Chapitre 1

Géométrie algorithmique

1.1 Distance minimale dans un nuage de points dans $\mathbb{R}^2$

Voir avec les scribes de la semaine passée.

1.2 Enveloppe convexe d'un ensemble fini de points de $\mathbb{R}^2$

Définition 1.1 L'enveloppe convexe d'un ensemble fini de points de $\mathbb{R}^2$ est la frontière de la plus petite (au sens de l'inclusion) partie convexe contenant tous les points.

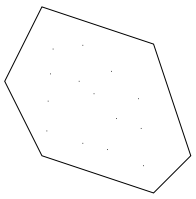

FIG. 1.1 – *Enveloppe convexe*

Remarque 1.1 Pour un ensemble fini de points, l'enveloppe convexe est un polygone.

Il y a plusieurs types d'algorithmes :
- incrémentaux, par balayage ;
- diviser pour régner ;
- par greffes successives (par exemple, on essaie de calculer des enveloppes incluses les unes dans les autres (voir FIG. 1.2)).

1.2.1 Algorithme naïf incrémental

Algorithme de base et structures de données

Points : $\forall i \in [\![1, n]\!], p_i = (x_i, y_i)$ dans un ordre quelconque.

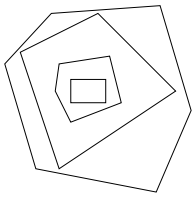

FIG. 1.2 – *Greffes successives*

Algorithme :

1. $env\_conv(p_1) \leftarrow (p_1)$
2. Pour i de 2 à n faire
3. $\quad env\_conv(p_1, \ldots, p_i) \leftarrow$ `ajout`$(env\_conv(p_1, \ldots, p_{i-1}), p_i)$
4. FinPour

Remarque 1.2 On note tous les points de l'enveloppe convexe dans l'ordre trigonométrique.

Test pour savoir si un point est à l'intérieur

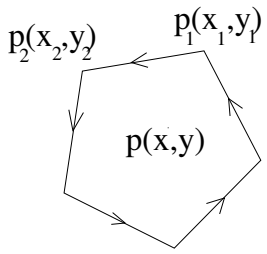

FIG. 1.3 – *Test*

On regarde le signe de :

$$\begin{vmatrix} x_2 - x_1 & x - x_1 \\ y_2 - y_1 & y - y_1 \end{vmatrix} = \|\overrightarrow{p_1p_2}\|.\|\overrightarrow{p_1p}\|.\sin(\overrightarrow{p_1p_2}, \overrightarrow{p_1p})$$

qui doit être positif.

Ajout

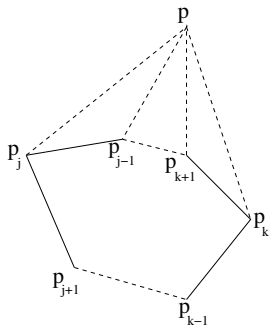

FIG. 1.4 – *Ajout*

`ajout`(C, p) =

1. Si p est à l'intérieur de C
2. Renvoyer C
3. Sinon
4. Trouver le segment $[p_i, p_{i+1}]$ le plus proche de p ;
5. Rechercher $p_j, j \geqslant i+1$, le premier sommet tel que $(\overrightarrow{pp_j}, \overrightarrow{pp_{j+1}}) \geqslant 0$ et $p_k, k \leqslant i$, le premier sommet tel que $(\overrightarrow{pp_k}, \overrightarrow{p_k p_{k-1}}) \leqslant 0$;
6. Renvoyer $(p, p_j, p_{j+1}, \ldots, p_{k-1}, p_k)$.

Complexité

La complexité est en $O(n^2)$:

$$n \times \begin{cases} \text{test intérieur} : O(n) \\ \text{segment le plus proche} : O(n) \\ \text{premier sommet} : O(n) \\ \ldots \end{cases} \Rightarrow O(n^2)$$

1.2.2 Amélioration : algorithme incrémental avec balayage de gauche à droite

Algorithme de base et structures de données

On utilise les mêmes structures de données.
Algorithme :

1. Tri des points par abscisses croissantes
2. $env\_conv(p_1) \leftarrow (p_1)$
3. Pour i de 2 à n faire
4. $env\_conv(p_1, \ldots, p_i) \leftarrow$ `ajout`$(env\_conv(p_1, \ldots, p_{i-1}), p_i)$
5. FinPour

Ajout

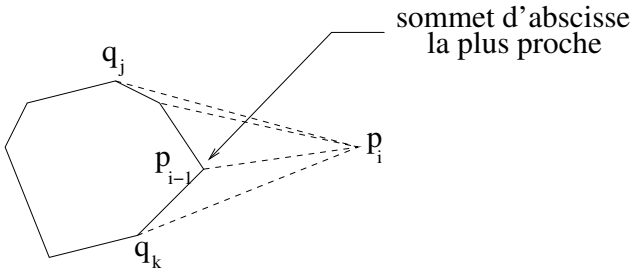

FIG. 1.5 – *Ajout*

On utilise un tableau de pointeurs indiquant, pour tout p_j, si $p_j \in env\_conv(p_1, \ldots, p_{i-1})$ et, si oui, donne sa position. Ainsi, p_{i-1} est récupérable dans $env\_conv(p_1, \ldots, p_{i-1})$.

On applique alors l'ajout décrit précédemment, sans le test, et en partant de p_{i-1}.

Complexité

En analyse amortie, on constate qu'on ne supprime un sommet de C que si celui-ci a déjà été ajouté ; en donnant un coût de 2 à l'ajout et un coût de 0 à la suppression, on obtient alors un algorithme en $O(n)$ pour l'ajout.

L'algorithme total est alors en $O(n \log(n))$ (à cause du tri).

1.2.3 Algorithme incrémental avec balayage circulaire

Algorithme

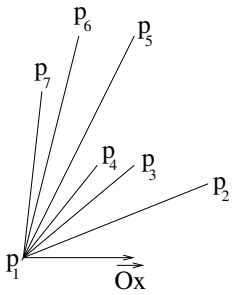

FIG. 1.6 – *Balayage circulaire*

On effectue une rotation à partir du point le plus bas p_1, qui est repérable en $O(n)$.

Algorithme :

1. Tri des $(p_i)_{i\geqslant 2}$ par ordre de $(\overrightarrow{Ox}, \overrightarrow{p_1p_i})$ croissants
2. $S \leftarrow \emptyset$ (pile)
3. `empiler`(p_1, S)
4. `empiler`(p_2, S)
5. `empiler`(p_3, S)
6. Pour i de 4 à n faire
7. Tant que $(\overrightarrow{sous\_haut(S)haut(S)}, \overrightarrow{haut(S)p_i}) < 0$ faire
8. `depiler`(S)
9. FinTantQue
10. `empiler`(p_i, S)
11. FinPour
12. Renvoyer S.

Complexité

On peut appliquer la même analyse amortie que précédemment. L'algorithme est donc en $O(n\log(n))$.

1.2.4 Diviser pour régner

On utilise une division au niveau du médian des abscisses des points, comme dans le paragraphe 1.1. Une analyse de complexité indique que :

$$T(n) \leqslant 2T\left(\frac{n}{2}\right) + 4$$

D'où : $T(n) = O(n)$.