Distances entre courbes

Épreuve pratique d'algorithmique et de programmation
Concours commun des écoles normales supérieures

Durée de l'épreuve: 3 heures 30 minutes

Juillet 2009

ATTENTION !

N'oubliez en aucun cas de recopier votre u_0
à l'emplacement prévu sur votre fiche réponse

Important.

Sur votre table est indiqué un numéro u_0 qui servira d'entrée à vos programmes. Les réponses attendues sont généralement courtes et doivent être données sur la fiche réponse fournie à la fin du sujet. À la fin du sujet, vous trouverez en fait deux fiches réponses. La première est un exemple des réponses attendues pour un $\widetilde{u_0}$ particulier (précisé sur cette même fiche et que nous notons avec un tilde pour éviter toute confusion!). Cette fiche est destinée à vous aider à vérifier le résultat de vos programmes en les testant avec $\widetilde{u_0}$ au lieu de u_0. Vous indiquerez vos réponses (correspondant à votre u_0) sur la seconde et vous la remettrez à l'examinateur à la fin de l'épreuve.

En ce qui concerne la partie orale de l'examen, lorsque la description d'un algorithme est demandée, vous devez présenter son fonctionnement de façon schématique, courte et précise. Vous ne devez en aucun cas recopier le code de vos procédures!
Quand on demande la complexité en temps ou en mémoire d'un algorithme en fonction d'un paramètre n, on demande l'ordre de grandeur en fonction du paramètre, par exemple: $O(n^2)$, $O(n \log n)$,...
Il est recommandé de commencer par lancer vos programmes sur de petites valeurs des paramètres et de ***tester vos programmes sur des petits exemples que vous aurez résolus préalablement à la main ou bien à l'aide de la fiche réponse type fournie en annexe***. Enfin, il est recommandé de lire l'intégralité du sujet avant de commencer afin d'effectuer les bons choix de structures de données dès le début.

1 Introduction

Certaines applications comme la reconnaissance de formes, par exemple de caractères dans une image, ont besoin de mesurer la ressemblance entre deux courbes. Il est possible de définir des distances pour quantifier ces ressemblances. Nous allons étudier des algorithmes efficaces pour calculer de telles distances, sur des courbes polygonales, et dans certaines configurations.

Étant donnés deux points du plan s et t, nous allons définir $d(s,t)$ comme le carré de la distance Euclidienne entre ces points. Dans cette épreuve, nous manipulerons des carrés de distance afin d'éviter des problèmes numériques avec des racines carrées qui sont ici inutiles.

Définition 1 (Distance) *Étant donnés maintenant deux ensembles compacts de points du plan, S et T, nous étendons la définition précédente de la manière suivante :*

$$d(S,T) = \inf \{\ d(s,t) \mid (s,t) \in (S,T)\ \}$$

De même entre un point s et un ensemble T :

$$d(s,T) = \inf \{\ d(s,t) \mid t \in T\ \}$$

Définition 2 (Distance de Hausdorff) *Définissons maintenant (le carré de) la distance de Hausdorff entre deux ensembles comme suit :*

$$d_H(S,T) = \max \{\ \sup_{t\in T}\ d(t,S)\ ,\ \sup_{s\in S}\ d(s,T)\ \}$$

C'est cette distance qui est généralement utilisée pour mesurer la proximité entre courbes, et que nous allons calculer sur des courbes polygonales.

Afin d'éviter des problèmes liés aux erreurs d'arrondis dans les fonctions numériques, on effectuera les calculs numériques avec des nombres flottants double precision.

2 Génération aléatoire de jeux de données

Considérons les suites d'entiers (u_k) et (v_k) définies pour $k \geqslant 0$ par :

$$u_k = \begin{cases} \textit{votre } \mathtt{u_0} \textit{ (à reporter sur votre fiche)} & \text{si } k = 0 \\ 15\,091 \times u_{k-1} \mod 64\,007 & \text{si } k \geqslant 1 \end{cases}$$

$$v_k = \begin{cases} \mathtt{u_0} & \text{si } k = 0 \\ 1\,129 \times v_{k-1} \mod 63\,997 & \text{si } k \geqslant 1 \end{cases}$$

Question 1 *Que valent :* ***a)*** u_{10} ***b)*** u_{100} ***c)*** v_{100} *?*

3 Distances entre ensembles finis de points

Soit p_k le point de coordonnées (u_k, v_k).
Nous noterons P_n et Q_n les ensembles de points suivants :

$$\begin{aligned} P_n &= \{\ p_{2i} \mid i \in \{0 \dots n-1\}\ \} \\ Q_n &= \{\ p_{2i+1} \mid i \in \{0 \dots n-1\}\ \} \end{aligned}$$

Question à développer pendant l'oral : Donnez la complexité de l'algorithme naïf qui calcule $d_H(P_n, Q_n)$.

Question 2 *Que valent :* ***a)*** $d(P_5, Q_5)$ ***b)*** $d(P_{100}, Q_{100})$ ***c)*** $d(P_{1000}, Q_{1000})$ *?*

Question 3 *Que valent :* ***a)*** $d_H(P_5, Q_5)$ ***b)*** $d_H(P_{100}, Q_{100})$ ***c)*** $d_H(P_{1000}, Q_{1000})$ *?*

Question à développer pendant l'oral : Proposez un algorithme de complexité moyenne sous-quadratique pour le calcul de $d(P_n, Q_n)$ dans le cas des distributions de points uniformes et estimez sa complexité. Vous vous baserez par exemple sur un tri des points selon l'ordre lexicographique.

4 Distances entre courbes polygonales

Soit $\overline{P}_n$ l'ensemble des segments formés par les paires de points consécutifs dans P_n, ainsi que le segment reliant le dernier au premier point de P_n. On définit de manière similaire $\overline{Q}_n$ à partir de Q_n.

Pour le calcul des distances entre ces polygones, on considèrera bien sûr les segments comme pleins. Afin de décomposer le problème, nous vous conseillons de programmer les fonctions intermédiaires suivantes :
- `orientation(p, q, r)` : calcule de quel côté de la droite orientée (`pq`) se situe le point `r` (ou si les trois points `p`, `q`, `r` sont collinéaires).
- `dist(p, q, r)` : calcule la distance $d([pq], r)$ du point `r` au segment `pq`.
- `dist(p, q, r, s)` : calcule la distance $d([pq], [rs])$ du segment `pq` au segment `rs`.

Question 4 *Que valent les moyennes de* $d(p_0, x)$ *pour* x *dans :* ***a)*** $\overline{Q}_4$ ***b)*** $\overline{Q}_{100}$ ***c)*** $\overline{Q}_{1000}$ *?*

Question 5 *Que valent :* ***a)*** $d(\overline{P}_4, \overline{Q}_4)$ ***b)*** $d(\overline{P}_5, \overline{Q}_5)$ ***c)*** $d(\overline{P}_{1000}, \overline{Q}_{1000})$ *?*

On remarquera que le cas de la distance de Hausdorff entre courbes polygonales est délicat. Nous essayons maintenant de simplifier le problème afin d'étudier des algorithmes plus efficaces. Nous allons donc étudier le cas des polygones convexes.

5 Vérification de convexité

Question à développer pendant l'oral : Proposez un algorithme de complexité optimale, qui vérifie qu'une suite de points décrit le bord d'un polygone convexe.

Question 6 *Les polygones suivants sont-ils convexes :* ***a)*** $\overline{P}_4$ ***b)*** $\overline{Q}_4$ ***c)*** $\overline{P}_5$ ***d)*** $\overline{Q}_5$ *?*

On note R_k l'ensemble des pentagones $\{ p_{2i}, p_{2i+2}, \ldots, p_{2i+8}, p_{2i} \mid i \in \{0 \ldots k-1\}\}$.

Question 7 *Parmi les ensembles de pentagones suivants, combien sont-ils convexes :* ***a)*** R_1 ***b)*** R_{10} ***c)*** R_{100} *?*

6 Génération aléatoire de polygones convexes

Dans un premier temps, nous allons nous attacher à la génération aléatoire de polygones convexes. Pour ce faire, nous allons faire la remarque suivante : les vecteurs correspondant aux segments formant un convexe "tournent" toujours dans le même sens : ils sont ordonnés selon leur direction, et leur somme est le vecteur nul. Nous allons donc générer des convexes en prenant cette remarque à l'envers, en produisant un tel ensemble de vecteurs.

On notera $\overline{C}_n$ le convexe obtenu à partir de P_n de la façon suivante :
- soit q le point de coordonnées celles du dernier point de P_n, multipliées par $\lfloor \frac{n}{4} \rfloor$,
- soit s_x et s_y les sommes respectives des coordonnées x et y des points de P_n,
- soit m le point de coordonnées $(n+1-(s_x \mod (n+1)), n+1-(s_y \mod (n+1)))$,
- soit P'_n l'ensemble $P_n \cup \{m\}$,
- soit b le barycentre des points de P'_n,
- soit V_n l'ensemble de vecteurs $\{ p-b \mid p \in P'_n \}$ (dont la somme est nulle).
- trier ces vecteurs dans un ordre qui suit le sens de rotation positif (inverse des aiguilles d'une montre), en partant du demi-axe (Ox) comme minimum. En cas d'égalité de direction, l'ordre lexicographique sera utilisé pour départager,
- soit finalement $\overline{C}_n$ le polygone obtenu en partant de q, et en additionnant successivement les vecteurs obtenus précédemment.

De manière similaire, les convexes $\overline{D}_n$ sont construits à partir de Q_n.

7 Séparation de polygones convexes

Le calcul de distances entre polygones convexes est facilité si l'on sait que les polygones ne s'intersectent pas, ni ne se contiennent. On dit dans ce cas qu'ils sont séparés.
On peut remarquer que dans ce cas, parmi les droites passant par un sommet de chacun des polygones, il en existe deux qui séparent les polygones, qu'on appelle bi-tangentes.

Question à développer pendant l'oral : Proposez un algorithme de complexité linéaire, qui teste si deux polygones convexes sont séparés.

Nous allons maintenant sélectionner des paires de polygones convexes séparés. Pour ce faire, nous allons noter $(\overline{E}_a, \overline{F}_a)$ les paires $(\overline{C}_b, \overline{D}_b)$, telles que b est le plus petit entier supérieur ou égal à a tel que $\overline{C}_b$ et $\overline{D}_b$ soient séparés.

Question 8 *Que vaut b pour $a =$* ***a)*** *10* ***b)*** *1000* ***c)*** *100000 ?*

8 Distances entre polygones convexes séparés

Question 9 *Que valent :* ***a)*** $d(\overline{E}_{10}, \overline{F}_{10})$ ***b)*** $d(\overline{E}_{1000}, \overline{F}_{1000})$ ***c)*** $d(\overline{E}_{100000}, \overline{F}_{100000})$ *?*

Question 10 *Que valent :* ***a)*** $d_H(\overline{E}_{10}, \overline{F}_{10})$ ***b)*** $d_H(\overline{E}_{1000}, \overline{F}_{1000})$ ***c)*** $d_H(\overline{E}_{100000}, \overline{F}_{100000})$ *?*

Question à développer pendant l'oral : Vous décrirez vos algorithmes et justifierez leur complexité, qui doit être au pire en $O(n)$.

9 Cas général

Question à développer pendant l'oral : Décrivez un algorithme de calcul de $d_H(\overline{P}_n, \overline{Q}_n)$.