Géométrie algorithmique, exercices
30 novembre 2005

1 Delaunay et échantillonage de courbes

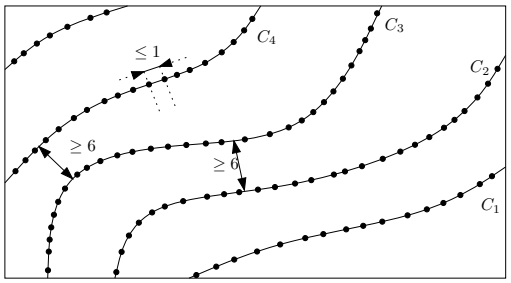

FIG. 1 –

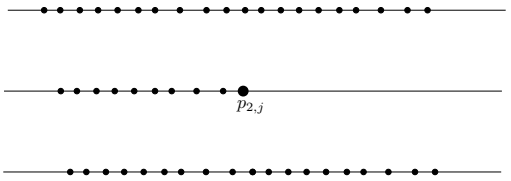

FIG. 2 –

On a un réseau de courbes à peu près parallèles C_1, C_2, C_3...On suppose que la distance entre deux courbes est toujours supérieure à 6, le rayon de courbure de ces courbes est également supérieur à 6. Ces courbes sont échantillonnées, deux points successifs sur une courbe ont une distance inférieure à 1 (Figure 1).

On calcule la triangulation de Delaunay de l'ensemble de ces points.

1.1 Delaunay «reconstruit» les courbes

Question 1

Montrer que deux points successifs $p_{i,j}$ et $p_{i,j+1}$ sur une courbe C_i sont reliés dans la triangulation de Delaunay.

1.2 Degré dans Delaunay

Supposons dans cette question, pour simplifier, que nos courbes sont des droites parfaitement parallèles avec une distance de 6. On considère la triangulation de Delaunay des points sur C_1 et C_3 et des points $p_{2,k}$ pour $k \leq j$ mais pas les $p_{2,k}$ pour $k > j$.

Question 2

Donner une borne inférieure sur le degré de $p_{2,j}$ dans la triangulation (Figure 2).

1.3 Insertion dans Delaunay

Supposons maintenant que les points sur chaque droite ont des écarts de longueur 1 exactement, que les points sur C_1 et C_3 soient déjà insérés dans la triangulation de Delaunay. Si on insère les points dans l'ordre $p_{2,1}$, $p_{2,2}$, $p_{2,3}$...$p_{2,k}$, donner une borne inférieure sur le nombre de triangles créés au cours de l'insertion de cette séquence de points.

Question 3

Proposer un autre ordre d'insertion, en donnant une estimation du nombre de triangle créés (et en essayant de minimiser ce nombre).

2 Recherche de plus proche voisin

Étant donné un ensemble $\mathcal{S}$ de n points dans le plan et un point du plan p on cherche à déterminer $NN_{\mathcal{S}}(p)$ le point de $\mathcal{S}$ le plus proche de p. On note $DT(\mathcal{S})$ la triangulation de Delaunay de $\mathcal{S}$.

w est initialisée avec un sommet quelconque de $DT(\mathcal{S})$.

```
N = un sommet de DT(S) ;
dmin = |pN| ;
Faire {
        v = N ;
        Pour chaque voisin w de v dans DT(S) {
                si ( dmin > |pw| ) {
                dmin = |pw| ;
                N = w ;
                }
        }
} tant que (N ≠ v) ;
return N ;
```

2.1

Démontrer que ce programme calcule le plus proche voisin de v.

2.2

Quelle est la complexité dans le cas le pire de cet algorithme ? Donner un exemple réalisant cette pire complexité ($\forall n$).

2.3

Si on modifie l'algorithme comme suit :
Variante :

```
N = un sommet de DT(S) ;
dmin = |pN| ;
Faire {
        v = N ;
        Pour chaque voisin w de v dans DT(S) {
                si ( dmin > |pw| ) {
                dmin = |pw| ;
                N = w ;
                break ; // Sortir de la boucle "Pour"
                }
        }
} tant que (N ≠ v) ;
return N ;
```

Qu'apportent la variante ?
Le nombre d'exécutions de l'instruction $N = w$ est-il modifié ?
Dans quel sens ?
Est-ce que l'on calcule bien toujours le plus proche voisin ?
Commentaires ?