PCSI 1 **Devoir à la maison d'informatique n° 2 :** 2023/2024

A rendre pour le : lundi 22 avril

Les fonctions doivent être commentées.
L'indentation doit être représentée par un trait vertical.
La copie doit être manuscrite.

Alliances d'entreprises

Dans tout le problème, une liste c d'entiers strictement positifs contient les chiffres d'affaires $c[0]$, $c[1]$, …, $c[n-1]$ en euros de n petites et moyennes entreprises ($n > 0$). On supposera toujours, sans le vérifier, que c est une telle liste.
Ces entreprises décident de s'allier pour devenir leader de leur secteur dominé par l'entreprise X qui a un chiffre d'affaire de Obj euros. Il s'agit donc de trouver un sous-ensemble d'entreprises I tel qu'on ait $\sum_{i\in I} c[i] \geq Obj$. Un tel ensemble d'entreprises est appelé une *alliance réussie*. Malheureusement, les alliances apportent parfois des désagréments. Pour minimiser ce risque, on cherche à réaliser une *alliance réussie stable* (une alliance *stable* en abrégé) vérifiant $\sum_{i\in I'} c[i] < Obj$ pour tout I' strictement inclus dans I.

Partie 1 : Préliminaires

1. Ecrire une fonction Verif(c,Obj) qui renvoie True si $\sum_{i=0}^{n-1} c[i] \geq Obj$ et False sinon.
   On supposera dans toute la suite de l'énoncé qu'une *alliance réussie stable* existe c'est-à-dire que $\sum_{i=0}^{n-1} c[i] \geq Obj$ et on dira que le chiffre d'affaires Obj est l'*objectif* des alliances considérées.

2. Ecrire une fonction estCroissante(c) qui renvoie True si les entreprises sont classées par chiffre d'affaires croissants ($c[0] \leq c[1] \leq \cdots \leq c[n-1]$) et False sinon.
   Déterminer la complexité de cette fonction.

3. On considère la fonction suivante :

```
def fonction1(c) :
    n=len(c)
    t=[0]*n
    for i in range(n) :
        for j in range(i+1) :
            t[i]=t[i]+c[j]
    return t
```

   Expliquer ce que renvoie cette fonction.
   Déterminer la complexité de la fonction fonction1.
   Ecrire une autre fonction fonction2(c) qui renvoie le même résultat que fonction1(c) mais qui ait une meilleure complexité.

Partie 2 : Alliances

Une alliance est représentée par un tableau a de booléens tel que $a[i]$ vaut True si l'entreprise est dans l'alliance et Faux sinon.

1. On considère la fonction suivante :

```
def fonction3(c,a) :
    if len(c) !=len(a) :
        print('a et c doivent être de la meme taille')
    else :
        s=0
        for i in range(len(c)) :
            if a[i]==True :
                s+=c[i]
        return s
```

   Expliquer ce que renvoie cette fonction.
   Déterminer la complexité de la fonction fonction3.

2. On considère la fonction suivante :

```
def fonction4(c,a,Obj) :
    return fonction3(c,a)>=Obj
```

Expliquer ce que renvoie cette fonction.
Déterminer la complexité de la fonction fonction4.

3. Ecrire une fonction estStable(c,a,Obj) qui retourne la valeur True si l'alliance représentée par le tableau a est une alliance *stable* pour l'objectif *Obj*. La valeur retournée est False sinon. Cette fonction devra être de complexité $O(n)$, cependant on ne demande pas de le prouver.
4. On suppose maintenant les entreprises triées selon leur chiffre d'affaire $c[0] \leq c[1] \leq \cdots \leq c[n-1]$. Ecrire une fonction allianceMin(c,Obj) qui renvoie la liste des numéros des entreprises d'une plus petite (en nombre d'entreprises) alliance *stable* pour l'objectif *Obj*. Cette fonction devra être de complexité $O(n)$, cependant on ne demande pas de le prouver.

Partie 3 : Calcul des alliances stables

Dans cette partie, on imprime toutes les *alliances stables* en énumérant toutes les alliances et en testant à chaque fois s'il s'agit d'une alliance *stable* pour l'objectif *Obj*. Au tableau a, on fait correspondre de manière unique le nombre *bin(a)* défini par :

$$bin(a) = a[0] * 2^0 + a[1] * 2^1 + \cdots + a[n-1] * 2^{n-1}$$

de représentation binaire $(a[0], a[1], \ldots, a[n-1])$. Ainsi, pour $n = 5$ et l'alliance des entreprises $\{1,3,4\}$, le nombre associé est 26 de représentation binaire $\overline{11010}^2$. Il suffit donc d'énumérer les nombres binaires qu'on peut écrire avec n bits pour énumérer toutes les alliances possibles.

1. Pour $n = 5$ et l'alliance des entreprises $\{2,4\}$, donner le nombre associé et sa représentation binaire.
2. Ecrire une fonction suivanteDe(a) qui modifie la liste a pour donner l'alliance suivante a' telle que $bin(a') = bin(a)+1$. Cette fonction retourne la valeur True si cette opération est possible et False sinon (dans ce cas, il n'y a pas d'alliance suivante). Cette fonction devra être de complexité $O(n)$, cependant on ne demande pas de le prouver.
3. Ecrire une fonction imprimer(a) qui imprime l'alliance correspondant au tableau a.
Ainsi, pour $n = 5$ et $a[0] = False, a[1] = True, a[2] = False, a[3] = True, a[4] = True$, l'impression donnera $[1,3,4]$.
4. Ecrire une fonction imprimerStables(c,Obj) qui affiche toutes les alliances *stables* pour l'objectif *Obj*.
Déterminer la complexité de cette fonction, qu'en pensez-vous ?