Metoda Divide et Impera

desparte şi stăpâneşte

Sortarea rapidă – Aplicație

Statistici de ordine

Problemă

Dat un vector a de n numere şi un indice k, $1 \le k \le n$, să se determine al k-lea cel mai mic element din vector.

Statistici de ordine

A i-a statistică de ordine a unei mulțimi = al i-lea cel mai mic element.

- **Minimul** = prima statistică de ordine
- **Maximul** = a n-a statistică de ordine

Statistici de ordine

- **Mediana** = punctul de la jumătatea unei mulțimi

  = o valoare v a.î. numărul de elemente din mulțime mai mici decât v este egal cu numărul de elemente din mulțime mai mari decât v.

Statistici de ordine

- **Mediana**

Dacă n este impar, atunci mediana este a $\lceil n/2 \rceil$–a statistică de ordine, altfel, prin convenție mediana este **media aritmetică** dintre a $\lfloor n/2 \rfloor$–a statistică și a $(\lfloor n/2 \rfloor + 1)$–a statistică de ordine

1	12	15	16	38	40

15,5

- **Mediană inferioară / superioară**

Statistici de ordine – utilitate

- **Statistică**
- **Mediana** – pentru o mulţime $A=\{a_1,\ldots,a_n\}$

  valoarea μ care minimizează expresia

$$\sum_{i=1}^{n}\left|\mu - a_i\right|$$

este mediana (inf/sup)

Statistici de ordine

Idee

Al k-lea minim – folosim poziţionarea de la quicksort (**pivot aleator**)

Statistici de ordine

Fie m poziţia pivotului

- Dacă m = k, pivotul este al k-lea minim
-
-

Statistici de ordine

Fie m poziţia pivotului

- Dacă m = k, pivotul este al k-lea minim
- Dacă m > k, al k-lea minim este în stânga pivotului (al k-lea minim din stanga)
-

Statistici de ordine

Fie m poziţia pivotului

- Dacă $m = k$, pivotul este al k-lea minim
- Dacă $m > k$, al k-lea minim este în stânga pivotului (al k-lea minim din stanga)
- Dacă $m < k$, al k-lea minim este în dreapta pivotului (al $(k-m)$-lea minim din dreapta)

```
//pentru numerotare de la 0
int selKMin(int a[], int p, int u){
    int m = pozRandom(p,u);
    if(m == k-1) return a[m];
    if(m < k-1) return selKMin(a,m+1,u);
    return selKMin(a,p,m-1);
}

int selKMin(){
    return selKMin(a,0,n-1);
}
```

- AlKMinim.java

$k = 2$

10	2	6	12	4	11

poziţionare pivot

4	2	6	10	12	11

$m = 4 > k \Longrightarrow$ stânga

$k = 2$

10	2	6	12	4	11

poziţionare pivot

4	2	6	10	12	11

$m = 4 > k \Longrightarrow$ stânga

4	2	6

poziţionare pivot

2	4	6

$m = 2 = k \Longrightarrow$ stop;

pivotul este al k-lea minim

Complexitate

- **Timpul mediu**

$$\begin{cases} T(n) \le n - 1 + \dfrac{1}{n} \displaystyle\sum_{m=1}^{n} T(\max\{m-1, n-m\}) \\ \qquad \le n - 1 + \dfrac{2}{n} \displaystyle\sum_{m=\frac{n}{2}}^{n-1} T(m) \end{cases}$$

$T(n) \le cn$ – se demonstrează prin inducţie

Statistici de ordine

Algoritm O(n) caz defavorabil – TEMĂ (1p)

- v. Cormen
- mai puţin eficient în practică

Statistici de ordine

Idee

În algoritmul anterior `selKMin` **se foloseşte în loc de pivot aleatoriu un pivot calculat astfel:**

Statistici de ordine

Idee

În algoritmul anterior `selKMin` **se foloseşte în loc de pivot aleatoriu un pivot calculat astfel:**

_ se împarte vectorul în grupe de 5 (cu cel mult o excepţie – ultimul grup)

– se formează un vector `mediane`[] având ca elemente mediana fiecărui grup

Statistici de ordine

Idee

În algoritmul anterior `selKMin` se foloseşte în loc de pivot aleatoriu un pivot calculat astfel:

_ se împarte vectorul în grupe de 5 (cu cel mult o excepţie – ultimul grup)

- se formează un vector `mediane[]` având ca elemente mediana fiecărui grup
- se calculează mediana acestui vector folosind aceeaşi funcţie `selKMin(mediane,` $\left\lceil \frac{n}{5} \right\rceil$, $\left\lceil \frac{n}{10} \right\rceil$)

Statistici de ordine

- **Complexitate** – O(n)
- **Tema** – pentru grupe de cate 3 nu se mai obţine tot O(n)

Mediana a doi vectori sortaţi

Mediana a doi vectori sortați

Se dau doi vectori a și b de lungime n, cu elementele ordonate crescător. Să se determine mediana vectorului obținut prin interclasarea celor doi vectori.

Mediana a doi vectori sortați

Exemplu: $n = 5$

1	12	15	16	38
2	13	17	30	45

Mediana a doi vectori sortați

Exemplu: n = 5

1 12 15 16 38

2 13 17 30 45

⇓

1 2 12 13 15 16 17 30 38 45

Mediana (15+16)/2 = 15,5

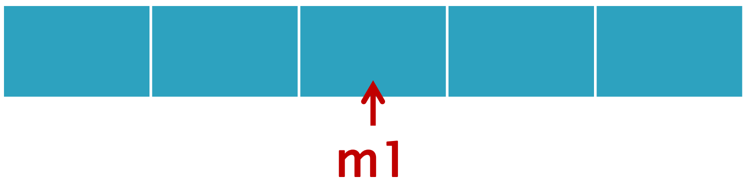
Mediana a doi vectori sortați

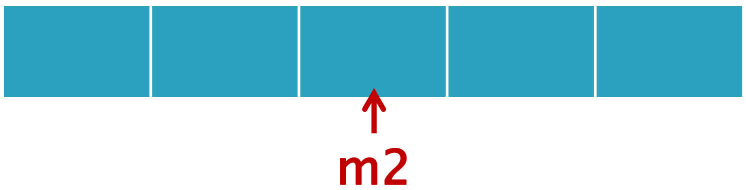
- **Algoritm O(n)** – interclasăm vectorii şi apoi aflăm mediana în timp constant (din elementele de la mijlocul vectorului, conform definiţiei)

Mediana a doi vectori sortați

- Algoritm O(log n)

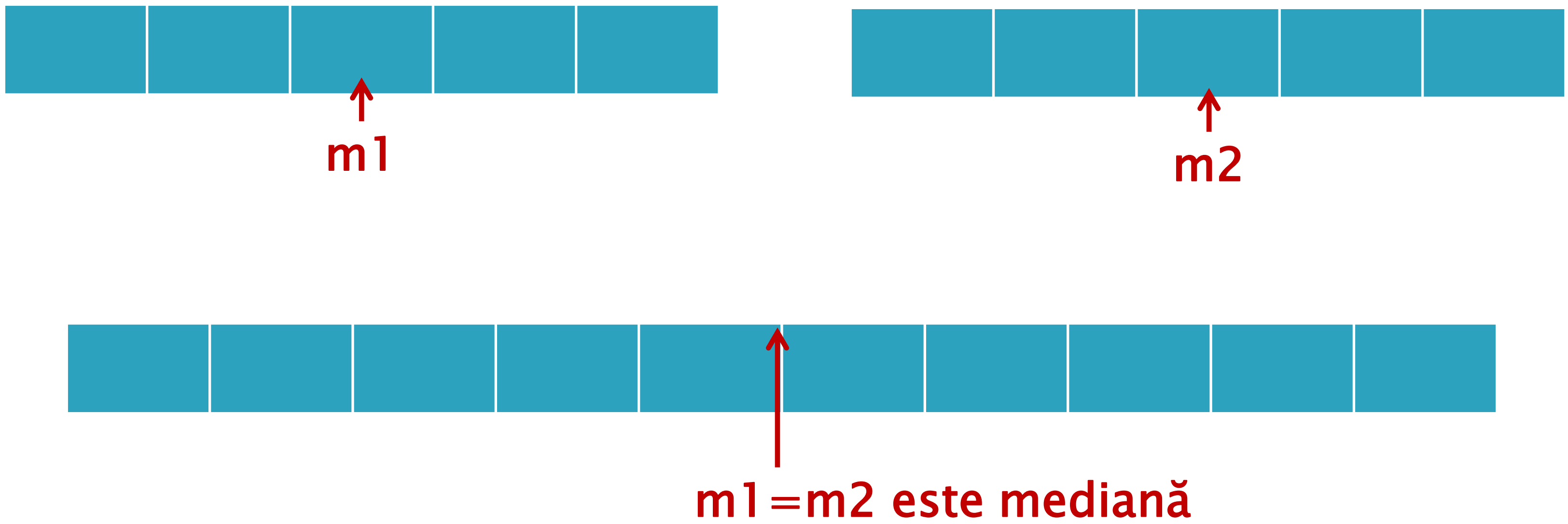
- Fie `m1` mediana vectorului `a` și `m2` mediana vectorului `b`



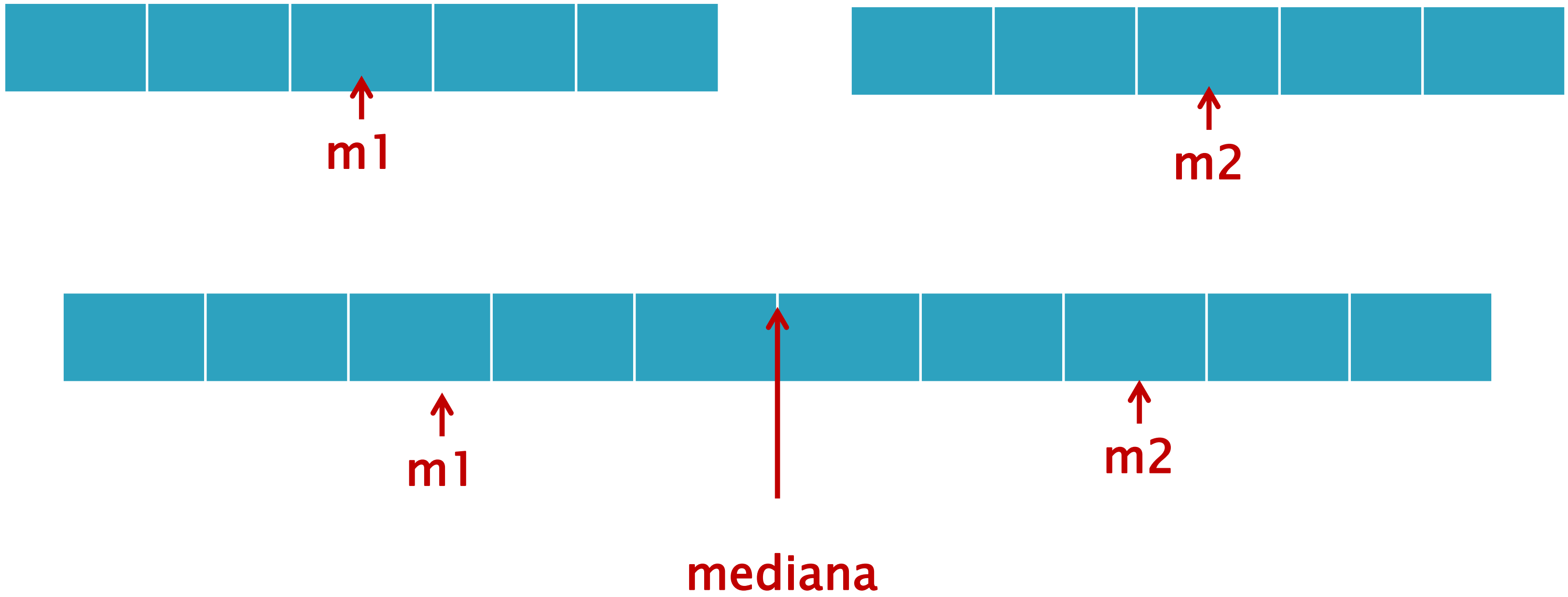


- Comparăm m1 şi m2

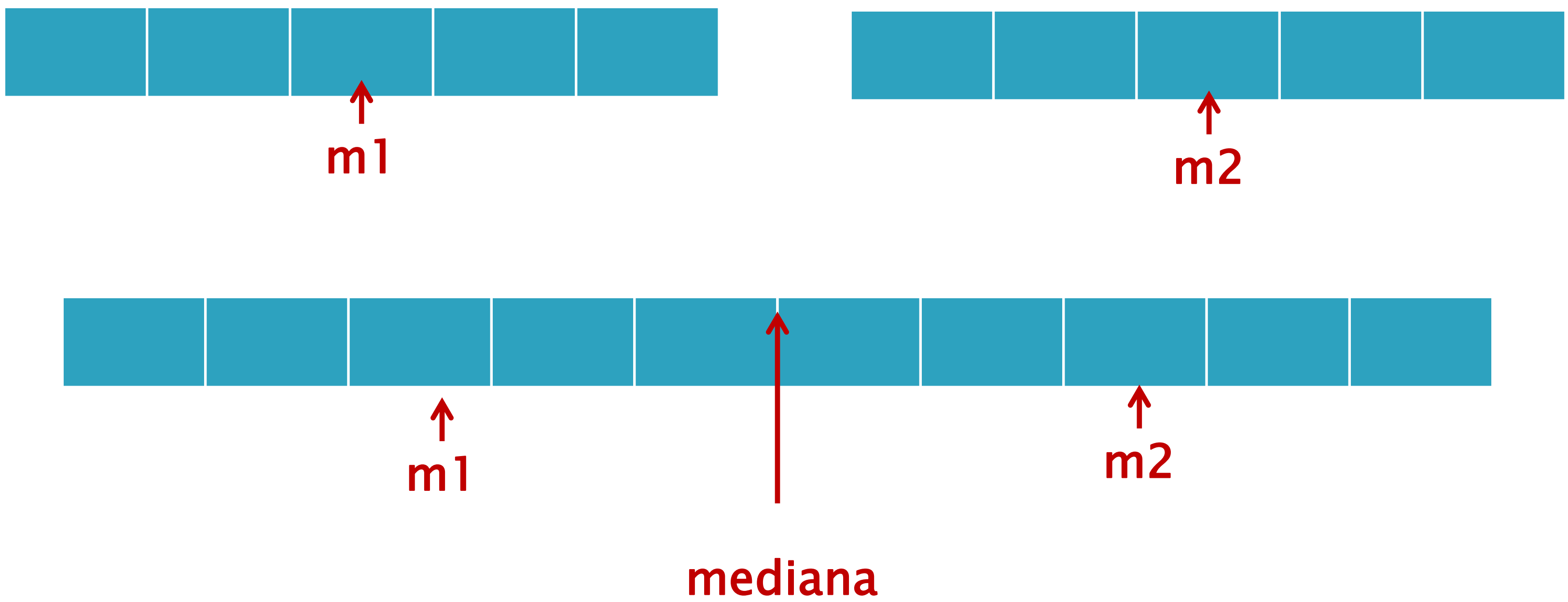
- Fie `m1` mediana vectorului `a` și `m2` mediana vectorului `b`



- Fie `m1` mediana vectorului `a` și `m2` mediana vectorului `b`



- Fie `m1` mediana vectorului `a` și `m2` mediana vectorului `b`



Pentru a determina mediana este suficient să considerăm:

- Subvectorul drept din primul vector (≥ m1)
- Subvectorul stâng din al doilea vector (≤ m2)

Mediana a doi vectori sortați

Corectitudine:

mediana noii probleme = mediana problemei inițiale ?

Pseudocod

- Fie `m1` mediana vectorului `a` și `m2` mediana vectorului `b`
  - Dacă `m1 = m2` atunci **această valoare este mediana**
  - Dacă `m1 > m2` atunci mediana se află în unul din **subvectorii**

    a[$\mathbf{0..\lfloor n/2 \rfloor}$], b[$\mathbf{\lfloor (n-1)/2 \rfloor..n-1}$]
  - Dacă `m1 < m2` atunci mediana se află în unul din **subvectorii**

    a[$\mathbf{\lfloor (n-1)/2 \rfloor..n-1}$], b[$\mathbf{0..\lfloor n/2 \rfloor}$]

1	12	15	16	38

2	13	17	30	45

1	12	15	16	38

15	16	38

2	13	17	30	45

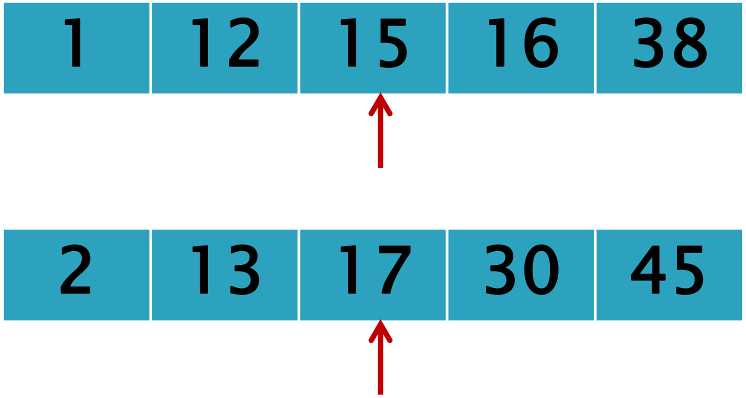
2	13	17

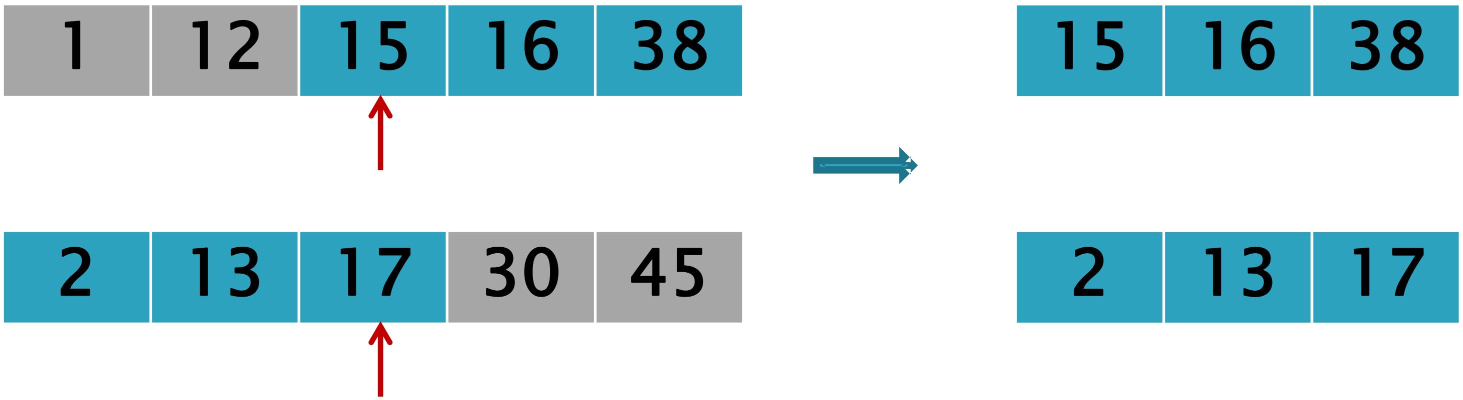
- **Ştim să rezolvăm direct:**
  - n = 1: (a[1]+b[1])/2
  - n = 2: (max{a[1],b[1]}+min{a[2],b[2]})/2

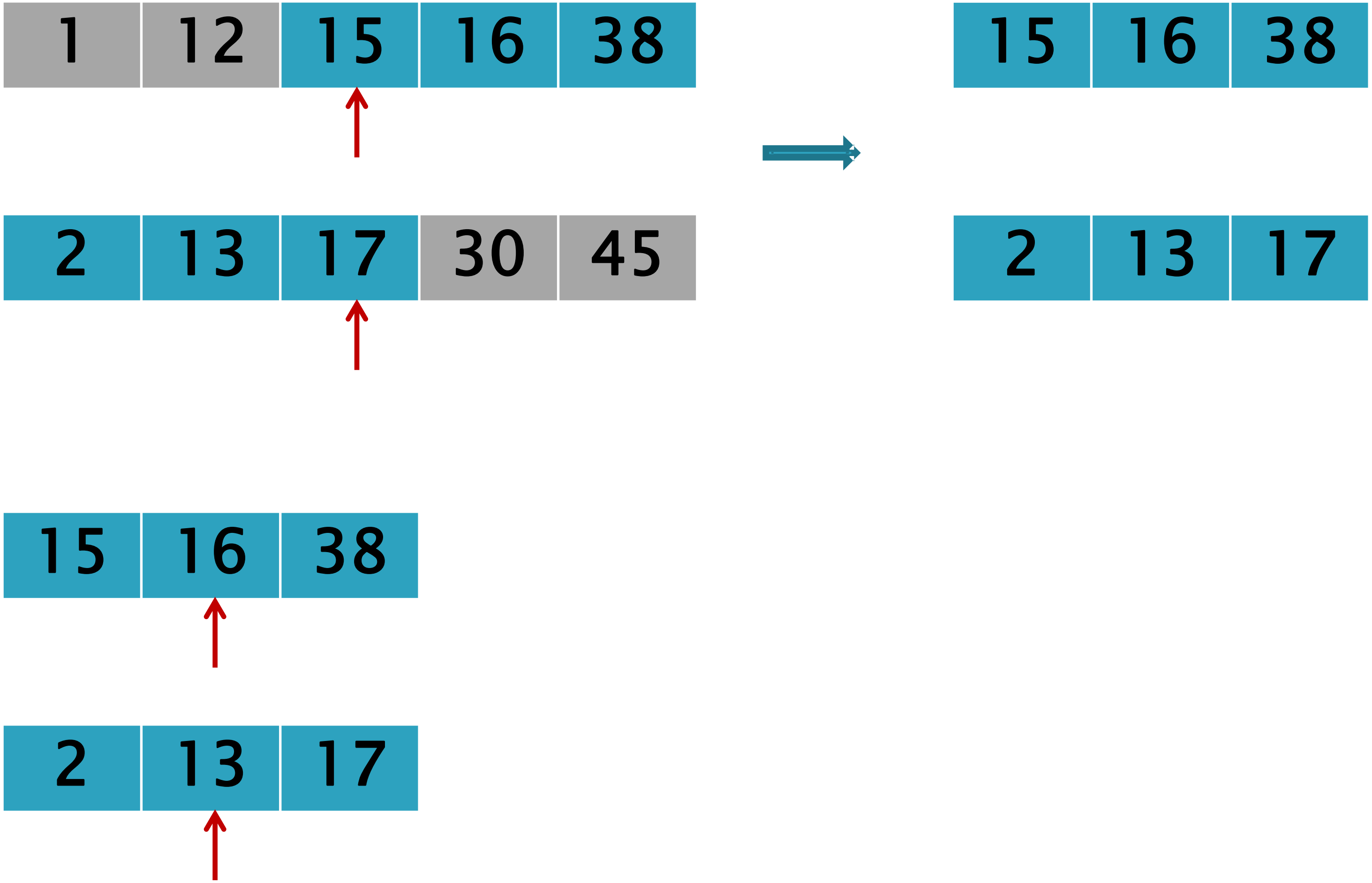
Exemplu

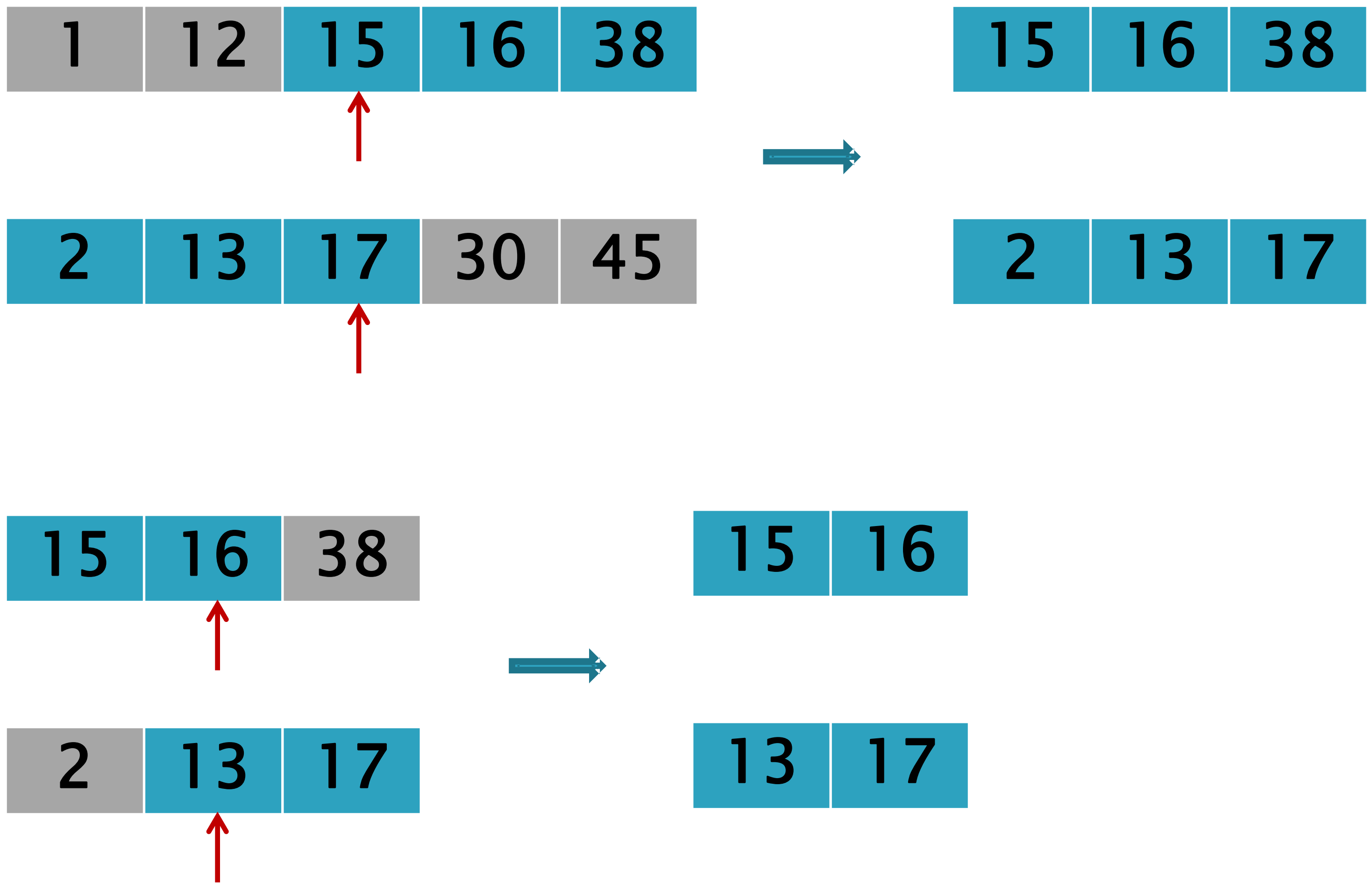
1	12	15	16	38

2	13	17	30	45

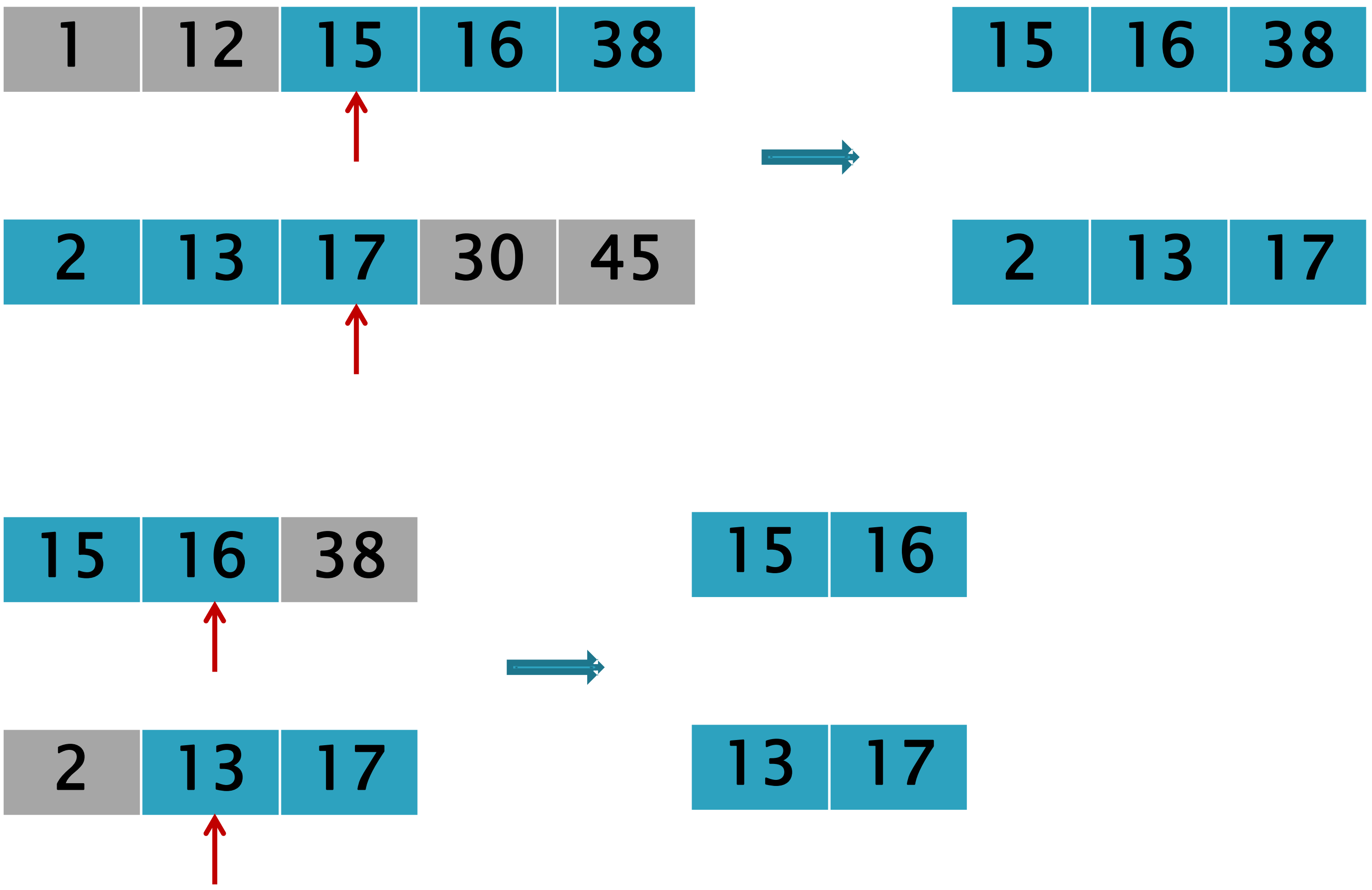

1 12 15 16 38
2 13 17 30 45


1
12
15
16
38
2
13
17
30
45
15
16
38
2
13
17


1 12 15 16 38
2 13 17 30 45
15 16 38
2 13 17
15 16 38
2 13 17



1 12 15 16 38
2 13 17 30 45
15 16 38
2 13 17
15 16 38
2 13 17
15 16
13 17



$$\text{Mediana} = \frac{\max\{13{,}15\}+\min\{16{,}17\}}{2} = \frac{15+16}{2}$$

$$= 15{,}5$$

Exemplul 2

1	12	15	16	38	40

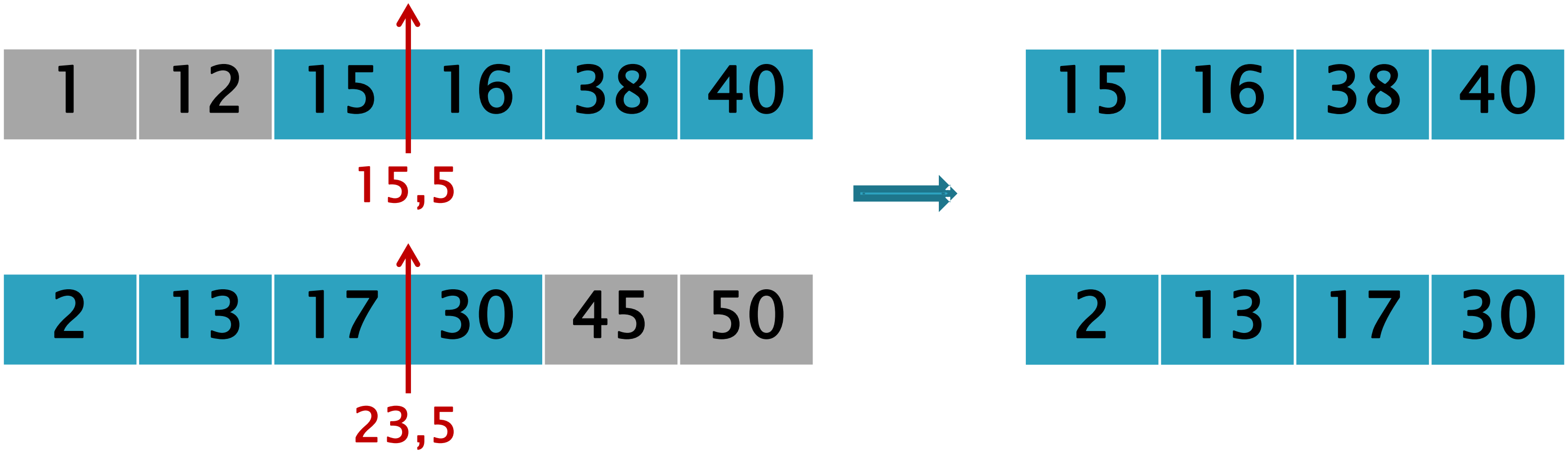
2	13	17	30	45	50

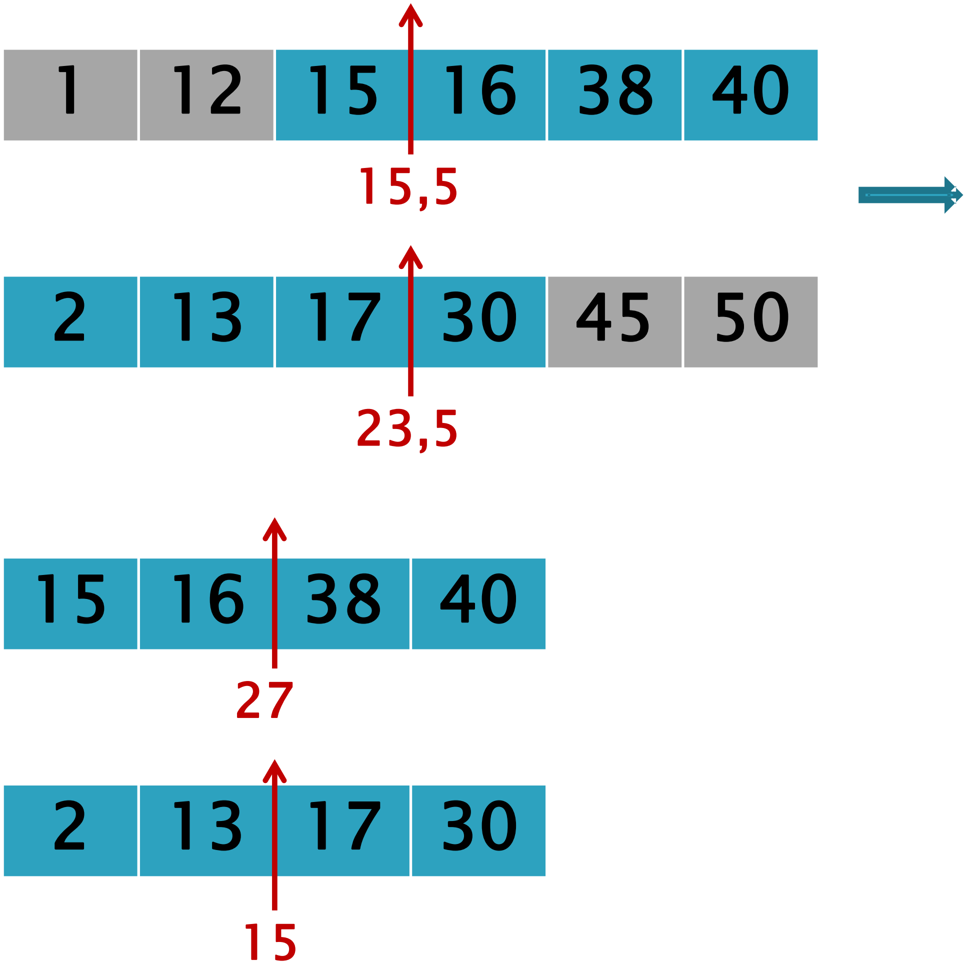
1	12	15	16	38	40

15,5

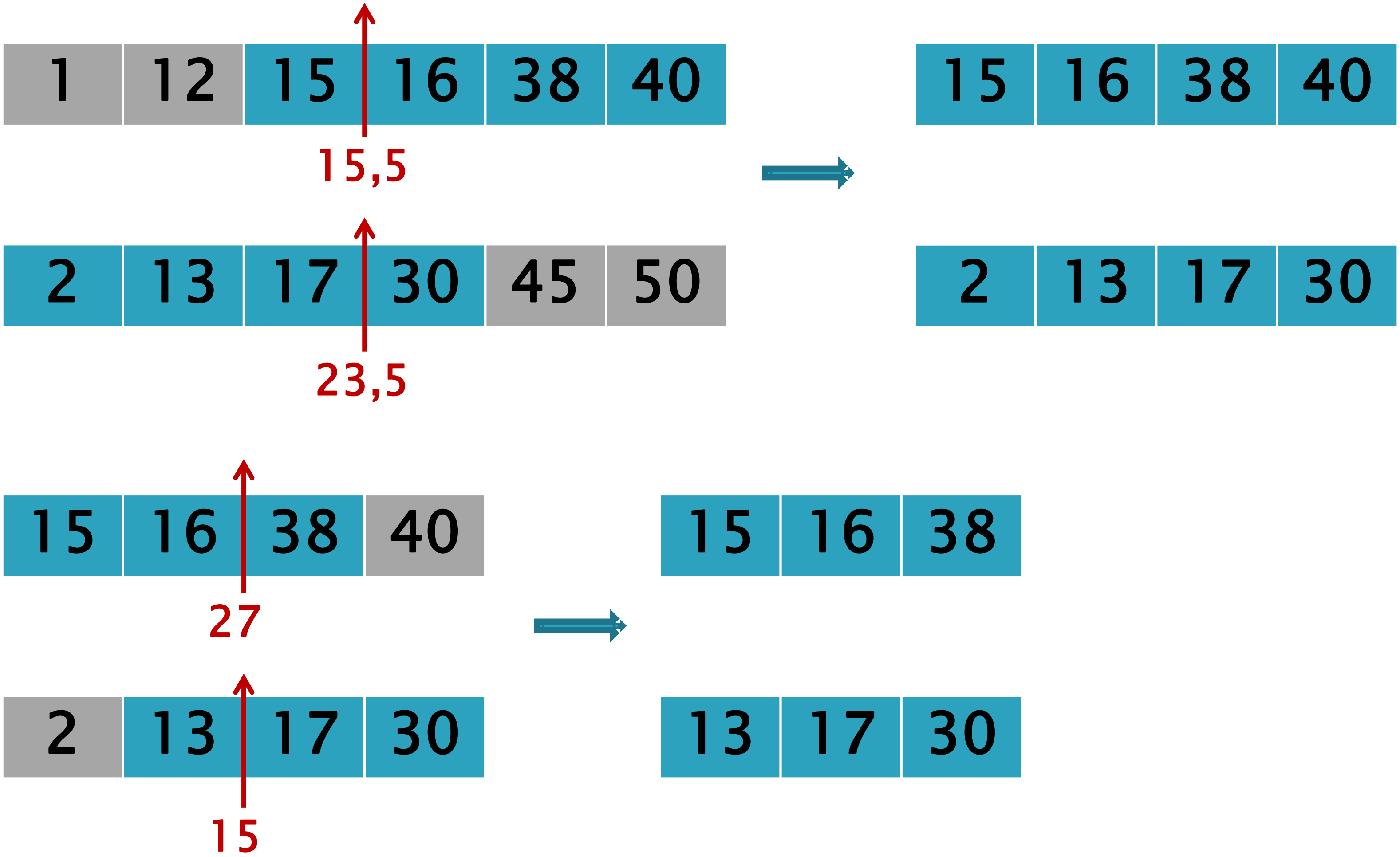
2	13	17	30	45	50

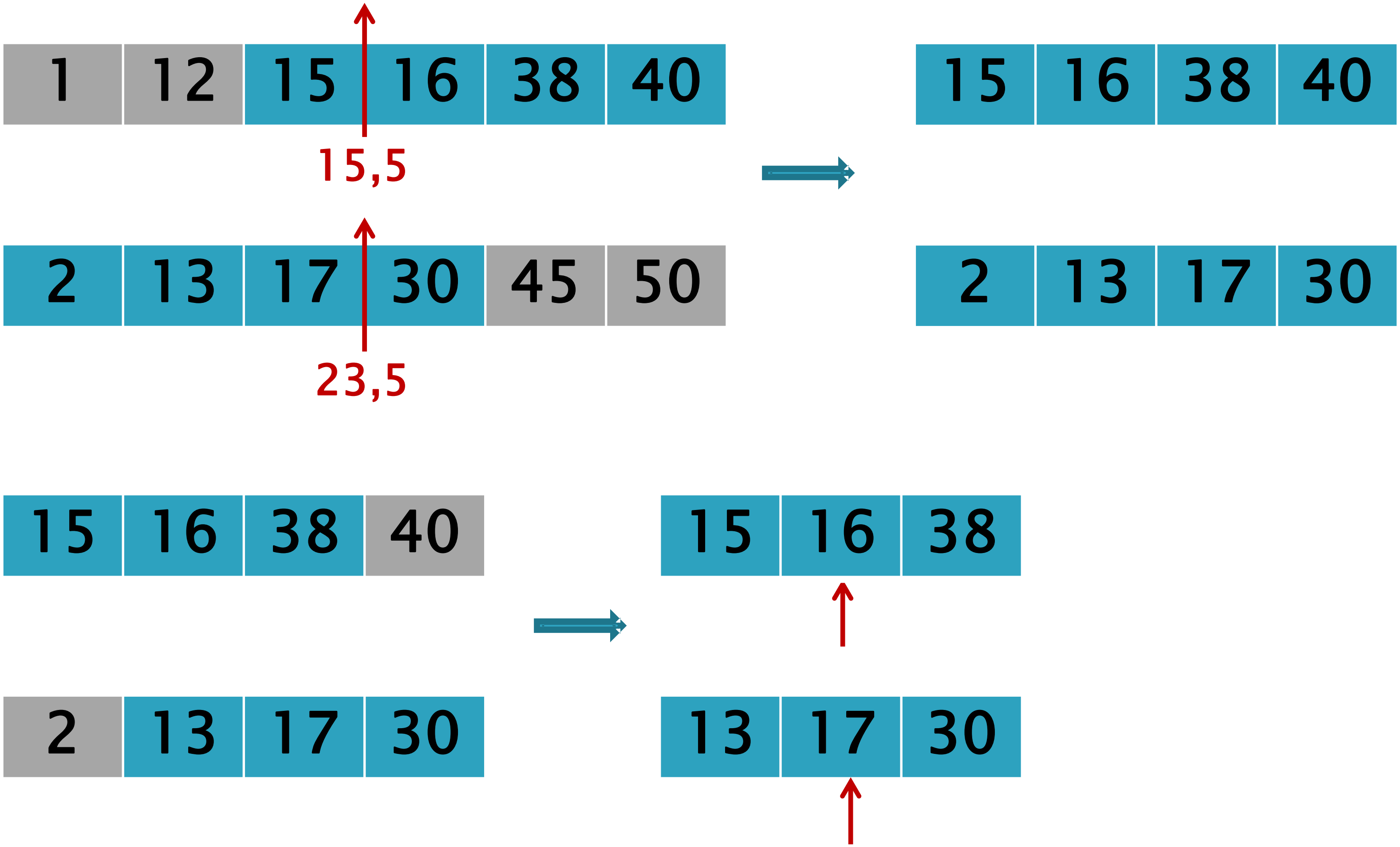
23,5

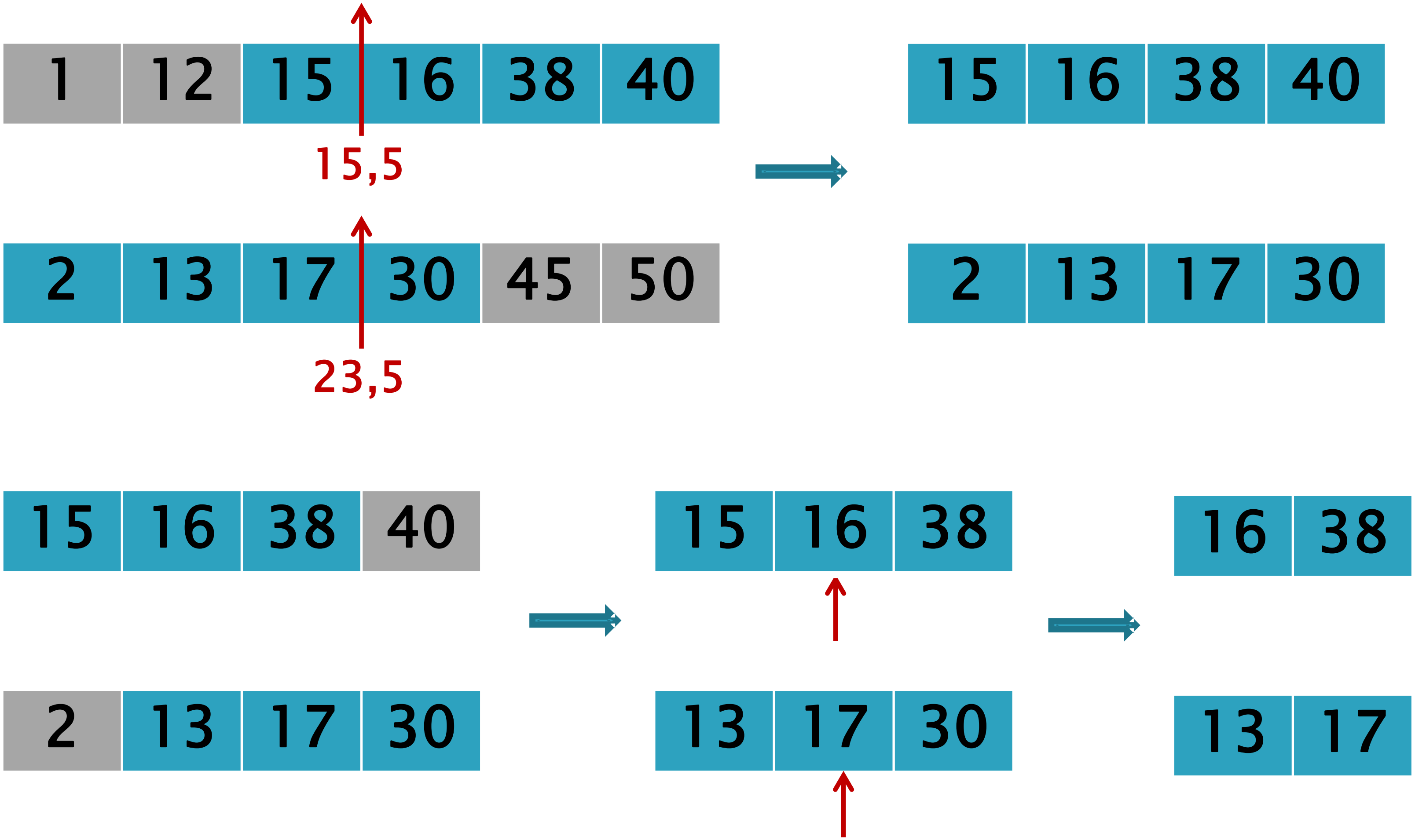

1 12 15 16 38 40
15,5
2 13 17 30 45 50
23,5
15 16 38 40
2 13 17 30

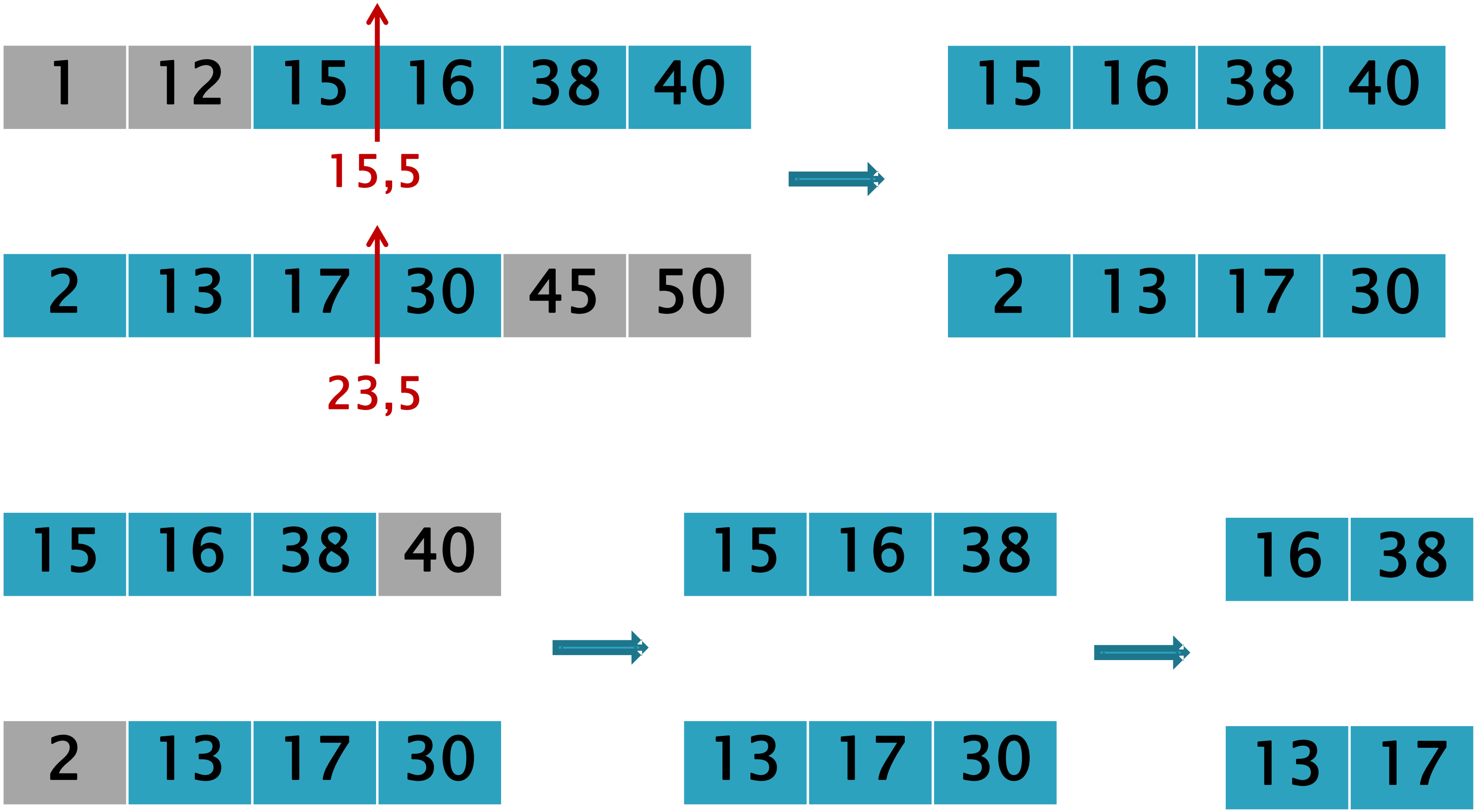


1 12 15 16 38 40
15,5
2 13 17 30 45 50
23,5
15 16 38 40
2 13 17 30
15 16 38 40
27
2 13 17 30
15


1 12 15 16 38 40
15,5
2 13 17 30 45 50
23,5
15 16 38 40
2 13 17 30
15 16 38 40
27
2 13 17 30
15
15 16 38
13 17 30


1 12 15 16 38 40
15,5
2 13 17 30 45 50
23,5
15 16 38 40
2 13 17 30
15 16 38 40
2 13 17 30
15 16 38
13 17 30


1 12 15 16 38 40
15,5
2 13 17 30 45 50
23,5
15 16 38 40
2 13 17 30
15 16 38 40
2 13 17 30
15 16 38
13 17 30
16 38
13 17



$$\textbf{Mediana} = \frac{\max\{13,16\}+\min\{17,38\}}{2} = \frac{16+17}{2}$$

$$= 16{,}5$$

Algoritm

```
double mediana(double[] v, int pv, int uv){
    int n=uv-pv+1;
    int m=(uv+pv)/2;
    if (n%2==0)
        return(v[m]+v[m+1])/2.0;
    else
        return v[m];
}
```

```
double calculMediana(int pa, int ua,int pb, int ub){
    int n = ua-pa+1;
    if (n<=2) //rezolv direct
      return (max(a[pa],b[pb])+min(a[ua],b[ub]))/2.0;

    double m1=mediana(a,pa,ua);//mediana lui a[pa..ua]
    double m2=mediana(b,pb,ub);//mediana lui b[pb..ub]
```

```
double calculMediana(int pa, int ua,int pb, int ub){
    int n = ua-pa+1;
    if (n<=2) //rezolv direct
      return (max(a[pa],b[pb])+min(a[ua],b[ub]))/2.0;

    double m1=mediana(a,pa,ua);//mediana lui a[pa..ua]
    double m2=mediana(b,pb,ub);//mediana lui b[pb..ub]

    if(m1==m2) return m1;
    if (m1>m2)
      return calculMediana(pa, pa+n/2, pb+(n-1)/2,ub);
    else
      return calculMediana(pa+(n-1)/2, ua, pb,pb+n/2);
}
```

```
double calculMediana(int pa, int ua,int pb, int ub){
    int n = ua-pa+1;
    if (n<=2) //rezolv direct
      return (max(a[pa],b[pb])+min(a[ua],b[ub]))/2.0;
    double m1=mediana(a,pa,ua);//mediana lui a[pa..ua]
    double m2=mediana(b,pb,ub);//mediana lui b[pb..ub]
    if(m1==m2) return m1;
    if (m1>m2)
      return calculMediana(pa, pa+n/2, pb+(n-1)/2,ub);
    else
      return calculMediana(pa+(n-1)/2, ua, pb,pb+n/2);
}

double calculMediana(){
     return calculMediana(0,n-1,0,n-1);
}
```

Mediana.java

Mediana a doi vectori sortați

- **Complexitate:** O(log n)

Mediana a doi vectori sortați

Mai este valabilă ideea pentru vectori de lungimi diferite (reducem problema la o problemă de acelaşi tip păstrând jumătate din fiecare vector)

Metoda 2

- **Idee:**
  - putem testa în timp constant dacă **un element fixat a[i] este (mai exact “face parte din”) mediana dorită:**

    $$b[j] \leq a[i] \leq b[j+1]$$

    pentru $j = n - i - 1$

Metoda 2

- **Idee:**
  - putem testa în timp constant dacă **un element fixat a[i] este (mai exact "face parte din") mediana dorită:**

    $$b[j] \le a[i] \le b[j+1]$$

    pentru **j = n - i - 1**

  - căutăm binar mediana în a = căutăm binar acel element a[i] cu proprietatea de mai sus
  - dacă nu o găsim căutăm binar mediana în b

1	3	4	7	11	29

i

2	8	13	17	30	45

j =n-i-1

a[i]<b[j] $\Longrightarrow$

7	11	29

i

2	8	13	17	30	45

j =n-i-1

b[j]<a[i]<b[j+1] STOP

a[i−1] = 7 < b[j] = 8

Mediana $= \dfrac{a[i]+b[j]}{2} = \dfrac{8+11}{2} =$ 9,5

Cele mai apropiate două puncte din plan

Cele mai apropiate puncte

Se dau n puncte în plan prin coordonatele lor.

Să se determine distanţa dintre cele mai apropiate două puncte.

Cele mai apropiate puncte

Se dau n puncte în plan prin coordonatele lor.

Să se determine distanţa dintre cele mai apropiate două puncte.

Aplicații:

- Geometria computațională
  - Computer vision
  - GIS – geographic information systems

Cele mai apropiate puncte

- **Variante 1**: Considerăm toate perechile de puncte

  $O(n^2)$

Cele mai apropiate puncte

- **Varianta 2:** Divide et Impera **O(n log n)**

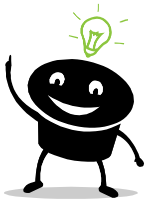
Cele mai apropiate puncte

- **Varianta 2:** Divide et Impera **O(n log n)**
- **Ipoteză**: Punctele au abscise distincte

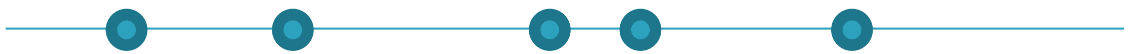
Cele mai apropiate puncte

Cum s-ar rezolva problema dacă punctele ar fi pe o dreaptă?

Cele mai apropiate puncte

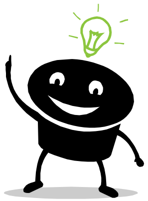


Idee: În cazul în care punctele se află pe o dreaptă, pentru a obține un algoritm mai eficient decât **O(n²)**, este utilă sortarea punctelor



Este suficient atunci să considerăm perechi de puncte adiacente **O(n log n)**

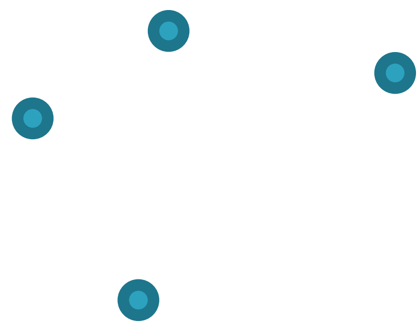
Cele mai apropiate puncte



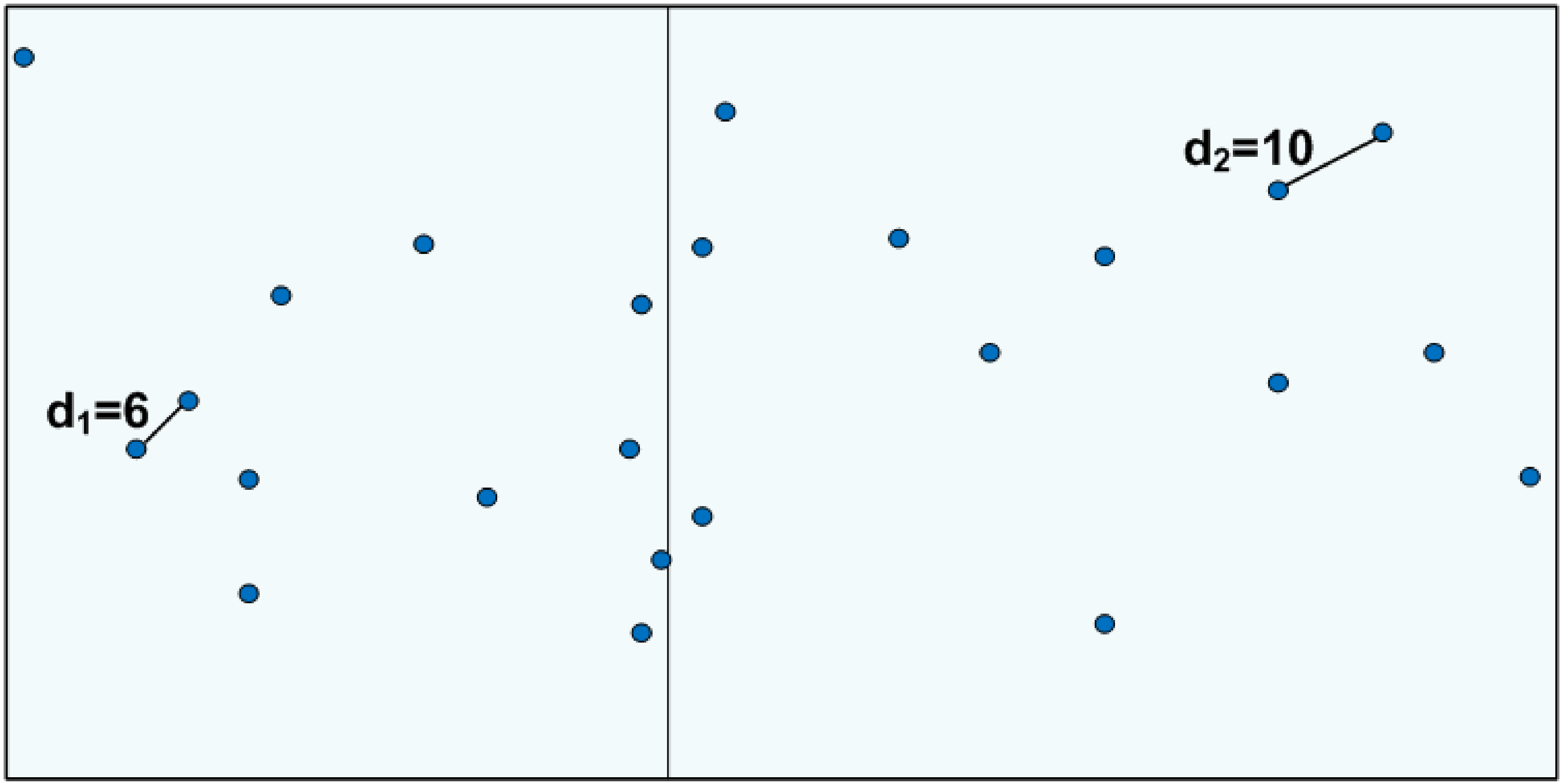
În plan ar putea fi util să sortăm vârfurile separat după abscisă și separat după ordonată, dar cele mai apropiate puncte nu sunt neapărat consecutive într-o astfel de ordonare

Cele mai apropiate puncte

În plan ar putea fi util să sortăm vârfurile separat după abscisă și separat după ordonată, dar cele mai apropiate puncte nu sunt neapărat consecutive într-o astfel de ordonare



Cele mai apropiate puncte



Cele mai apropiate puncte

- Împărţim mulţimea de puncte în două submulţimi cu n/2 puncte, **printr-o dreaptă verticală L**
-
-
-

Cele mai apropiate puncte

- Împărţim mulţimea de puncte în două submulţimi cu n/2 puncte, **printr-o dreaptă verticală L**
- Rezolvăm problema pentru cele două submulţimi şi obţinem distanţele minime $\mathbf{d_1}$, respectiv $\mathbf{d_2}$
-
-

Cele mai apropiate puncte

- Împărţim mulţimea de puncte în două submulţimi cu n/2 puncte, **printr-o dreaptă verticală L**
- Rezolvăm problema pentru cele două submulţimi şi obţinem distanţele minime $\mathbf{d_1}$, respectiv $\mathbf{d_2}$
- Determinăm distanţa minimă $\mathbf{d_3}$ între două puncte din submulţimi diferite
-

Cele mai apropiate puncte

- Împărţim mulţimea de puncte în două submulţimi cu n/2 puncte, **printr-o dreaptă verticală L**
- Rezolvăm problema pentru cele două submulţimi şi obţinem distanţele minime $\mathbf{d_1}$, respectiv $\mathbf{d_2}$
- Determinăm distanţa minimă $\mathbf{d_3}$ între două puncte din submulţimi diferite
- Returnăm minimul dintre distanţele $\mathbf{d_1}$, $\mathbf{d_2}$ şi $\mathbf{d_3}$

Cele mai apropiate puncte

Cum determinăm eficient distanţa minimă d_3 între două puncte din submulţimi diferite?

Cele mai apropiate puncte

Cum determinăm eficient distanţa minimă d_3 între două puncte din submulţimi diferite?

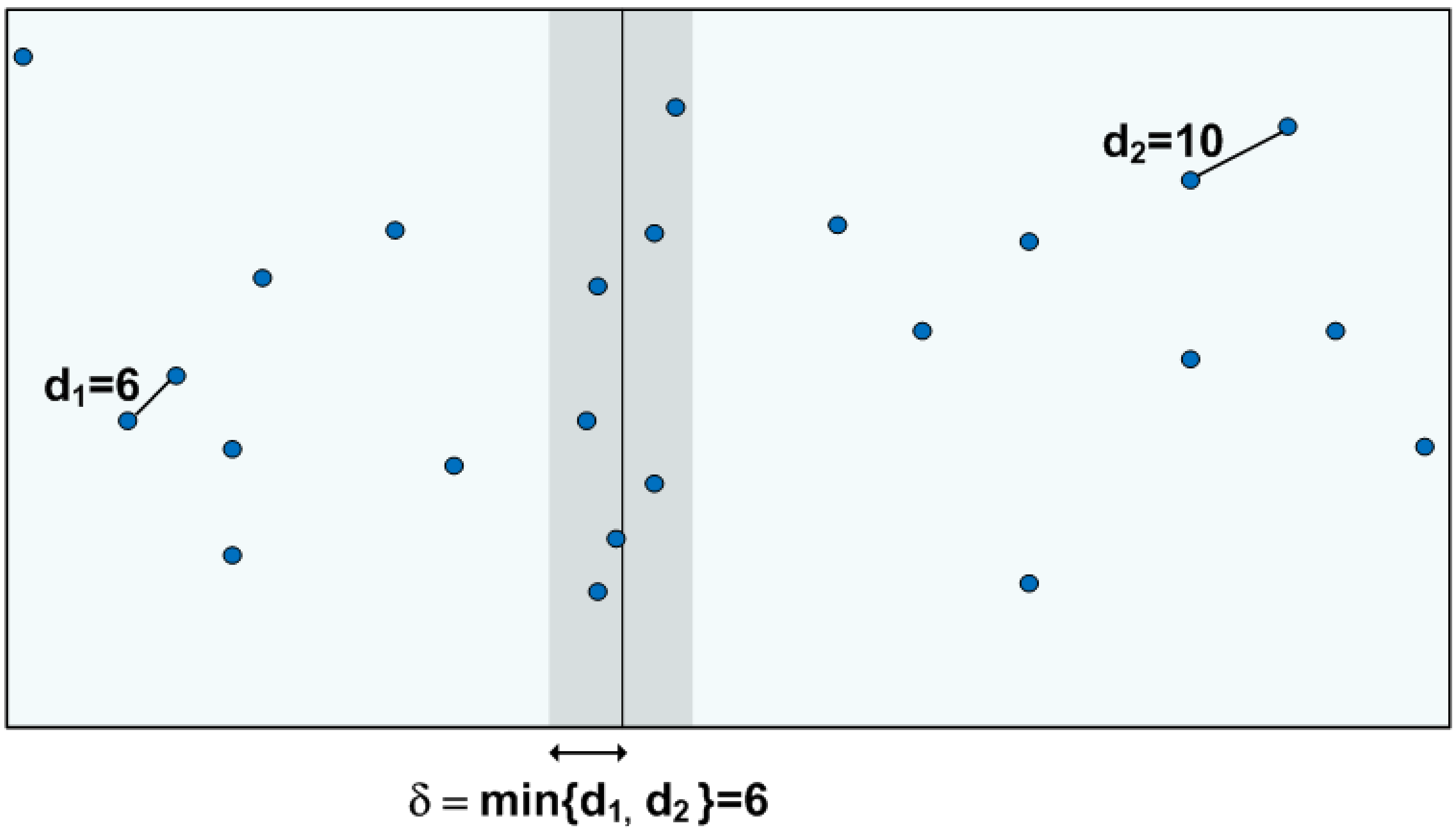
- Considerând fiecare pereche de puncte (i,j) cu i într-o submulțime și j în cealaltă - ineficient

(ca și la numărarea inversiunilor)

Cele mai apropiate puncte

Fie $\delta = \min\{d_1, d_2\}$.

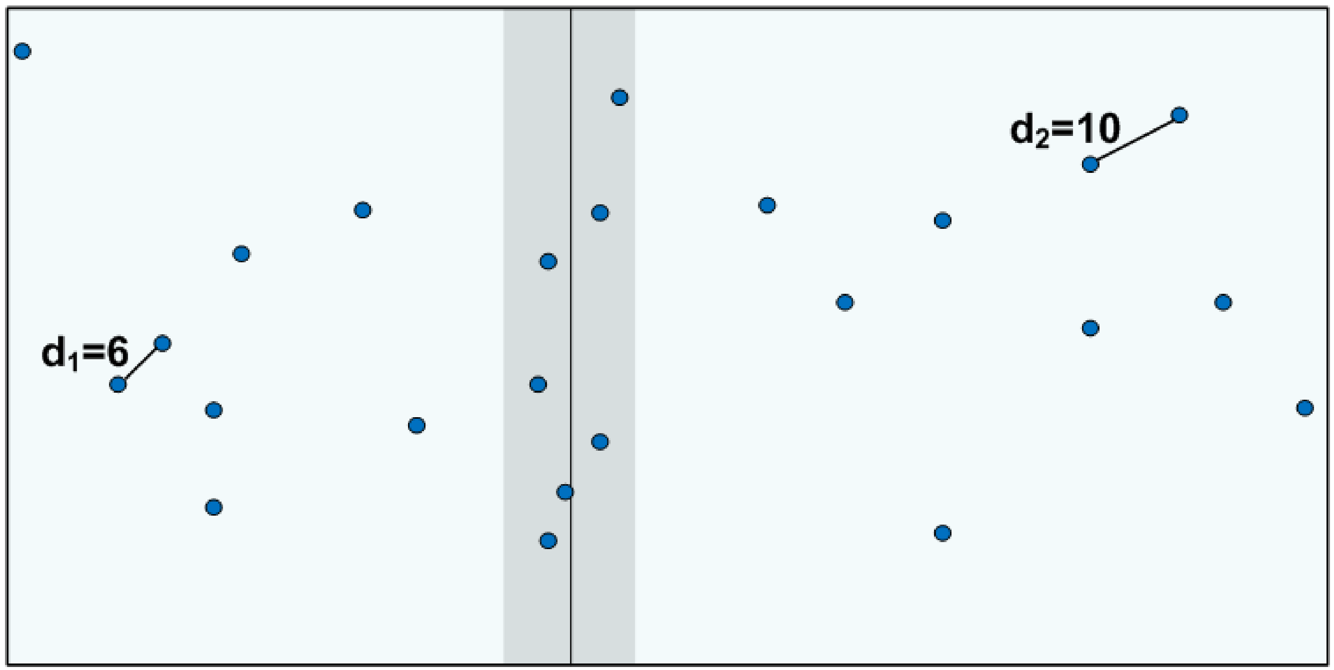
1. **Este suficient să considerăm puncte la distanţă cel mult δ de dreapta L**



Cele mai apropiate puncte

Fie $\delta = \min\{d_1, d_2\}$.

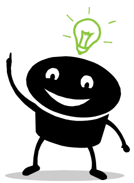
1. **Este suficient să considerăm puncte la distanţă cel mult δ de dreapta L**

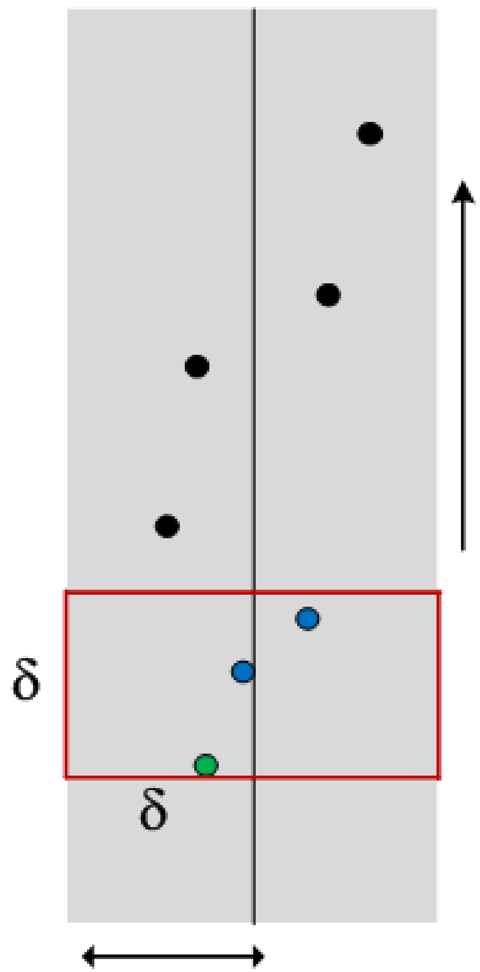


Dacă analizăm toate perechile de puncte din bandă – tot ineficient, pot fi mult puncte în bandă

Cele mai apropiate puncte

Fie $\delta = \min\{d_1, d_2\}$.



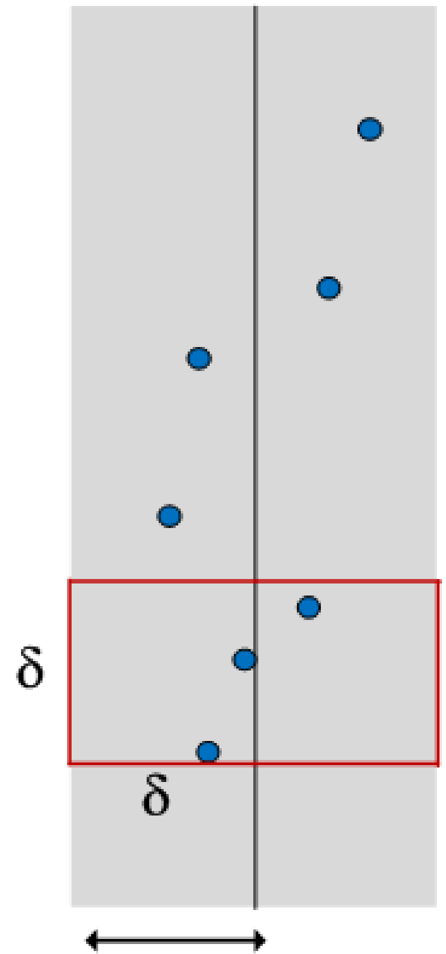


puncte la distanță mai mare ca δ de punctul verde – nu trebuie considerate în perechi cu punctul verde

Cele mai apropiate puncte

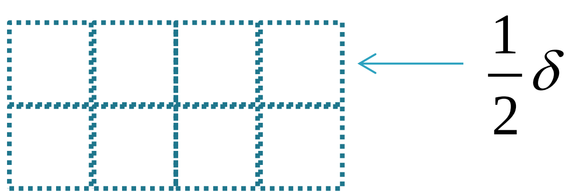
2. Două puncte din submulţimi diferite aflate la o distanţă mai mică decât δ se situează într-un dreptunghi de dimensiuni $\delta \times 2\delta$, centrat pe dreapta L

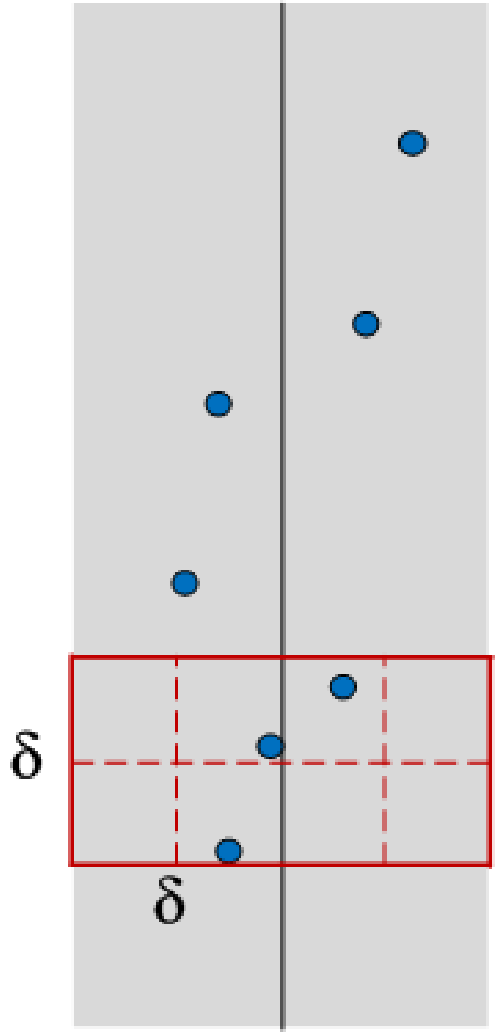
Cât de multe puncte se pot afla într-un astfel de dreptunghi?



Cele mai apropiate puncte

3. Deoarece d_1, $d_2 \geq \delta$, într-un dreptunghi de dimensiuni $\delta \times 2\delta$, centrat pe dreapta L se pot afla maxim 8 puncte



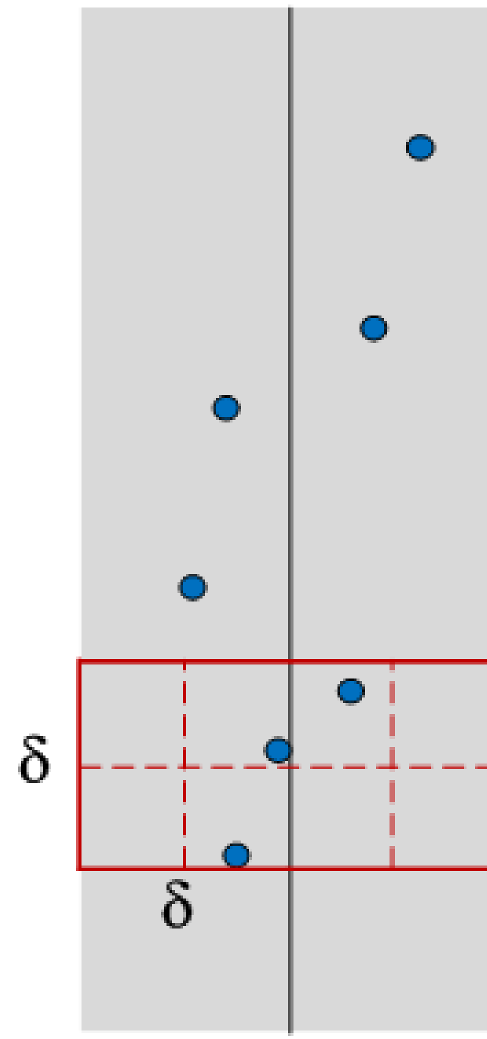


Cele mai apropiate puncte

3. Deoarece d_1, $d_2 \geq \delta$, într-un dreptunghi de dimensiuni $\delta \times 2\delta$, centrat pe dreapta L se pot afla maxim 8 puncte

$\frac{1}{2}\delta$ 

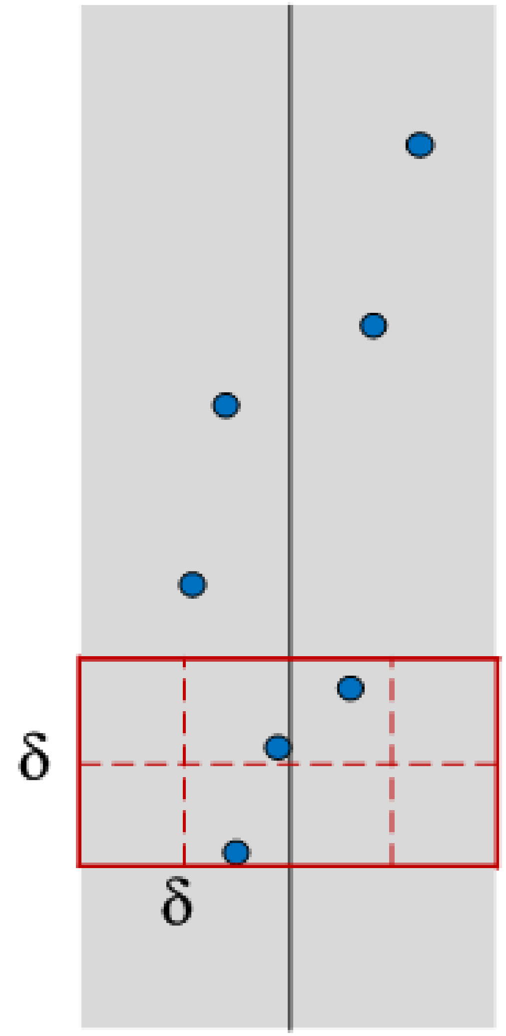
Un astfel de pătrat este inclus fie în partea stângă, fie în cea dreaptă.



Cele mai apropiate puncte

3. Deoarece d_1, $d_2 \geq \delta$, într-un dreptunghi de dimensiuni $\delta \times 2\delta$, centrat pe dreapta L se pot afla maxim 8 puncte

Consecinţă: **Pentru a calcula d_3 avem nevoie doar de 7 puncte care urmează după fiecare punct p din bandă, în şirul punctelor din bandă <u>sortate crescător după ordonată</u>**



Cele mai apropiate puncte

- Împărţim mulţimea de puncte în două submulţimi cu n/2 puncte, **printr-o dreaptă verticală L**
- Rezolvăm problema pentru cele două submulţimi şi obţinem distanţele minime $\mathbf{d_1}$, respectiv $\mathbf{d_2}$
- Determinăm distanţa minimă $\mathbf{d_3}$ între două puncte din submulţimi diferite, considerând doar puncte p din banda de lățime $\delta = \min\{\mathbf{d_1}, \mathbf{d_2}\}$ și perechile formate de p cu fiecare din cele 7 puncte din bandă care îi urmează în **ordonarea după coordonata y**
- Returnăm minimul dintre distanţele $\mathbf{d_1}$, $\mathbf{d_2}$ şi $\mathbf{d_3}$

Algoritm

Varianta 1

- X – mulţimea punctelor sortate după abscisă
- Y – mulţimea punctelor **sortate după ordonată**
- **DivImp**(st, dr, Y)

```
        //X[st..dr]
        //Y=punctele din X[st..dr] sortate dupa ordonata
```

- X – mulţimea punctelor sortate după abscisă
- Y – mulţimea punctelor **sortate după ordonată**
- **DivImp**(st, dr, Y) //X[st..dr]

```
  dacă |X|<4 atunci
     d = min(perechi de elemente din X[st..dr])
  altfel
     mij = (st+dr)/2
     //dreapta verticala trece prin X[mid].x
```

- X – mulţimea punctelor sortate după abscisă
- Y – mulţimea punctelor **sortate după ordonată**
- **DivImp**(st, dr, Y) //X[st..dr]

```
  dacă |X|<4 atunci
     d = min(perechi de elemente din X[st..dr])
  altfel
     mij = (st+dr)/2
     //dreapta verticala trece prin X[mid].x
     Calculeaza in O(n), n=dr-st+1=|Y| :
        SY= mulțimea punctelor din Y din stanga dreptei
            (cu coordonata x <= X[mid].x)
        DY= mulțimea punctelor din Y din dreapta
```

- X – mulţimea punctelor sortate după abscisă
- Y – mulţimea punctelor **sortate după ordonată**
- **DivImp**(st, dr, Y) //X[st..dr]

```
  dacă |X|<4 atunci
     d = min(perechi de elemente din X[st..dr])
  altfel
     mij = (st+dr)/2
     SY= mulțimea punctelor din Y ∩ X[st..mij] (stanga)
     DY= mulțimea punctelor din Y ∩ X[mij+1..dr] (dreapta)
     d1 = divimp(st, mij, SY) //X[st..mij]
     d2 = divimp(mij+1,dr, DY) //X[mij+1..dr]
     d = min{d1, d2}
```

- X – mulţimea punctelor sortate după abscisă
- Y – mulţimea punctelor sortate după ordonată
- **DivImp**(st, dr, Y) //X[st..dr]

```
  dacă |X|<4 atunci
     d = min(perechi de elemente din X[st..dr])
  altfel
     mij = (st+dr)/2
     SY= mulțimea punctelor din Y ∩ X[st..mij] (stanga)
     DY= mulțimea punctelor din Y ∩ X[mij+1..dr] (dreapta)
     d1 = divimp(st, mij, SY) //X[st..mij]
     d2 = divimp(mij+1, dr, DY) //X[mij+1..dr]
     d = min{d1, d2}
     LY= Y ∩ banda (cu abscisa la distanta ≤ d de X[mid].x)
```

- X – mulţimea punctelor sortate după abscisă
- Y – mulţimea punctelor **sortate după ordonată**
- **DivImp**(st, dr, Y) //X[st..dr]

```
  dacă |X|<4 atunci
    d = min(perechi de elemente din X[st..dr])
  altfel
    mij = (st+dr)/2
    SY= mulțimea punctelor din Y ∩ X[st..mij] (stanga)
    DY= mulțimea punctelor din Y ∩ X[mij+1..dr] (dreapta)
    d1 = divimp(st, mij, SY) //X[st..mij]
    d2 = divimp(mij+1, dr, DY) //X[mij+1..dr]
    d = min{d1, d2}
    LY= Y ∩ banda (cu abscisa la distanta ≤ d de X[mid].x)
    calculează d3 considerând punctele p din LY si
      perechile formate de p cu fiecare din cele 7
      puncte care îi urmează în LY
    d = min{d, d3}
    return d
```

Varianta 2

Cele mai apropiate puncte

Idee – Pentru a nu sorta de la început sau în fiecare etapă punctele crescător după ordonată se pot **interclasa** şirurile deja sortate (**!** în etapa de divide) după ordonată ale punctelor din cele două submulţimi

– Se poate folosi un singur vector

- X – mulţimea punctelor sortate după abscisă
- Y=X – (sortate tot dupa abscisa)
- **DivImp**(&X, &Y, st, dr)

```
    // X[st..dr]
    // Y -devin sortate după ordonată
    //Obs: X[st..dr]=Y[st..dr] ca multimi de puncte
```

- X – mulţimea punctelor sortate după abscisă
- Y=X – (sortate tot dupa abscisa)
- **DivImp**(&X, &Y, st, dr)**//Y -devin sortate după ordonată**

```
  dacă |X|<4 atunci
     sorteaza dupa ordonata (Y ,st, dr)
     d = min(perechi de elemente din X[st..dr])
```

- X – mulţimea punctelor sortate după abscisă
- Y=X
- **DivImp**(&X, &Y, st, dr)

```
  dacă |X|<4 atunci
    sorteaza dupa ordonata (Y ,st, dr)
    d = min(perechi de elemente din X[st..dr])
  altfel
    mij = (st+dr)/2
    d1 = divimp(X, Y, st, mij)
    d2 = divimp(X, Y, mij+1,dr)
    d = min{d1, d2}
```

- X – mulţimea punctelor sortate după abscisă
- Y=X
- **DivImp**(&X, &Y, st, dr)

```
  dacă |X|<4 atunci
    sorteaza dupa ordonata (Y ,st, dr)
    d = min(perechi de elemente din X[st..dr])
  altfel
    mij = (st+dr)/2
    d1 = divimp(X, Y, st, mij)
    d2 = divimp(X, Y, mij+1,dr)
    d = min{d1, d2}
    interclaseaza(Y, st, mij, dr) //sortare pe Oy
```

- X - mulţimea punctelor sortate după abscisă
- Y=X
- **DivImp**(&X, &Y, st, dr)

```
  dacă |X|<4 atunci
     sorteaza dupa ordonata (Y ,st, dr)
     d = min(perechi de elemente din X[st..dr])
  altfel
     mij = (st+dr)/2
     d1 = divimp(X, Y, st, mij)
     d2 = divimp(X, Y, mij+1,dr)
     d = min{d1, d2}
     interclaseaza(Y, st, mij, dr)
     LY = Y ∩ banda (cu abscisa la distanta ≤ d de X[mid].x)
     calculează d3 considerând punctele p din LY si
       perechile formate de p cu fiecare din cele 7
       puncte care îi urmează în LY
     d = min{d, d3}
     return d
```

Cele mai apropiate puncte

- **Complexitate:** $O(n \log n)$

$$T(n) = 2T(n/2)+cn, \text{ pentru } n>1$$

Cele mai apropiate puncte

- **Unde a intervenit ipoteza** “Punctele au abscise distincte”

- Se poate renunţa la ea?

Înmulțirea a două numere

Înmulțirea a două numere

Se dau două numere cu n cifre, x și y.

Propuneți un algoritm pentru calculul produsului xy cu un număr cât mai mic de operații elementare (adunări și înmulțiri de cifre)

Înmulțirea a două numere

- **Algoritmul clasic**

$$\begin{array}{r} 923 \ \times \\ 512 \\ \hline 1846 \\ 923 \\ 4615 \\ \hline 472576 \end{array}$$

n

$\leq 2n$

$O(n^2)$

Înmulțirea a două numere

- Divide et Impera
  - $x = a \cdot 10^{n/2} + b$
  - $y = c \cdot 10^{n/2} + d$
  - $xy = ac \cdot 10^{n} + (ad + bc)\ 10^{n/2} + bd$

Înmulțirea a două numere

- Divide et Impera
  - $\mathbf{x = a \cdot 10^{n/2} + b}$
  - $\mathbf{y = c \cdot 10^{n/2} + d}$
  - $\mathbf{xy = ac \cdot 10^{n} + (ad + bc)\ 10^{n/2} + bd}$

4 subprobleme de același tip

$$T(n) = 4T(n/2) + cn = ?$$

Înmulțirea a două numere

- Divide et Impera
  - $x = a \cdot 10^{n/2} + b$
  - $y = c \cdot 10^{n/2} + d$
  - $xy = ac \cdot 10^{n} + (ad + bc)\ 10^{n/2} + bd$

4 subprobleme de același tip

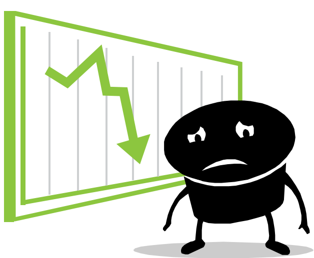
$$T(n) = 4T(n/2) + cn = O(n^2)$$

Înmulțirea a două numere

- Divide et Impera
  - $x = a \cdot 10^{n/2} + b$
  - $y = c \cdot 10^{n/2} + d$
  - $xy = ac \cdot 10^{n} + (ad + bc)\ 10^{n/2} + bd$

4 subprobleme de același tip

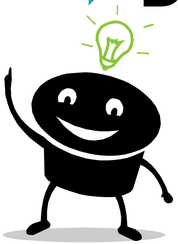
$$T(n) = 4T(n/2) + cn = O(n^2)$$



tot $O(n^2)$

Înmulțirea a două numere

- Divide et Impera – Algoritmul lui <u>KARATSUBA</u>



  - **Idee:** Încercăm să reducem la mai puține subprobleme

Calculăm produsul

p = (a+b)(c+d) = ac + ad + bc + bd

Avem

- **ad + bc = p – ac – bd**

Înmulțirea a două numere

- **Divide et Impera**

  Amintim

  - $xy = ac \cdot 10^n + (ad + bc)\ 10^{n/2} + bd$

  Este suficient să calculăm

  1. $p = (a+b)(c+d)$
  2. ac
  3. bd

Înmulțirea a două numere

- **Divide et Impera**

  Amintim

  - $xy = ac \cdot 10^n + (ad + bc)\ 10^{n/2} + bd$

  Este suficient să calculăm

  1. $p = (a+b)(c+d)$
  2. ac
  3. bd

  Combinarea rezultatelor: **O(n)**

  $xy = ac \cdot 10^n + (p-ac-bd)\ 10^{n/2} + bd$

Înmulțirea a două numere

- **Divide et Impera**

  Amintim

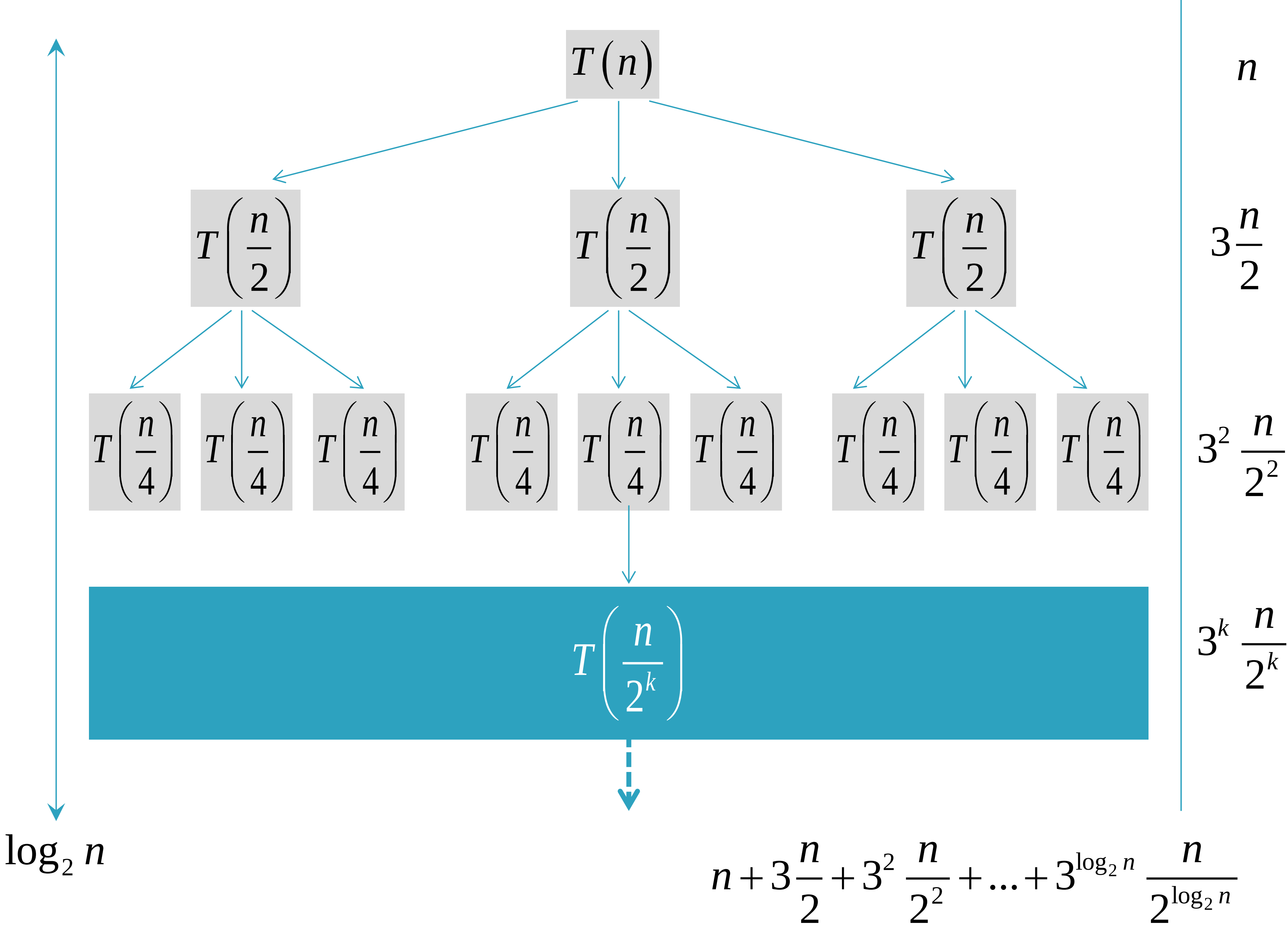
  - $xy = ac \cdot 10^n + (ad + bc)\ 10^{n/2} + bd$

  Este suficient să calculăm

  1. $p = (a+b)(c+d)$
  2. ac
  3. bd

$$T(n) = 3T(n/2) + cn \quad \Rightarrow O(n^{1,59})$$

Justificare:



$T(n)$
$T\left(\frac{n}{2}\right)$
$T\left(\frac{n}{4}\right)$
$T\left(\frac{n}{2^k}\right)$
n
$3\frac{n}{2}$
$3^2\frac{n}{2^2}$
$3^k\frac{n}{2^k}$
$\log_2 n$
$n+3\frac{n}{2}+3^2\frac{n}{2^2}+\ldots+3^{\log_2 n}\frac{n}{2^{\log_2 n}}$

- $$n + 3\frac{n}{2} + 3^2\frac{n}{2^2} + \ldots + 3^{\log_2 n}\frac{n}{2^{\log_2 n}} = n\sum_{k=0}^{\log_2 n}\left(\frac{3}{2}\right)^k$$
- $$O\left(n\left(\frac{3}{2}\right)^{\log_2 n}\right)$$
- $$\left(\frac{3}{2}\right)^{\log_2 n} = \frac{3^{\log_2 n}}{n}$$
- $$T(n) = O(3^{\log_2 n}) = O(n^{\log_2 3}), \quad \log_2 3 \simeq 1,59$$

Înmulțirea a două matrice pătratice

Înmulțirea a două matrice

Se dau două matrice pătratice de dimensiune n, X și Y. Propuneți un algoritm pentru calculul produsului XY cu un număr cât mai mic de operații elementare (adunări și înmulțiri de numere)

Înmulțirea a două matrice

- **Algoritmul clasic – $O(n^3)$**

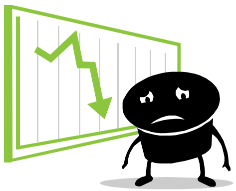
Înmulțirea a două matrice

- **Divide et Impera**
  - $$X = \left(\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right) \qquad Y = \left(\begin{array}{c|c} E & F \\ \hline G & H \end{array}\right)$$
  - $$XY = \left(\begin{array}{c|c} AE + BC & AF + BH \\ \hline CE + DG & CF + DH \end{array}\right)$$

Înmulțirea a două matrice

- Divide et Impera
  - $$X = \left(\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right) \qquad Y = \left(\begin{array}{c|c} E & F \\ \hline G & H \end{array}\right)$$
  - $$XY = \left(\begin{array}{c|c} AE + BC & AF + BH \\ \hline CE + DG & CF + DH \end{array}\right)$$

$$\mathtt{T(n)} = \mathtt{8T(n/2)} + \mathtt{cn}^2 = \mathtt{O(n}^3\mathtt{)}$$



Înmulțirea a două MATRICE

- **Divide et Impera – STRASSEN**
  - **Idee:** Încercăm să reducem la mai puține subprobleme

Calculăm produsele

- $p_1 = A(F-H)$
- $p_2 = (A+B)H$
- $p_3 = (C+D)E$
- $p_4 = D(G-E)$
- $p_5 = (A+D)(E+H)$
- $p_6 = (B-D)(G+H)$
- $p_7 = (A-C)(E+F)$

Înmulțirea a două matrice

$$XY = \left(\begin{array}{c|c} p_5 + p_4 - p_2 + p_6 & p_1 + p_2 \\ \hline p_3 + p_4 & p_1 + p_5 - p_3 - p_7 \end{array} \right)$$

T(n) = 7T(n/2) + cn² = O(n³)

$$T(n) = O(n^{\log_2 7}), \quad \log_2 7 \cong 2{,}8074$$

Divide et Impera

- aplicativitate în calculul paralel