INF4705 – Analyse et conception d’algorithmes

TP3 – Automne 2018

|  |  |
| --- | --- |
| **Nom, prénom, matricule des membres** | Badirou Bissola, 1770039  Badirou Salim, 1733231 |
| **Note finale / 14** | 0 |

# Informations techniques

* Répondez directement dans ce document ODT avec LibreOffice. Utilisez LibreOffice et non Word, sinon ce document sera corrompu.
* La correction se fait à même le rapport.
* Avant le 4 décembre à 23h59, vous devez faire une remise électronique en suivant les instructions suivantes :
  + Le dossier remis doit se nommer matricule1\_matricule2\_tp3 et doit être compressé sous format zip.
  + À la racine de ce dernier, on doit retrouver :
    - Ce rapport sous format ODT.
    - Un script nommé *tp.sh* servant à exécuter les différents algorithmes du TP. L’interface du script est décrite à la fin du rapport.
    - Un fichier texte nommé *emails.txt* contenant le courriel de chaque membre de l’équipe
    - Le code source et les exécutables
* Vous avez le choix du langage de programmation utilisé. Le code et les exécutables soumis devront être compatible avec les ordinateurs de la salle L-4714. Conseil pour le TP3 : Compilez avec -O3 si vous utilisez C++.
* Si vous utilisez des extraits de codes (programmes) trouvés sur Internet, vous devez en mentionner la source, sinon vous serez sanctionnés pour plagiat.

# Mise en situation

Le dernier travail pratique se fera dans le cadre du concours du meilleur algorithme pour la session d'automne 2018. Le travail demandé consiste à concevoir et implanter un algorithme de votre cru pour résoudre un problème combinatoire. Le classement des équipes déterminera votre note pour la qualité de l'algorithme. Votre algorithme sera exécuté sur 3 exemplaires de notre choix pendant 3 minutes.

Vous êtes administrateur d’une compagnie de voyages organisés, et dans le cadre de vos fonctions vous devez planifier des parcours pour des groupes de touristes dans différentes villes. Chaque ville compte plusieurs centres d’intérêt comme des musées, des monuments, etc, qui sont dispersés un peu partout. Les centres d’intérêt ont des niveaux d’appréciation variés (par exemple, la tour Eiffel aurait un niveau d’appréciation très élevé). De plus, le temps alloué pour les parcours est limité. Vous devez créer des parcours de la meilleure qualité possible (dont la somme des niveaux d’appréciation des centres d’intérêt visités soit la plus élevée possible) tout en respectant la limite de temps allouée. Vos parcours doivent débuter et terminer à l’hôtel où logent les touristes. Évidemment, vous ne pouvez pas visiter un même centre d’intérêt plus d’une fois.

On peut représenter le problème sous la forme d'un graphe complet, la valeur d'un noeud indiquant le niveau d’appréciation de ce centre d’intérêt, et la valeur d'une arête représentant la combinaison du temps nécessaire pour passer d'un centre d’intérêt à un autre avec le temps nécessaire pour faire la visite (par conséquent, l'inégalité triangulaire ne tient pas). L'objectif est de trouver un chemin d'une durée inférieure ou égale à la limite de temps, qui maximise la somme des niveaux d’appréciation des noeuds visités. Le chemin doit être fermé (le noeud de départ est le même que le noeud d'arrivée, i.e., l'hôtel) et simple (les noeuds ne peuvent être visités qu'une seule fois).

Le rapport pour ce dernier travail pratique est assez succinct. Vous êtes encouragés à terminer ce travail assez tôt afin de ne pas compromettre la préparation à vos examens finaux.

# Jeu de données

Pour tester votre algorithme, vous disposez d’un jeu de données de 10 exemplaires. La première ligne contient le nombre de centres d’intérêt *n*. Les *n* lignes suivantes représentent la matrice d’adjacence du graphe (l’hôtel est à *n*=0). L’avant-dernière ligne indique le temps maximal alloué pour le parcours, et la dernière ligne représente le niveau d’appréciation de chaque centre d’intérêt.

### Description de votre algorithme

|  |  |
| --- | --- |
| 0 | / 1 pt |

*Décrivez brièvement votre algorithme en quelques phrases.*

### Présentation de votre algorithme

|  |  |
| --- | --- |
| 0 | / 2 pt |

*Sous forme de pseudo-code et incluant une analyse de complexité théorique des principales fonctions. Si vous préférez écrire vos équations en Latex, vous pouvez ajouter un pdf à la remise avec la réponse à cette question et le mentionner ici.*

### Justification de l’originalité de votre algorithme

|  |  |
| --- | --- |
| 0 | / 3 pt |

*La conception de votre algorithme sera jugée avec les critères suivants :*

* *Lien avec le contenu du cours*
* *Originalité*
* *Initiative*

### Votre algorithme est-il assuré de trouver une solution optimale?

|  |  |
| --- | --- |
| 0 | / 1 pt |

*Répondez simplement “oui” ou “non”. Aucune justification requise.*

# Autres critères de correction

### Respect de l’interface tp.sh

|  |  |
| --- | --- |
| 0 | / 1 pt |

Utilisation :

./tp.sh -e [path\_vers\_exemplaire]

Lorsque exécuté sans le paramètre -p, le programme affiche uniquement la qualité d’un parcours à chaque fois qu’une meilleure solution est trouvée. Votre programme est sensé s’exécuter tant et aussi longtemps qu’il n’est pas manuellement interrompu.

Argument optionnel :

-p Chaque fois qu’une meilleure solution est trouvée, le programme affiche le nouveau parcours (au lieu de simplement afficher la qualité de la solution).

Important : l’option -e doit accepter des fichiers avec des paths absolus.

Le script tp.sh ne vous est pas fourni, mais vous pouvez facilement adapter celui du TP2.

Voici un exemple d’affichage (dans cet exemple, après avoir trouvé un parcours initial, le programme en a par la suite trouvé un meilleur) :

0 32 9 41 12 15 39 3 0

0 6 22 31 13 8 12 41 45 17 0

Un script de vérification de solution vous est fourni (sol\_check.py, les instructions d’utilisation sont dans le code source). Ce script vous indiquera si l’affichage est correct et si votre solution est valide. C’est ce script qui sera utilisé pour la correction, donc assurez-vous qu’il reconnaisse vos solutions.

### Qualité de l’algorithme

|  |  |
| --- | --- |
| 0 | / 4 pt |

### Qualité du code

|  |  |
| --- | --- |
| 0 | / 1 pt |

### Présentation générale (concision, qualité du français, etc.)

|  |  |
| --- | --- |
| 0 | / 1 pt |

### Pénalité pour retard ou autre

|  |
| --- |
| 0 |

* -1 pt / journée de retard, arrondi vers le haut. Les TPs ne sont plus acceptés après 3 jours.