TAP
Curs 2: DIVIDE ET IMPERA

Tehnici avansate de programare

Lect.dr. Iulia Banu
Departamentul de Informatică,
Universitatea din București

semestrul 1, 2017

Rezumat curs Divide et Impera

1. Prezentare generală
2. Algortimul general
3. Teorema master
4. Exemple și aplicații
   - Cautare binară
     - Cautare intr-o matrice sortata
   - Mediana a doi vectori sortati
   - Sortarea prin interclasare
     - Numărul de inversiuni dintr-un vector
   - Quicksort
     - Mediana unui vector, al k-lea minim
   - Cea mai apropiata pereche de puncte

Prezentare generală

Constă în:

- **împărțirea** problemei inițiale în subprobleme de **același tip**;
- **rezolvarea** subproblemelor (prin aceeași metodă sau direct);
- **combinarea** rezultatelor obținute pentru a determina rezultatul problemei inițiale.

Prezentare generală

- Subproblemele sunt rezolvate similar, prin împărțirea în subprobleme
  $\Rightarrow$ **caracter recursiv**.
- Subprobleme independente
  $\Rightarrow$ **aplicativitate în calcul paralel**.
- Analiza complexității, analiza relațiilor de recurență
  $\Rightarrow$**substitutie, teorema master**.

Prezentare generală - Reprezentare ca arbore

- Putem reprezenta **subproblemele** ca **noduri într-un arbore**, construit dinamic.
- Problema inițială este **rădăcina arborelui**.
- O subproblemă are ca **fii**, subproblemele în care se împarte pentru a fi rezolvată, **frunzele** sunt problemele care se rezolvă direct, fără a fi reîmpărțite în subprobleme.
- Rezolvarea constă în **parcurgerea în postordine** a arborelui.

Algoritmul general

$$
\begin{array}{l}
\textit{function}\ \ DivImp(p,u) \\
\qquad \textit{if}\ \ (u-p) < \epsilon \\
\qquad\qquad r \leftarrow Rezolva(p,u) \\
\qquad \textit{else} \\
\qquad\qquad m \leftarrow Interm(p,u); \\
\qquad\qquad r1 \leftarrow DivImp(p,m); \\
\qquad\qquad r2 \leftarrow DivImp(m+1,u); \\
\qquad\qquad r \leftarrow Combina(r1,r2) \\
\qquad \textit{return}\ \ r \\
\textit{end}
\end{array}
$$

Teorema master

Fie T(n) o funcție monotonă care satisface relația de recurență:

$$T(n) = aT(n/b) + f(n)$$

$a >= 1, b > 1$ constante și

$f(n) \in \Theta(n^d)$ cu $d >= 0$. Atunci:

$$T(n) = \Theta(n^d) \text{ daca } a < b^d;$$

$$T(n) = \Theta(n^d \ logn) \text{ daca } a = b^d;$$

$$T(n) = \Theta(n^{log_b a}) \text{ daca } a > b^d;$$

Algoritmul de căutare binară

Recurența: $\mathbf{T(n) = T(n/2) + c}$, unde c este o constantă.

Se consideră un vector sortat crescător $a = (a_0, a_1, \ldots, a_n, a_{n+1})$ cu $a_0 = -\infty$ si $a_n = \infty$ și o valoare x.

Să se determine

- indicele i din vectorul la care este memorată valoarea x sau
- intervalul $[a_{i-1}, a_i]$ care îl conține pe x: $a_{i-1} < x < a_i$.

Algoritmul se reduce la o singură subproblemă.

Algoritmul de căutare binară

Rezolvarea recurenței: $\mathbf{T(n) = T(n/2) + c}$

Complexitate: **O(log n)**

$$
\begin{aligned}
T(n) = {} & T(n/2) + c = \\
& = [T(n/4) + c] + c = T(n/2^2) + 2c \\
& \dots \\
& = T(n/2^k) + kc
\end{aligned}
$$

$k = log_2 n$

Algoritmul de căutare binară, aplicații

- Se consideră un vector $a = (a_1, \ldots, a_{2p+1})$ care conține p + 1 numere distincte, oricare două elemente egale sunt situate pe poziții consecutive. Să se găsească valoarea x care apare o singură dată în vector.

Algoritmul de căutare binară, aplicații

- Se consideră un vector $a = (a_1, \ldots, a_{2p+1})$
  care conține p $+$ 1 numere distincte, oricare două elemente egale sunt situate pe poziții consecutive. Să se găsească valoarea x care apare o singură dată în vector.

- Se consideră vectorul $a = (a_1, \ldots, a_n)$ obținut dintr-un vector sortat prin mutarea circulară a primelor k poziții pe pozițiile
  $n - k + 1, \ldots, n$. Să se găsească maximul în vectorul a.

Algoritmul de căutare binară, aplicații

- Se consideră un vector $a = (a_1, \dots, a_{2p+1})$ care conține p + 1 numere distincte, oricare două elemente egale sunt situate pe poziții consecutive. Să se găsească valoarea x care apare o singură dată în vector.

- Se consideră vectorul $a = (a_1, \dots, a_n)$ obținut dintr-un vector sortat prin mutarea circulară a primelor k poziții pe pozițiile $n - k + 1, \dots, n$. Să se găsească maximul în vectorul a.

- Se considera o functie strict descrescătoare $f : \mathbb{N} \to \mathbb{Z}$. Să se găsească prima valoare n pentru care $f(n) < 0$. Presupunem că timpul în care este calculată valoarea funcției într-un punct este constant.

Cautare intr-o matrice sortata

Se considera o matrice A in care toate linii, respectiv toate coloanele sunt sortate crescator.
Sa se determine daca valoare x se afla intre valorile matricei A.

Solutii:

- căutare binară pe linii/coloane O(l*log(c)) în cazul în care numărul de linii este mai mic decât numărul de coloane;
- strategie divide et impera, comparam elementul cautat cu valoarea din mijlocul matricei: $T(n*m) = 1 + 3*T(n*m/4)$.
  $O(n*m^{log_4 3})$: se aplica teorema master, cazul al 3-lea;
- eliminarea succesiva a unei linii sau a unei coloane $O(n + m)$.
- Optim: O(l*log(c/l))

Mediana a doi vectori sortati

Problema: Se dau doi vectori a, b de dimensiuni egale *n*. Sa se obtina mediana vectorului obtinut prin interclasarea celor doi vectori.

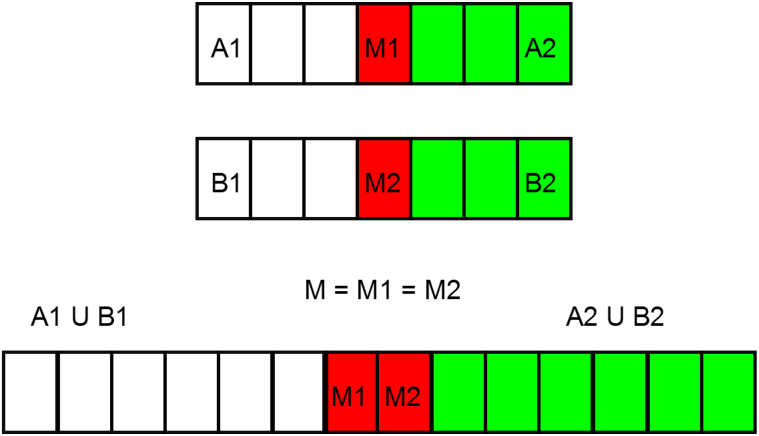
Soluția 1: Se interclasează vectorii și se obține mediana în timp constant. Complexitate **O(n)**

Exemplul: n = 5
a = 14 17 20 22 30
b = 12 31 40 42 45
v = 12 14 17 20 22 30 31 40 42 45 vectorul obținut prin interclasare
Mediana m = (22 + 30)/2 = 26

Mediana a doi vectori sortati

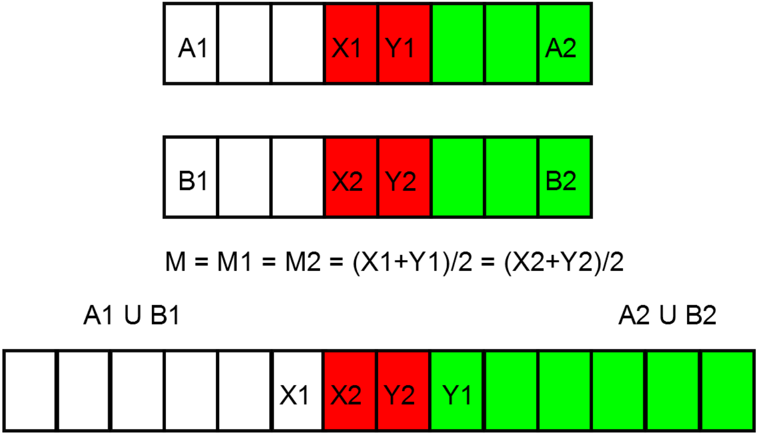
Soluția 2: Se compara medianele M1 si M2 ale celor doi vectori.

Daca **a) M1 = M2** am gasit mediana vectorului obtinut prin interclasare.



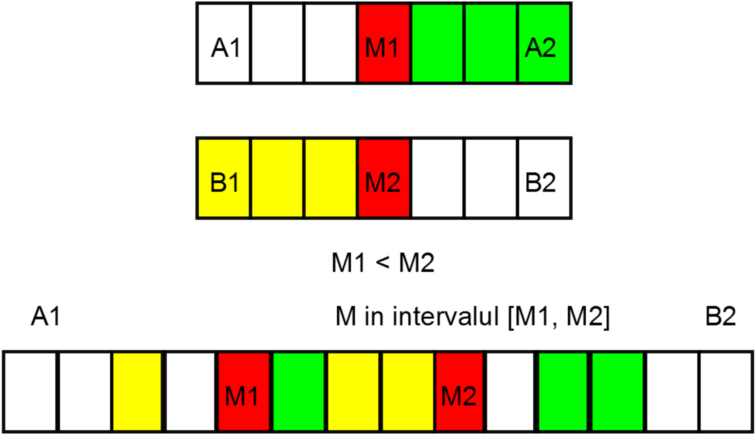
Mediana a doi vectori sortati

Daca m1 $=$ m2 am gasit mediana vectorului obtinut prin interclasare.



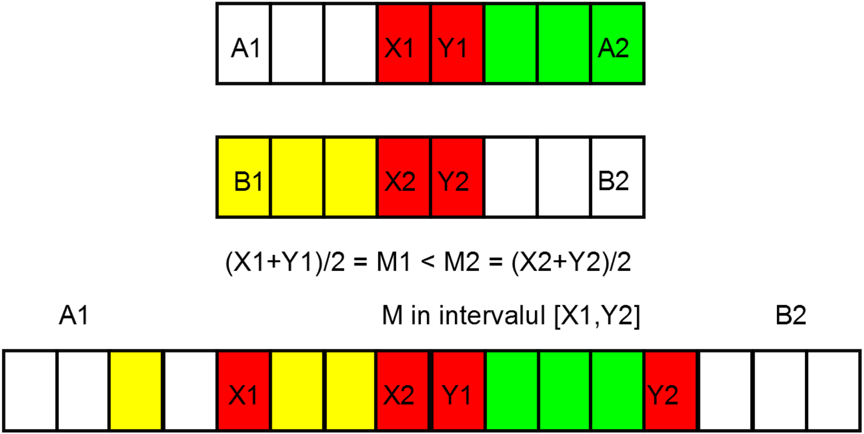
Mediana a doi vectori sortati

Daca **a) M1 $<$ M2**, *n* impar, mediana se află în intervalul [M1, M2].

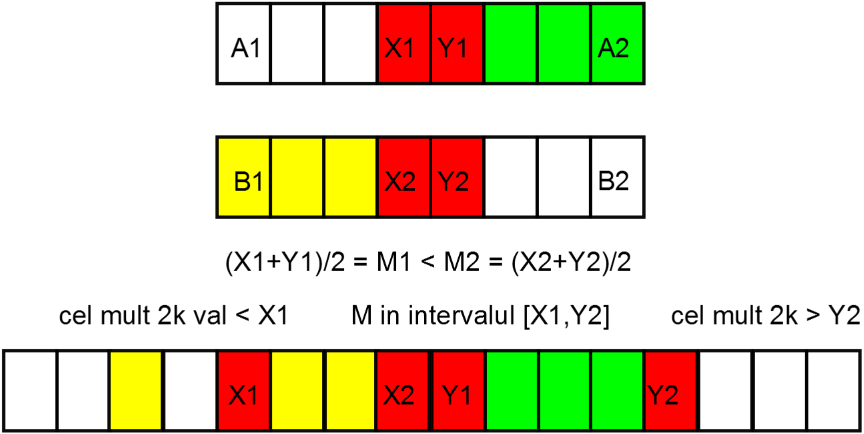


Mediana a doi vectori sortati

Daca **b) M1 $<$ M2**, *n* par, mediana se află în intervalul [X1, Y2].



Daca **b) M1 $<$ M2**, *n* par, mediana se află în intervalul [X1, Y2].



Mediana a doi vectori sortati

Complexitate **O(log n)**

Sortarea prin interclasare

Recurența: $\mathbf{T(n) = 2T(n/2) + cn}$, unde c este o constantă.

Se sortează vectorul $a = (a_1, \ldots, a_n)$ astfel:
Împărțim vectorul în doi subvectori, ordonăm crescător fiecare subvector și asamblăm rezultatele prin interclasare.

$$
\begin{array}{l}
\textit{procedure} \quad SortInter(p, u) \\
\qquad \textit{if} \;\; p = u \\
\qquad \textit{else} \;\; m \leftarrow \lfloor (p+u)/2 \rfloor; \\
\qquad\qquad SortInter(p, m); \\
\qquad\qquad SortInter(m+1, u); \\
\qquad\qquad Inter(p, m, u);. \\
\textit{end}.
\end{array}
$$

Rezolvarea recurenței: $\mathbf{T(n) = 2T(n/2) + cn}$

$k = log_2 n$

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

$$
\begin{aligned}
\mathrm{T}(n) = {} & 2\mathrm{T}(2^{k-1}) + \mathrm{c}2^k = \\
& = 2[2\mathrm{T}(2^{k-2}) + \mathrm{c}2^{k-2}] + \mathrm{c}2^k = 2^2\mathrm{T}(2^{k-2}) + 2\mathrm{c}2^k. \\
& \dots \\
& = 2^i\mathrm{T}(2^{k-i}) + i\mathrm{c}2^k. \\
& \dots \\
& = 2^k\mathrm{T}(1) + k\mathrm{c}2^k = \mathrm{n}(\mathrm{T}(1) + k\mathrm{c}).
\end{aligned}
$$

Sortarea prin interclasare, aplicații

- Se consideră un vector $a = (a_1, \ldots, a_n)$
  Să se găsească subsecvența de suma maximă. O(n logn) O(n)

Sortarea prin interclasare, aplicații

- Se consideră un vector $a = (a_1, \ldots, a_n)$
  Să se găsească subsecvența de suma maximă. O(n logn) O(n)

- Se consideră vectorul $a = (a_1, \ldots, a_n)$
  Să se determine numărul de inversiuni din vectorul a. O(n logn)

Numărul de inversiuni dintr-un vector

Aplicații:

- gradul de ordonare al unui vector
- analiza a clasificărilor (ranking): Preferințele utilizatorilor memorate ca permutări. Se pot compara preferințele a doi utilizatori.

Numărul de inversiuni dintr-un vector
= numărul de inversiuni din subvectorul stâng (prima jumătate)
+ numărul de inversiuni din subvectorul drept
+ numărul de inversiuni (i, j) cu $i < m$ indice în subvectorul stâng și j m indice în subvectorul drept (am notat cu m indicele elementului din mijlocul vectorului).

Mediana unui vector, al k-lea minim

Dat un vector a de n numere și un indice k,$1 \leq k \leq n$, să se determine al k-lea cel mai mic element din vector.
Complexitate: **O(n)**

Mediana:

7 14 **9** 15 8 5 10 : **9**
7 14 **9** 15 8 5 **10** 12 : **9.5**

Se alege un pivot care va ajunge pe pozitia m in vectorul sortat. Daca $m = k$, pivotul este al k-lea minim. Daca $m > k$, al k-lea minim se afla intre valorile din stanga pivotului. Daca $m < k$, al k-lea minim se afla intre valorile din dreapta pivotului.

Mediana unui vector, al k-lea minim

Căutăm al 4-lea minim (mediana pentru un vector cu 7 elemente)

7 14 9 15 8 5 **10** pivot 10

Mediana unui vector, al k-lea minim

Căutăm al 4-lea minim (mediana pentru un vector cu 7 elemente)

7 14 9 15 8 5 **10** pivot 10

7 5 9 8 **10** 15 14 10 este al 5-lea minim.

Mediana unui vector, al k-lea minim

Căutăm al 4-lea minim (mediana pentru un vector cu 7 elemente)

7 14 9 15 8 5 **10** pivot 10

7 5 9 8 **10** 15 14 10 este al 5-lea minim.
Mediana este in stanga pivotului

Mediana unui vector, al k-lea minim

Căutăm al 4-lea minim (mediana pentru un vector cu 7 elemente)

7 14 9 15 8 5 **10**	pivot 10
7 5 9 8 **10** 15 14	10 este al 5-lea minim. Mediana este in stanga pivotului
7 5 9 **8**	pivot 8

Mediana unui vector, al k-lea minim

Căutăm al 4-lea minim (mediana pentru un vector cu 7 elemente)

7 14 9 15 8 5 **10**	pivot 10
7 5 9 8 **10** 15 14	10 este al 5-lea minim. Mediana este in stanga pivotului
7 5 9 **8**	pivot 8
7 5 **8** 9	8 este al 3-lea minim.

Mediana unui vector, al k-lea minim

Căutăm al 4-lea minim (mediana pentru un vector cu 7 elemente)

7 14 9 15 8 5 **10**	pivot 10
7 5 9 8 **10** 15 14	10 este al 5-lea minim. Mediana este in stanga pivotului
7 5 9 **8**	pivot 8
7 5 **8** 9	8 este al 3-lea minim. Mediana este in dreapta pivotului

Mediana unui vector, al k-lea minim

Căutăm al 4-lea minim (mediana pentru un vector cu 7 elemente)

7 14 9 15 8 5 **10**	pivot 10
7 5 9 8 **10** 15 14	10 este al 5-lea minim. Mediana este in stanga pivotului
7 5 9 **8**	pivot 8
7 5 **8** 9	8 este al 3-lea minim. Mediana este in dreapta pivotului
9	pivot 9 este al 4-lea minim.

Mediana unui vector, al k-lea minim

Căutăm al 4-lea minim (mediana pentru un vector cu 7 elemente)

7 14 9 15 8 5 **10**	pivot 10
7 5 9 8 **10** 15 14	10 este al 5-lea minim. Mediana este in stanga pivotului
7 5 9 **8**	pivot 8
7 5 **8** 9	8 este al 3-lea minim. Mediana este in dreapta pivotului
9	pivot 9 este al 4-lea minim.

Mediana unui vector, al k-lea minim

```
function kmin(p,u)
    m = pozRand(p,u);
    if (m = k) return a[m]
    if(m < k)
        return kmin(m+1,u)
    else
        return kmin(p,m-1)
end
```

Mediana unui vector, al k-lea minim

Timpul mediu de executie:

$$T(n) \leq (n-1) + \frac{1}{n}\sum_{k=1}^{n} T(max\{k-1, n-k\})$$

$$\leq \frac{2}{n}\sum_{k=\lceil n/2 \rceil}^{n-1} T(k) \leq cn$$

Pentru a demonstra

$$T(n) \leq cn$$

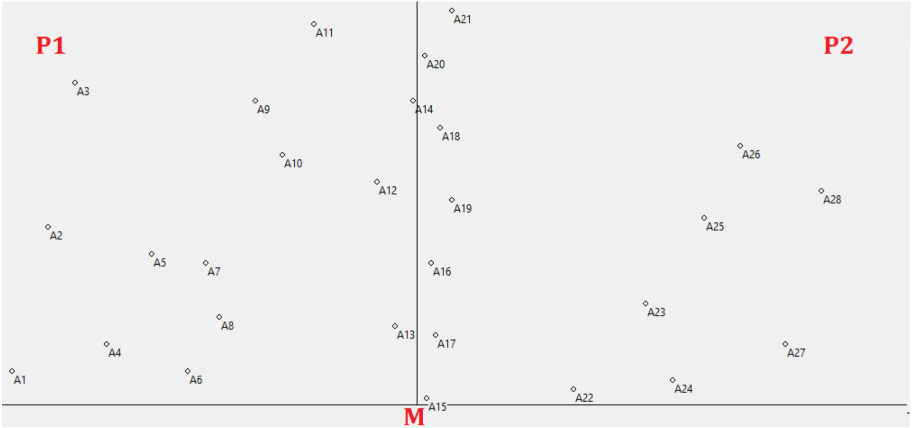
aplicăm substituții în relația de mai sus.

Cea mai apropiata pereche de puncte

Fiind date n puncte in plan, prin coordonatele lor (x_i, y_i) sa se gaseasca cea mai apropiata pereche de puncte.

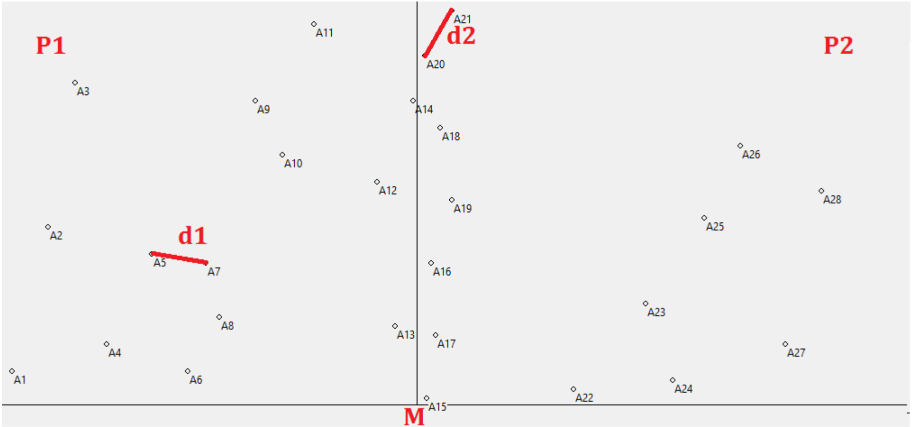
Complexitate **$O(n \log n)$: $T(n) = 2T(n/2) + O(n)$**


P0
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
A11
A12
A13
A14
A15
A16
A17
A18
A19
A20
A21
A22
A23
A24
A25
A26
A27
A28

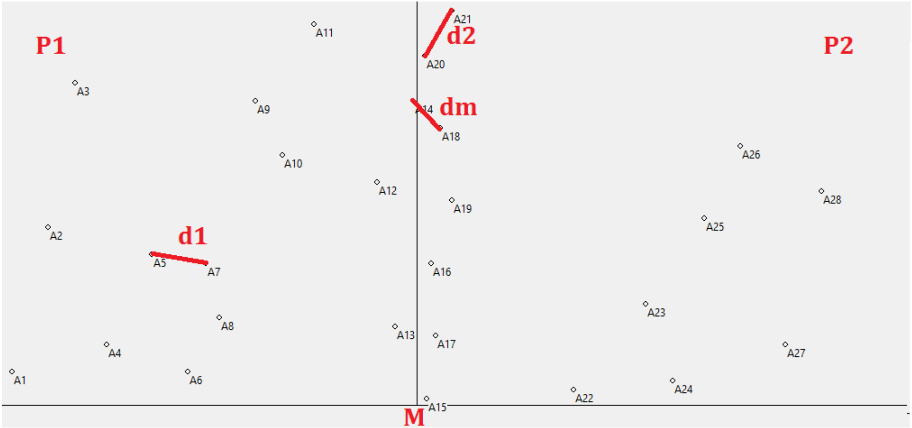


Se imparte multimea de puncte in doua submultimi P1 si P2 cu numar egal (n/2) de puncte. Se cauta mediana M. Dreapta verticala x = M va fi cea va delimita cele doua multimi de puncte.

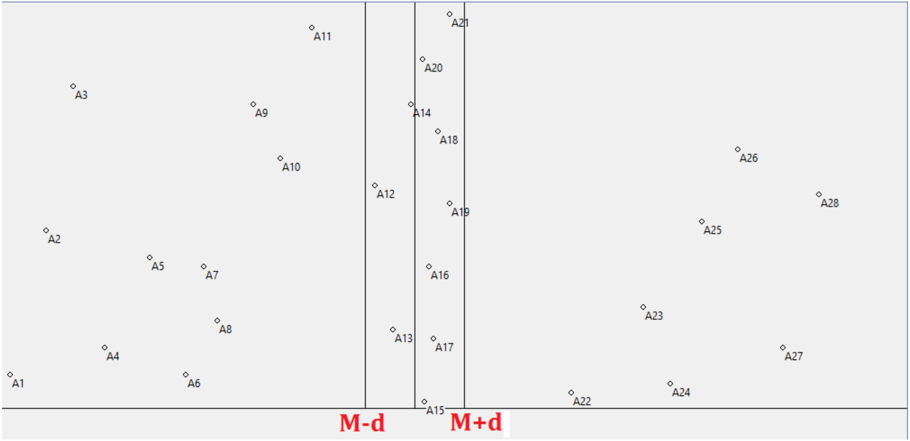
Complexitatea acestui pas este O(n).



Se gaseste cea mai apropiata pereche de puncte din P1 intre care exista distanta d_1, cea mai apropiata pereche de puncte din P2, intre care exista distanta d_2.

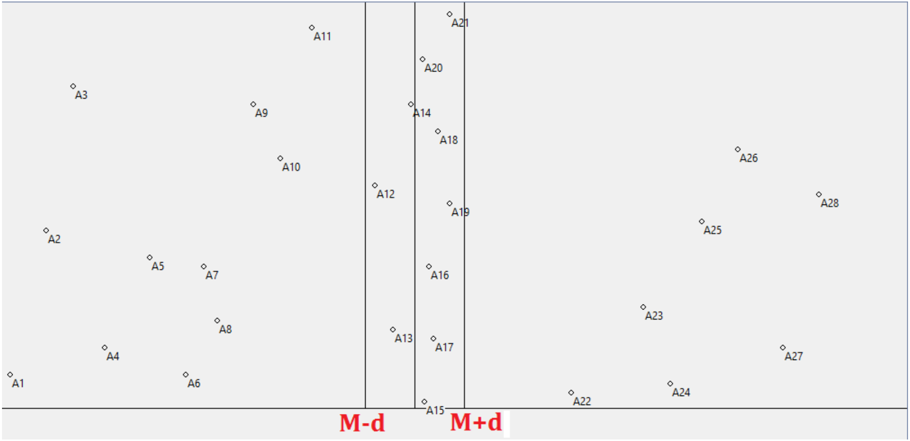


Dupa rezolvarea probelmelor P_1, P_2 nu este calculata distanta $d_m = min\{d(A, B)$, pentru $A \in P_1, B \in P_2\}$.
Se va alege minimul dintre distantele d_1, d_2 si d_m.



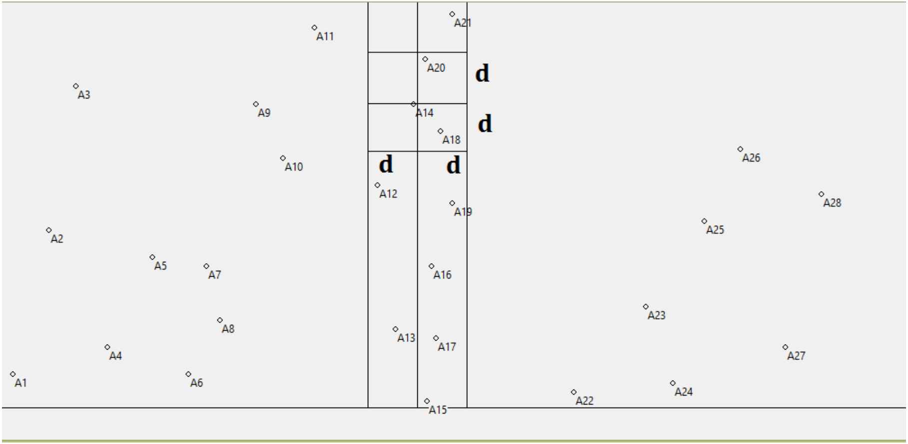
Cum calculam eficient d_m?

Fie $d = min(d_1, d_2)$. Pentru a calcula d_m este suficient sa consideram perechi de puncte care au abscisa in intervalul $[M - d, M + d]$.



Cum calculam eficient d_m?

Fie $d = min(d_1, d_2)$. Pentru a calcula d_m este suficient sa consideram perechi de puncte care au abscisa in intervalul $[M - d, M + d]$.
Nu vrem ca dm să fie mai mare decât d.

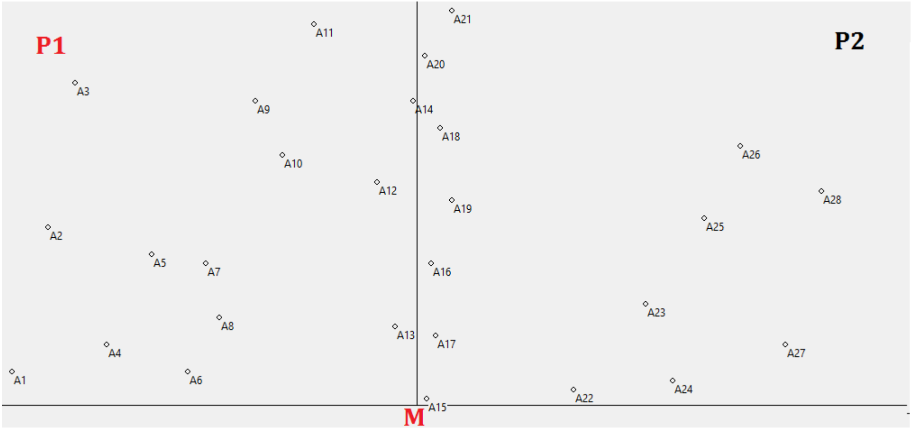


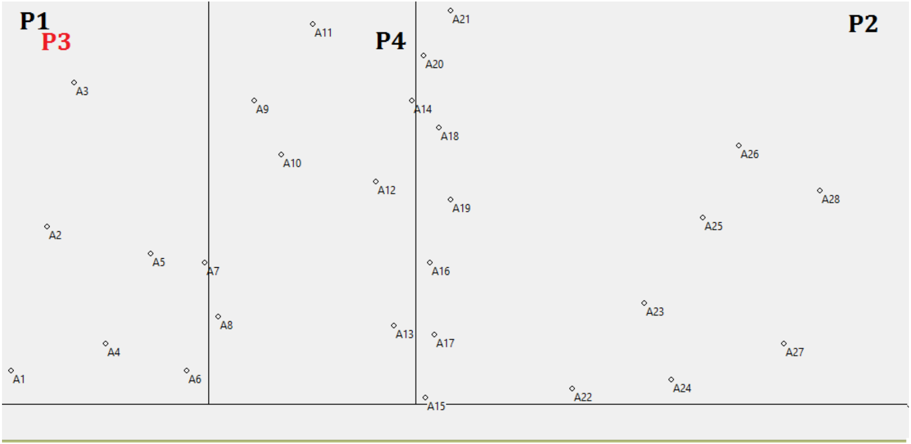
Cum calculam eficient d_m? O(n)
Pana acum nu am luat in considerare decat distantele intre abscisele punctelor.

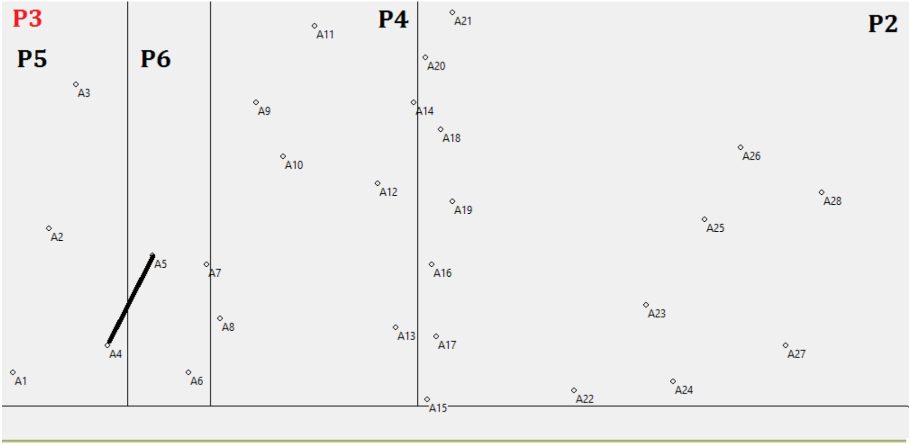
Este suficient sa consideram pentru fiecare punct $A(x_A, y_A)$ cu $|M - x_A| \leq \delta$ puncte $B(x_B, y_B)$ din dreptunghiul de dimensiuni $[2dxd]$ centrat pe dreapta x = M. Intr-un astfel de dreptunghi pot fi maxim 8 puncte.

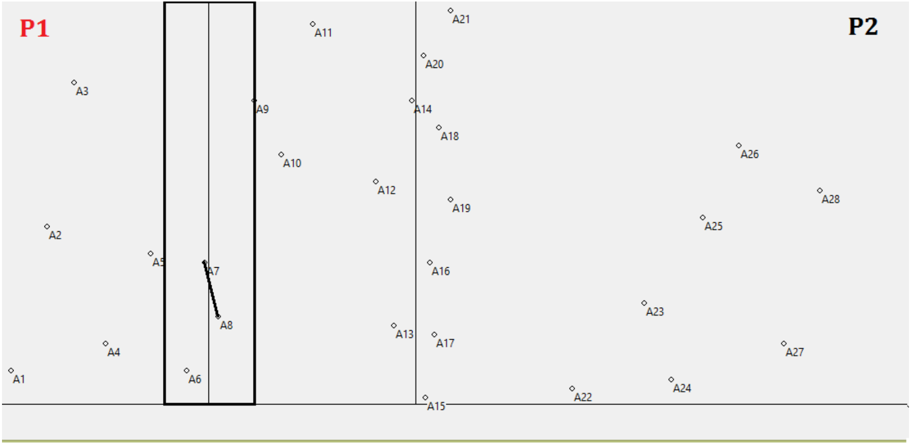
Este suficient sa consideram pentru fiecare punct $A(x_A, y_A)$ cu $|M - x_A| \leq \delta$ distantele fata de urmatoarele 7 puncte daca punctele sunt sortate dupa ordonata.

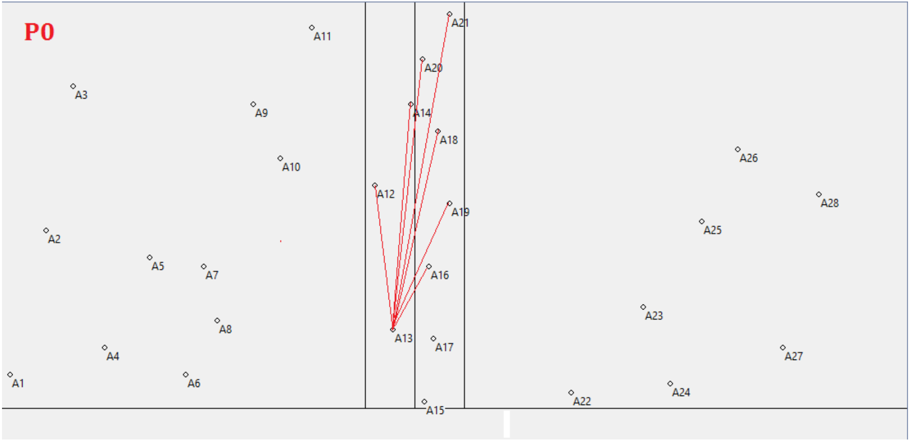
```
function dmin(X , Y, st, dr)
    daca (|X| < 4) d = min(perechi de elemente din X[st..dr])
    altfel
        mid=(st+dr)/2
        SY= multimea punctelor din Y ∩ X[st..mij]
        DY= multimea punctelor din Y ∩ X[mij+1..dr]
        d1=dmin(X[st..mij], SY, st, mij)
        d2=dmin(X[mij+1..dr], DY, mij+1,dr)
        d=min(d1, d2)
        LY= Y ∩ B (cu abscisa la distanta <d de abscisa punctului X[mid])
        calculeaza(dm) /* considernd punctele p din LY si
            perechile formate de p cu fiecare din cele 7 puncte din LY*/
        d=min(d, dm)
    return d
end
```


P1
P2
M
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
A11
A12
A13
A14
A15
A16
A17
A18
A19
A20
A21
A22
A23
A24
A25
A26
A27
A28


P1
P3
P4
P2
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
A11
A12
A13
A14
A15
A16
A17
A18
A19
A20
A21
A22
A23
A24
A25
A26
A27
A28


P3
P5
P6
P4
P2
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
A11
A12
A13
A14
A15
A16
A17
A18
A19
A20
A21
A22
A23
A24
A25
A26
A27
A28


P1
P2
A1
A2
A3
A4
A6
A7
A8
A9
A10
A11
A12
A13
A14
A15
A16
A17
A18
A19
A20
A21
A22
A23
A24
A25
A26
A27
A28


P0
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
A11
A12
A13
A14
A15
A16
A17
A18
A19
A20
A21
A22
A23
A24
A25
A26
A27
A28