Devoir maison

Recherche opérationnelle S3.

Guangyue CHEN
Numéro d'étudiant : 20170020

Ce devoir maison a été généré automatiquement et aléatoirement pour Guangyue CHEN. Il contient 3 exercices : un exercice d'application du cours sur 1 point, un exercice intermédiaire sur 1 point et un algorithme à coder sur 2 points. Pour répondre à ce devoir, vous devez rendre, sur `exam.ensiie.fr`*, un fichier texte nommé* `chenguangyue_20170020.txt`*. Ce fichier devra contenir* ***dans l'ordre*** *les solutions des 3 exercices tels que demandé dans chaque exercice. Toute ligne vide dans le fichier sera ignorée (vous pouvez donc sauter des lignes entre 2 exercices, ou au sein d'un exercice). ATTENTION : un non respect du format pour un exercice entraînera la note de 0 pour l'exercice. Il vous est possible de tester votre format ainsi que votre dernier exercice avec le script* `check` *qui vous a été fourni avec ce sujet. Vous pouvez par exemple le copier avec votre fichier solution sur une machine de l'école et exécuter*

```
./check chenguangyue_20170020.txt
```

Si votre fichier est au bon format, le script vous le dira explicitement. Le code de votre dernier exercice peut être dans ce cas testé sur 100 tests unitaires. Si un test ne passe pas, le script vous l'affichera pour que vous puissiez corriger votre code. Vous pouvez demander de l'aide aux professeur/chargés de TD, ou aux autres élèves, mais sachez que le sujet est différent pour tout le monde.

Exercice — ***#1-7-1 Probabilités d'une file d'attente***

On considère une file d'attente à 1 guichet avec un taux de naissance constant $\lambda = 65$ et un taux de mort constant $\mu = 73$. Calculez la probabilité P_9 qu'il y ait 9 personnes dans la file en régime permanent, si ce régime existe.

Vous donnerez cette réponse au format suivant : vous noterez en premier le numéro de l'exercice précédé par un `#`,
puis une ligne contenant un flottant égal à la probabilité P_9 si le régime permanent existe. Ce nombre doit être écrit avec la notation scientifique, arrondi 2 chiffres après la virgules. Si le régime permanent n'existe pas alors la ligne doit contenir à la place le nombre `-1`.

Par exemple :

```
# 1-7-1
3.80E-1
```

Exercice — ***#2-5-3 Algorithme du simplexe***

Soit le programme linéaire suivant :

$$\begin{cases} \min & -x_1 & -2x_2 & & & & & \\ & x_1 & & x_3 & & & = 3 \\ & 3x_1 & +2x_2 & & x_4 & & = 11 \\ & x_1 & +x_2 & & & x_5 & = 5 \\ & x_1, & x_2, & x_3, & x_4, & x_5, & \geq 0 \end{cases}$$

Donnez les différentes solutions de base réalisables explorées par l'algorithme du simplexe en partant du point $(x_1, x_2) = (0, 0)$. Ces solutions de bases sont toutes à coordonnées entières. A chaque itération, au moment de choisir la variable qui rentrera dans la base, si l'algorithme peut indifféremment choisir parmi deux variables car elles ont même coût réduit, il choisira celle de plus petit indice. De même quand il doit choisir la variable sortant de la base. Remarque : vous devez indiquer $(0, 0)$ comme étant la première solution visitée.

Vous donnerez cette réponse au format suivant : vous noterez en premier le numéro de l'exercice précédé par un `#`,
puis au plus 3 lignes contenant 2 entiers x et y séparés par une espace où le i^{e} couple (x, y) est la i^{e} solution de base réalisable visitée par l'algorithme du simplexe.

Par exemple :

```
# 2-5-3
0 0
0 1
0 2
1 0
...
```

Exercice — ***#3-6-2 Algorithme de calcul des classes d'états communiquants***

Coder en C un algorithme calculant les classes d'état communiquants d'une chaîne de Markov. Pour cela, vous coderez une fonction avec la signature suivante :
`void communicating_classes_algorithm(int n, float** transitions, int* comclasses);`

- `n` est le nombre d'états de la chaîne de Markov numérotés de 1 à n.
- `transitions` est un tableau de `n` × `n` flottants où `transitions[i-1][j-1]` est la probabilité de transition de l'état i vers l'état j.
- `comclasses` est un tableau de `n` entiers. En entrée, `comclasses[i-1]` vaut -1 pour tout $1 \leq i \leq n$. Votre fonction doit affecter dans `comclasses[i-1]` un entier entre 1 et n, quel qu'il soit, tel que, si deux états i et j sont dans la même classe d'états communiquants si et seulement si `comclasses[i-1]` = `comclasses[j-1]`.

On garantit, que, quelque soit l'entrée $2 \leq n \leq 100$.

Votre fonction doit compiler avec `gcc -Wall -Wextra -ansi` et se terminer en moins de 1 seconde. Vous pouvez déclarer d'autres fonctions ou des structures mais ne pouvez ni écrire de `#define` ou de `#include`. Vous pouvez utiliser tout ce qui est défini dans `stdlib.h` à l'exception de `system`. Cette bibliothèque sera inclue automatiquement. Ne mettez pas de commentaire. Elle sera testée sur 100 cas. Votre note est de 2 si tous les tests sont réussis, 1 si plus de 10 pourcents mais pas tous, et 0 sinon. **Conseil** : n'utilisez pas de tableaux, utilisez des pointeurs à la place.

Vous donnerez cette réponse au format suivant : vous noterez en premier le numéro de l'exercice précédé par un `#`, puis écrivez votre code.

Par exemple :

```
# 3-6-2
void communicating_classes_algorithm(int n, float** transitions, int* comclasses){
}
```