Algorithme géométrique, FMIN215 Année 2013-2014
Algos
M1 Info, M1 Maths-Info.

- Algorithmique géométrique -

- Algorithmes du cours -

Chapitre 1 : Intersections de segments

Algorithme : ALGORITHME PAR BALAYAGE (Bentley, Ottman, 1979)

Données : Un ensemble de n segments du plan $(S_i)_{1\leq i\leq n}$. Pour simplifier, on suppose qu'aucun segment est vertical et que trois segments ne se coupent jamais en un même point.

Résultat : *OUI*, si il existe deux segments S_i et S_j qui s'intersectent, *NON* sinon.

début
  Trier les extrémités des segments $(S_i)_{1\leq i\leq n}$ par abscisses croissantes pour former l'échéancier *Ech* ;
  `// géré par un tableau ou une liste chaînée`
  $T \longleftarrow \emptyset$; `// l'ordre vertical dynamique sur les segments`
  pour tous les $p \in Ech$*, suivant l'ordre sur Ech* **faire**
    si *p est l'extrémité gauche d'un segment* S_i **alors**
      $Inserer(T, S_i)$; `// découverte du segment` S_i
      si $Intersection(AuDessus(T, S_i), S_i)$ *vraie* **alors retourner** *OUI*;
      si $Intersection(AuDessous(T, S_i), S_i)$ *vraie* **alors retourner** *OUI*;
    si *p est l'extrémité droite d'un segment* S_i **alors**
      si $Intersection(AuDessus(T, S_i), AuDessous(T, S_i))$ *vraie* **alors retourner** *OUI*;
      $Supprimer(T, S_i)$; `// on quitte le segment` S_i
  retourner *NON*; `// pas d'intersection détectée`
fin

Complexité : $O(n^2)$ si *Ech* est géré par un tableau, $O(n \log n)$ si il est géré par un arbre de recherche équilibré.

Chapitre 2 : Enveloppes convexes

Algorithme : ALGORITHME DE JARVIS (1973)

Données : Un ensemble de n points du plan $(P_i)_{1\leq i\leq n}$ en position générale.

Résultat : La liste L des sommets de l'enveloppe convexe de $\mathcal{P}$ donnée dans le sens direct.

début
  $P_{min} \longleftarrow$ le point d'ordonnée minimale;
  $P_{courant} \longleftarrow P_{min}$;
  $L \longleftarrow (P_{courant})$; `// la liste à retourner`
  répéter
    Choisir $P \neq P_{courant}$;
    Prendre $P_{suivant}$ le point le plus à droite de $[P_{courant}P)$;
    Ajouter $P_{suivant}$ à L;
    $P_{courant} \longleftarrow P_{suivant}$;
  jusqu'à $P_{courant} = P_{min}$;
  retourner L
fin

Complexité : $O(n|EC(\mathcal{P})|)$

Algorithme : ALGORITHME DE GRAHAM (1972)

Données : Un ensemble de n points du plan $(P_i)_{1 \leq i \leq n}$ en position générale.
Résultat : La pile L des sommets de l'enveloppe convexe de $\mathcal{P}$ donnée dans le sens direct.

1 **début**
2     $P_{min} \longleftarrow$ le point d'ordonnée minimale;
3   On note $Q_1, \ldots, Q_{n-1}$ les points restants triés par angle polaire croissant autour de P_{min};
4   Empiler(P_{min}, L) ; Empiler(Q_1, L) ; Empiler(Q_2, L); `// le début de` $EC(\mathcal{P})$
5   **pour tous les** *i de 3 à $n-1$* **faire**
6   **tant que** *Q_i est à droite de $[AvantDernier(L), Dernier(L))$* **faire**
7   Dépiler L ;
8   Empiler(Q_i, L) ;
9   **retourner** L
10 **fin**

Complexité : La ligne 3 correspondant au tri prend un temps $O(n \log(n))$, les autres lignes prennent un temps $O(n)$. En tout, $O(n \log(n))$.

Chapitre 3 : Graphes planaires

Algorithme : DESSIN D'UN GRAPHE PLANAIRE (Demoucron-Malgrange-Pertuiset, 1964)

Données : Un graphe $G = (V, E)$ non-orienté et simple à n sommets.
Résultat : Une représentation plane de G si il est planaire, '*NON PLANAIRE*' sinon.

début
    Choisir C un cycle de G, noter $G_0 = C$ et $\widetilde{G}_0$ sa représentation plane;
    $i \longleftarrow 0$;
    tant que $G_i \neq G$ **faire**
        $B^\star \longleftarrow \emptyset$;
        pour tous les *G_i-pont B* **faire**
            Calculer le nombre $a(B)$ de faces de $\widetilde{G}_i$ qui sont incidentes à tous les points d'attachement de B;
            si $a(B) = 0$ **alors retourner** *NON-PLANAIRE;*
            si $a(B) = 1$ **alors** $B^\star \longleftarrow B$; `// on la sélectionne`
        si $B^\star = \emptyset$ **alors**
            Choisir B un G_i-pont quelconque;
            $B^\star \longleftarrow B$;
        Choisir un chemin P entre deux points d'attahement de $B^\star$;
        Ajouter P à G_i pour obtenir G_{i+1}; `// on étend le dessin`
        Représenter P dans $\widetilde{G}_i$ pour obtenir $\widetilde{G}_{i+1}$;
        $i \longleftarrow i + 1$;
    retourner $\widetilde{G}_i$
fin

Complexité : $O(n^3)$ (à discuter en cours...).

Algorithme : CONSTRUCTION D'UNE CARTE TRAPZÉOÏDALE ET DU GRAPHE DE LOCALISATION ASSOCIÉ

Données : Un ensemble $S = \{s_1, \ldots, s_n\}$ de segments du plan.
Résultat : La carte trapézoïdale $\mathcal{T}(S)$ etle graphe (orienté) de localisation D associé.

début
  Déterminer une zone rectangulaire R contenant tous les segments s_i;
  Initialiser $\mathcal{T}(S)$ à R et D à un seul sommet étiqueté R;
  pour tous les *i de 1 à n* **faire**
    Soient p et q les extrémités respectivement gauche et droite de s_i;
    Trouver dans D le trapèze Δ_1 contenant p ; `// On veut` $\Delta_1, \ldots, \Delta_p$ `intersectés par le segment` s_i
    $j \leftarrow 1$;
    tant que *q est à droite de* `BordDroit`(Δ_j) **faire**
      si Δ_j *a un seul trapèze voisin à sa droite* **alors**
        Nommer Δ_{j+1} ce trapèze;
      sinon
        Soit r le point par lequel passe `BordDroit`(Δ_j);
        si *r est au dessus de s_i* **alors**
          $\Delta_{j+1} \leftarrow$ le voisin en bas à droite de Δ_j
        sinon
          $\Delta_{j+1} \leftarrow$ le voisin en haut à droite de Δ_j;
        $j \leftarrow j + 1$;
    Supprimer $\Delta_1, \ldots, \Delta_p$ de $\mathcal{T}(S)$ et les remplacer par les trapèzes créés par l'intersection avec s_i;
    Supprimer les feuilles étiquetées $\Delta_1, \ldots, \Delta_p$ dans D et créer les nouvelles feuilles correspondants aux trapèzes créés ainsi possiblement que de nouveaux nœuds internes;
  retourner *$\mathcal{T}(S)$ et D;*
fin

Complexité : $O(n \log n)$ en moyenne (à discuter en cours...).

Chapitre 4 : Triangulations

Algorithme : TRIANGULATION INCRÉMENTALE

Données : Un ensemble $P = \{p_1, \dots, p_n\}$ de points du plan.

Résultat : $\mathcal{T}$ une triangulation de P.

début
- Trier les points de P par ordre léxicographique (par abscisse croissante puis en cas d'égalité par ordonnée croissante). On note $q_1, \dots, q_n$ les points de P dans cet ordre;
- $\mathcal{T} \leftarrow \{q_1 q_2 q_3\}$;
- Ecrire $EC = (r_1, r_2, r_3, r_1)$ dans le sens direct avec $r_1 = q_3$ et $\{r_2, r_3\} = \{q_1, q_2\}$;
- **pour** $i = 3$ *à* $n - 1$ *;* `// on ajoute` q_{i+1}
- **faire**
  - On note $EC = (r_1, \dots, r_k, r_{k+1})$ dans le sens direct avec $r_1 = r_{k+1} = q_i$;
  - $j \leftarrow 1$;
  - **tant que** $\det(\overrightarrow{q_{i+1}r_j}, \overrightarrow{q_{i+1}r_{j+1}}) < 0$ *;* `// on cherche le point visible depuis` q_{i+1} `le plus à droite possible`
  - **faire**
    - Ajouter le triangle $r_j r_{j+1} q_{i+1}$ à $\mathcal{T}$;
    - $j \leftarrow j + 1$;
  - $k_{\text{droite}} \leftarrow j$;
  - $j \leftarrow k + 1$;
  - **tant que** $\det(\overrightarrow{q_{i+1}r_j}, \overrightarrow{q_{i+1}r_{j-1}}) > 0$ *;* `// on cherche le point visible depuis` q_{i+1} `le plus à gauche possible`
  - **faire**
    - Ajouter le triangle $r_j r_{j-1} q_{i+1}$ à $\mathcal{T}$;
    - $j \leftarrow j - 1$;
  - $k_{\text{gauche}} \leftarrow j$;
  - Remplacer dans EC l'intervalle $(r_1, \cdots, r_{k_{\text{droite}}-1})$ par q_{i+1} et l'intervalle $(r_{k_{\text{gauche}}+1}, \cdots, r_{k+1})$ par q_{i+1};
- **retourner** $\mathcal{T}$*;*

fin

Complexité : $O(n \log n)$, le nombre de triangles et segments créés étant linéaires en n.

Algorithme : DELAUNAY PAR FLIPS

Données : une triangulation $\mathcal{T} = \{\Delta_1, \cdots, \Delta_t\}$ quelconque d'un ensemble P de points du plan, codée de façon à avoir, pour chaque triangle Δ_i, la liste de ses (au plus) trois triangles voisins.

Résultat : $\mathcal{T}'$ une triangulation de Delaunay de P.

début
- $flip \leftarrow vrai$;
- **tant que** *flip est vrai* **faire**
  - $flip \leftarrow faux$;
  - **pour tous les** *i de 1 à t* **faire**
    - **si** *un 'flip' est possible entre Δ_i et un de ses triangles voisins* **alors**
      - Faire ce 'flip' dans $\mathcal{T}$;
      - $flip \leftarrow vrai$;
- **retourner** $\mathcal{T}$*;*

fin

Complexité : Le nombre de 'flips' à faire est au plus $\binom{n}{2}$, donc en tout une complexité en $O(n^3)$.

Algorithme : ALGORITHME DE S.FORTUNE, CACUL DU DIAGRAMME DE VORONOÏ

Données : Un ensemble P de n points du plan. On suppose qu'il n'existe pas deux points de P ayant la même abscisse.

Résultat : Le diagramme de Voronoï de P : les points du diagramme, les cellules et leur adjacence.

1 **début**
2   Trier les points de P par abscisse croissante pour intialiser l'échéancier Ech;
3   $Vor \leftarrow \emptyset$;
4   $Lp \leftarrow \emptyset$; `// La ligne de plage, les arcs de parabole sont rangées de bas en haut`
5   **tant que** $Ech \neq \emptyset$ **faire**
6   Prendre ev le premier évènement de Ech;
7   **si** *ev est un évènement de site, correspondant au point x de P* **alors**
8   Déterminer la portion de parabole P_y de Lp intersectant la demi-droite horizontale à la gauche de x; `// Si` x `est le premier sommet de` Ech`, prendre` $P_y = \emptyset$`.`
9   Noter P_w (resp. P_z) l'arc de parabole précédant (resp. succédant à) P_y dans Lp;
10   Séparer P_y en deux portions de parabole P_y^1 et P_y^2 avec P_y^2 sous P_y^1;
11   Insérer P_x entre P_y^1 et P_y^2 dans Lp;
12   Supprimer l'évènement de cercle $P_wP_yP_z$ dans Ech ; `// si les évènements de cercle considérés n'existent pas, ne pas faire les opérations correspondantes.`
13   Insérer les évènements de cercle $P_wP_y^1P_x$, $P_y^2P_xP_y^1$ et $P_xP_y^1P_z$ dans Ech;
14   Créer $Vor(x)$ dans Vor et la reliée à $Vor(y)$; `// pour l'instant, on ne connait pas les sommets du diagramme bordant` $Vor(x)$`.`
15   **si** *ev est un évènement de cercle, correspondant à la disparition de P_x dans Lp* **alors**
16   Noter P_y le succésseur de P_x dans Lp;
17   Noter P_u le succésseur de P_y dans Lp;
18   Noter P_z le prédécesseur de P_x dans Lp;
19   Noter P_w le prédécesseur de P_z dans Lp;
20   Effacer les évènements de cercles contenant x, c-à-d $P_wP_zP_x$, $P_zP_xP_y$ et $P_xP_yP_u$;
21   Ajouter les évènements de cercles $P_wP_zP_y$ et P_zPyP_u;
22   Noter p le centre du cercle circonscrit au triangle xyz;
23   Ajouter p comme un point de Vor incident aux cellules $Vor(x)$, $Vor(y)$ et $Vor(z)$;
24   Relier $Vor(y)$ et $Vor(z)$;
25   **retourner** *Vor;*
26 **fin**

Complexité : La ligne **2** demande un temps $O(n \log n)$. Ensuite, on a un nombre linéaire d'objets. Si Ech et Lp sont gérés par des arbres de recherche équilibrés, chaque opération (Insertion, Supression, Prédécesseur, Successeur) demande un temps en $O(\log n)$. Donc, en tout, l'algorithme prend un temps en $O(n \log n)$.