

Cours 10

Géométrie algorithmique

Jean-Jacques.Levy@inria.fr

<http://jeanjacqueslevy.net>

tel: 01 39 63 56 89

secrétariat de l'enseignement:

Catherine Bensoussan

cb@lix.polytechnique.fr

Aile 00, LIX

tel: 01 69 33 34 67

<http://www.enseignement.polytechnique.fr/informatique/>

Plan

1. Entrée graphiques
2. Enveloppe convexe
3. Recherche de points dans des intervalles
4. Intersection de segments orthogonaux
5. Intersection de segments

Click – Double Click

- **Front descendant (ne pas oublier d'attendre la remontée)**

```
for(;;) {  
    while (!g.button())  
        ;  
    p = getMouse ();  
    while (g.button())  
        ;  
    action (p.h, p.v);  
}
```

- **Front montant (plus sûr)**

```
for (;;) {  
    while (!g.button())  
        ;  
    while (g.button())  
        ;  
    p = getMouse ();  
    action (p.h, p.v);  
}
```

Toute information supplémentaire se trouve en

<http://www.enseignement.polytechnique.fr/profs/informatique/Philippe.Chassignet/MACLIB/Java/doc/tree.html>

Programmation des événements clavier/souris

- `in.readLine()` est bloquant \Rightarrow problème pour lire simultanément le clavier et la souris.
- problème bien connu de la **séquentialité**. Il faut savoir gérer l'**asynchronisme**.
- solution 1: avoir un read non bloquant
- solution 2: faire des threads et gérer du parallélisme asynchrone.
- solution 3: avoir une structure d'événements gérés par le système. Par exemple le driver d'événements de X-window.
- plus sophistiqué: des callbacks ou des langages pour gérer les interactions (Squeak, Cardelli-Pike) ou en déterminisant les interactions (Esterel, Berry-Cosserat-Gonthier).
- pour double-click, il faut estampiller les événements par le temps.

Ordre trigonométrique

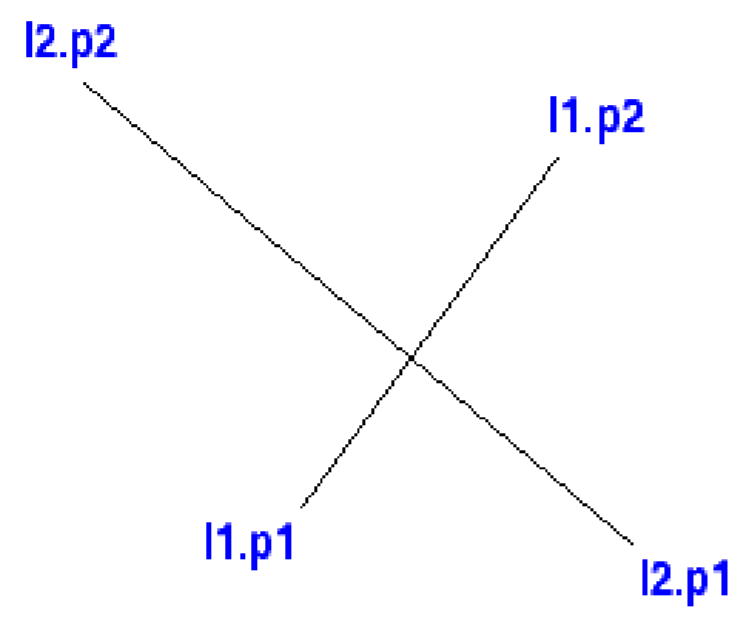
On cherche à savoir si l'angle $\widehat{P_1P_0P_2} > 0$?

En calculant le produit vectoriel $\overrightarrow{P_0P_1} \wedge \overrightarrow{P_0P_2}$. Si l'angle est nul, par convention on compare les normes.

```
static int ccw (Point p0, Point p1, Point p2) {
    int dx1 = p1.x - p0.x; int dy1 = p1.y - p0.y;
    int dx2 = p2.x - p0.x; int dy2 = p2.y - p0.y;
    if (dx1 * dy2 > dy1 * dx2) return 1;
    else if (dx1 * dy2 < dy1 * dx2) return -1;
    else {
        if (dx1 * dx2 < 0 || dy1 * dy2 < 0) return -1;
        else if (dx1*dx1 + dy1*dy1 >= dx2*dx2 + dy2*dy2) return 0;
        else return 1;
    }
}
```

Intersection de segments

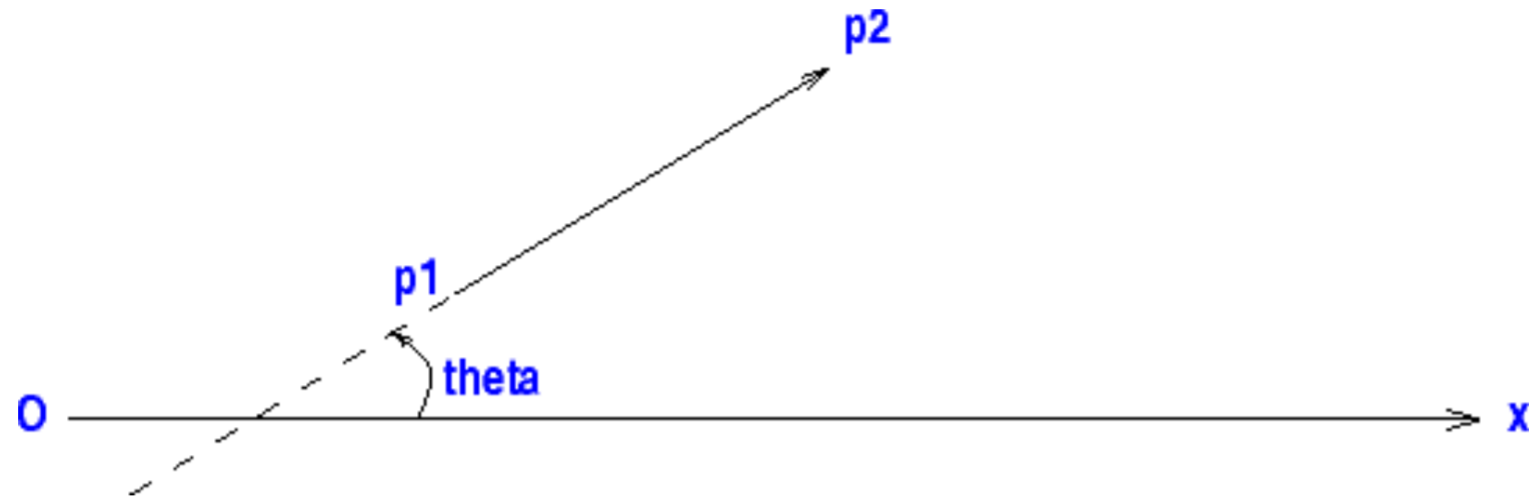
```
static boolean intersect (Line l1, Line l2) {  
    return ccw (l1.p1, l1.p2, l2.p1) * ccw (l1.p1, l1.p2, l2.p2) <= 0  
        && ccw (l2.p1, l2.p2, l1.p1) * ccw (l2.p1, l2.p2, l1.p2) <= 0;  
}
```



Pente d'un vecteur

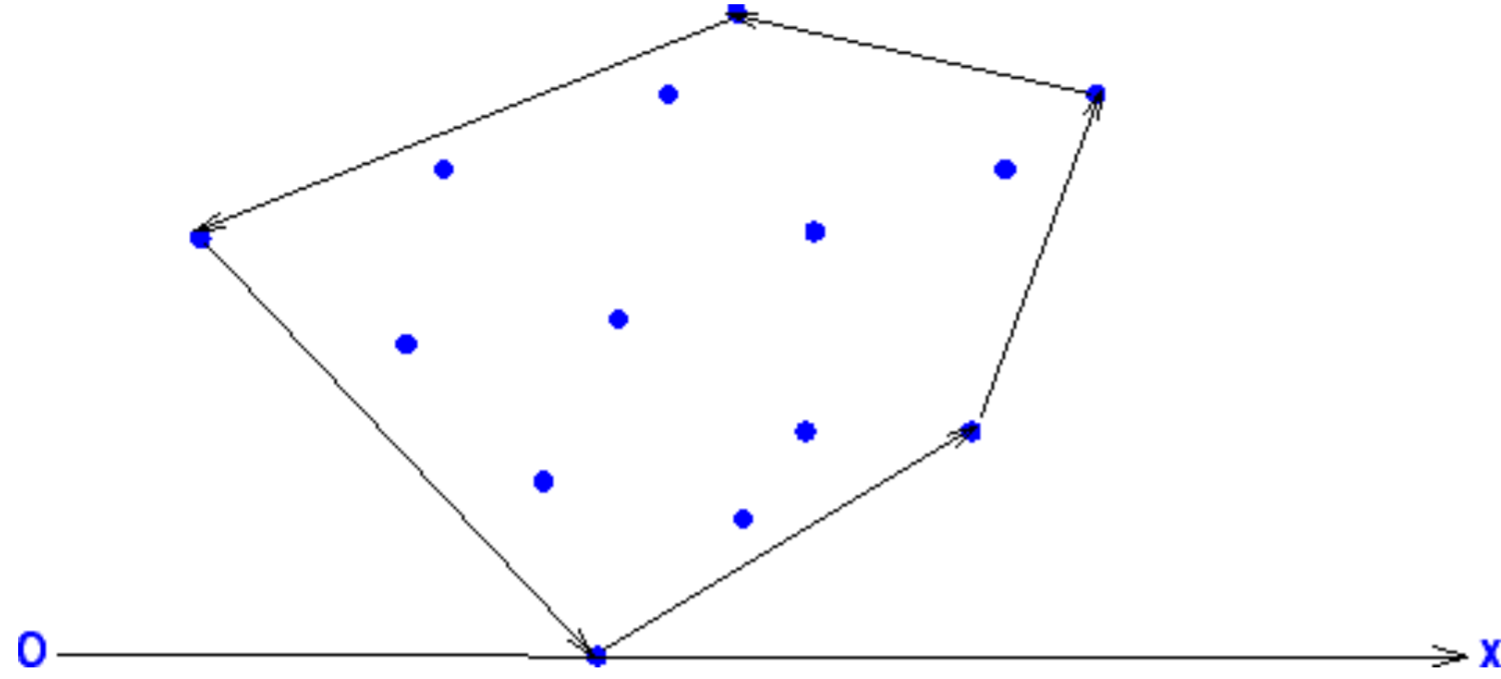
```
static float theta (Point p1, Point p2) {  
    float t; int dx = p2.x - p1.x; int dy = p2.y - p1.y;  
    if (dx == 0 && dy == 0) t = 0;  
    else t = (float) dy / (Math.abs(dx) + Math.abs(dy));  
    if (dx < 0) t = 2 - t;  
    else if (dy < 0) t = 4 + t;  
    return t * 90.f;  
}
```

Cette fonction évite le long calcul de `Math.atan(dy/dx)`.



Enveloppe convexe (marche de Jarvis)

- Chercher un point avec y minimum.
- Chercher les points successifs de l'enveloppe dans l'ordre trigonométrique.
- Un point P_{m+1} sur l'enveloppe est le point P_i dont l'angle $(\overrightarrow{P_m P_{m-1}}, \overrightarrow{P_m P_i})$ est minimum et positif.



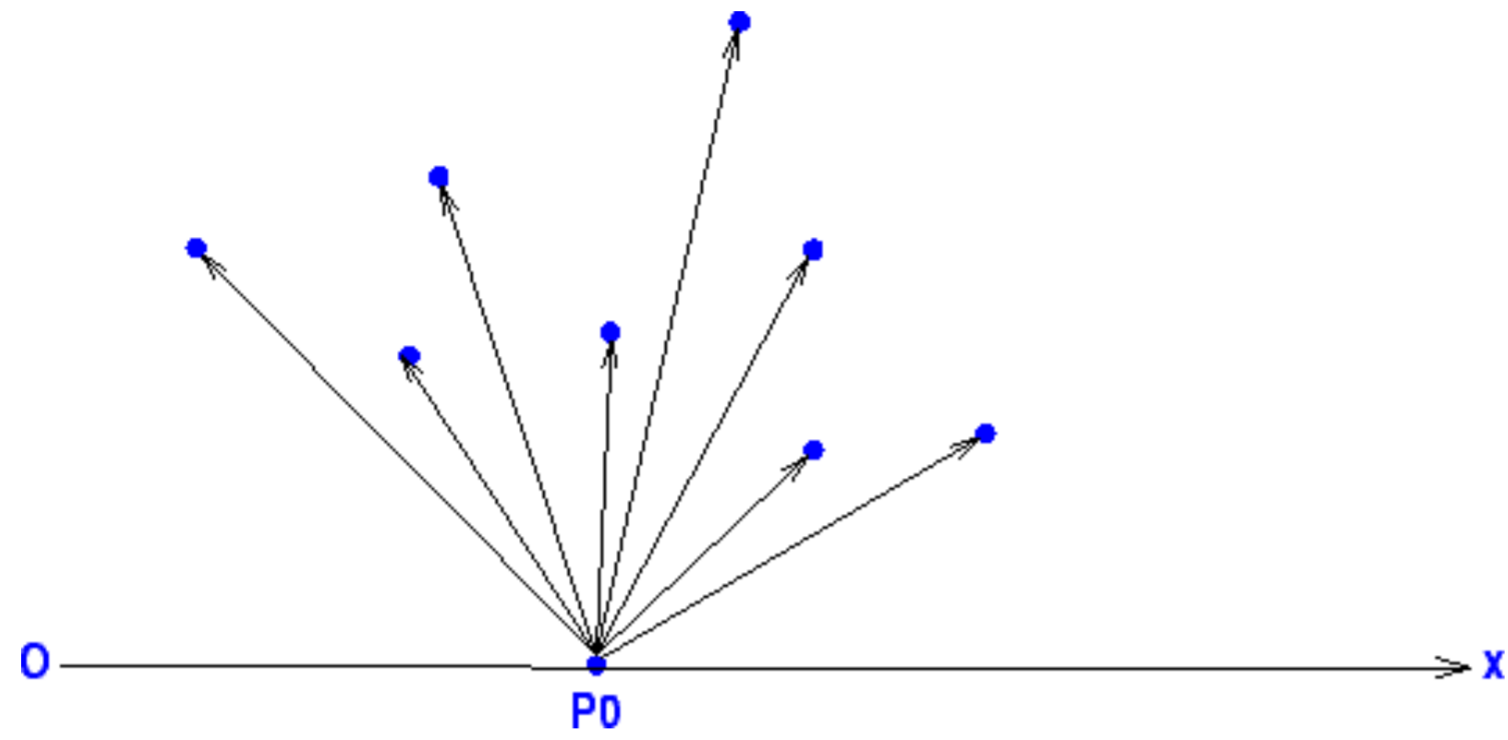
Enveloppe convexe (marche de Jarvis)

```
static int enveloppe (Point[] p) {
    int m = 0; int n = p.length;
    if (n > 0) {
        int min = 0;
        for (int i = 1; i < n; ++i) if (p[i].y < p[min].y) min = i;
        float angle1 = 0, angle2 = 360; do {
            Point t = p[m]; p[m] = p[min]; p[min] = t;
            ++m; min = 0;
            for (int i = m; i < n; ++i) {
                float alpha = theta(p[m-1], p[i]);
                if (angle1 < alpha && alpha < angle2) {
                    min = i; angle2 = alpha;
                }
            }
            angle1 = angle2; angle2 = theta(p[min], p[0]);
        } while (min != 0);
    }
    return m;
}
```

Exercice 1 Complexité?

Enveloppe convexe (marche de Graham)

- Chercher un point P_0 avec y minimum.
- Trier les points P_i sur l'angle formé avec P_0 .
- Partir de ce point en tournant toujours à gauche.



Enveloppe convexe (marche de Graham)

```
static int enveloppe (Point[] p) {
    int m = 0; int n = p.length;
    if (n <= 2) return n;
    else {
        int min = 0;
        for (int i = 1; i < n; ++i) if (p[i].y < p[min].y) min = i;
        for (int i = 0; i < n; ++i)
            if (p[i].y == p[min].y && p[i].x > p[min].x) min = i;
        Point t = p[0]; p[0] = p[min]; p[min] = t;
        tri(p); m = 2;
        for (int i = 3; i < n; ++i) {
            while (ccw(p[m], p[m-1], p[i]) >= 0)
                --m;
            ++m;
            t = p[m]; p[m] = p[i]; p[i] = t;
        }
        return m+1;
    }
}
```

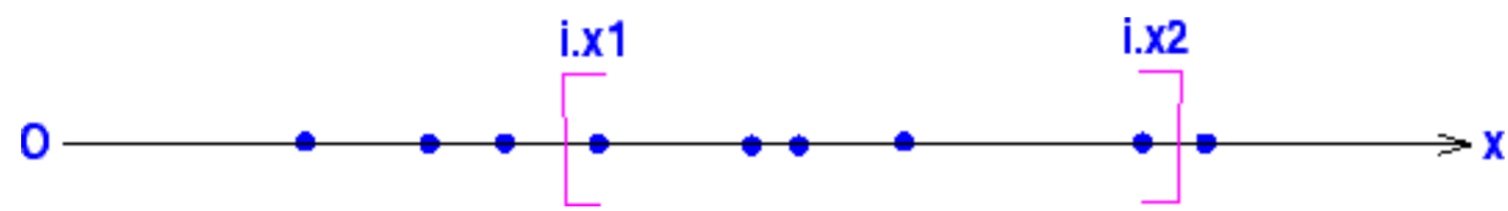
Exercice 2 Complexité?

Recherche de points dans des intervalles

- Dimension 1: représenter les points avec un arbre de recherche.

```
static void rechercher (Arbre a, intervalle i) {  
    if (a != null) {  
        boolean bg = i.x1 <= a.x, bd = a.x <= i.x2;  
        if (bg) rechercher (a.gauche, i);  
        if (bg && bd) System.out.print (a.x + " ");  
        if (bd) rechercher (a.droite, i);  
    }  
}
```

Exercice 3 Complexité?



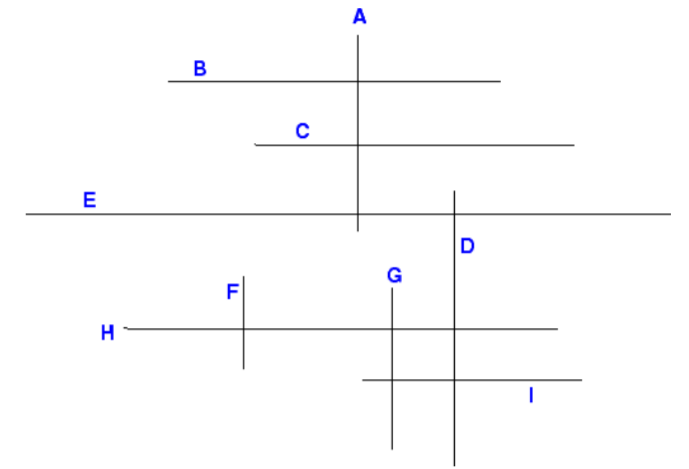
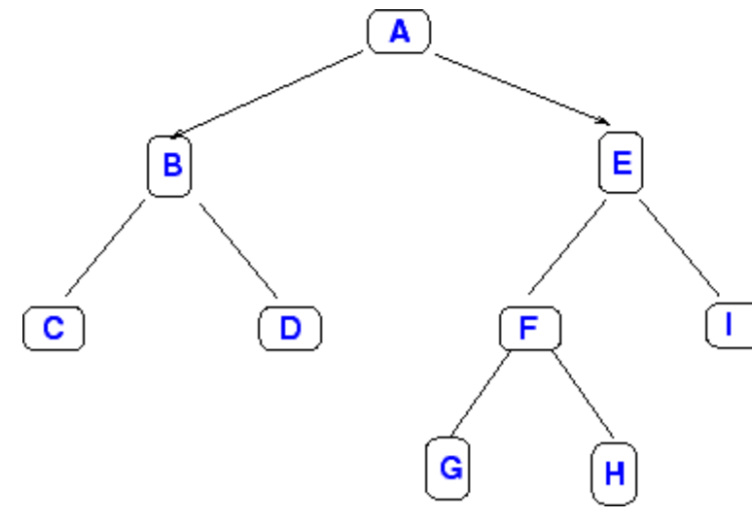
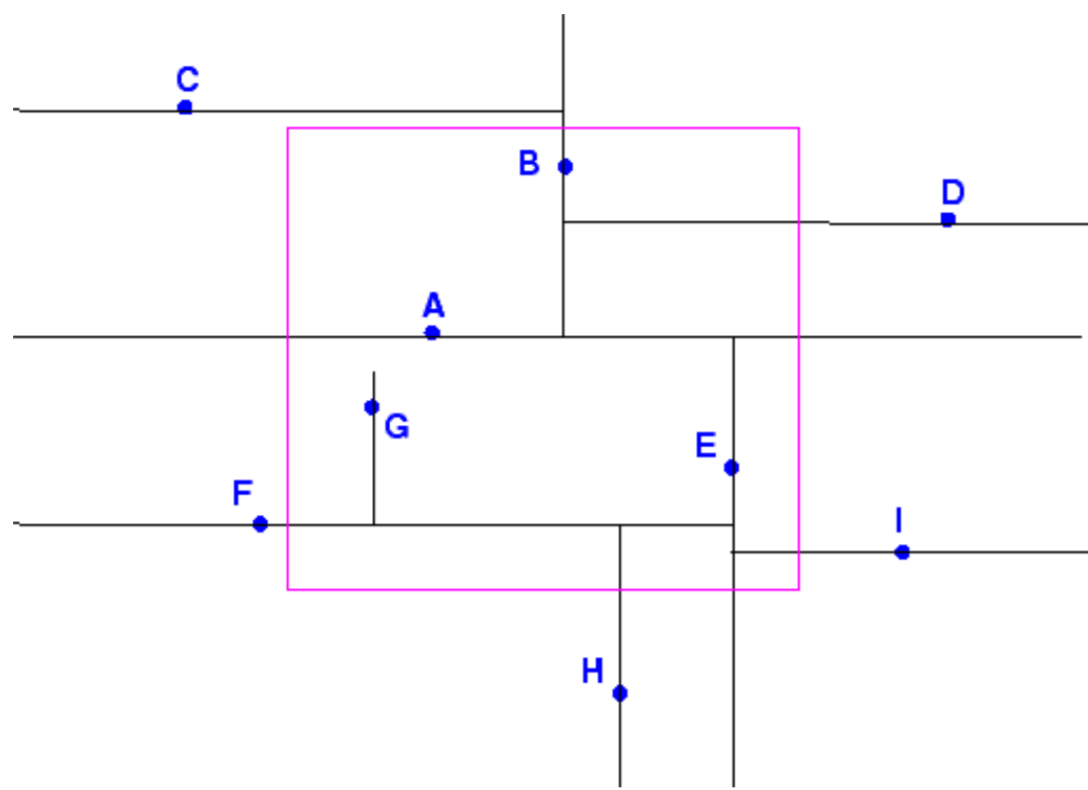
Recherche de points dans des intervalles

- Dimension 2: arbre de recherche en alternant le rangement sur x et y .

```
static void rechercher (Arbre a, Rect r, Boolean d) {
    boolean t1, t2;
    if (a != null) {
        boolean bx1 = r.x1 <= a.p.x, bx2 = a.p.x <= r.x2;
        boolean by1 = r.y1 <= a.p.y, by2 = a.p.y <= r.y2;
        if (d) { b1 = bx1; b2 = bx2; }
        else { b1 = by1; b2 = by2; }
        if (b1)
            rechercher (a.gauche, r, !d);
        if (dansRect (a.p, r))
            System.out.print (a.p + " ");
        if (b2)
            rechercher (a.droite, r, !d);
    }
}
```

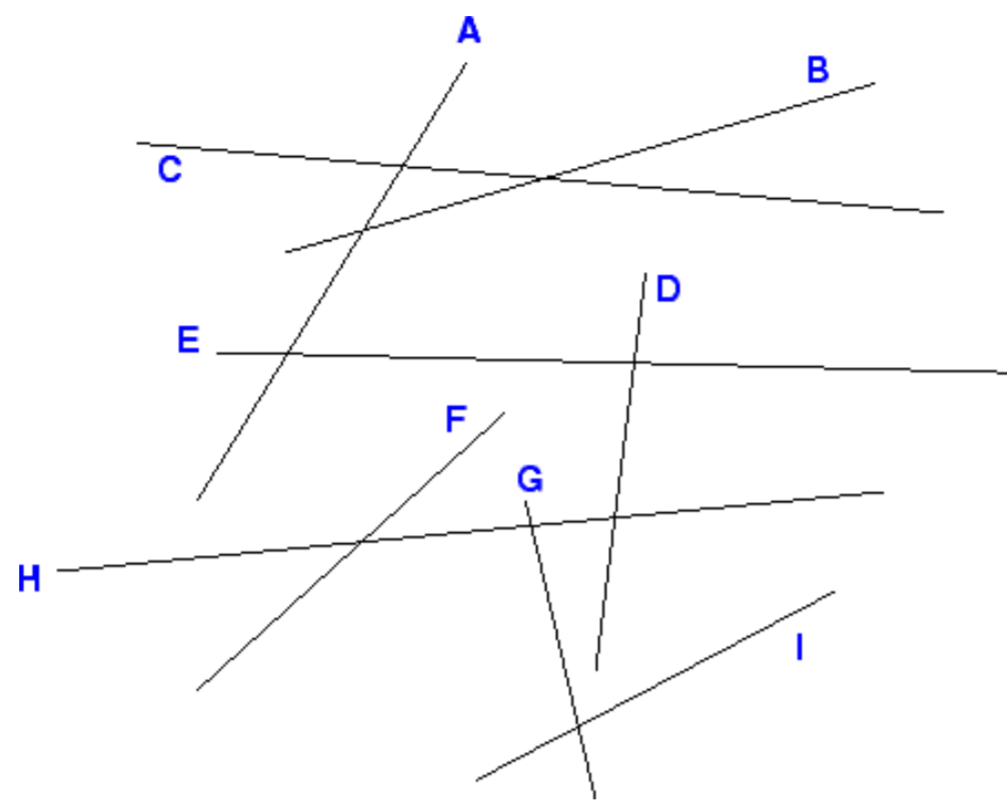
Exercice 4 Complexité?

Graphiquement



Intersection de segments orthogonaux

- On balaie le plan avec une ligne horizontale de bas en haut *scan line*. Il faut donc trier les extrémités des segments sur y .
- A chaque point début de segment vertical, on rajoute dans l'arbre de recherche sa coordonnée x .
- A chaque segment horizontal, on fait une recherche des points dans l'intervalle correspondant au segment.
- A chaque fin de segment vertical, on retire de l'arbre de recherche la coordonnée x .
- $O(k + N \log N)$.



Intersection de segments quelconques (Shamos-Hoey)

Recherche d'au moins 1 intersection:

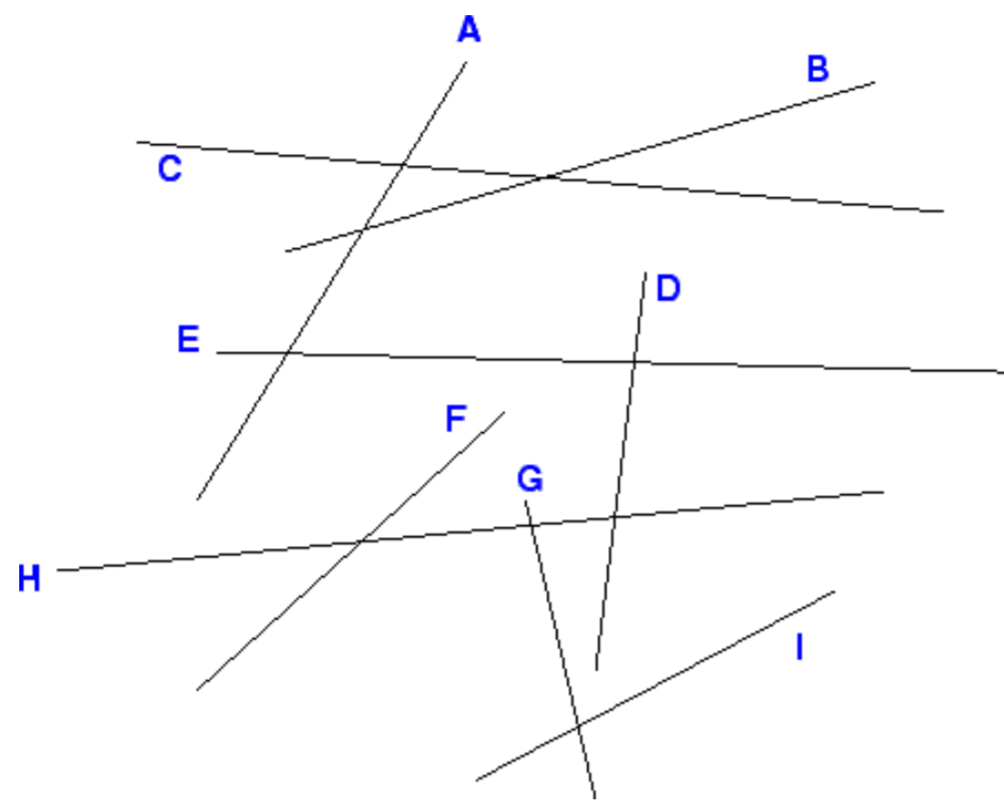
- On balaie le plan avec une ligne horizontale de bas en haut scan line. Il faut donc trier les extrémités des segments. Soit Q cet ensemble. Au début, $R = \emptyset$.
- A chaque point p dans Q dans l'ordre des y croissants,
 1. si p est le début du segment s . On insère s dans R . On teste si s intersecte le segment de gauche ou de droite dans R , et on retourne cette intersection.
 2. si p est la fin du segment s . On teste si les segments à gauche de s et à droite de s dans R s'intersectent, et on retourne cette intersection. On enlève s de R .
- $O(N \log N)$.

Intersections de segments quelconques (Bentley-Ottmann, 79)

- On balaie le plan avec une ligne horizontale de bas en haut scan line. Il faut donc trier les extrémités des segments. Soit Q cet ensemble. Au début, $R = \emptyset$.
- A chaque point p dans Q dans l'ordre des y croissants,
 1. si p est le début du segment s . On insère s dans R . Si s intersecte le segment de gauche ou de droite t dans R , on rajoute le point d'intersection de s et t dans Q (en respectant l'ordre des y croissants).
 2. si p est la fin du segment s . Si l'intersection des segments à gauche de s et à droite de s dans R n'est pas dans Q , on teste l'intersection et on l'ajoute à Q . On enlève s de R .
 3. si p est l'intersection de s et t , on écrit p et on échange s et t dans R . (Remarque: ils sont alors adjacents). On teste si le segment de gauche s s'intersecte avec le segment à gauche de lui dans R , et le segment à droite de t , et on rajoute cette intersection à Q .

Intersections de segments quelconques (Bentley-Ottmann, 79)

- $O(N \log N + k \log N)$, en utilisant des arbres équilibrés pour R et Q comme une file de priorité.



La vision et la synthèse d'images sont enseignées en Majeure 1 et 2.

Problèmes aux quels on a échappé

- compression
- flots dans les graphes
- analyse d'algorithme
- problèmes NP complets
- complexité abstraite (P-SPACE, NC, etc)
- algorithmes randomisés
- géométrie algorithmique
- algorithmes distribués
- correction des programmes
- concurrence
- programmation système
- relations et bases de données
- compilation ...

N'oubliez pas les majeures!

Majeure 1: Algèbre et informatique

Majeure 2: Informatique

Travaillez votre
projet info
avec
mesure