INF8775 – Analyse et conception d’algorithmes

TP3 – Automne 2021

|  |  |
| --- | --- |
| **Nom, prénom, matricule des membres** | Nom, Prénom, matricule  Nom, Prénom, matricule |
| **Note finale / 15** | **0** |

Informations sur la correction

* Répondez directement dans ce document. La correction se fait à même le rapport.
* Le date limite pour rendre ce TP est le 7 décembre à 23h59. Vous devez faire une *remise électronique sur Moodle* en suivant les instructions suivantes :
  + Le dossier remis doit se nommer matricule1\_matricule2\_tp1 et doit être compressé sous format zip.
  + À la racine de ce dernier, on doit retrouver :
    - Ce rapport au format odt ou docx
    - Un script nommé *tp.sh* servant à exécuter l’algorithme du TP. L’interface du script est décrite à la fin du rapport.
    - Le code source et l'exécutable.
* Vous avez le choix du langage de programmation utilisé. Notez que le code et l'exécutable soumis seront testés sur les ordinateurs de la salle L-4714 et doivent être compatibles avec cet environnement. En d’autres mots, tout doit fonctionner correctement lorsque le correcteur exécute votre script *tp.sh* sur un des ordinateurs de la salle.
* La commande *chmod +x tp.sh* rendra le script *mon\_script.sh* exécutable. Pour l’exécuter il s’agira de faire *./tp.sh*

Mise en situation

Ce dernier travail pratique se fera dans le cadre du concours du meilleur algorithme pour la session d’automne 2023. 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 chacun.

Le problème à résoudre est le suivant :

Un groupe d’écoliers du primaire doit être placé en rang (l’un derrière l’autre). On vous demande d’accomplir cette tâche en tenant compte de la consigne suivante : certains écoliers vont se chamailler si placés immédiatement un derrière l’autre. On vous donne donc une liste des paires d’écoliers qui ne peuvent pas se succéder sans se chamailler. On souhaite également que le plus grand nombre possible d’écoliers n’aient pas leur vue obstruée par un élève plus grand situé devant eux (directement ou plus loin devant) : vous devez donc minimiser le nombre d’écoliers ayant leur vue obstruée et le nombre d’écoliers se chamaillant selon le coût suivant : écoliers obstrués + 10\*(paires d’écoliers se chamaillant).

Jeu de données

Un fichier gen.py vous est fourni pour générer des exemplaires avec la commande :   
**python gen.py -n {number of students} -p {number of pairs}**

Chaque fichier d’exemplaire est organisé comme suit :

<nb d’écoliers>  
<nb de paires d’écoliers qui se chamaillent>  
<taille de l’écolier 1>  
 .  
 .  
 .  
<taille de l’écolier *n*>  
<écolier *a* de la paire 1> <écolier *b* de la paire 1>  
 .  
 .  
 .  
<écolier *a* de la paire *m*> <écolier *b* de la paire *m*>

Q1 – Description de votre algorithme

*Décrivez en quelques phrases votre algorithme.*

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

Q2 – Présentation de votre algorithme

*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. Pas besoin de faire une analyse empirique de la complexité.*

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

Q3 – Justification de l’originalité de votre algorithme

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

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

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

Q4 – Votre algorithme est-il assuré de trouver la solution optimale ?

*Répondez simplement “oui” ou “non”. Si oui, justifiez.*

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

Autres critères de correction

Respect de l’interface tp.sh

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

Utilisation :

tp.sh -e [chemin\_vers\_exemplaire] [-p]

Lorsque le script est exécuté sans le paramètre -p, le programme affiche uniquement le nombre d’élèves dont la vue est obstruée plus le nombre de paires incompatibles (multiplié par 10) sur une nouvelle ligne, à chaque fois qu’une meilleure solution est trouvée.

**Argument optionnel** :

-p Chaque fois qu’une meilleure solution est trouvée, le programme affiche cette nouvelle solution. Le format de cette solution est décrit ci-dessous

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

**Format de la solution** : Pour une même solution, il faut afficher tous les élèves sur une même ligne, un à la suite de l’autre, séparés par un espace. Leur ordre dans la ligne correspond à l’ordre dans lequel ils sont installés en classe. Pour afficher une nouvelle solution il faut passer à la ligne suivante.  
L’exemple ci-dessous présente deux solutions pour un exemplaire avec 5 élèves. Dans la deuxième solution on voit que l’élève au premier rang est le numéro 2, celui au deuxième rang est le numéro 0, etc…

0 1 2 3 4  
2 0 3 1 4

Seulement le score de la dernière solution trouvée sera pris en compte.

Il est d’une grande importance de respecter le format de sortie, puisque l’évaluation de la qualité de votre algorithme passera d’abord par le test de l’intégrité de la solution. Ce dernier échouera si votre format de sortie n’est pas respecté et votre programme serait considéré non fiable ce qui entraîne une note de 0 sur 4 pour la qualité de l’algorithme. Un script *check\_sol.py* vous est fourni avec le sujet. C’est ce script qui sera utilisé pour lire ce que retournera vos programmes. Donc utilisez-le pour vérifier que votre sortie est bien au bon format.

Lors de l’évaluation, votre programme sera exécuté sur 3 exemplaires de notre choix. Nous utiliserons alors les ordinateurs du laboratoire L-4714 pour tester vos algorithmes. Nous considérerons la dernière solution affichée après 3 minutes de calcul. Dès qu’une solution (sous-optimale) a été trouvée, affichez-la à la sortie standard dans le format décrit plus haut ; votre programme sera interrompu une fois la limite de temps atteinte.

Qualité de l’algorithme

*Les points de cette question sont répartis comme suit :  
Nous allons exécuter votre code sur 3 exemplaires de notre choix. Pour chaque exemplaire la sortie de votre code sera envoyée vers le script de vérification, et nous classerons sur chaque exemplaire les différents binômes.  
Pour chacun des 3 exemplaires, le quart des groupes ayant la meilleure solution remportera 1 point, le deuxième quart 0.75, le troisième quart 0.5 et le dernier quart 0.25 points.*

*Calcul du pointage pour le classement :*

*Une bonne solution obtenue très rapidement peut être aussi intéressante qu’une solution légèrement meilleure obtenue beaucoup plus lentement. C’est pourquoi nous allons accorder un bonus aux algorithmes qui n’utilisent pas le maximum du temps imparti. Le score final se calculera donc de la façon suivante : Sglobal = -Sexemplaire / a(t)*

*Ou Sexemplaire représente le coût de la solution à l’exemplaire (Sexemplaire = écoliers obstrués + 10\*(paires d’écoliers se chamaillant)).*

*a(t) représente le facteur lié au temps t (s), défini comme : a(t) = min(1.5, 5/t + 1),*

*Sglobal, le score utilisé pour le classement des équipes.*

*Un bonus de 1 point est donné si vous trouvez une solution* ***qui bat un algorithme baseline*** *dans le temps imparti.*

***Cela signifie que si vous avez une solution de relativement bonne qualité mais un algorithme peu performant vous aurez quand même 1.75/4.***

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

Qualité du code et présentation générale

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

Pénalités

|  |
| --- |
| 0 |

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