Algorithmique et Programmation

—

Examen sur machine

—

G1: keriven*(at)*certis.enpc.fr G2: thorstensen*(at)*certis.enpc.fr
G3: aganj*(at)*certis.enpc.fr G4: etyngier*(at)*certis.enpc.fr
G5: segonne*(at)*certis.enpc.fr G6: chariot*(at)*certis.enpc.fr

—

08/12/06

—

[Conseil : bien lire chaque question avant de tenter d'y répondre]

1 Enoncé

1.1 Voyageur de commerce par recuit simulé

Soient n villes situées aux points $M_1, ..., M_n$, le problème du voyageur de commerce consiste à trouver un circuit fermé de longueur minimale passant par toutes les villes. Il s'agit donc de permuter les villes entre elles pour minimiser

$$l(M_1, ..., M_n) = \Sigma_{i=1}^{n-1} d(M_i, M_{i+1}) + d(M_n, M_1)$$

où $d(A, B)$ est la distance entre A et B. Une méthode classique pour trouver une solution approchée à ce problème est de procéder à un "recuit simulé" :

1. Partir d'un circuit quelconque C
2. Pour un "grand" nombre de fois
   (a) Modifier aléatoirement C en un circuit D
   (b) Si $l(D) < l(C)$ alors remplacer C par D
   (c) Sinon, remplacer C par D avec une probabilité $e^{-(l(D)-l(C))/T}$, où T est une constante à choisir.

1.2 Travail demandé

1. **Travailler en local dans** `D:\nom_prenom` ;
2. Y créer une solution `Examen` et lui ajouter un projet "Winlib" de nom `voyageur` ;
3. Un circuit sera mémorisé comme un tableau de `Pixel`[1] de taille constante `n` (valeur raisonnable : $n = 20$) ;
4. Faire et utiliser une fonction qui génère un circuit correspondant à n villes situées en des positions aléatoires dans la fenêtre ;
5. Faire et utiliser une fonction qui affiche ce circuit[2] ;
6. Faire et utiliser une fonction qui calcule la longueur de ce circuit ;
7. Faire et utiliser une fonction qui transforme un circuit en un autre par échange de deux villes choisies au hasard ;
8. Implémenter le recuit simulé sans l'étape (c). Afficher le circuit et sa longueur à chaque fois qu'il change. L'algorithme devrait rester coincé dans un minimum local ;
9. Rajouter l'étape (c). On rappelle que `double(rand())/RAND_MAX` est un nombre aléatoire entre 0 et 1. Ajuster T pour ne pas toujours remplacer C par D !
10. Choisir maintenant une valeur de T qui décroît en $1/\sqrt{t}$, où t est le nombre d'essais, de façon à accepter de plus en plus rarement un D plus long que C ;
11. Pour vraiment faire marcher l'algorithme, il faut programmer une nouvelle façon de transformer un circuit en un autre. La voici : choisir deux villes au hasard et retourner le chemin entre les deux villes, c'est-à-dire

    $$\text{transformer } (M_1, ..., M_n) \text{ en } (M_1, ..., M_{i-1}, M_j, M_{j-1}, ..., M_{i+1}, M_i, M_{j+1}, ..., M_n)$$

    Programmer cette nouvelle façon de faire ;
12. A la fin de l'examen, **ne pas vous déconnecter et attendre que le surveillant passe recupérer votre travail**.

[1] On rappelle que `Pixel` est une structure de la Winlib, contenant les champs `double x` et `double y`
[2] Les fonctions de la Winlib telles que "OpenWindow" ou "DrawLine" (confère Annexe C du poly) seront *très* utiles.