



# G om trie algorithmique

B. Meyer

Cours INF280, d partement INFRES





# Contenu

Les objets simples

Les polygones

Algorithmes sophistiqués



## En quelques mots

- ▶ Beaucoup de raisonnements mathématiques de collégiens, mais revus à la sauce algorithmique.
- ▶ Nombreux *risques d'erreurs* numériques :
  - Dans le doute, utiliser des `double`.
- ▶ Cas limites ou cas dégénérés pénibles :
  - ▶ droites parallèles au lieu de sécantes,
  - ▶ triangles plats,
  - ▶ cercles de rayon nul,
  - ▶ droites de pente  $\infty$ ,
  - ▶ etc.

## Quelques bonnes pratiques

- ▶ Ne jamais faire le test  $a==b$  mais tester  $fabs(a-b) < EPS$  où  $EPS$  est petit ( $1e-9$  par exemple).
- ▶ Travailler avec des *entiers* aussi longtemps que possible.

### Quelques exemples

- ▶ Utiliser  $\|\mathbf{v}\|^2$  plutôt que  $\|\mathbf{v}\|$  (ce qui évite  $\sqrt{\cdot}$ ).
- ▶ Comparer les angles via leur cosinus s'il est plus facile à calculer.
- ▶ Utiliser une classe de pseudo-rationnels (par ex : couples  $[num, denom]$  avec opérateurs  $=$  et  $\leq$ ).



## Principe du balayage par une droite

Contexte : problème portant sur  $n$  points.

**Paradigme** de géométrie algorithmique fréquent  
(intersection de segments, diagrammes de Voronoï, etc.).

- ▶ Solution **naïve** en  $O(n^2)$  :  
Une action ou un test est effectué pour chaque couple de points.
- ▶ Solution par **balayage** en  $O(n \ln n)$  :  
On imagine une droite qui balaye le plan. Action ou test effectué sur les couples de points ou événements rencontrés le long de la droite.

(Ex : UVa10613, UVa10691)



# Contenu

Les objets simples

Les polygones

Algorithmes sophistiqués

## Les points

- ▶ Méritent souvent une classe propre.
- ▶ Penser à surcharger les opérateurs == ou < (si besoin de tri).

### Une structure de points

```
struct point {
    double x, y;
    point() {
        x=0.0; y=0.0;
    }
    point(double in_x, double in_y): x(in_x), y(in_y) {}
    bool operator == (point pt2) const {
        return (fabs(x - pt2.x) < EPS
            && (fabs(y - pt2.y) < EPS));
    }
};
```

# Les vecteurs

Prévoir dans son code :

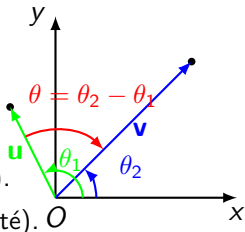
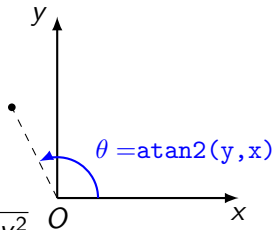
- ▶ structure ad hoc,
- ▶ fonction **produit scalaire** ( $xx' + yy'$ ),
- ▶ fonction **produit vectoriel** ( $xy' - x'y$ ).

Attributs :

- ▶ Norme : `hypot(x,y)` renvoie  $\sqrt{x^2 + y^2}$ .
- ▶ Angle avec l'axe ( $Ox$ ) :
  - ▶ Éviter `atan(y/x)` (danger : division par 0).
  - Utiliser `atan2(y,x)` (valeur dans  $[-\pi, \pi]$ ).

Angle entre deux vecteurs **u** et **v** :

- ▶ Soustraire leur angle avec ( $Ox$ ) (angle orienté).
- ▶ Calculer `arccos(u · v / (||u|| ||v||))` (angle non-orienté).





## Les droites, les segments, les coniques, ...

Formes possibles de droites :

- ▶ équation cartésienne  $ax + by + c = 0$ ,
- ▶ pente et ordonnée à l'origine  $y = ax + b$  (sauf droite verticale),
- ▶ abscisse et ordonnée à l'origine  $x/a + y/b = 1$ ,
- ▶ point & vecteur directeur,
- ▶ couple de points.

Coordonnées de l'**intersection** de deux droites sécantes

$D : ax + by + c$  et  $D : a'x + b'y + c'$  (via les formules de Cramer) :

$$\Delta = \begin{vmatrix} a & b \\ a' & b' \end{vmatrix}, \quad x = -\frac{\begin{vmatrix} c & b \\ c' & b' \end{vmatrix}}{\Delta} \quad \text{et} \quad y = -\frac{\begin{vmatrix} a & c \\ a' & c' \end{vmatrix}}{\Delta}.$$

Les droites sont **effectivement sécantes** ssi  $\Delta$  est non-nul.

(Ex : UVa837, UVa10678)

## Les coniques

Tranche d'un cône par un plan ou ensemble des points  $M$  tels que

$$\text{dist}(F, M) = e \cdot \text{dist}(F, \Delta)$$

où  $F$  est le **foyer**,  $\Delta$  la **directrice** et  $e$  l'**excentricité**.

Équation polaire :

$$\rho = \text{cste} / (1 + e \cos \theta).$$

**Cas**  $e = 0$  : **cercle** de centre  $\Omega$  et rayon  $R$  :

- ▶ équation  $(x - x_\Omega)^2 / R^2 + (y - y_\Omega)^2 / R^2 = 1$
- ▶ paramétrisation  $x(t) = R \cos(t)$ ,  $y(t) = R \sin(t)$ ,  $t \in [0, 2\pi[$ .
- ▶ périmètre  $2\pi R$ , aire  $\pi R^2$

**Cas**  $e = 1$  : **parabole**

## Les coniques

**Cas**  $0 < e < 1$  : **ellipse** de centre  $\Omega$ , de foyers  $F$  et  $F'$ , demi-grand axe  $a$ , demi-petit axe  $b$  :

- ▶ équation  $(x - x_\Omega)^2/a^2 + (y - y_\Omega)^2/b^2 = 1$
- ▶ paramétrisation  $x(t) = a \cos(t)$ ,  $y(t) = b \sin(t)$ ,  $t \in [0, 2\pi[$ .
- ▶ points  $M$  tel que  $\text{dist}(M, F) + \text{dist}(M, F') = \text{constante}$ .
- ▶ aire  $\pi ab$

**Cas**  $1 < e$  : **hyperbole** de centre  $\Omega$ , de foyers  $F$  et  $F'$ , demi-grand axe  $a$ , demi-petit axe  $b$  :

- ▶ équation  $(x - x_\Omega)^2/a^2 - (y - y_\Omega)^2/b^2 = 1$
- ▶ paramétrisation  $x(t) = a \cosh(t)$ ,  $y(t) = b \sinh(t)$ ,  $t \in [0, 2\pi[$ .
- ▶ points  $M$  tel que  $|\text{dist}(M, F) - \text{dist}(M, F')| = \text{constante}$ .

## Distances sur la sphère

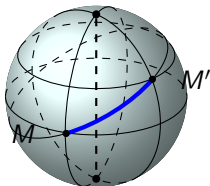
Distance orthodromique (ou distance du plus grand cercle) entre :

- ▶ premier point  $M$  de latitude  $\delta$  et longitude  $\lambda$ ,
- ▶ second point  $M'$  de latitude  $\delta'$  et longitude  $\lambda'$ ,

avec la formule suivante (*Haversine formula* en anglais) :

$$d = 2R \arcsin \sqrt{\sin^2 \left( \frac{\delta' - \delta}{2} \right) + \cos \delta \cos \delta' \sin^2 \left( \frac{\lambda' - \lambda}{2} \right)}$$

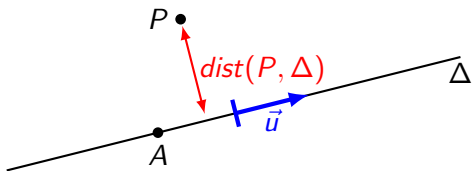
où  $R$  = rayon de la sphère.



## Distance point – droite

- Distance entre le point  $P$  et la droite  $\Delta$  passant par  $A$  dirigée par  $\vec{u}$  :

$$\text{dist}(P, \Delta) = \|\overrightarrow{AP} \times \vec{u}\| / \|\vec{u}\|.$$



(Ex : UVa10263)

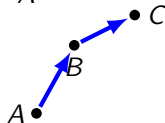
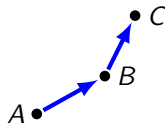
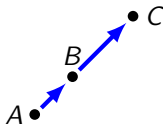
## Alignements et virages

► Trois points  $A$ ,  $B$ ,  $C$  sont

1. colinéaires ssi  $\overrightarrow{AB} \times \overrightarrow{BC} = 0$ ,

2. en virage à gauche ssi  $\overrightarrow{AB} \times \overrightarrow{BC} > 0$ ,

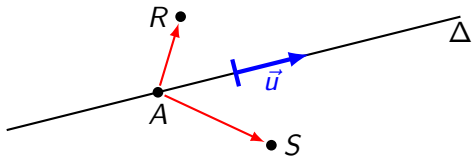
3. en virage à droite ssi  $\overrightarrow{AB} \times \overrightarrow{BC} < 0$ .



## Intersection de segments

- ▶ Un bipoint  $(R, S)$  est **de part et d'autre** de la droite  $\Delta$  passant par  $A$  et dirigée par le vecteur  $\vec{u}$  si

$$(\overrightarrow{AR} \times \vec{u}) \cdot (\overrightarrow{AS} \times \vec{u}) < 0.$$



- ▶ Deux segments  $[AB]$  et  $[RS]$  sont **strictement sécants** ssi le bipoint  $(A, B)$  est de part et d'autre de la droite  $(RS)$  et le bipoint  $(R, S)$  est de part et d'autre de la droite  $(AB)$ .

Pour calculer l'intersection, voir transparent 9.

(Ex : UVa191, UVa10902)



# Contenu

Les objets simples

Les polygones

Algorithmes sophistiqués



## Grandeurs du triangle

- ▶ **Longueur** des côtés (Al-Kashi) :

$$c^2 = a^2 + b^2 - 2ab \cos \gamma.$$

- ▶ **Aire du triangle**  $ABC$  :

$$A = \frac{\text{base} \times \text{hauteur}}{2} = \frac{\|\vec{AB} \times \vec{AC}\|}{2} = \sqrt{s(s-a)(s-b)(s-c)}$$

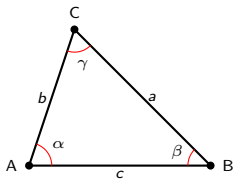
où  $s = (a + b + c)/2$  (formule d'Héron).

- ▶ **Centre de gravité**, intersection des médianes :

$$G = \text{barycentre}((A, 1), (B, 1), (C, 1)).$$

- ▶ **Orthocentre**, intersection des hauteurs :

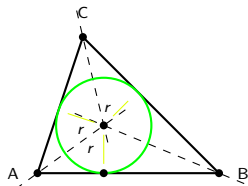
$$H = \text{barycentre}((A, \tan \alpha), (B, \tan \beta), (C, \tan \gamma)).$$



# Cercles des triangles

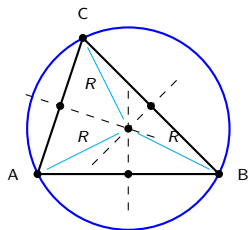
## ► Cercle inscrit :

- Centre  $I$  : intersection des bissectrices :  
 $I = \text{barycentre}((A, a), (B, b), (C, c))$ .
- Rayon  $r = \frac{A}{s}$  où  $A = \text{aire}$ ,  $s = \text{demi-périmètre}$ .



## ► Cercle circonscrit :

- Centre  $O$  : intersection des médiatrices :  
 $O = \text{barycentre}((A, \tan \beta + \tan \gamma), (B, \tan \alpha + \tan \gamma), (C, \tan \alpha + \tan \beta))$ .
- Rayon  $R = \frac{a \cdot b \cdot c}{4A} = \frac{a}{2 \sin \alpha} = \frac{b}{2 \sin \beta} = \frac{c}{2 \sin \gamma}$ .

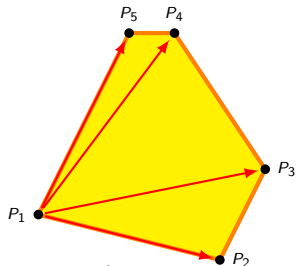


## Aire d'un polygone

Polygone  $\Pi = (P_1 P_2 \dots P_k)$  :  
 (même formule que le triangle)

$$A = \frac{1}{2} \left\| \sum_{i=2}^{k-1} \overrightarrow{P_1 P_i} \times \overrightarrow{P_1 P_{i+1}} \right\|$$

$$= \frac{1}{2} \begin{vmatrix} x_1 & y_1 & - \\ x_2 & y_2 & + \\ \vdots & \vdots & + \\ x_n & y_n & + \\ x_1 & y_1 & + \end{vmatrix} = \frac{1}{2} \left| \sum_{i=1}^n x_i y_{i+1} - x_{i+1} y_i \right|$$



Valable que le polygone soit convexe ou non.

(Ex : UVa11265)

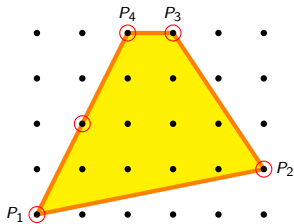
## Formule de Pick

### Théorème

Soit  $\Pi$  un polygone à sommets entiers,  $a$  son aire,  $b$  le nombre de points entiers sur sa frontière et  $i$  le nombre de points entiers dans son intérieur strict, alors

$$a = i + \frac{b}{2} - 1.$$

(Ex : UVa10088)

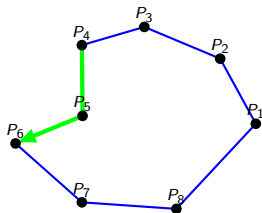
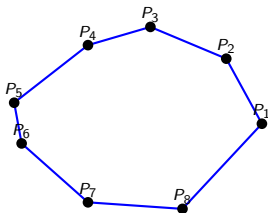


$$\begin{aligned} a &= \frac{1}{2} \left\| \binom{5}{1} \times \binom{3}{4} + \binom{3}{4} \times \binom{2}{4} \right\| \\ &= 9 + \frac{5}{2} - 1 = \frac{21}{2} \end{aligned}$$

# Convexité d'un polygone ?

## Théorème

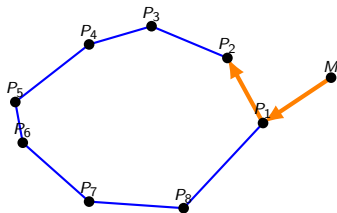
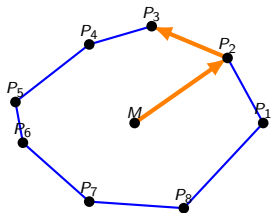
Pour qu'un polygone donné dans le sens trigonométrique  $(P_1P_2 \dots P_n)$  soit convexe, il faut et il suffit que pour tout  $1 \leq i \leq n$ ,  $P_{i-1} \rightarrow P_i \rightarrow P_{i+1}$  soit un virage à gauche.



## Point à l'intérieur d'un polygone ?

### Théorème

Un point  $M$  appartient à l'intérieur du polygone convexe  $(P_1P_2 \dots P_n)$  ssi, pour tout  $1 \leq i \leq n$ ,  $M \rightarrow P_i \rightarrow P_{i+1}$  est un virage à gauche.



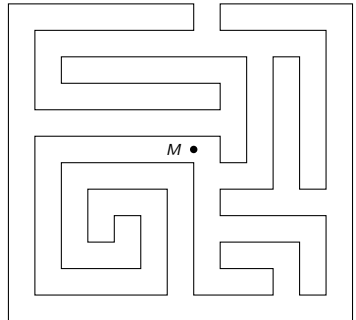
(Ex : UVa748, UVa10112)

## Point à l'intérieur d'un polygone ?

### Théorème

Un point  $M$  appartient à l'intérieur d'un polygone  $\Pi$  ssi une demi-droite issue de  $M$  intersecte les côtés de  $\Pi$  un nombre impair de fois.

Attention aux cas dégénérés : rencontre avec un sommet de  $\Pi$  ou inclusion d'une arête de  $\Pi$ .

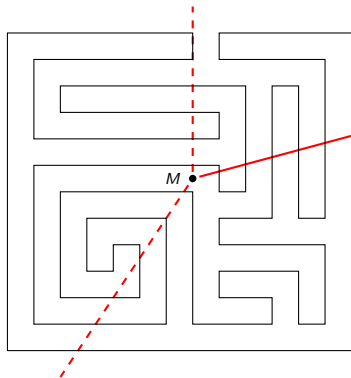


## Point à l'intérieur d'un polygone ?

### Théorème

Un point  $M$  appartient à l'intérieur d'un polygone  $\Pi$  ssi une demi-droite issue de  $M$  intersecte les côtés de  $\Pi$  un nombre impair de fois.

Attention aux cas dégénérés : rencontre avec un sommet de  $\Pi$  ou inclusion d'une arête de  $\Pi$ .



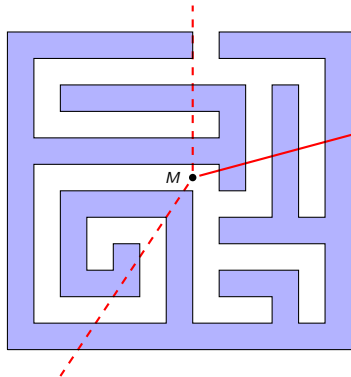


## Point à l'intérieur d'un polygone ?

### Théorème

Un point  $M$  appartient à l'intérieur d'un polygone  $\Pi$  ssi une demi-droite issue de  $M$  intersecte les côtés de  $\Pi$  un nombre impair de fois.

Attention aux cas dégénérés : rencontre avec un sommet de  $\Pi$  ou inclusion d'une arête de  $\Pi$ .





# Contenu

Les objets simples

Les polygones

Algorithmes sophistiqués

# Enveloppe convexe : la marche de Jarvis

Entrée : points  $(M_i)_{i \leq n}$ .

Sortie : Enveloppe convexe  $(P_k)_{k \leq h}$ .

Complexité :  $O(nh)$ .

(Ex : UVa10652, UVa596)

## Emballage de cadeaux

$P_0 \leftarrow \min_i \{M_i\}$  (ordre lexicographique).

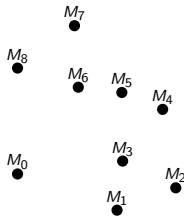
Pour  $k \geq 0$ ,

$P_{k+1} \leftarrow$  point le plus à droite de  $P_k$ .

Si  $P_{k+1} = P_0$ , s'arrêter.

NB :  $P_{k+1}$  est le plus à droite de  $P_k$  si :

$$\forall M \neq P_k, P_{k+1}, \overrightarrow{P_k P_{k+1}} \times \overrightarrow{P_k M} > 0.$$



# Enveloppe convexe : la marche de Jarvis

Entrée : points  $(M_i)_{i \leq n}$ .

Sortie : Enveloppe convexe  $(P_k)_{k \leq h}$ .

Complexité :  $O(nh)$ .

(Ex : UVa10652, UVa596)

## Emballage de cadeaux

$P_0 \leftarrow \min_i \{M_i\}$  (ordre lexicographique).

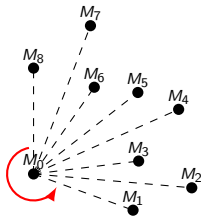
Pour  $k \geq 0$ ,

$P_{k+1} \leftarrow$  point le plus à droite de  $P_k$ .

Si  $P_{k+1} = P_0$ , s'arrêter.

NB :  $P_{k+1}$  est le plus à droite de  $P_k$  si :

$$\forall M \neq P_k, P_{k+1}, \overrightarrow{P_k P_{k+1}} \times \overrightarrow{P_k M} > 0.$$



# Enveloppe convexe : la marche de Jarvis

Entrée : points  $(M_i)_{i \leq n}$ .

Sortie : Enveloppe convexe  $(P_k)_{k \leq h}$ .

Complexité :  $O(nh)$ .

(Ex : UVa10652, UVa596)

## Emballage de cadeaux

$P_0 \leftarrow \min_i \{M_i\}$  (ordre lexicographique).

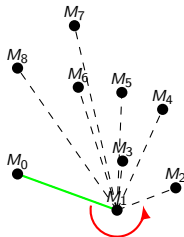
Pour  $k \geq 0$ ,

$P_{k+1} \leftarrow$  point le plus à droite de  $P_k$ .

Si  $P_{k+1} = P_0$ , s'arrêter.

NB :  $P_{k+1}$  est le plus à droite de  $P_k$  si :

$$\forall M \neq P_k, P_{k+1}, \overrightarrow{P_k P_{k+1}} \times \overrightarrow{P_k M} > 0.$$



# Enveloppe convexe : la marche de Jarvis

Entrée : points  $(M_i)_{i \leq n}$ .

Sortie : Enveloppe convexe  $(P_k)_{k \leq h}$ .

Complexité :  $O(nh)$ .

(Ex : UVa10652, UVa596)

## Emballage de cadeaux

$P_0 \leftarrow \min_i \{M_i\}$  (ordre lexicographique).

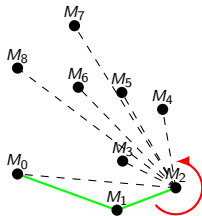
Pour  $k \geq 0$ ,

$P_{k+1} \leftarrow$  point le plus à droite de  $P_k$ .

Si  $P_{k+1} = P_0$ , s'arrêter.

NB :  $P_{k+1}$  est le plus à droite de  $P_k$  si :

$$\forall M \neq P_k, P_{k+1}, \overrightarrow{P_k P_{k+1}} \times \overrightarrow{P_k M} > 0.$$



# Enveloppe convexe : la marche de Jarvis

Entrée : points  $(M_i)_{i \leq n}$ .

Sortie : Enveloppe convexe  $(P_k)_{k \leq h}$ .

Complexité :  $O(nh)$ .

(Ex : UVa10652, UVa596)

## Emballage de cadeaux

$P_0 \leftarrow \min_i \{M_i\}$  (ordre lexicographique).

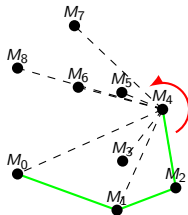
Pour  $k \geq 0$ ,

$P_{k+1} \leftarrow$  point le plus à droite de  $P_k$ .

Si  $P_{k+1} = P_0$ , s'arrêter.

NB :  $P_{k+1}$  est le plus à droite de  $P_k$  si :

$$\forall M \neq P_k, P_{k+1}, \overrightarrow{P_k P_{k+1}} \times \overrightarrow{P_k M} > 0.$$



# Enveloppe convexe : la marche de Jarvis

Entrée : points  $(M_i)_{i \leq n}$ .

Sortie : Enveloppe convexe  $(P_k)_{k \leq h}$ .

Complexité :  $O(nh)$ .

(Ex : UVa10652, UVa596)

## Emballage de cadeaux

$P_0 \leftarrow \min_i \{M_i\}$  (ordre lexicographique).

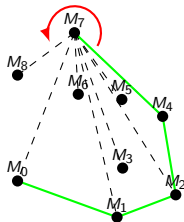
Pour  $k \geq 0$ ,

$P_{k+1} \leftarrow$  point le plus à droite de  $P_k$ .

Si  $P_{k+1} = P_0$ , s'arrêter.

NB :  $P_{k+1}$  est le plus à droite de  $P_k$  si :

$$\forall M \neq P_k, P_{k+1}, \overrightarrow{P_k P_{k+1}} \times \overrightarrow{P_k M} > 0.$$





# Enveloppe convexe : la marche de Jarvis

Entrée : points  $(M_i)_{i \leq n}$ .

Sortie : Enveloppe convexe  $(P_k)_{k \leq h}$ .

Complexité :  $O(nh)$ .

(Ex : UVa10652, UVa596)

## Emballage de cadeaux

$P_0 \leftarrow \min_i \{M_i\}$  (ordre lexicographique).

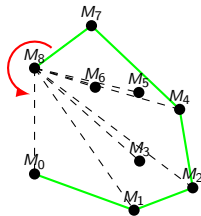
Pour  $k \geq 0$ ,

$P_{k+1} \leftarrow$  point le plus à droite de  $P_k$ .

Si  $P_{k+1} = P_0$ , s'arrêter.

NB :  $P_{k+1}$  est le plus à droite de  $P_k$  si :

$$\forall M \neq P_k, P_{k+1}, \overrightarrow{P_k P_{k+1}} \times \overrightarrow{P_k M} > 0.$$



# Enveloppe convexe : la marche de Jarvis

Entrée : points  $(M_i)_{i \leq n}$ .

Sortie : Enveloppe convexe  $(P_k)_{k \leq h}$ .

Complexité :  $O(nh)$ .

(Ex : UVa10652, UVa596)

## Emballage de cadeaux

$P_0 \leftarrow \min_i \{M_i\}$  (ordre lexicographique).

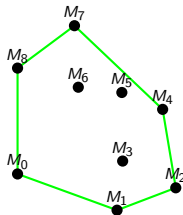
Pour  $k \geq 0$ ,

$P_{k+1} \leftarrow$  point le plus à droite de  $P_k$ .

Si  $P_{k+1} = P_0$ , s'arrêter.

NB :  $P_{k+1}$  est le plus à droite de  $P_k$  si :

$$\forall M \neq P_k, P_{k+1}, \overrightarrow{P_k P_{k+1}} \times \overrightarrow{P_k M} > 0.$$



# Enveloppe convexe : le parcours de Graham

Entrée : points  $(M_i)_{i \leq n}$ .

Sortie : Enveloppe convexe  $(P_k)_{k \leq h}$ .

Complexité :  $O(n \log n)$ .

## Graham

P.push( $\min_i \{M_i\}$ ) (ordre lexicographique).

Trier  $\{M_i\}$  par pente croissante de  $(P_0 M_i)$ .

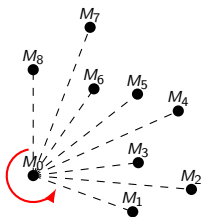
P.push( $M_1$ ).

Pour  $i \leq n$ ,

Tant que  $P_{-2} \rightarrow P_{-1} \rightarrow M_i$  en virage à droite,

P.pop().

P.push( $M_i$ ).



# Enveloppe convexe : le parcours de Graham

Entrée : points  $(M_i)_{i \leq n}$ .

Sortie : Enveloppe convexe  $(P_k)_{k \leq h}$ .

Complexité :  $O(n \log n)$ .

## Graham

P.push( $\min_i \{M_i\}$ ) (ordre lexicographique).

Trier  $\{M_i\}$  par pente croissante de  $(P_0 M_i)$ .

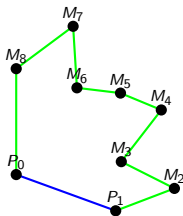
P.push( $M_1$ ).

Pour  $i \leq n$ ,

Tant que  $P_{-2} \rightarrow P_{-1} \rightarrow M_i$  en virage à droite,

P.pop().

P.push( $M_i$ ).



# Enveloppe convexe : le parcours de Graham

Entrée : points  $(M_i)_{i \leq n}$ .

Sortie : Enveloppe convexe  $(P_k)_{k \leq h}$ .

Complexité :  $O(n \log n)$ .

## Graham

P.push( $\min_i \{M_i\}$ ) (ordre lexicographique).

Trier  $\{M_i\}$  par pente croissante de  $(P_0 M_i)$ .

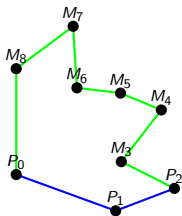
P.push( $M_1$ ).

Pour  $i \leq n$ ,

Tant que  $P_{-2} \rightarrow P_{-1} \rightarrow M_i$  en virage à droite,

P.pop().

P.push( $M_i$ ).



# Enveloppe convexe : le parcours de Graham

Entrée : points  $(M_i)_{i \leq n}$ .

Sortie : Enveloppe convexe  $(P_k)_{k \leq h}$ .

Complexité :  $O(n \log n)$ .

## Graham

P.push( $\min_i \{M_i\}$ ) (ordre lexicographique).

Trier  $\{M_i\}$  par pente croissante de  $(P_0 M_i)$ .

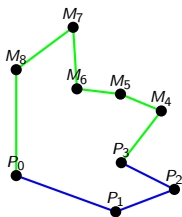
P.push( $M_1$ ).

Pour  $i \leq n$ ,

Tant que  $P_{-2} \rightarrow P_{-1} \rightarrow M_i$  en virage à droite,

P.pop().

P.push( $M_i$ ).



# Enveloppe convexe : le parcours de Graham

Entrée : points  $(M_i)_{i \leq n}$ .

Sortie : Enveloppe convexe  $(P_k)_{k \leq h}$ .

Complexité :  $O(n \log n)$ .

## Graham

P.push( $\min_i \{M_i\}$ ) (ordre lexicographique).

Trier  $\{M_i\}$  par pente croissante de  $(P_0 M_i)$ .

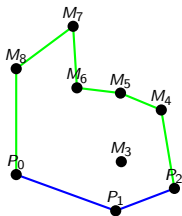
P.push( $M_1$ ).

Pour  $i \leq n$ ,

Tant que  $P_{-2} \rightarrow P_{-1} \rightarrow M_i$  en virage à droite,

P.pop().

P.push( $M_i$ ).



# Enveloppe convexe : le parcours de Graham

Entrée : points  $(M_i)_{i \leq n}$ .

Sortie : Enveloppe convexe  $(P_k)_{k \leq h}$ .

Complexité :  $O(n \log n)$ .

## Graham

P.push( $\min_i \{M_i\}$ ) (ordre lexicographique).

Trier  $\{M_i\}$  par pente croissante de  $(P_0 M_i)$ .

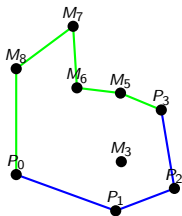
P.push( $M_1$ ).

Pour  $i \leq n$ ,

Tant que  $P_{-2} \rightarrow P_{-1} \rightarrow M_i$  en virage à droite,

P.pop().

P.push( $M_i$ ).





# Enveloppe convexe : le parcours de Graham

Entrée : points  $(M_i)_{i \leq n}$ .

Sortie : Enveloppe convexe  $(P_k)_{k \leq h}$ .

Complexité :  $O(n \log n)$ .

## Graham

P.push( $\min_i \{M_i\}$ ) (ordre lexicographique).

Trier  $\{M_i\}$  par pente croissante de  $(P_0 M_i)$ .

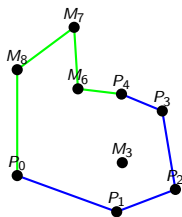
P.push( $M_1$ ).

Pour  $i \leq n$ ,

Tant que  $P_{-2} \rightarrow P_{-1} \rightarrow M_i$  en virage à droite,

P.pop().

P.push( $M_i$ ).



# Enveloppe convexe : le parcours de Graham

Entrée : points  $(M_i)_{i \leq n}$ .

Sortie : Enveloppe convexe  $(P_k)_{k \leq h}$ .

Complexité :  $O(n \log n)$ .

## Graham

P.push( $\min_i \{M_i\}$ ) (ordre lexicographique).

Trier  $\{M_i\}$  par pente croissante de  $(P_0 M_i)$ .

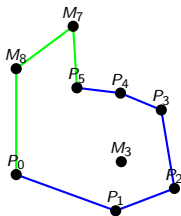
P.push( $M_1$ ).

Pour  $i \leq n$ ,

Tant que  $P_{-2} \rightarrow P_{-1} \rightarrow M_i$  en virage à droite,

P.pop().

P.push( $M_i$ ).



# Enveloppe convexe : le parcours de Graham

Entrée : points  $(M_i)_{i \leq n}$ .

Sortie : Enveloppe convexe  $(P_k)_{k \leq h}$ .

Complexité :  $O(n \log n)$ .

## Graham

P.push( $\min_i \{M_i\}$ ) (ordre lexicographique).

Trier  $\{M_i\}$  par pente croissante de  $(P_0 M_i)$ .

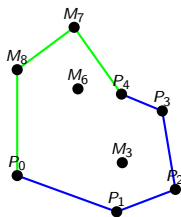
P.push( $M_1$ ).

Pour  $i \leq n$ ,

Tant que  $P_{-2} \rightarrow P_{-1} \rightarrow M_i$  en virage à droite,

P.pop().

P.push( $M_i$ ).



# Enveloppe convexe : le parcours de Graham

Entrée : points  $(M_i)_{i \leq n}$ .

Sortie : Enveloppe convexe  $(P_k)_{k \leq h}$ .

Complexité :  $O(n \log n)$ .

## Graham

P.push( $\min_i \{M_i\}$ ) (ordre lexicographique).

Trier  $\{M_i\}$  par pente croissante de  $(P_0 M_i)$ .

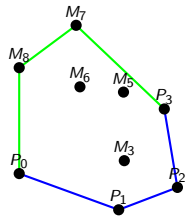
P.push( $M_1$ ).

Pour  $i \leq n$ ,

Tant que  $P_{-2} \rightarrow P_{-1} \rightarrow M_i$  en virage à droite,

P.pop().

P.push( $M_i$ ).



# Enveloppe convexe : le parcours de Graham

Entrée : points  $(M_i)_{i \leq n}$ .

Sortie : Enveloppe convexe  $(P_k)_{k \leq h}$ .

Complexité :  $O(n \log n)$ .

## Graham

P.push( $\min_i \{M_i\}$ ) (ordre lexicographique).

Trier  $\{M_i\}$  par pente croissante de  $(P_0 M_i)$ .

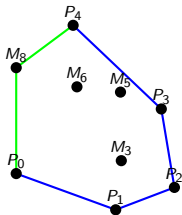
P.push( $M_1$ ).

Pour  $i \leq n$ ,

Tant que  $P_{-2} \rightarrow P_{-1} \rightarrow M_i$  en virage à droite,

P.pop().

P.push( $M_i$ ).



# Enveloppe convexe : le parcours de Graham

Entrée : points  $(M_i)_{i \leq n}$ .

Sortie : Enveloppe convexe  $(P_k)_{k \leq h}$ .

Complexité :  $O(n \log n)$ .

## Graham

P.push( $\min_i \{M_i\}$ ) (ordre lexicographique).

Trier  $\{M_i\}$  par pente croissante de  $(P_0 M_i)$ .

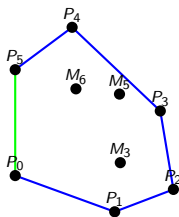
P.push( $M_1$ ).

Pour  $i \leq n$ ,

Tant que  $P_{-2} \rightarrow P_{-1} \rightarrow M_i$  en virage à droite,

P.pop().

P.push( $M_i$ ).



# Enveloppe convexe : le parcours de Graham

Entrée : points  $(M_i)_{i \leq n}$ .

Sortie : Enveloppe convexe  $(P_k)_{k \leq h}$ .

Complexité :  $O(n \log n)$ .

## Graham

P.push( $\min_i \{M_i\}$ ) (ordre lexicographique).

Trier  $\{M_i\}$  par pente croissante de  $(P_0 M_i)$ .

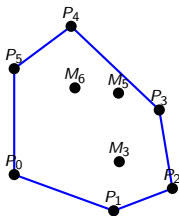
P.push( $M_1$ ).

Pour  $i \leq n$ ,

Tant que  $P_{-2} \rightarrow P_{-1} \rightarrow M_i$  en virage à droite,

P.pop().

P.push( $M_i$ ).



# Plus proches points du plan euclidien

Entrée : points  $(M_i)_{i \leq n}$ .

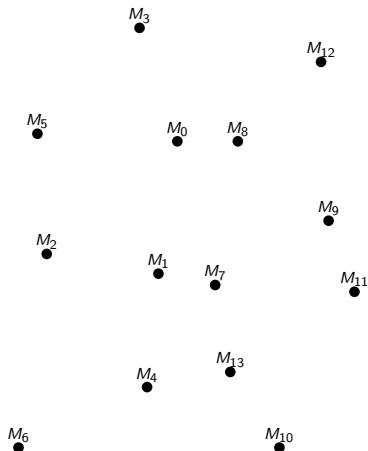
Sortie :  $\min_{i \neq j} d(M_i, M_j)$ .

Complexité :  $O(n \log n)$ .

(Ex : UVa10245, UVa11378)

## Algorithme diviser pour régner

- ▶ Classer les points par abscisse croissante.  
Les scinder par la droite médiane  $\Delta$ .
- ▶ Résoudre à gauche et à droite récursivement. Minima :  $d_g$  et  $d_d$ .
- ▶ Dans une fenêtre balayante de taille  $(2\delta, \delta)$ , où  $\delta = \min(d_g, d_d)$ , centrée en  $\Delta$ , chercher la distance minimum  $d_\Delta$  entre un point gauche et droit.
- ▶ Renvoyer  $\min(d_g, d_d, d_\Delta)$ .





# Plus proches points du plan euclidien

Entrée : points  $(M_i)_{i \leq n}$ .

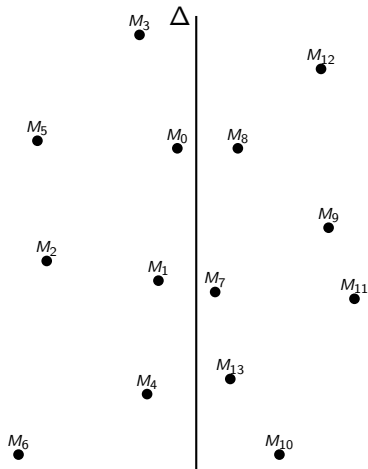
Sortie :  $\min_{i \neq j} d(M_i, M_j)$ .

Complexité :  $O(n \log n)$ .

(Ex : UVa10245, UVa11378)

## Algorithme diviser pour régner

- ▶ Classer les points par abscisse croissante.  
Les scinder par la droite médiane  $\Delta$ .
- ▶ Résoudre à gauche et à droite récursivement. Minima :  $d_g$  et  $d_d$ .
- ▶ Dans une fenêtre balayante de taille  $(2\delta, \delta)$ , où  $\delta = \min(d_g, d_d)$ , centrée en  $\Delta$ , chercher la distance minimum  $d_\Delta$  entre un point gauche et droit.
- ▶ Renvoyer  $\min(d_g, d_d, d_\Delta)$ .



## Plus proches points du plan euclidien

Entrée : points  $(M_i)_{i \leq n}$ .

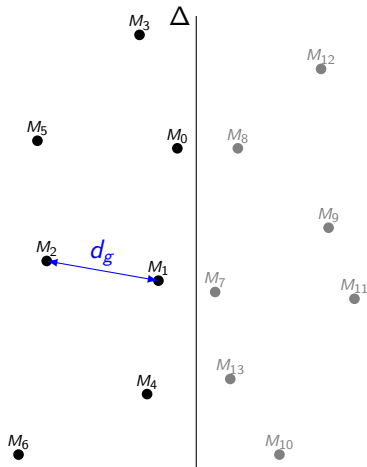
Sortie :  $\min_{i \neq j} d(M_i, M_j)$ .

Complexité :  $O(n \log n)$ .

(Ex : UVa10245, UVa11378)

### Algorithme diviser pour régner

- ▶ Classer les points par abscisse croissante.  
Les scinder par la droite médiane  $\Delta$ .
- ▶ Résoudre à gauche et à droite récursivement. Minima :  $d_g$  et  $d_d$ .
- ▶ Dans une fenêtre balayante de taille  $(2\delta, \delta)$ , où  $\delta = \min(d_g, d_d)$ , centrée en  $\Delta$ , chercher la distance minimum  $d_\Delta$  entre un point gauche et droit.
- ▶ Renvoyer  $\min(d_g, d_d, d_\Delta)$ .



# Plus proches points du plan euclidien

Entrée : points  $(M_i)_{i \leq n}$ .

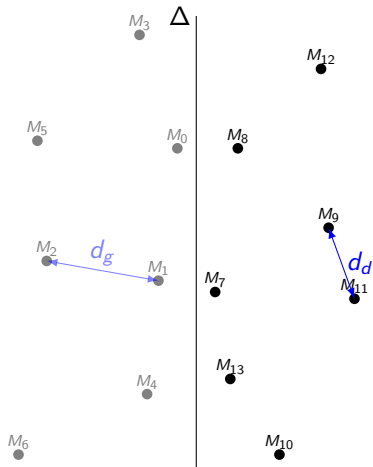
Sortie :  $\min_{i \neq j} d(M_i, M_j)$ .

Complexité :  $O(n \log n)$ .

(Ex : UVa10245, UVa11378)

## Algorithme diviser pour régner

- ▶ Classer les points par abscisse croissante.  
Les scinder par la droite médiane  $\Delta$ .
- ▶ Résoudre à gauche et à droite récursivement. Minima :  $d_g$  et  $d_d$ .
- ▶ Dans une fenêtre balayante de taille  $(2\delta, \delta)$ , où  $\delta = \min(d_g, d_d)$ , centrée en  $\Delta$ , chercher la distance minimum  $d_\Delta$  entre un point gauche et droit.
- ▶ Renvoyer  $\min(d_g, d_d, d_\Delta)$ .



# Plus proches points du plan euclidien

Entrée : points  $(M_i)_{i \leq n}$ .

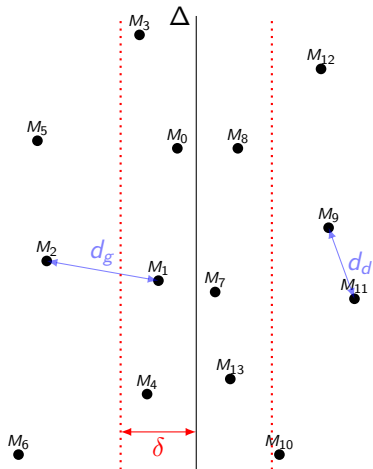
Sortie :  $\min_{i \neq j} d(M_i, M_j)$ .

Complexité :  $O(n \log n)$ .

(Ex : UVa10245, UVa11378)

## Algorithme diviser pour régner

- ▶ Classer les points par abscisse croissante.  
Les scinder par la droite médiane  $\Delta$ .
- ▶ Résoudre à gauche et à droite récursivement. Minima :  $d_g$  et  $d_d$ .
- ▶ Dans une fenêtre balayante de taille  $(2\delta, \delta)$ , où  $\delta = \min(d_g, d_d)$ , centrée en  $\Delta$ , chercher la distance minimum  $d_\Delta$  entre un point gauche et droit.
- ▶ Renvoyer  $\min(d_g, d_d, d_\Delta)$ .



# Plus proches points du plan euclidien

Entrée : points  $(M_i)_{i \leq n}$ .

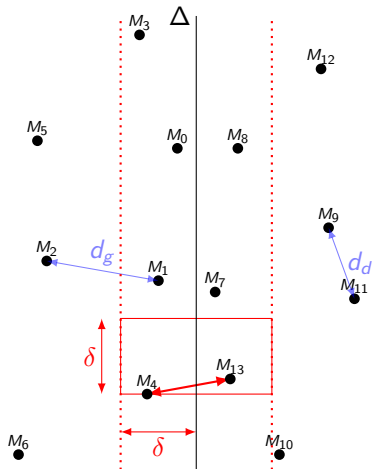
Sortie :  $\min_{i \neq j} d(M_i, M_j)$ .

Complexité :  $O(n \log n)$ .

(Ex : UVa10245, UVa11378)

## Algorithme diviser pour régner

- ▶ Classer les points par abscisse croissante.  
Les scinder par la droite médiane  $\Delta$ .
- ▶ Résoudre à gauche et à droite récursivement. Minima :  $d_g$  et  $d_d$ .
- ▶ Dans une fenêtre balayante de taille  $(2\delta, \delta)$ , où  $\delta = \min(d_g, d_d)$ , centrée en  $\Delta$ , chercher la distance minimum  $d_\Delta$  entre un point gauche et droit.
- ▶ Renvoyer  $\min(d_g, d_d, d_\Delta)$ .



# Plus proches points du plan euclidien

Entrée : points  $(M_i)_{i \leq n}$ .

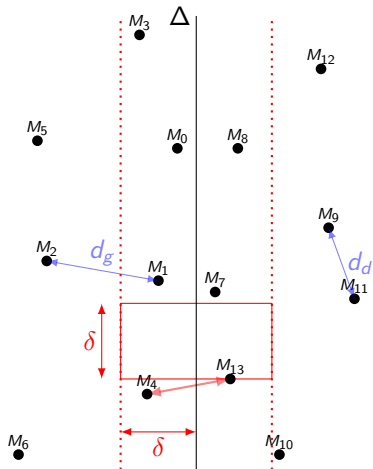
Sortie :  $\min_{i \neq j} d(M_i, M_j)$ .

Complexité :  $O(n \log n)$ .

(Ex : UVa10245, UVa11378)

## Algorithme diviser pour régner

- ▶ Classer les points par abscisse croissante.  
Les scinder par la droite médiane  $\Delta$ .
- ▶ Résoudre à gauche et à droite récursivement. Minima :  $d_g$  et  $d_d$ .
- ▶ Dans une fenêtre balayante de taille  $(2\delta, \delta)$ , où  $\delta = \min(d_g, d_d)$ , centrée en  $\Delta$ , chercher la distance minimum  $d_\Delta$  entre un point gauche et droit.
- ▶ Renvoyer  $\min(d_g, d_d, d_\Delta)$ .



# Plus proches points du plan euclidien

Entrée : points  $(M_i)_{i \leq n}$ .

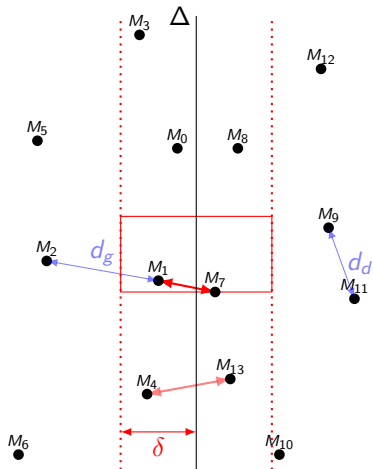
Sortie :  $\min_{i \neq j} d(M_i, M_j)$ .

Complexité :  $O(n \log n)$ .

(Ex : UVa10245, UVa11378)

## Algorithme diviser pour régner

- ▶ Classer les points par abscisse croissante.  
Les scinder par la droite médiane  $\Delta$ .
- ▶ Résoudre à gauche et à droite récursivement. Minima :  $d_g$  et  $d_d$ .
- ▶ Dans une fenêtre balayante de taille  $(2\delta, \delta)$ , où  $\delta = \min(d_g, d_d)$ , centrée en  $\Delta$ , chercher la distance minimum  $d_\Delta$  entre un point gauche et droit.
- ▶ Renvoyer  $\min(d_g, d_d, d_\Delta)$ .



# Plus proches points du plan euclidien

Entrée : points  $(M_i)_{i \leq n}$ .

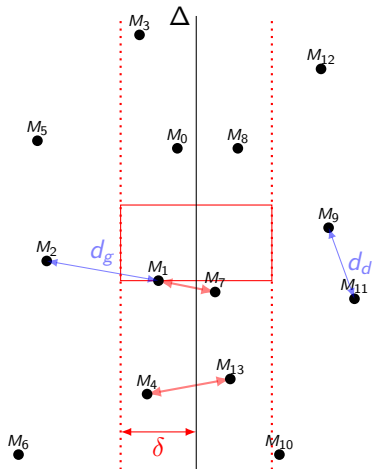
Sortie :  $\min_{i \neq j} d(M_i, M_j)$ .

Complexité :  $O(n \log n)$ .

(Ex : UVa10245, UVa11378)

## Algorithme diviser pour régner

- ▶ Classer les points par abscisse croissante.  
Les scinder par la droite médiane  $\Delta$ .
- ▶ Résoudre à gauche et à droite récursivement. Minima :  $d_g$  et  $d_d$ .
- ▶ Dans une fenêtre balayante de taille  $(2\delta, \delta)$ , où  $\delta = \min(d_g, d_d)$ , centrée en  $\Delta$ , chercher la distance minimum  $d_\Delta$  entre un point gauche et droit.
- ▶ Renvoyer  $\min(d_g, d_d, d_\Delta)$ .





# Plus proches points du plan euclidien

Entrée : points  $(M_i)_{i \leq n}$ .

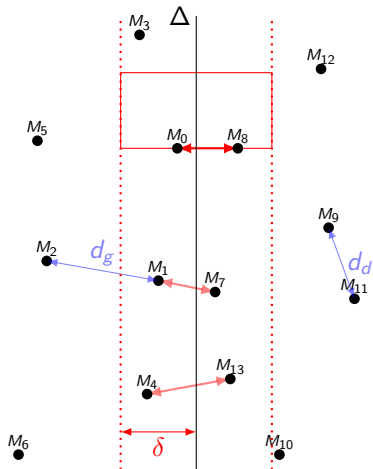
Sortie :  $\min_{i \neq j} d(M_i, M_j)$ .

Complexité :  $O(n \log n)$ .

(Ex : UVa10245, UVa11378)

## Algorithme diviser pour régner

- ▶ Classer les points par abscisse croissante.  
Les scinder par la droite médiane  $\Delta$ .
- ▶ Résoudre à gauche et à droite récursivement. Minima :  $d_g$  et  $d_d$ .
- ▶ Dans une fenêtre balayante de taille  $(2\delta, \delta)$ , où  $\delta = \min(d_g, d_d)$ , centrée en  $\Delta$ , chercher la distance minimum  $d_\Delta$  entre un point gauche et droit.
- ▶ Renvoyer  $\min(d_g, d_d, d_\Delta)$ .



# Plus proches points du plan euclidien

Entrée : points  $(M_i)_{i \leq n}$ .

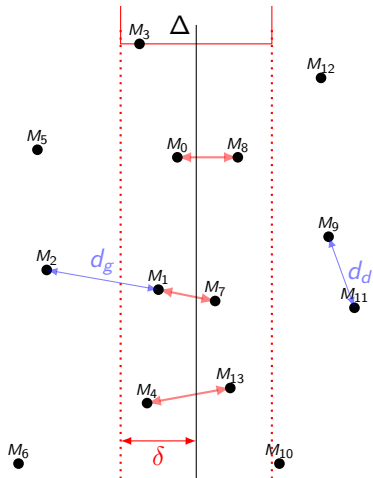
Sortie :  $\min_{i \neq j} d(M_i, M_j)$ .

Complexité :  $O(n \log n)$ .

## Algorithme diviser pour régner

- ▶ Classer les points par abscisse croissante.  
Les scinder par la droite médiane  $\Delta$ .
- ▶ Résoudre à gauche et à droite récursivement. Minima :  $d_g$  et  $d_d$ .
- ▶ Dans une fenêtre balayante de taille  $(2\delta, \delta)$ , où  $\delta = \min(d_g, d_d)$ , centrée en  $\Delta$ , chercher la distance minimum  $d_\Delta$  entre un point gauche et droit.
- ▶ Renvoyer  $\min(d_g, d_d, d_\Delta)$ .

(Ex : UVa10245, UVa11378)



# Plus proches points du plan euclidien

Entrée : points  $(M_i)_{i \leq n}$ .

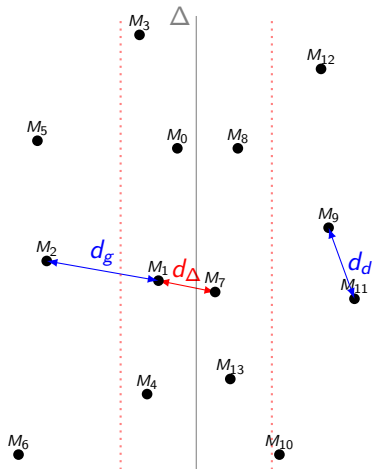
Sortie :  $\min_{i \neq j} d(M_i, M_j)$ .

Complexité :  $O(n \log n)$ .

(Ex : UVa10245, UVa11378)

## Algorithme diviser pour régner

- ▶ Classer les points par abscisse croissante.  
Les scinder par la droite médiane  $\Delta$ .
- ▶ Résoudre à gauche et à droite récursivement. Minima :  $d_g$  et  $d_d$ .
- ▶ Dans une fenêtre balayante de taille  $(2\delta, \delta)$ , où  $\delta = \min(d_g, d_d)$ , centrée en  $\Delta$ , chercher la distance minimum  $d_\Delta$  entre un point gauche et droit.
- ▶ Renvoyer  $\min(d_g, d_d, d_\Delta)$ .





## Crédits

- ▶ Transparents développés pour la préparation au concours à Hong Kong University 2010 :  
<https://i.cs.hku.hk/~provinci/>
- ▶ Dessins sphériques :  
<https://tex.stackexchange.com/questions/46850/>
- ▶ [Halim&Halim, *Competitive programming 3*] pour les pointeurs vers UVa