TAP
Curs 4,5: DIVIDE ET IMPERA

Tehnici avansate de programare

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

semestrul 1, 2019

Rezumat curs Divide et Impera

Exemplul 1

- Să se afle simultan maximul și minimul dintr-un șir de n numere întregi. Propuneți un algoritm care să facă un număr minim de comparații și care să garanteze aflarea minimului și maximului.

Exemplul 1

- Să se afle simultan maximul și minimul dintr-un șir de *n* numere întregi. Propuneți un algoritm care să facă un număr minim de comparații și care să garanteze aflarea minimului și maximului.
- Există un algoritm care calculează maximul și minimul folosind doar **ceil(3$\star$n/2) - 2** comparații? Este acesta optim?

Exemplul 1

- Să se afle simultan maximul și minimul dintr-un șir de *n* numere întregi. Propuneți un algoritm care să facă un număr minim de comparații și care să garanteze aflarea minimului și maximului.
- Există un algoritm care calculează maximul și minimul folosind doar **ceil(3$\star$n/2) - 2** comparații? Este acesta optim?
- Există un algoritm divide et impera pentru rezolvarea problemei? Cum calculăm complexitatea timp?

Exemplul 1

- Să se afle simultan maximul și minimul dintr-un șir de n numere întregi. Propuneți un algoritm care să facă un număr minim de comparații și care să garanteze aflarea minimului și maximului.

- Există un algoritm care calculează maximul și minimul folosind doar **ceil(3$\star$n/2) - 2** comparații? Este acesta optim?

- Există un algoritm divide et impera pentru rezolvarea problemei? Cum calculăm complexitatea timp?

$$\mathbf{T(n) = 2\ T(n/2) + 2}$$

Aplicând substituții succesive, se obține pentru $n = 2^p$:

$\mathbf{T}(2^p) = 2^k T(2^{p-k}) + 2(2^k - 1)$. Pentru $k = p - 1$ avem $\mathbf{T}(n) = 3 \star \frac{n}{2} - 2$

Exemplul 2

- Să se afle subsecvența de sumă maximă dintr-un șir de n numere întregi.

Exemplul 2

- Să se afle subsecvența de sumă maximă dintr-un șir de n numere întregi.
- Există un algoritm divide et impera optim pentru rezolvarea problemei? Cum calculam complexitatea timp?

Exemplul 2

- Să se afle subsecvența de sumă maximă dintr-un șir de n numere întregi.
- Există un algoritm divide et impera optim pentru rezolvarea problemei? Cum calculam complexitatea timp?

$$\mathbf{T(n) = 2\ T(n/2) + n}$$

Recurența este asemănătoare cu cea obținută la merge-sort.

Aplicând teorema master se obține complexitatea timp:

$$T(n) \in \mathcal{O}(n \log n)$$

Prezentare generală

Metoda Divide et Impera constă în:

- **împărțirea** problemei inițiale în subprobleme de **același tip**;
- **rezolvarea** subproblemelor recursiv, 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
  $\Rightarrow$ **substituție, 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. Sunt adevărate afirmațiile:

Cazul 1: Pentru $f(n) \in \Theta(n^c)$ cu $c < \log_b a$ avem $T(n) = \Theta(n^{\log_b a})$

Cazul 2: Pentru $f(n) \in \Theta(n^c \log^k n)$ cu $c = \log_b a$ avem $T(n) = \Theta(n^c \log^{k+1} n)$

Cazul 3: Pentru $f(n) \in \Theta(n^c)$ cu $c > \log_b a$ avem $T(n) = \Theta(f(n))$

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: $\mathcal{O}(n \log n)$

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

$k = log_2 n$

- 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 < n (k nu este cunoscut) poziții pe pozițiile $n - k + 1, \ldots, n$. Să se găsească maximul în vectorul a.

Algoritmul de căutare binară, aplicații

- Se consideră un vector $a = (a_1, \ldots, a_{2p+1})$ 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 < n (k nu este cunoscut) poziții pe pozițiile $n - k + 1, \ldots, n$. Să se găsească maximul în vectorul a.

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

Căutare într-o matrice sortată

Se consideră o matrice A în care toate liniile, respectiv toate coloanele sunt sortate crescător.
Să se determine dacă valoare x se afla între valorile matricei A.

Soluții:

Căutare într-o matrice sortată

Se consideră o matrice A în care toate liniile, respectiv toate coloanele sunt sortate crescător.
Să se determine dacă valoare x se afla între valorile matricei A.

Soluții:

- căutare binară pe linii/coloane. În cazul în care numărul de linii este mai mic decât numărul de coloane se obține complexitatea $\mathcal{O}(l * \log(c))$;

Căutare într-o matrice sortată

Se consideră o matrice A în care toate liniile, respectiv toate coloanele sunt sortate crescător.
Să se determine dacă valoare x se afla între valorile matricei A.

Soluții:

- căutare binară pe linii/coloane. În cazul în care numărul de linii este mai mic decât numărul de coloane se obține complexitatea $\mathcal{O}(l * \log(c))$;
- strategie divide et impera, comparăm elementul căutat cu valoarea din mijlocul matricei: T$(l \star c) = 1 + 3$*T$(l \star c/4)$.
$\mathcal{O}(l \star c^{log_4 3})$: se aplica teorema master, cazul 1;

Căutare într-o matrice sortată

Se consideră o matrice A în care toate liniile, respectiv toate coloanele sunt sortate crescător.
Să se determine dacă valoare x se afla între valorile matricei A.

Soluții:

- căutare binară pe linii/coloane. În cazul în care numărul de linii este mai mic decât numărul de coloane se obține complexitatea $\mathcal{O}(l * \log(c))$;
- strategie divide et impera, comparăm elementul căutat cu valoarea din mijlocul matricei: T$(l \star c) = 1 + 3$*T$(l \star c/4)$.
  $\mathcal{O}(l \star c^{log_4 3})$: se aplica teorema master, cazul 1;
- eliminarea succesiva a unei linii sau a unei coloane $\mathcal{O}(l + c)$.

Căutare într-o matrice sortată

Se consideră o matrice A în care toate liniile, respectiv toate coloanele sunt sortate crescător.
Să se determine dacă valoare x se afla între valorile matricei A.

Soluții:

- căutare binară pe linii/coloane. În cazul în care numărul de linii este mai mic decât numărul de coloane se obține complexitatea $\mathcal{O}(l * \log(c))$;
- strategie divide et impera, comparăm elementul căutat cu valoarea din mijlocul matricei: $\mathrm{T}(l \star c) = 1 + 3$*$\mathrm{T}(l \star c/4)$.
  $\mathcal{O}(l \star c^{log_4 3})$: se aplica teorema master, cazul 1;
- eliminarea succesiva a unei linii sau a unei coloane $\mathcal{O}(l + c)$.
- Optim: $\mathcal{O}(l * log(c/l))$

Mediana a doi vectori sortați

- Se dau doi vectori A, B de dimensiuni egale n. Să se obțină mediana vectorului obținut prin interclasarea celor doi vectori.

Soluția 1: Se interclasează vectorii și se obține mediana în timp constant. Complexitate $\mathcal{O}(n)$

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 Ma si Mb ale celor doi vectori.

Daca **ma = mb** am gasit mediana vectorului obtinut prin interclasare Mc = Ma = Mb.

Daca **ma > mb** perechea a carei medie da mediana vectorului obtinut prin interclasare se afla intre elementele subvectorilor:

$a[0 \ldots \lfloor n/2 \rfloor]$ si $b[\lfloor (n-1)/2 \rfloor \ldots n-1]$

Daca **ma < mb** perechea a carei medie da mediana vectorului obtinut prin interclasare se afla intre elementele subvectorilor:

$b[0 \ldots \lfloor n/2 \rfloor]$ si $a[\lfloor (n-1)/2 \rfloor \ldots n-1]$

Complexitate: $\mathcal{O}(\log n)$

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.

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

Sortarea prin interclasare

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

Sortarea prin interclasare, aplicații

- 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 v
> $=$ numărul de inversiuni din subvectorul stâng (prima jumătate)
> $+$ numărul de inversiuni din subvectorul drept
> $+$ numărul de inversiuni (i, j) a.i $v[i] > v[j]$ 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.

```
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
```

Timpul mediu de executie:

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

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

Pentru a demonstra

$$T(n) \le cn$$

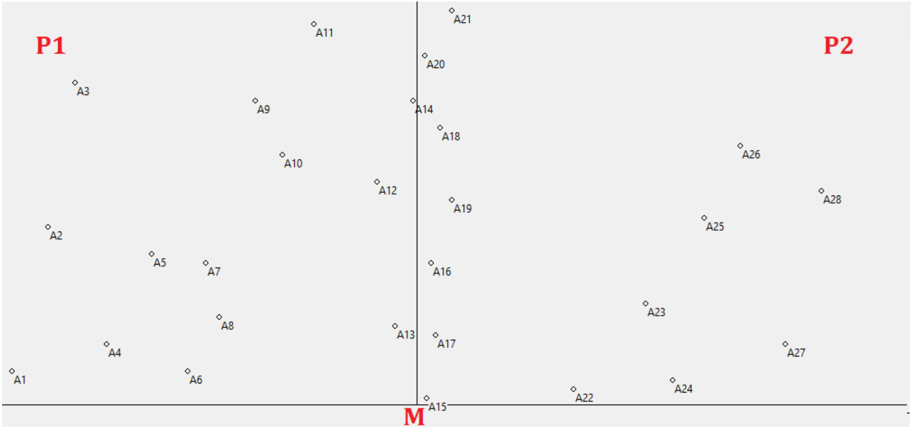
aplicăm substituții în relația de mai sus.

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.

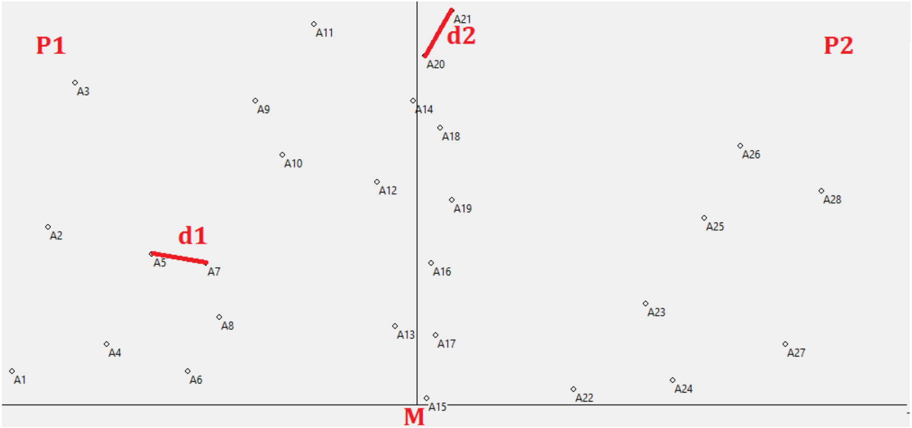
Complexitate **$O(n \log n)$: $T(n) = 2T(n/2) + O(n)$**


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

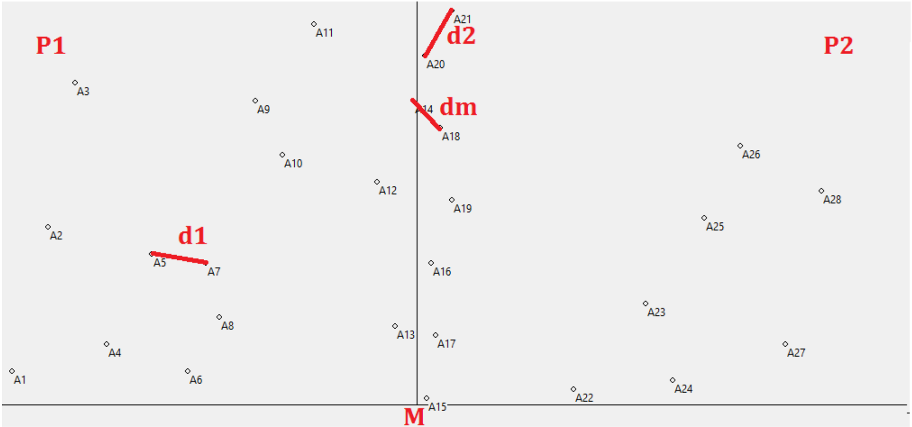


Se imparte multimea de puncte in doua submultimi P1 si P2 cu numar egal (n/2) de puncte. Se cauta mediana M. Dreapta verticala x = M va fi cea va delimita cele doua multimi de puncte.

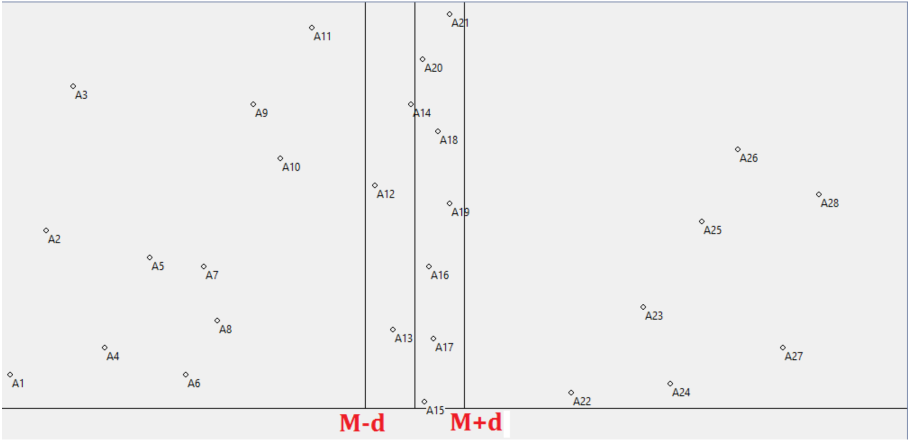
Complexitatea acestui pas este O(n).



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.

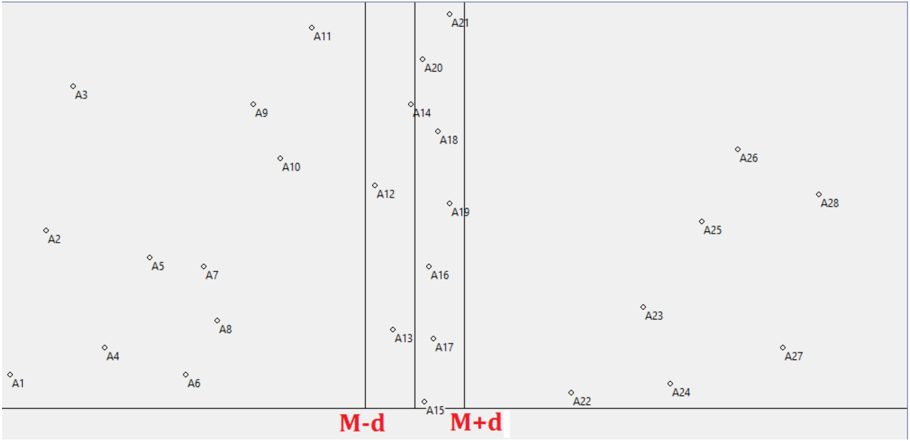


Dupa rezolvarea probelmelor P_1, P_2 nu este calculata distanta $d_m = min\{d(A, B)$, pentru $A \in P_1, B \in P_2\}$.
Se va alege minimul dintre distantele d_1, d_2 si d_m.



Cum calculam eficient d_m?

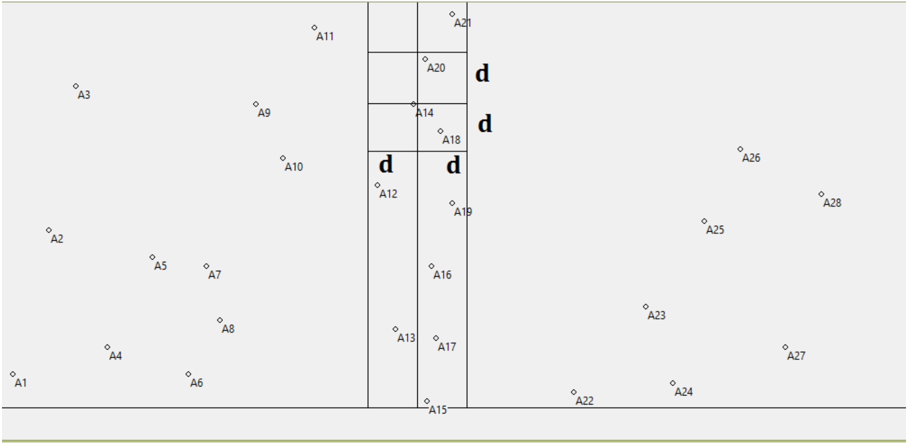
Fie $d = min(d_1, d_2)$. Pentru a calcula d_m este suficient sa consideram perechi de puncte care au abscisa in intervalul $[M - d, M + d]$.



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.

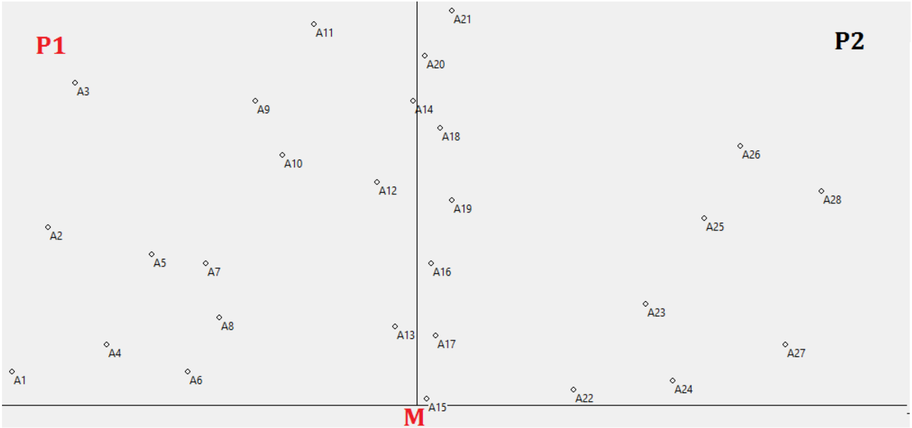


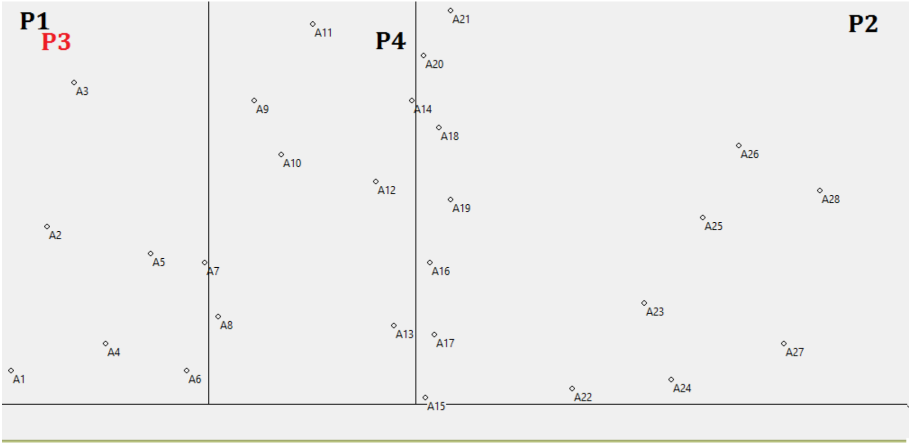
Cum calculam eficient d_m? O(n)
Pana acum nu am luat in considerare decat distantele intre abscisele punctelor.

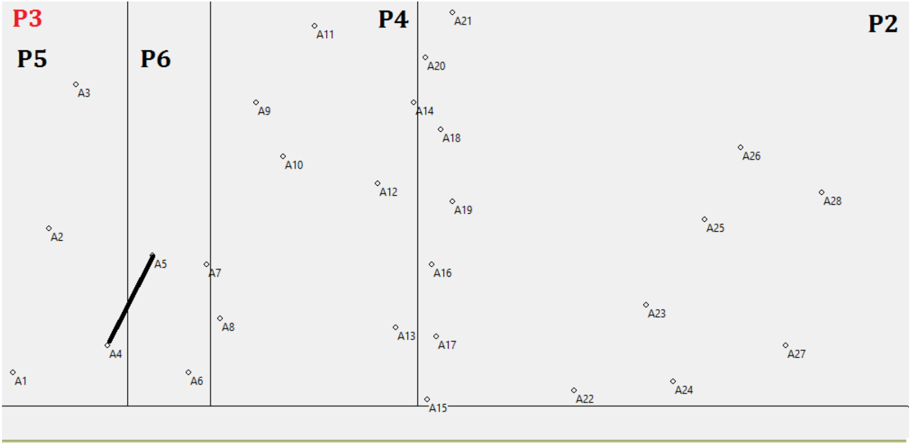
Este suficient sa consideram pentru fiecare punct $A(x_A, y_A)$ cu $|M - x_A| \leq \delta$ puncte $B(x_B, y_B)$ din dreptunghiul de dimensiuni $[2dxd]$ centrat pe dreapta x = M. Intr-un astfel de dreptunghi pot fi maxim 8 puncte.

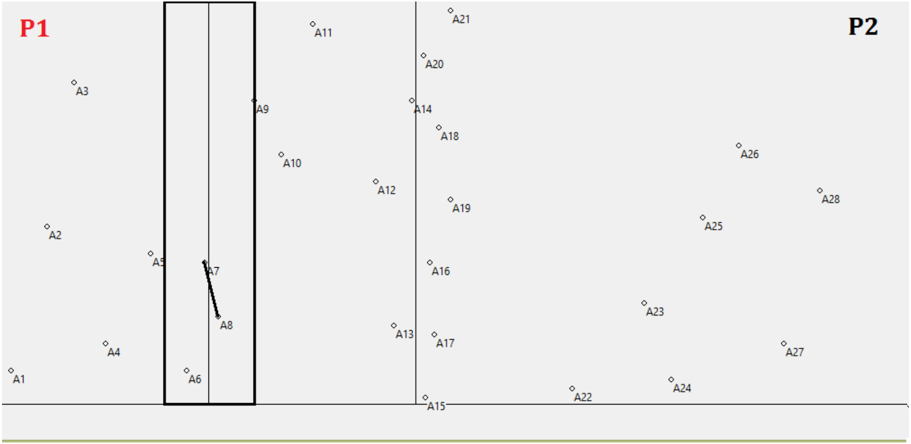
Este suficient sa consideram pentru fiecare punct $A(x_A, y_A)$ cu $|M - x_A| \leq \delta$ distantele fata de urmatoarele 7 puncte daca punctele sunt sortate dupa ordonata.

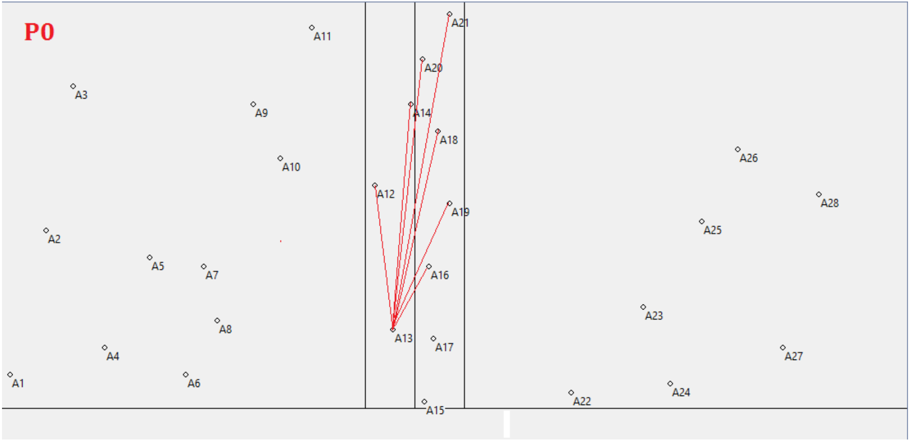
```
function dmin(X , Y, st, dr)
    daca (|X| < 4) d = min(perechi de elemente din X[st..dr])
    altfel
        mid=(st+dr)/2
        SY= multimea punctelor din Y ∩ X[st..mij]
        DY= multimea punctelor din Y ∩ X[mij+1..dr]
        d1=dmin(X[st..mij], SY, st, mij)
        d2=dmin(X[mij+1..dr], 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
A5
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