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 — ***#1-1-1 Problème de partition d'entiers : méthode itérative***

On souhaite partager l'ensemble d'entiers $X = \{x_1, x_2, \ldots, x_8\}$ en deux parties de sommes égales. On utilise pour cela la version itérative l'algorithme de programmation dynamique vu en cours. Pour tout $0 \leq i \leq 8$ et tout $0 \leq j \leq (\sum_{x \in X} x)/2$, on calcule $f(i, j)$ qui vaut TRUE s'il existe, parmi $x_1, x_2, \ldots, x_i$, des entiers dont la somme fait j et FALSE sinon. (si $i = 0$, la somme fait 0)

$$X = \{2, 10, 9, 9, 4, 2, 1, 3\}$$

A l'itération 2, on obtient les résultats suivants :

	0	1	2	3	4	5	6	7	8	9	10
$f(2, j)$	True	False	True	False	False	False	False	False	False	False	True
		11	12	13	14	15	16	17	18	19	20
$f(2, j)$		False	True	False	False	False	False	False	False	False	False

Donnez la valeur de $f(3, j)$ pour tout $1 \leq j \leq 20$.

Vous donnerez cette réponse au format suivant : vous noterez en premier le numéro de l'exercice précédé par un `#`,
puis 1 ligne contenant 21 entiers binaires 0 ou 1 séparés par 1 espace où le j^{e} chiffre de la ligne est 0 si $f(3, j-1)$ est FALSE, et 1 sinon.

Par exemple :

```
# 1-1-1
0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

Exercice — ***#2-4-1 Couverture minimum par méthode de séparation et d'évaluation***

On veut couvrir l'ensemble $U = [\![1; 5]\!]$ suivant avec un minimum d'ensemble parmi ceux de S. Autrement dit, combien d'ensemble de S peut-on laisser au minimum tel que chaque entier de $[\![1; 5]\!]$ soit dans au moins un ensemble ?

S	1	2	3	4	5
s_1	1	0	0	0	0
s_2	0	1	0	1	0
s_3	0	1	0	0	1
s_4	1	0	1	0	1
s_5	0	0	1	1	0
s_6	0	1	0	0	0
s_7	1	0	0	1	1

Appliquez l'algorithme suivant EXPLORE(U, 1, 0) et donnez la valeur des variables `explore`, `cut`, `update` et `best` à la fin de l'algorithme.

$explore \leftarrow 0$; $cut \leftarrow 0$; $update \leftarrow 0$; $best \leftarrow 8$
procedure EXPLORE(U, i, c)
  Si($\bigcup_{j=i}^{7} s_i \neq U$) $cut \leftarrow cut + 1$; **Renvoyer**
  $msize \leftarrow \left(\max_{j=i}^{7}(|s_j \cap U|)\right)$
  Si($msize \neq 0$) $bound \leftarrow c + \lceil |U|/msize \rceil$ **Else** $bound \leftarrow c$
  Si($bound \geq best$) $cut \leftarrow cut + 1$; **Renvoyer**
  $explore \leftarrow explore + 1$
  Si U est vide **Alors**
    Si($best > c$) $best = c$; $update \leftarrow update + 1$
  Sinon
    EXPLORE($U \backslash s_i, i + 1, c + 1$)
    Si($bound \geq best$) $cut \leftarrow cut + 1$; **Renvoyer**
    EXPLORE($U, i + 1, c$)

Vous donnerez cette réponse au format suivant : vous noterez en premier le numéro de l'exercice précédé par un `#`,
puis 1 ligne contenant 4 entiers séparés par 1 espace égaux dans l'ordre à la valeur des variables `explore`, `cut`, `update` et `best` à la fin de l'algorithme.

Par exemple :

```
# 2-4-1
12 3 5 4
```

Exercice — ***#3-2-1 Algorithme de Johnson : ordre optimal***

Coder en C l'algorithme de Johnson pour déterminer l'ordre optimal d'exécution de tâches du deux machines.

Pour cela, vous coderez une fonction avec la signature suivante :

```
void johnson_ordering_algorithm(int n, int* t1, int* t2, int* ordering);
```

— `n` est le nombre de tâches, numérotées de 1 à n.

— `t1` est un tableau de `n` entiers où `t1[i-1]` est la durée d'exécution de la tâche i sur la machine M_1.

— `t2` est un tableau de `n` entiers où `t2[i-1]` est la durée d'exécution de la tâche i sur la machine M_2.

— `ordering` est un tableau de `n` entiers. En entrée, `ordering[i-1]` $= 0$ pour tout $1 \leq i \leq n$. Pour tout $1 \leq i \leq n$, votre fonction doit affecter dans `ordering[i-1]` le numéro de la i^e tâche dans l'ordre qui minimise la durée totale d'exécution. Par exemple, si la première tâche est la tâche 3, vous devez affecter 3 à `ordering[0]`.

On garantit, que, quelque soit l'entrée $n \geq 2$, les durées sont positives, chaque tâche a une durée différente sur chaque machine, et deux tâches ont des durées distinctes sur une même machine.

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.

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 :

```
#3-2-1
void johnson_ordering_algorithm(int n, int* t1, int* t2, int* ordering){
}
```