Mini-projet
d'algorithmique avancée

ENSSAT INFO 2

2019-2020

Présentation du problème

On considère le problème de la triangulation d'un polygone convexe. On se donne les n sommets d'un polygone plan et une mesure de la distance séparant chaque couple de sommets. Le problème consiste à sélectionner un ensemble de cordes (segments joignant deux sommets non adjacents) tel qu'il n'y ait aucune intersection de cordes et que le polygone entier soit divisé en triangles. La longueur totale (somme des distances entre les extrémités de toutes les cordes choisies) doit être minimale. On appelle un tel ensemble de cordes une triangulation minimale.

Exemple.

La figure suivante montre un heptagone et les coordonnées $(x,\ y)$ de ses sommets. Les lignes en pointillé représentent une triangulation, qui n'est d'ailleurs pas minimale. Son poids est la somme des longueurs des cordes $(s_0,\ s_2)$, $(s_0,\ s_3)$, $(s_0,\ s_5)$ et $(s_3,\ s_5)$ soit :

$$\sqrt{8^2+16^2}+\sqrt{15^2+16^2}+\sqrt{22^2+2^2}+\sqrt{7^2+14^2}=77.56$$

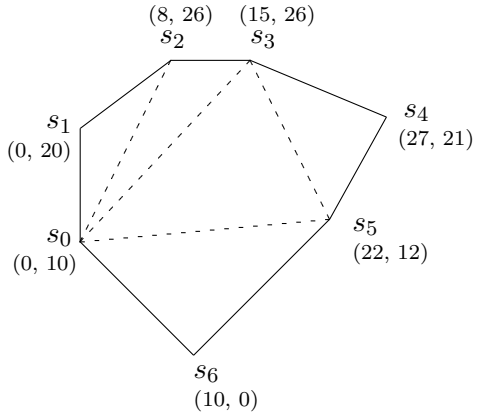

Fig. 1. Une des triangulations possibles d'un heptagone

Nous supposerons dans la suite que le contour du polygone étudié possède n sommets classés par ordre de parcours rétrograde (contraire du sens trigonométrique) notés s_0, s_1, ..., s_{n-1}. On va étudier trois méthodes pour traiter ce problème : les essais successifs, la programmation dynamique et l'utilisation d'un algorithme glouton. L'objectif est de

comparer ces méthodes et de faire une recommandation quant à un choix.

A. Questions préliminaires

1. Combien de cordes distinctes peut-on dénombrer dans un polygone à n sommets ?

2. Montrer que toutes les triangulations d'un polygone à n sommets comportent le même nombre de cordes.

B. Essais successifs

1. Ecrire une fonction *validecorde*(i, j) qui rend vrai si la corde joignant les sommets s_i et s_j n'a pas déjà été tracée et si elle ne coupe aucune corde déjà tracée (on supposera que les cordes déjà tracées sont stockées dans un tableau).

2. On considère un algorithme par essais successifs basé sur la stratégie suivante : à l'étape i, on trace l'une des cordes valides issues de s_i ou on ne trace rien. Calcule-t-on plusieurs fois la même triangulation et si oui pourquoi ? Montrer que cette méthode ne permet pas toujours d'obtenir toutes les triangulations.

3. On veut pouvoir construire toute triangulation une fois et une seule.

   a. En supposant disponible le vecteur des cordes C tel que $C[i]$ contient les numéros des sommets de départ et d'arrivée de la i^{e} corde parmi toutes celles possibles ainsi que sa longueur, proposer une stratégie par essais successifs. Ecrire le programme correspondant.

   b. Donner la complexité de cet algorithme en terme de nombre d'appels récursifs engendrés.

   c. Proposer des condition d'élagage simple pour déterminer qu'une branche de l'arbre des appels récursifs ne peut plus mener à une solution. Mettre en oeuvre ces conditions d'élagage, évaluer les gains obtenus et commenter.

   d. En utilisant la version optimisée, jusqu'à quel nombre de sommets peut-on aller en pratique pour obtenir un résultat en moins de 2 minutes ?

C. Programmation dynamique

On envisage maintenant une solution de type programmation dynamique. On définit le sous-problème de taille t débutant au sommet s_i, noté $T_{i,t}$, comme la triangulation minimale du polygone s_i, s_{i+1}, ..., s_{i+t-1}, formé par la suite des sommets du contour original débutant en s_i et continuant dans l'ordre rétrograde. La corde du polygone initial formant une arête de $T_{i,t}$ est (s_i, s_{i+t1}). Pour résoudre $T_{i,t}$, nous considérons les trois cas suivants :

- Nous pouvons choisir le sommet s_{i+t-2} pour former un triangle avec les cordes (s_i, s_{i+t-2}) et (s_i, s_{i+t-1}) et le troisième côté (s_{i+t-2}, s_{i+t-1}) puis résoudre le problème $T_{i,t-1}$.

- Nous pouvons choisir le sommet s_{i+1} pour former un triangle avec les cordes (s_i, s_{i+t-1}) et (s_{i+1}, s_{i+t-1}) et le troisième côté (s_i, s_{i+1}) puis résoudre le problème $T_{i+1,t-1}$.
- Pour tout k entre 2 et $t-3$, nous pouvons choisir le sommet s_{i+k} et former un triangle de côtés (s_i, s_{i+k}), (s_{i+k}, s_{i+t-1}) et (s_i, s_{i+t-1}) puis résoudre les sous-problèmes $T_{i,k+1}$ et $T_{i+k,t-k}$.

Comme la résolution de tout sous-problème de taille inférieure à trois ne demande aucune action, il est possible de résumer les trois cas précédents en disant que si l'on choisit un k entre 1 et $t-2$, on est conduit à résoudre les sous-problèmes $T_{i,k+1}$ et $T_{i+k,t-k}$.

Il convient de préciser que cette stratégie est à la fois valide (on examine toute triangulation possible) et minimale (on examine chaque triangulation une seule fois).

1. Établir la formule de calcul d'une triangulation minimale d'un polygone convexe de n côtés sous forme d'une récurrence complète.

2. Proposer un algorithme de type programmation dynamique découlant de la formule précédente.

3. Donner la complexité spatiale de l'algorithme, ainsi que sa complexité temporelle en terme de nombre de comparaisons (de valeurs de triangulations).

4. On constate que, dans cette stratégie, deux sous-problèmes ont toujours un sommet commun. Expliquer en quoi cette façon de décomposer est intéressante du point de vue des triangulations considérées. Quelles modifications devraient être apportées à l'algorithme si la décomposition se faisait en tirant deux cordes quelconques ?

D. Algorithme glouton

On appelle corde extérieure d'un polygone, toute corde permettant de tracer un triangle dont les deux côtés restants sont des arêtes du polygone. On considère la stratégie gloutonne suivante. A chaque étape, on retient la corde extérieure du polygone restant à trianguler dont la longueur est minimale.

1. Mettre en œuvre cette méthode.

2. Que peut-on dire du caractère exact de cette stratégie gloutonne ?

E. Recommandation argumentée

Au vu de ce qui précède, faire une recommandation pour résoudre ce problème.

Livrables attendus

Le mini-projet est effectué en binôme. Deux livrables par binôme sont attendus à l'issue du mini-projet :

1. Compte-rendu du mini-projet. Le compte-rendu contiendra *a minima* : une page de garde, une introduction, les réponses à toutes les questions de l'énoncé incluant les explications claires des solutions développées, les algorithmes écrits en **pseudo-langage** et **commentés**, une étude de la complexité de chaque algorithme, les jeux d'essai motivés (complets, avec étude expérimentale permettant d'exhiber empiriquement les gains découlant de la mise en œuvre des critères d'élagage choisis), une conclusion incluant un véritable bilan du projet (rappel et comparaison des solutions proposées, perspectives), les listings commentés en annexe. Le rapport hors annexes ne devra pas dépasser 15 pages.
ATTENTION : Le compte-rendu hors annexes doit pouvoir être lu indépendamment du code (contenir toutes les informations nécessaires à la compréhension et l'évaluation des solutions proposées).

2. Sources du mini-projet. Les fichiers sources commentés de votre code devront être fournis, incluant un fichier README (le fichier README expliquera comment installer et utiliser votre/vos programme(s)). L'utilisation du langage de développement Java vous est recommandé pour l'implantation des algorithmes (mais ce n'est pas une obligation).

Les livrables devront être envoyés par mail à votre encadrant de TP, dans un mail intitulé *"[INFO2 Algorithmique Avancée] Mini-projet du binome B_1 / B_2"* où B_1 et B_2 sont à remplacer par les noms des binômes.