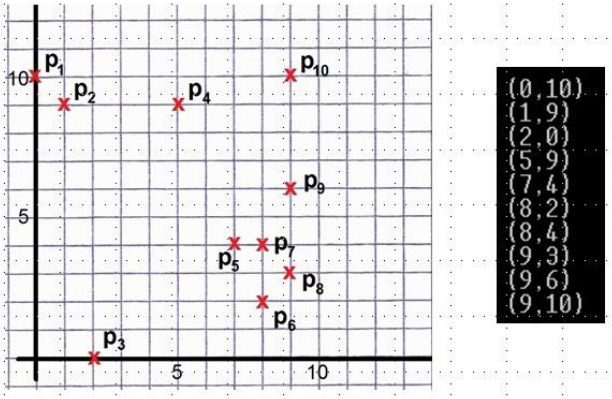
$$|\mathcal{S}_l| = \left\lfloor \frac{|\mathcal{S}|}{2} \right\rfloor \qquad |\mathcal{S}_r| = \left\lceil \frac{|\mathcal{S}|}{2} \right\rceil$$



Vremenska složenost

- sortiranje točaka u $\mathcal{S}$
- $\mathcal{O}$(nlogn)

1. Sortiraj točke iz $\mathcal{S}$ rastuće po x-koordinati

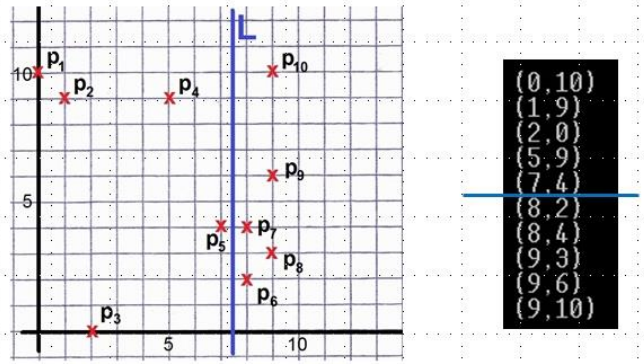


- combine step - sortira točke iz $\mathcal{T}$ i uspoređuje svaku točku s najviše 7 drugih (najviše 7n usporedbi)
- Θ(nlogn) (u najgorem slučaju $|\mathcal{T}| = n$, $|\mathcal{T}|\log|\mathcal{T}|=n\log n$)

- podjela točaka u $\mathcal{S}_l$ i $\mathcal{S}_r$
- Θ(1) (točke su sortirane)

2. Podijeli $\mathcal{S}$ pravcem L (paralelnim y-osi) u $\mathcal{S}_l$ i $\mathcal{S}_r$ tako da

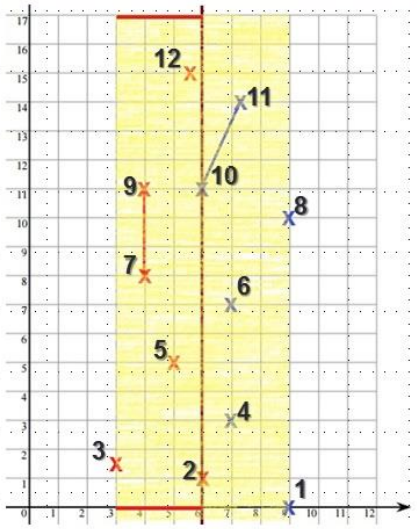
$$|\mathcal{S}_l| = \left\lfloor \frac{|\mathcal{S}|}{2} \right\rfloor \qquad |\mathcal{S}_r| = \left\lceil \frac{|\mathcal{S}|}{2} \right\rceil$$



Točku p treba usporediti sa „susjedima" u $\mathcal{T}$

→ sortiramo točke iz $\mathcal{T}$ rastuće po y-koordinati

→ svaku točku uspoređujemo sa 7 točaka nakon p ∈ $\mathcal{T}$ u rastućem poretku njihovih y-koordinata



$$T(n) = \begin{cases} 1 & , \ n=2 \\ 3 & , \ n=3 \\ 2T(\frac{n}{2}) + \mathcal{O}(n\log n), & n>3 \end{cases}$$

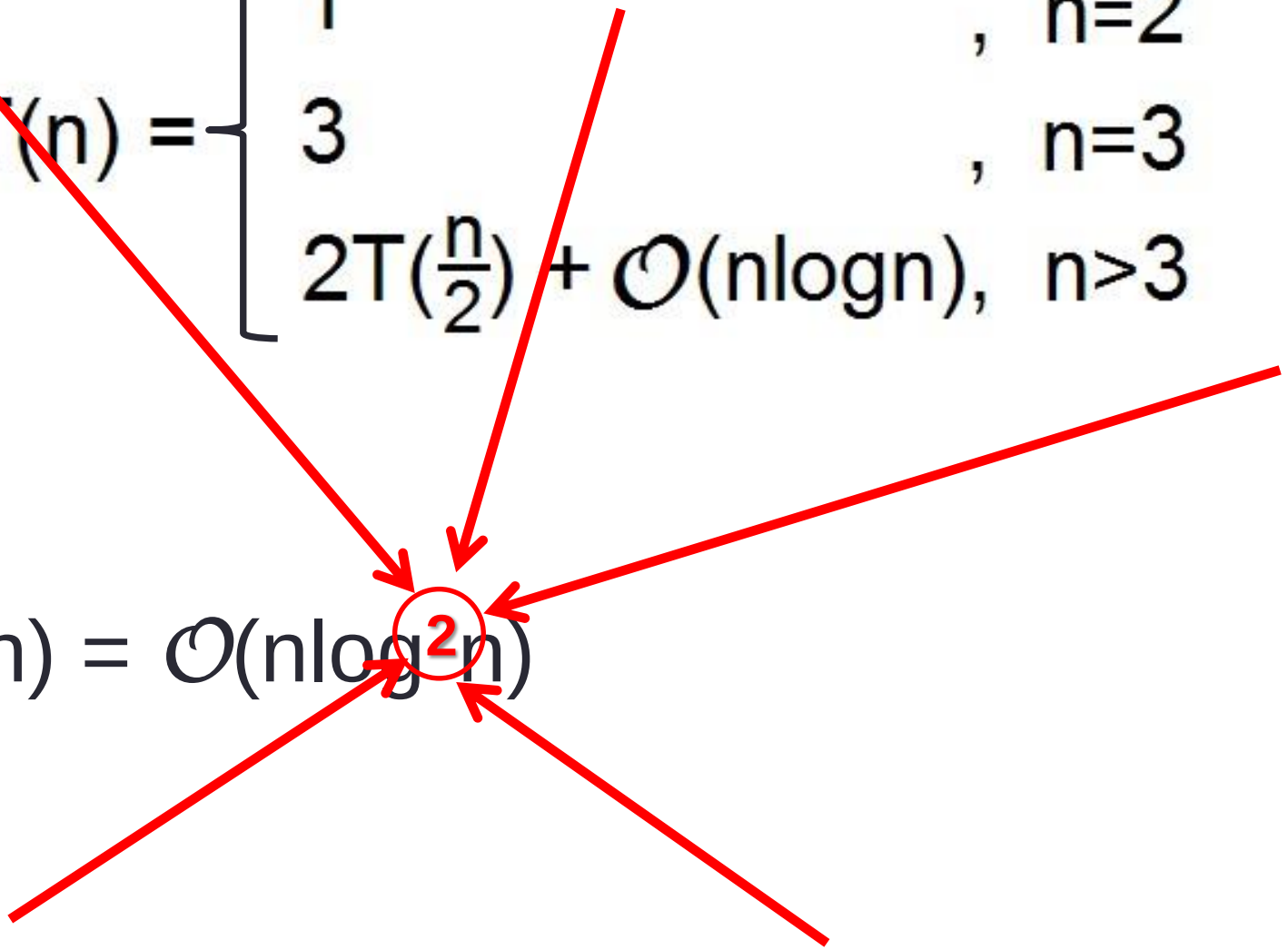
...

$$T(n) = \mathcal{O}(n\log^2 n)$$

$$T(n) = \begin{cases} 1 & , \ n=2 \\ 3 & , \ n=3 \\ 2T(\frac{n}{2}) + \mathcal{O}(n\log n), & n>3 \end{cases}$$

...

$$T(n) = \mathcal{O}(n\log^2 n)$$



- Primijetimo,

  ako vrijeme za combine step smanjimo na Θ(n), tada je vremenska složenost algoritma Θ(nlogn)
- To se postiže „presortiranjem”

  - elementi u $\mathcal{S}$ su sortirani po svojim y-koordinatama samo jednom i spremljeni u polje Y
  - svaki put umjesto sortiranja $\mathcal{T}$ u combine stepu, samo „izvadimo” elemente iz Y u Θ(n) (jednostavno, točke u $\mathcal{T}$ su one u Y unutar udaljenosti δ od L)

$$T(n) = \begin{cases} 1 & , \ n=2 \\ 3 & , \ n=3 \\ 2T(\frac{n}{2}) + \mathcal{O}(n\log n), & n>3 \end{cases}$$

...

$T(n) = \mathcal{O}(n\log^2 n)$

PRIJE

SADA

$$T(n) = \begin{cases} 1 & , \ n=2 \\ 3 & , \ n=3 \\ 2T(\frac{n}{2}) + \Theta(n) & , \ n>3 \end{cases}$$

...

$T(n) = \Theta(n\log n)$

Algoritam CLOSESTPAIR

- **Ulaz:** Skup S od n točaka u ravnini
- **Izlaz:** Minimalna udaljenost dvije točke u S
  1. Sortiraj točke iz S u nepadajućem poretku x-koordinata
  2. Y ← sortirane točke iz S u nepadajućem poretku y-koordinata
  3. δ ← cp(1,n)

Napomena

U algoritmu, za točku p, x(p) označava x-koordinatu točke p (projekcija na x-os)

cp(low,high)

1. **if** high-low+1<=3 **then** izračunaj δ direktno
2. **else**
3. mid ← (low+high)/2
4. x_0 ← x(S[mid])
5. δ_l ← cp(low,mid)
6. δ_r ← cp(mid+1,high)
7. δ ← min{δ_l,δ_r}
8. k ← 0
9. **for** i ← 1 **to** n {Izvadi T iz Y}
10. **if** |x(Y[i])-x_0| <= δ **then**
11. k ← k+1
12. T[k] ← Y[i]
13. **end if**
14. **end for** {k je veličina od T}

cp(low,high)

```
15.        δ'=2δ          {Postavi δ' na bilo koji broj veći od δ}
16.        for i ← 1 to k-1                    {Izračunaj δ'}
17.                for j ← i+1 to min{i+7,k}
18.                        if d(T[i],T[j]) < δ' then δ' ← d(T[i],T[j])
19.                end for
20.        end for
21.        δ ← min{δ,δ'}
22. end if
23. return δ
```

Main

```
int main(void)
{
    ifstream f("primjer.txt");
    int n;
    f >> n;
    tocka S[n], Y[n];
    for(int i=0; i<n; i++)  {    f >> S[i].x;   f >> S[i].y;    }
    qsort(S,n,sizeof(tocka),comparex);
    for(int i=0; i<n; i++)  {   Y[i].polozaj=S[i].polozaj=i;  Y[i].x=S[i].x; Y[i].y=S[i].y;   }
    qsort(Y,n,sizeof(tocka),comparey);
    tocka p;
    tocka q;
    cout << "Najmanja udaljenost = " << closestpair(0,n-1,S,Y,n,&p,&q) << endl;
    cout << "To su tocke (" << p.x << "," << p.y << ") i (";
    cout << q.x << "," << q.y << ")." << endl;
    f.close();
    return 0;
}
```

closestpair

```
double closestpair(int low, int high, tocka* S, tocka* Y, int n, tocka* p, tocka* q)
{
    if((high-low)==1)     … // Tražene točke su S[low] i S[high], vrati tu udaljenost
    if((high-low)==2)     … // Imamo tri točke, direktno računaj mi. udaljenost

    int mid=(low+high)/2;
    double L=S[mid].x;

    tocka Yl[mid-low+1], Yr[high-mid];
    int lijevi=0, desni=0;
    for(int brojac=0; brojac<n; brojac++)
    {
        if(Y[brojac].polozaj>mid) …// Točku Y[brojac] stavi u Yr[desni], desni++
        else                      …// Stavi tu točku u  Yr[lijevi], lijevi++
    }
```

closestpair

```
tocka t1,t2;
double deltal=closestpair(low,mid,S,Yl,lijevi,p,q);
double deltar=closestpair(mid+1,high,S,Yr,desni,&t1,&t2);
double delta;
if(deltar<deltal)       … // Točke p i q su zapravo t1 i t2
      else              delta=deltal;
int k=0;
tocka T[n];
for(int i=0; i<n; i++) //izvadi T iz Y
{               if( aps(Y[i].x-L)<=delta )
                {       T[k].x=Y[i].x;   T[k].y=Y[i].y; k++;     }
}       //k je velicina od T
```

closestpair

```
    double delta2=2*delta; //inicijaliziraj delta2 na bilo koji broj veci od delta
    for(int i=0;i<k;i++)
        for(int j=i+1; j<i+8 && j<k; j++)
            if(udaljenost(T[i],T[j])<delta2)  {
                    t1.x=T[i].x;            t1.y=T[i].y;
                    t2.x=T[j].x;            t2.y=T[j].y;
                    delta2=udaljenost(T[i],T[j]);  }
    if(delta2<delta)        {           p->x=t1.x;          p->y=t1.y;
                                        q->x=t2.x;          q->y=t2.y;
                                        return delta2;    }
    return delta;
}
```

Ostalo

```
struct tocka {     double x;
                   double y;
                   int polozaj;    };

double aps(double a) {   if(a>=0) return a;
                         return -a;    }

int comparex(const void* a, const void* b) {
          if(((tocka*)a)->x < ((tocka*)b)->x) return -1;
          return 1;           }

int comparey(const void* a, const void* b) {
    if(((tocka*)a)->y < ((tocka*)b)->y) return -1;
    return 1; }

double udaljenost(tocka A, tocka B) {
    return sqrt((A.x-B.x)*(A.x-B.x)+(A.y-B.y)*(A.y-B.y));  }
```

Teorem

Za dani skup $\mathcal{S}$ od n točaka u ravnini,

vremenska složenost algoritma CLOSESTPAIR koji nalazi par točaka iz $\mathcal{S}$ sa najmanjom udaljenošću je $\Theta(n \log n)$.

Primjeri (rezultati)

Primjer 1.

```
Najmanja udaljenost = 1
To su tocke (7,4) i (8,4).
```

primjer2 - Notepad

File Edit Format View Help

```
10
5.0 9.0
9.0 3.0
2.0 0.0
8.0 4.0
7.0 4.0
9.0 10.0
1.0 9.0
8.0 2.0
0.0 10.0
9.0 6.0
```

Primjer 2.

primjer - Notepad

File Edit Format View Help

```
10
0.654682 0.925557
0.409382 0.619391
0.891663 0.888594
0.716629 0.9962
0.477721 0.946355
0.925092 0.81822
0.624291 0.142924
0.211332 0.221507
0.293786 0.691701
0.839186 0.72826
```

```
Najmanja udaljenost = 0.0779102
To su tocke (0.891663,0.888594) i (0.925092,0.81822).
```

Testiranje

Intel Celeron CPU 847 @ 1.10 GHz

test - Notepad

File Edit Format View Help

```
0.0876068 0.666667 0.233103 1.33148 2.06882 4.82877 0.339372 1.95325 1.05514 1.89245
0.646281 5.85714 0.762402 0.587169 0.801339 0.614856 1.43043 0.951807 2.22333 0.0391061
0.865764 0.964072 1.01203 0.198034 0.291139 0.848062 0.873351 0.0430233 0.974394 0.679076
8.77778 0.225386 2.69159 0.0424178 0.40678 0.55335 1.21918 1.05413 0.0588235 0.715847
1.00801 0.0879581 0.89893 2.56233 3.01294 2.14545 1.9321 0.996289 1.42169 1.71402
7.18103 0.969651 0.756176 3.18241 1.73902 0.0281501 12.6575 0.325301 1.35366 0.189084
3.38832 0.252747 0.46224 1.13913 0.584906 2.31788 1.29793 2.2413 0.557292 0.0207101
1.58681 1.96094 1.03846 0.275362 0.375212 0.445618 16.3226 2.44379 1.52027 1.16159
1.13889 1.15985 1.13223 14.1667 1.7151 0.34767 17.8095 27.0909 1.74 1.37732
0.382313 0.053 0.445154 1.14068 0.271368 0.814318 1.33884 1.91232 0.501618 1.57864
0.50081 2.06858 2.4 0.931777 2.625 24.8889 1.38361 1.40683 0.401235 0.270349
0.889456 0.623016 2.22886 0.739903 0.727729 2.82979 0.737171 0.056962 0.757624 1.83618
0.211111 0.288754 4.98958 0.91676 0.304688 0.31811 4.85714 0.506955 0.408117 2.01613
6.07692 2.02398 0.881882 0.493865 0.03 1.16771 2.28718 0.105189 1.17417 0.00344828
1.24964 0.507595 0.601415 0.00341297 0.636364 0.206089 0.813984 0.891747 1.09032 3.79474
2.95758 0.531025 0.764388 0.789091 0.859649 2.37705 0.952255 0.32783 0.27984 1.29623
1.25342 0.911579 0.984694 0.71223 2.69811 1.72438 0.619048 0.162393 0.470238 2.97727
0.879213 3.42593 1.12732 0.879493 0.413474 0.57424 0.657172 3.31724 0.878924 0.796296
1.18625 0.37045 0.0666667 0.663636 10.12 0.30137 1.14615 0.592357 20.6087 8.84211
0.868653 2.44388 0.322684 1.1494 0.380606 0.381714 2.325 11.7324 1.63423 2.91011
2.8524 1.14051 0.194726 0.212058 0.339172 8.02 0.841853 0.587027 0.0236382 0.335165
0.00692841 0.850168 22.6563 3.44056 0.773519 0.0710322 0.518006 0.42359 1.57895 0.281775
0.934297 1.34132 0.517241 0.201439 1.112 0.150794 0.828777 2.17162 0.794433 1.14141
1.75464 0.0387097 0.777778 9.09091 0.0647249 0.0131926 0.537975 2.52632 0.0566535 1.49091
10.1379 1.15779 1.21474 0.995223 0.677072 0.8119 0.936656 0.20603 2.81298 0.370722
1.01149 1.7265 0.0917431 0.0142487 0.75 0.293666 0.854054 0.246073 1.10329 0.79759
1.2605 1.65458 45.625 2.37313 0.131818 0.39816 1.09846 4.37615 0.127252 0.941246
1.07226 2.57922 0.748614 0.157447 2.33894 0.205128 1.69342 2.56221 1.7486 1.46927
1.23897 2.40055 38.9565 5.46018 1.02869 13.9286 3.25385 0.404594 0.599143 0.345794
0.0550363 0.794558 0.672148 1.23514 0.551867 1.25413 1.43239 0.0789164 0.71885 1.55944
1.47266 4.21023 0.55569 0.253731 0.669519 0.26148 2.13033 0.397436 0.262585 1.19065
0.559356 0.249292 1.75228 2.92691 0.643302 0.998832 0.306695 0.6959 0.624448 1.69369
0.439169 0.127389 11.9863 1.33879 0.0182208 0.16092 0.203125 0.0817996 7.52747 0.842373
6.78082 1.12063 0.878543 0.169884 0.440789 0.316832 1.45055 0.606024 1.27258 0.292585
```

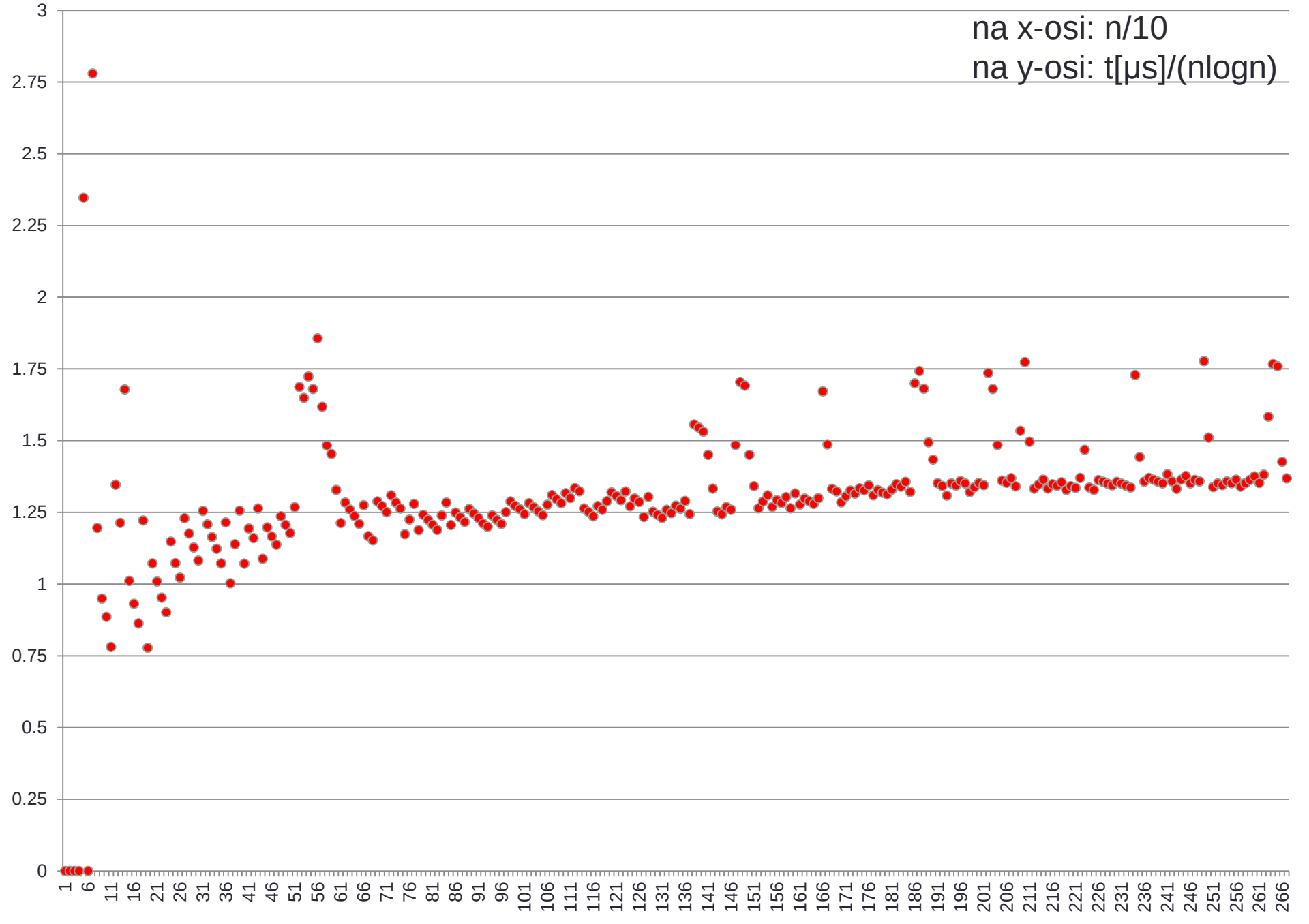
HR 10:29 23.1.2017.

20 000 slučajnih brojeva u test.txt

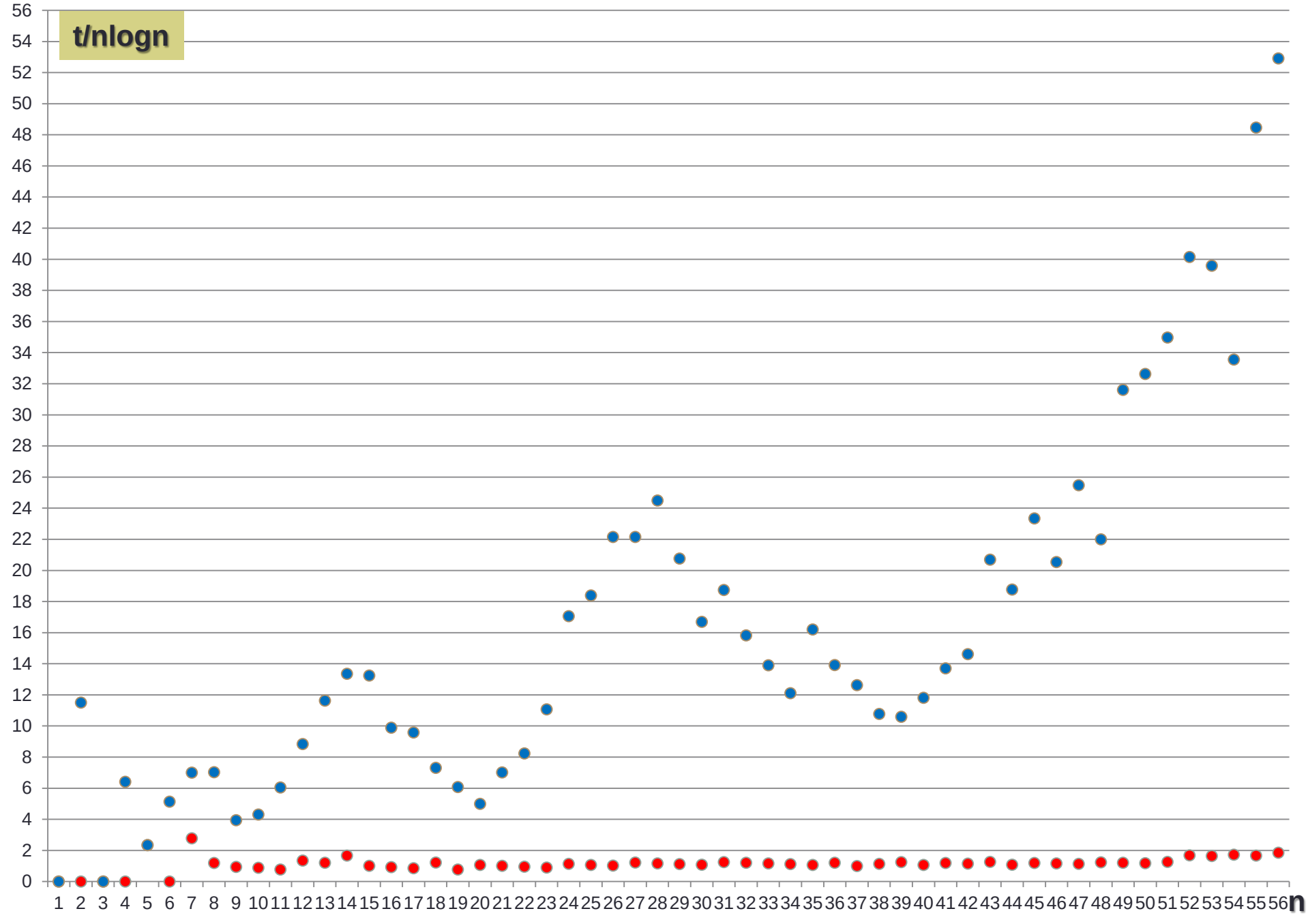
(naravno, nisu svi iskorišteni)

NOT TESTED ON ANIMALS

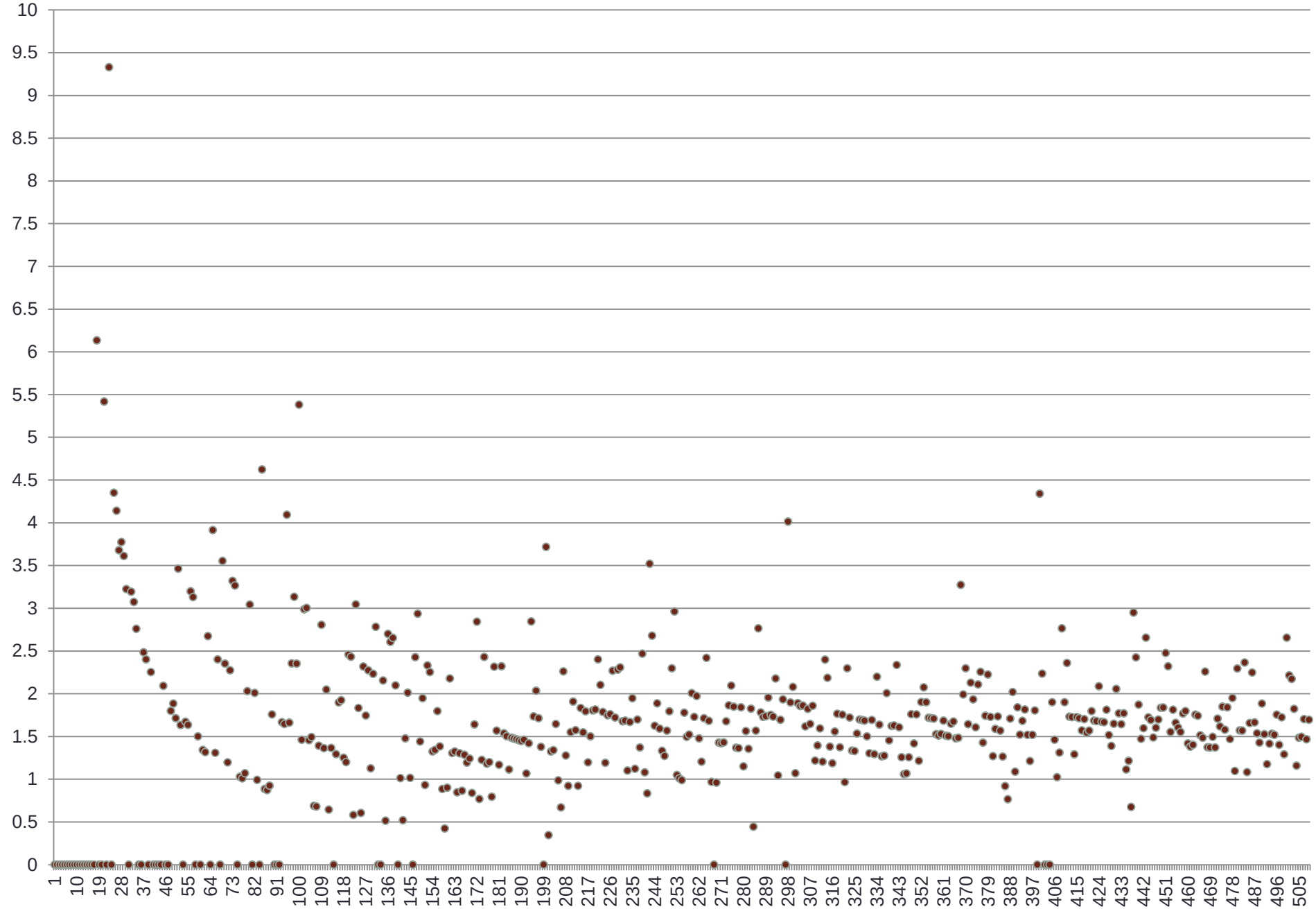
Rezultati


na x-osi: n/10
na y-osi: t[µs]/(nlogn)
3
2.75
2.5
2.25
2
1.75
1.5
1.25
1
0.75
0.5
0.25
0
1
6
11
16
21
26
31
36
41
46
51
56
61
66
71
76
81
86
91
96
101
106
111
116
121
126
131
136
141
146
151
156
161
166
171
176
181
186
191
196
201
206
211
216
221
226
231
236
241
246
251
256
261
266

Rezultati - realizacija pseudokoda vs. kod



Rezultati (koristi T ’(n)=T(n)+sort)



Pitanja

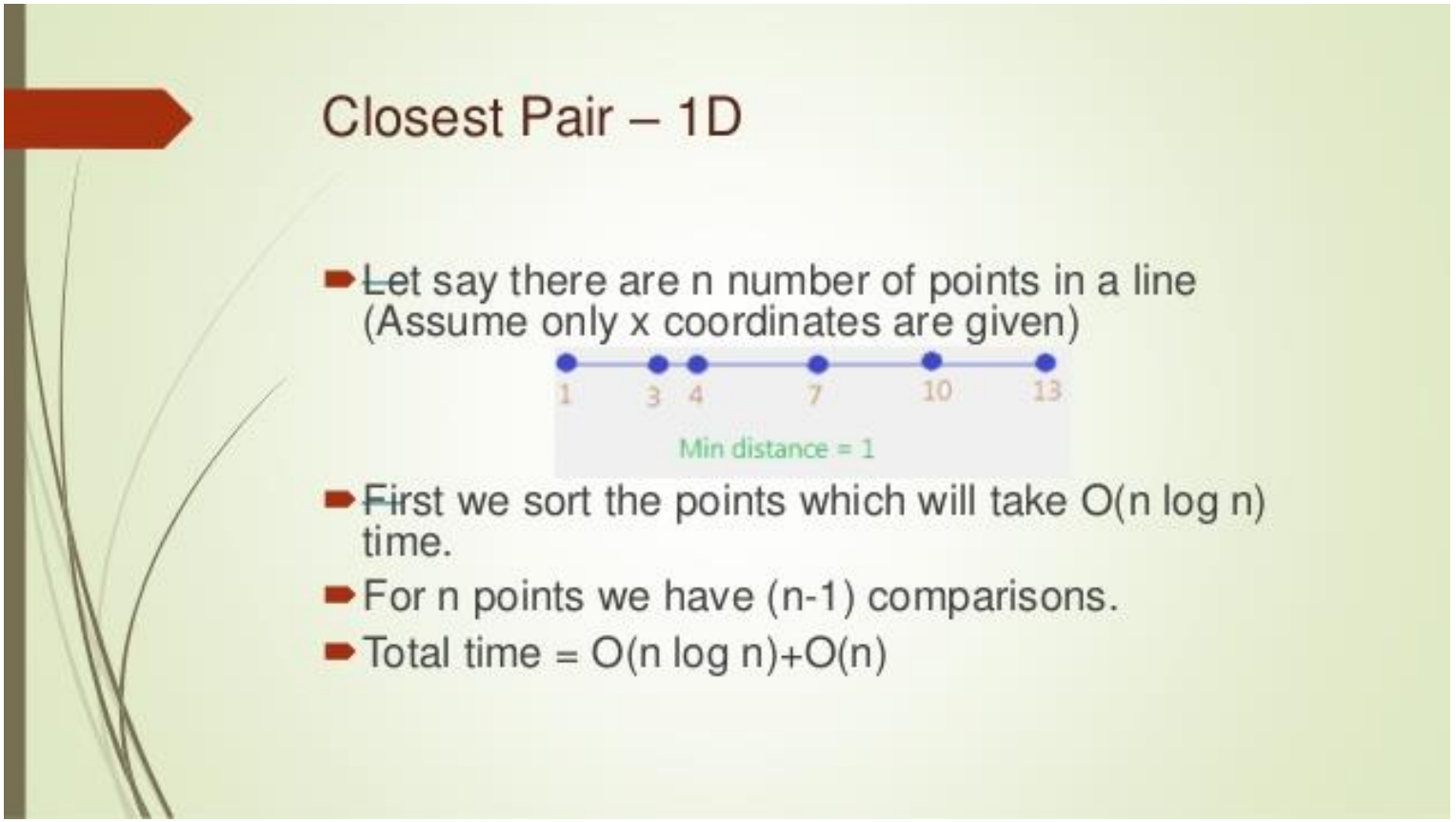
1. Ideja: Točke koje leže na pravcu L uvijek staviti u $\mathscr{S}_l$. Sada svaku točku iz $\mathscr{T}$ treba usporediti sa najviše 5 točaka (umjesto dosadašnjih 7), tj. najviše 6 točaka može biti u δx2δ pravokutniku. Točno?
   - Ne.
2. Ako osiguramo da se nikoje dvije zadane točke ne podudaraju, koliko sada treba biti usporedbi - 7 i dalje?
   - Ne, 6.

Pitanja

3. Promijenimo algoritam tako da ne uspoređujemo svaku točku iz $\mathcal{T}$ sa sedam točaka iz $\mathcal{T}$, nego svaku točku lijevo od L usporedimo sa nekim točkama njoj zdesna. Kako to postići? Koliko točaka desno od L treba usporediti sa svakom točkom slijeva?
4. Algoritam bez presortiranja, uz vremensku složenost Θ(nlogn)? (MERGESORT)

Pitanja

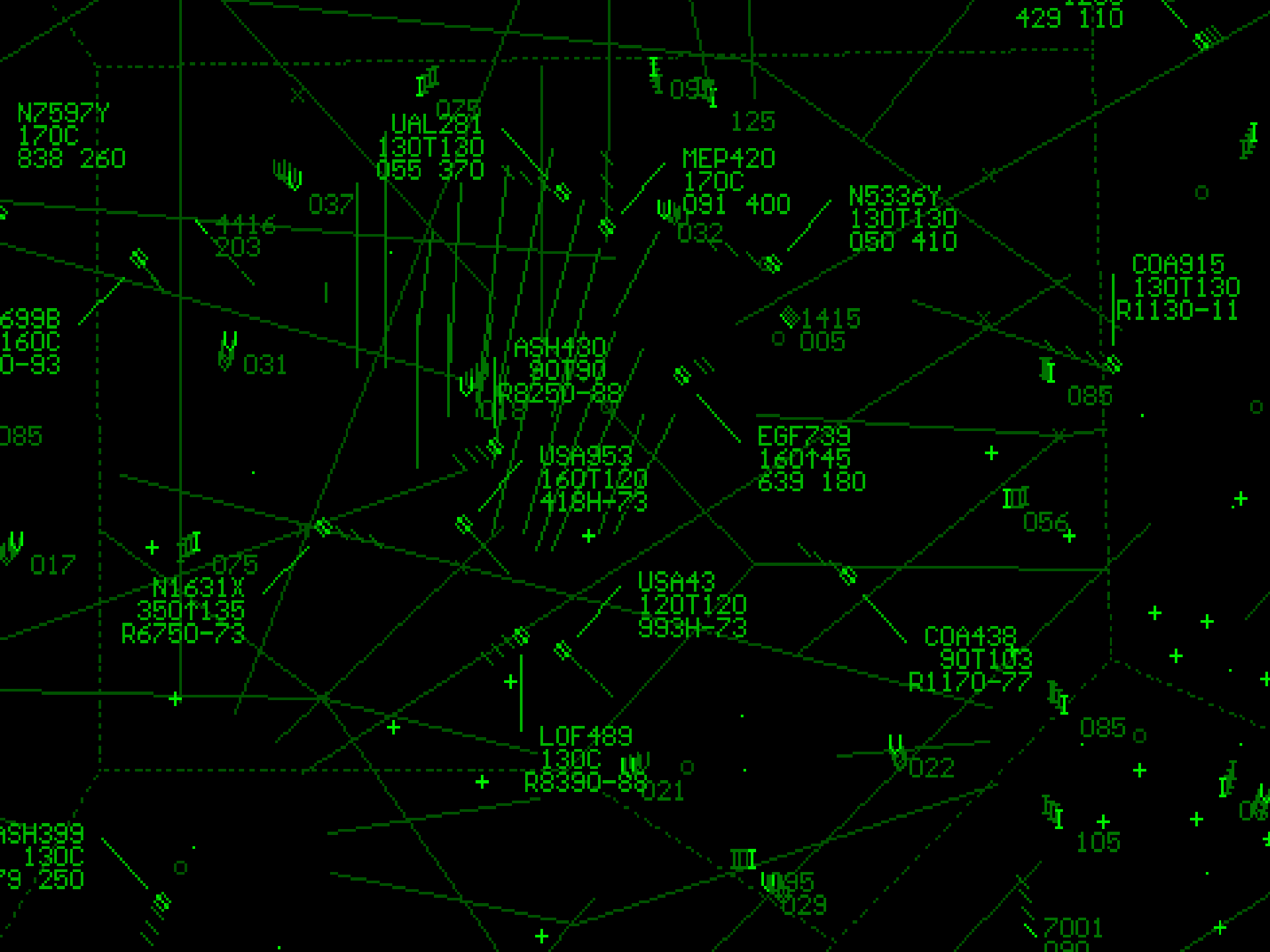
5. Jednodimenzionalna varijanta problema?



preuzeto sa www.slideshare.net/aruneel1/closest-pair-final-53246545 (pristupljeno 22. siječnja 2017.)

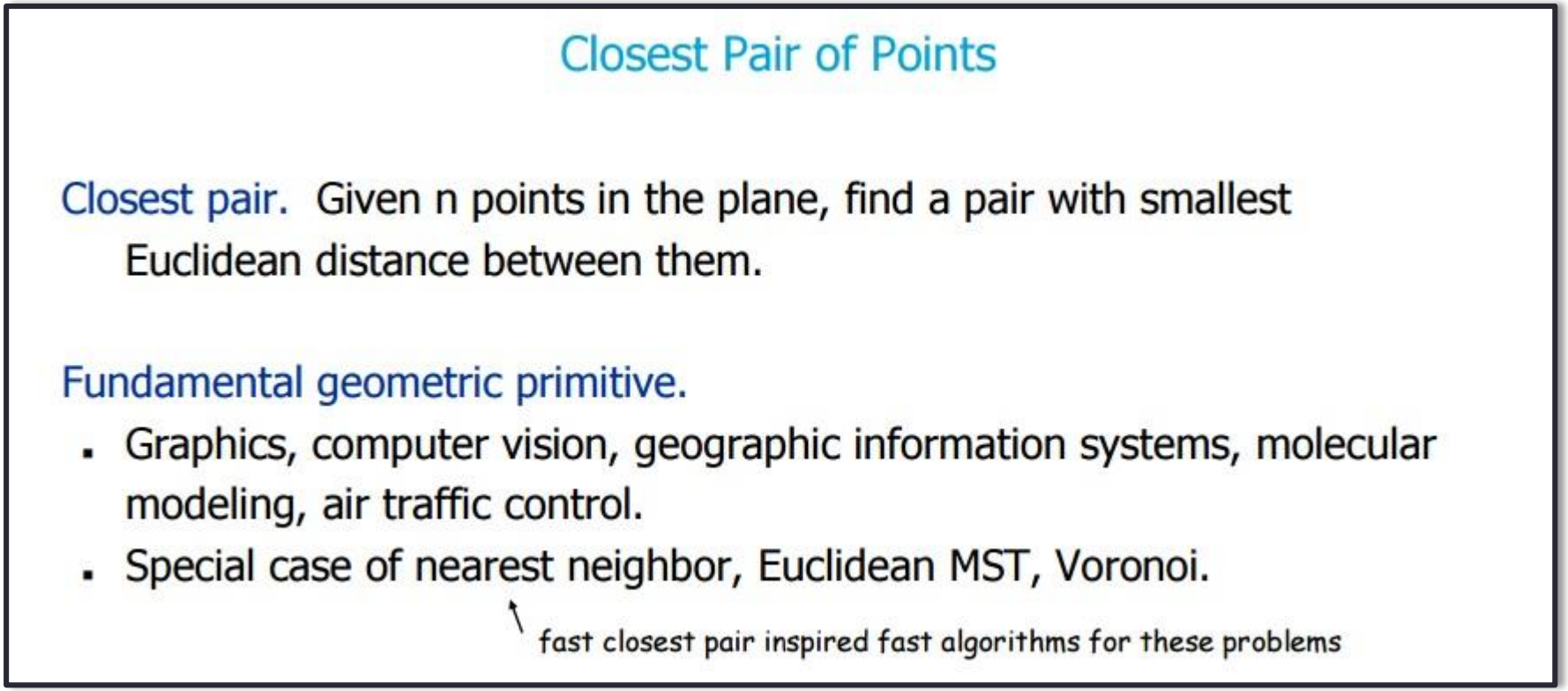
Pitanja

6. Moguće primjene?
   - Kontrola zračnog/morskog prometa (izbjegavanje mogućih kolizija - pratimo dva najbliža vozila)


429 110
N7597Y
170C
838 260
075
UAL281
130T130
055 370
125
MEP420
170C
091 400
032
037
4416
203
N5336Y
130T130
050 410
COA915
130T130
R1130-11
1415
005
085
031
ASH480
90T90
R8250-88
EGF739
160T145
639 180
VSA953
160T120
418H-73
056
017
075
N1631X
350T135
R6750-73
USA43
120T120
993H-23
COA438
90T103
R1170-77
085
LOF489
130C
R8390-88
021
022
105
095
029
7001

Pitanja

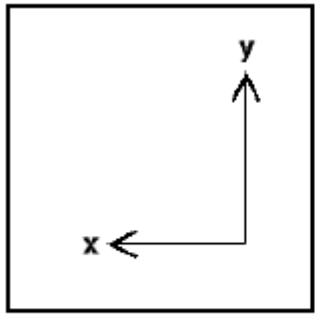
7. Ostale primjene?
   - Da, spominju se i ostale…



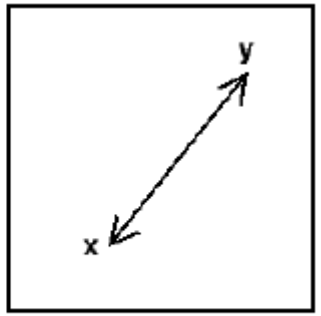
preuzeto sa ocw.tudelft.nl/wp-content/uploads/Algoritmiek_Closest_Pair_of_Points.pdf (pristupljeno 22. siječnja 2017.)

Pitanja

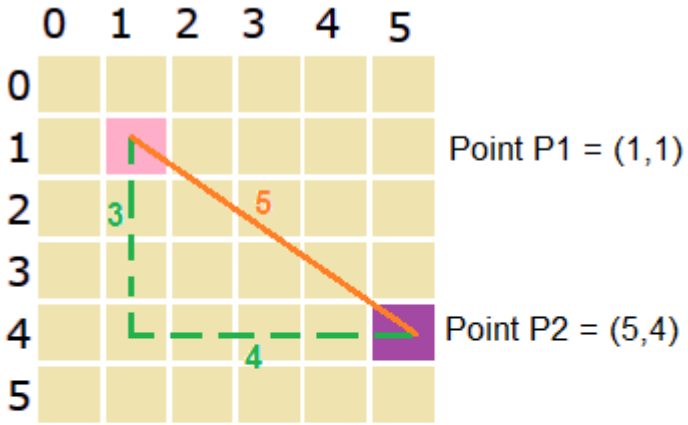
8. Nužno euklidska udaljenost?
   - Naravno da ne.



Manhattan

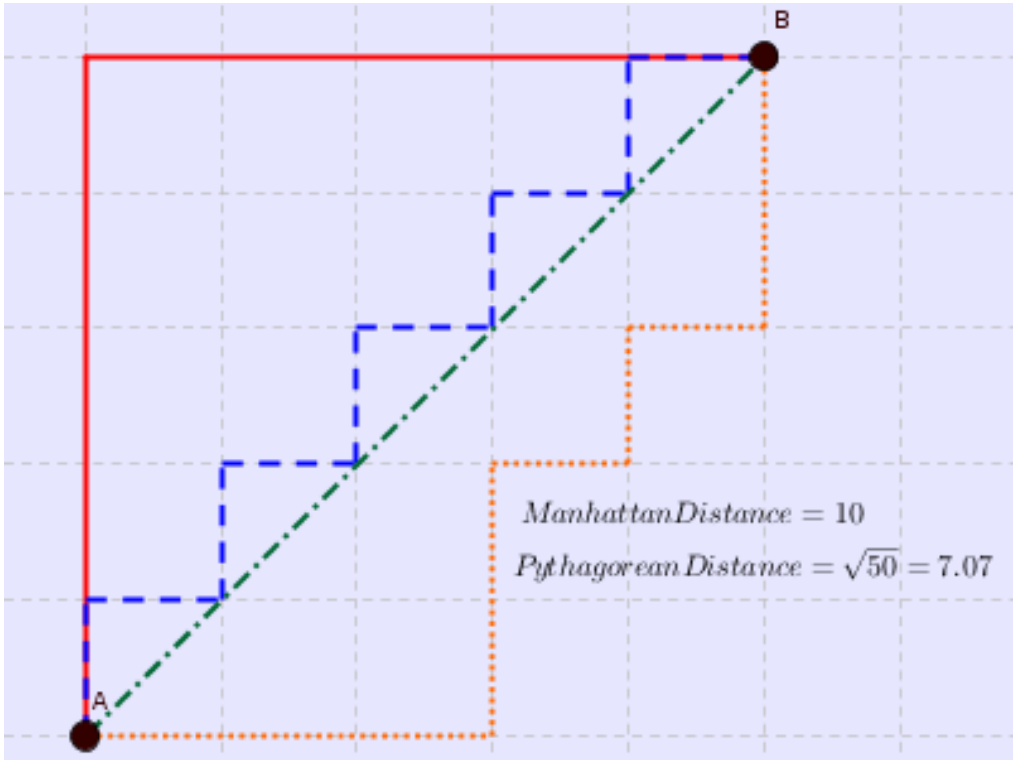


Euclidean



Euclidean distance = $\sqrt{(5-1)^2 + (4-1)^2} = 5$

Manhattan distance = $|5-1| + |4-1| = 7$




Terminal 5
Central Park Zoo
Intrepid Sea Air and Space Museum
Clinton
Clearview Ziegfeld
21 Club
Hell's Kitchen
Midtown East
Chelsea
Korea Town
Murray Hill
High Line Park
Fashion Institute of Technology
St Vartan Park
The Joyce Theater
Kips Bay
NYU Medical Center
Flatiron District
12th Ave
10th Ave
9th Ave
W 59th St
5th Ave
E 60th St
E 59th St
W 43rd St
W 42nd St
E 57th St
E 56th St
W 34th St
Dyer Ave
8th Ave
11th Ave
Madison Ave
Park Ave
3rd Ave
9A
495
A
B

Voronoi dijagram

Not logge

WIKIPEDIA
The Free Encyclopedia

Article Talk Read Edit View history Sea

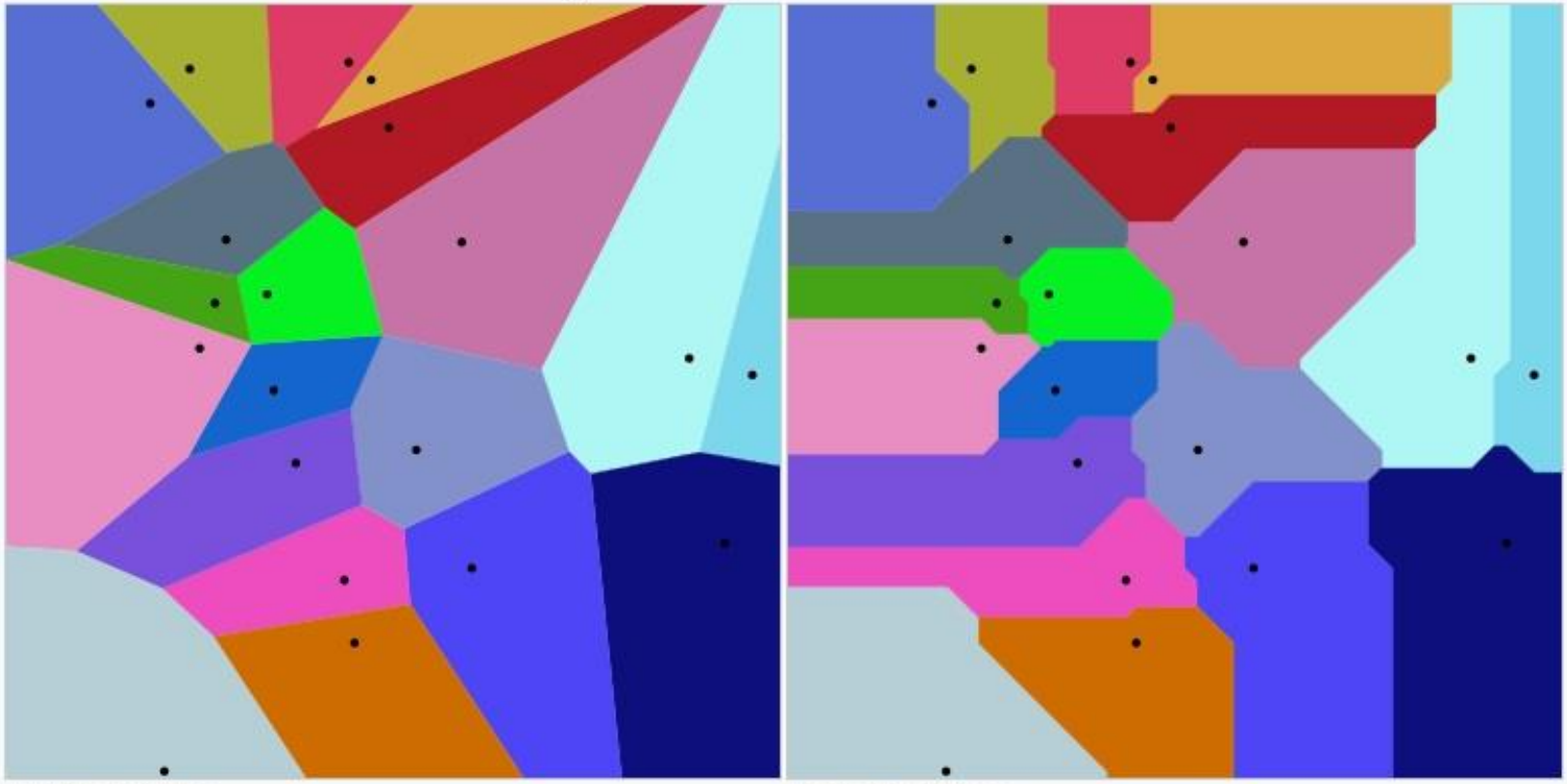
Voronoi diagram

From Wikipedia, the free encyclopedia

Main page
Contents
Featured content

In mathematics, a **Voronoi diagram** is a partitioning of a plane into regions based on distance to points in a specific subset of the plane. That set of points (called seeds, sites, or generators) is specified beforehand, and for each seed there is a corresponding region consisting of all points closer to that seed than to any other. These regions are called Voronoi cells. The Voronoi diagram of a set of point

Voronoi diagrams of 20 points under two different metrics



Euclidean distance

Manhattan distance

Voronoi dijagram

Voronoi dijagram je geometrijski koncept koji se temelji na blizini diskretnih skupova podataka. Uobičajeno se kaže da je Voronoi dijagram unija rubova između Voronoi regija definiranih kao skup točaka koje su bliže jednom elementu prostora (npr. u slici 3.2.1. element prostora je točka) nego nekom drugom. Moguće je i formalno definirati na sljedeći način:

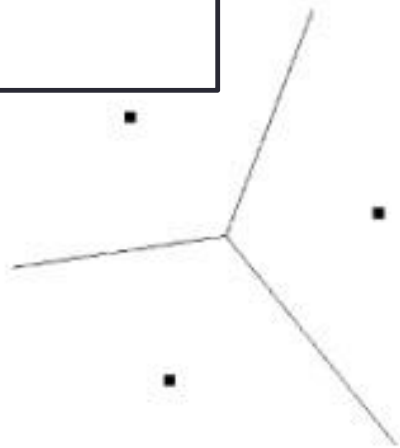
Voronoi regija jednog elementa e iz skupa elemenata E, oznake VR(e), je skup točaka koje su bliže elementu e nego nekom drugom elementu iz E

$$VR(e) = \{ p \in \mathbb{R}^n \mid d(p,e) \leq d(p,e'), \forall e' \neq e, e' \in E \}$$

gdje je d uobičajena Euklidova udaljenost između dva elementa. Voronoi dijagram iz E, oznake VD(E), definira se kao unija granica Voronoi regija

$$VD(E) = \cup_i \delta VR(e_i)$$

gdje je simbol δ granica elementa.



Slika 3.2.1. Jednostavan Voronoi dijagram triju točaka

Mrkonjić, R. (2011) *Postupci određivanja kostura modela na osnovi poligonalnog modela* [online]. Zagreb: Fakultet elektronike i računarstva. Dostupno na: bib.irb.hr/datoteka/519129.Postupci_odreivanja_kostura_modela_na_osnovi_poligonalnog_modela_ver_1.pdf [22. siječnja 2017.]

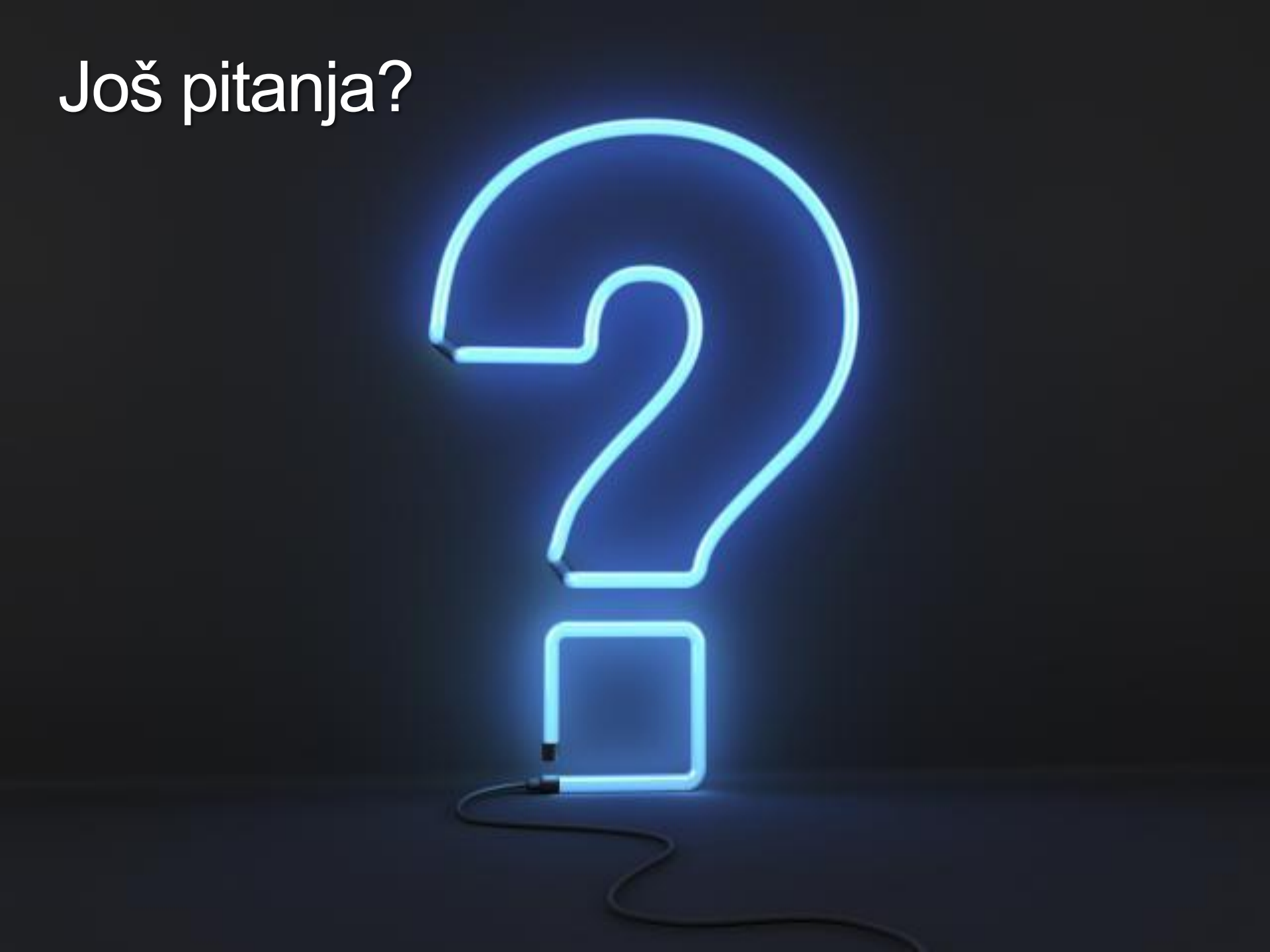
Pitanja

9. Može li brže?
   - Može - i to u $\mathcal{O}(n)$.
   - (Detalji u Kleinberg-Tardos: Algorithm Design,
     §13.7 Finding the Closest Pair of Points: A Randomized Approach)

Literatura

- Alsuwaiyel M. H. *Algorithms: Design Techniques and Analysis*. Singapore: World Scientific. 1999.
- Cormen T. H., Leiserson C. E. i Rivest R. L. *Introduction to Algorithms*. Massachusetts: The MIT Press. 2002.
- Kleinberg J. i Tardos E. *Algorithm Design*, Cornell University: Pearson Education. 2006.

Još pitanja?



Kraj dijaprojekcije.

Hvala na pažnji!

Zagreb, 24. siječnja 2017.

Prirodoslovno-matematički fakultet
Sveučilište u Zagrebu