Exercice 1 - Graphes de plus proche voisin

Soit S un ensemble de n points du plan. Fixons $q_0 \in S$. Soit q_1 le plus proche voisin de q_0 dans S, c'est à dire le point de $S \setminus \{q_0\}$ à distance minimale de q_0. Soit q_2 le second plus proche voisin de q_0 dans S, c'est à dire le plus proche voisin de q_0 dans $S \setminus \{q_0, q_1\}$. Pour $i \geq 3$, on définit de même manière q_i comme le $i$ème plus proche voisin de q_0 dans S.

1. Prouvez que q_0q_1 est une arête de la triangulation de Delaunay de S.

   The disk centered at q_0 passing through q_1 contains only q_0, thus the disk of diameter q_0q_1, which is included in the previous one is empty. By the empty circle property, q_0q_1 is a Delaunay edge.

2. Prouvez que q_0q_2 ou q_1q_2 est une arête de la triangulation de Delaunay de S.

   The disk D_2 centered at q_0 passing through q_2 contains only q_0 and q_1, thus we consider the two disks Z_0 and Z_1 passing through q_2 tangent in q_2 to D_2 and respectively passing through q_0 and q_1. We have to cases:
   — $Z_0 \subset Z_1 \subset D_2$ and Z_0 is empty, by the empty circle property, q_0q_2 is a Delaunay edge.
   — $Z_1 \subset Z_0 \subset D_2$ and Z_1 is empty, by the empty circle property, q_1q_2 is a Delaunay edge.

3. Prouvez que pour tout k il existe $i < k$ tel que q_kq_i est une arête de la triangulation de Delaunay de S.

   The disk of center q_0 through q_k verifies $D_k \cap S = \{q_0, q_1 \dots q_{k-1}\}$. Consider the pencil of circles through q_k tangent to D_k The bigest empty circle of that pencil inside D_k pass through a point inside D_k that is some q_i with $i < k$ and by the empty circle property, q_iq_k is a Delaunay edge.

4. Le graphe dirigé de plus proche voisin est le graphe dont les sommets sont les points de S et où $p \to q$ est une arête si q est le plus proche voisin de p.

   Écrivez un algorithme qui prend en entrée la triangulation de Delaunay de S et produit le graphe dirigé de plus proche voisin de S.

   > Vous pouvez écrire des choses comme :
   > `pour` v `énumérant les sommets de` $DT(S)$,
   > `pour` w `énumérant les voisins de` v `dans` $DT(S)$,
   > ou `créer` $v \to w$,
   > ou $v.couleur = rouge$ pour ajouter des informations à un sommet (ou arête ou...)

```
for u enumerating all vertices of DT(S)  {
      d = ∞;
      for w enumerating the neighbor of u in DT(S)  {
            if ‖uw‖ < d then {nn = w;   d = ‖uw‖;   }
      }
      output edge(u, nn),
}
```

5. Quelle est la complexité de votre algorithme ?

   The inside loop costs $d^\circ(u)$, thus the total cost of the algorithm is $\sum_{u\in S} d^\circ(u) < 6n$.

6. Écrivez un algorithme qui prend en entrée la triangulation de Delaunay de S et produit le graphe dirigé de *second* plus proche voisin de S, et indiquez sa complexité.

```
for u enumerating all vertices of DT(S)  {
        u.d = ∞;
        for w enumerating the neighbor of u in DT(S)  {
              if ‖uw‖ < d then {u.nn = w;   d = ‖uw‖;   }
        }
}
for u enumerating all vertices of DT(S)  {
        d = ∞;
        for w enumerating the neighbor of u in DT(S)  {
              if (‖uw‖ < d and w ≠ u.nn) then {sn = w;   d = ‖uw‖;   }
        for w enumerating the neighbor of u.nn in DT(S)  {
              if (‖uw‖ < d and w ≠ u) then {sn = w;   d = ‖uw‖;   }
        output edge(u, sn),
        }
}
```

   The cost is

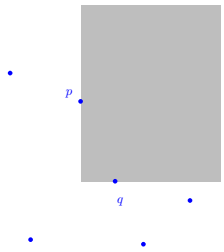
$$\begin{aligned}
\sum_{u\in S}(d^\circ_{DT}(u) + d^\circ_{DT}(u.nn)) &= \sum_{u\in S} d^\circ_{DT}(u) + \sum_{u\in S}\sum_{v\in\{u.nn\}} d^\circ_{DT}(v) \\
&= \sum_{u\in S} d^\circ_{DT}(u) + \sum_{v\in S}\sum_{u \text{ such that } v=u.nn} d^\circ_{DT}(v) \\
&= \sum_{u\in S} d^\circ_{DT}(u) + \sum_{v\in S} d^\circ_{NN}(v)\cdot d^\circ_{DT}(v) \\
&= \sum_{u\in S} d^\circ_{DT}(u) + \sum_{v\in S} 6 d^\circ_{DT}(v) \le 42n
\end{aligned}$$

Exercice 2 - Points maximaux

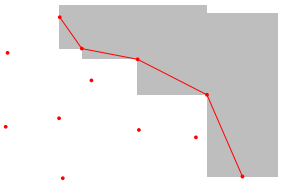
Soit S un ensemble de n points. On supposera que l'on n'a pas deux points à la même abcisse ou à la même ordonnée. Pour deux points p et q de S, on dira que $p \to q$ est une arête (orientée) dominante si

$$x_p < x_q; \quad y_p > y_q; \quad \forall r \in S \setminus \{p, q\} x_r < x_p \text{ ou } y_r < y_q$$

c'est à dire que la région grisée sur la figure ne contient pas de points de S



1. Dessiner les arêtes maximales pour l'ensemble de points suivants :



2. Montrer que les arêtes maximales peuvent être organisées en une séquence $p_1 \to p_2 \to \ldots \to p_k$ avec $x_{p_1} < x_{p_2} < \ldots < x_{p_k}$. Caractériser p_1 et p_k.

   Si q est le point le plus haut, il ne peut pas y avoir de point p tel que pq soit une arête dominante (une arête dominante est orientée vers le bas et vers la droite).

   On va choisir p_1 le point le plus haut. Si p_1 n'est pas le point le plus à droite, on va choisir p_2 le point le plus haut des points à droite de p_1. Il est clair que p_1p_2 est dominante.

   On peut construire une arête suivante jusqu'à ce que l'on arrive à p_k le point le plus à droite.

3. Proposer un algorithme (en pseudo-code) permettant de trouver les points maximaux d'un ensemble S.

   Trier les points par ordonnée décroissante.
   Soit p le premier point (dans cet ordre)
   Pour chaque point q de S à partir du suivant de p
   si p à droite de q,
   afficher pq;
   $p := q$;
   fin si
   fin pour

4. Analyser la complexité asymptotique de votre algorithme en fonction de n le nombre de points de S.

   On a un tri en $O(n \log n)$ suivi d'une simple boucle en temps linéaire. La complexité totale est donc $O(n \log n)$.

5. Montrer une borne inférieure pour le problème du calcul de la séquence des arêtes maximales.

   On va construire un algorithme stupide de tri utilisant comme boîte noire une algorithme de points maximaux:
   Pour trier n nombres $x_i, 1 \leq i \leq n$
   Contruire les n points $(-x_i, x_i)$
   Trouver la séquence des arêtes maximales
   Afficher l'ordonnée des points dans l'ordre.

   Cet algorithme permet de trier les valeurs x_i. À part la boîte noire calculant les arêtes maximales la complexité est linéaire mais comme on sait que l'on ne peut pas trier plus vite que $\Omega(n \log n)$ c'est que le calcul des arêtes maximales est en $\Omega(n \log n)$.