Épreuve d'informatique de l'X - 2006 - PSI

Alliance d'entreprises

Question 1 : On rappelle que n est supposé être une constante du programme.

```
1  def somme(c,a):
2      global n
3      res = 0
4      for i in range(n):
5          if a[i] :
6              res = res + c[i]
7      return(res)
```

Question 2 :

```
1  def estReussie(c, a, Obj):
2      return(somme(c,a)>=Obj)
```

Question 3 : Pour vérifier qu'une alliance est stable, il faut de vérifier que l'alliance est réussie (ligne 3) et que l'objectif n'est pas réalisé dès que l'on enlève un élément de l'alliance. Le calcul de la somme correspondante se fait en retirant du chiffre d'affaire de l'alliance le chiffre d'affaire de l'entreprise (test de la ligne 5). Si le résultat est supérieur à l'objectif, l'alliance n'est pas stable (ligne 6).

```
1  def estStable(c,a,Obj):
2      sommeAlliance = somme(c,a)
3      res = estReussie(c,a,Obj)
4      for i in range(n):
5          if a[i] and (sommeAlliance - c[i] >= Obj) :
6              return(False)
7      return(res)
```

Question 4 : Une alliance a plus de chance d'être stable si les chiffres d'affaire des sociétés qui la composent sont importants. Il suffit d'afficher les numéros des entreprises en partant de la fin jusqu'à ce que leur somme dépasse l'objectif. On suppose que l'objectif est réalisable.

```
1  def allianceMin(c, Obj):
2      global n
3      i = n-1
4      s = 0
5      while s < Obj :
6          print(i, sep=' ',end=' ')
7          s = s + c[i]
8          i = i-1
9      print()
10     return()
```

Question 5 : On balaye le tableau a tant qu'il y a des « `vrai` » pour atteindre la position du premier « `faux` ». Dans le même temps on met « `faux` » à la place de « `vrai` » dans les cases visitées. À la fin de la boucle `while` (lignes 4 à 6) la variable i correspond à l'indice de la première case non visitée. Si i est différent de n, l'opération est l'opération est possible, on met « `vrai` » dans la i-ème case et retourne « `vrai` » (ligne 9 à 11). Sinon, l'opération n'est pas possible, on retourne « `faux` » (ligne 8).

```
1  def suivantDe(a):
2      global n
3      i=0
4      while  i < n and a[i] :
5          a[i] = False
6          i = i+1
7      if i == n and a[n-1] == False :
8          return(False)
9      else :
10         a[i] = True
11         return(True)
```

Question 6 :

```
1  def imprimer(a):
2      global n
3      for i in range(n):
4          if a[i] :
5              print(i,end=' ')
6      print();
```

Question 7 :

```
1  def imprimerStables(c,Obj):
2      global n
3      a = n*[False]
4      while suivantDe(a) :
5          if estStable(c,a,Obj) :
6              imprimer(a)
```

Pour l'ordre de grandeur du nombre d'opérations, la boucle des lignes 4 à 6 étant parcourue autant de fois qu'il y a d'alliances possibles, on aura 2^n appels à la fonction `estStable` dont le coût est en $O(n)$ et au plus 2^n appels à la fonction `imprimer` dont le coût est aussi en $O(n)$. Le nombre d'opérations effectuées sera de l'ordre de $O(n.2^n)$.

Question 8 :

```
1  def cumuls(c, t):
2      global n
3      t[0] = c[0]
4      for i in range(1,n):
5          t[i] = c[i] + t[i-1]
```

Question 9 : Dans la boucle des lignes 4 à 9, on complète notre sélection d'entreprises jusqu'à ce que l'objectif soit réalisé. À chaque fois que l'on sélectionne une entreprise, on modifie le tableau a (ligne 8) et l'objectif à réaliser (ligne 9). On retourne enfin le numéro de la dernière entreprise ajoutée et le paramètre `Obj` modifié (ligne 11).

Les paramètres d'une fonction Python étant passés par valeurs ne sont pas modifiés en dehors de la fonction. C'est pour cette raison que je rajoute `Obj` comme paramètre de sortie.

```
1  def completer(t, c, a, Obj, k):
2      global n
3      i = k
4      while Obj > 0 :
5          i = 0
6          while t[i] < Obj :
7              i = i+1
8          a[i] = True
9          Obj = Obj - c[i]
10     return(i, Obj)
```

Question 10 : On commence par rechercher le ℓ de l'énoncé (boucle des lignes 4 et 5). Si $k = i + l + 1$ est égal à n, on s'arrête en retournant la valeur $(n, 0)$ (ligne 8), sinon on enlève les entreprises dont les numéros sont compris entre i et $i + l$ de notre sélection et on rajoute leur chiffre d'affaire à l'objectif (ligne 12). Puis on rajoute l'entreprise k à notre sélection (ligne 15) et on retranche son chiffre d'affaire de l'objectif (ligne 15). Enfin, on retourne le couple (k, Obj).

```
1  def prelude(t, c, a, Obj, i):
2      global n
3      l = 0
4      while i+l+1 < n and a[i+l+1] :
5          l = l+1
6      k = i+l+1
7      if k == n :
8          return(n,0)
9      else :
10         for j in range(i,k) :
11             if a[j]:
12                 Obj = Obj + c[j]
13                 a[j] = False
14         a[k] =True
15         Obj = Obj - c[k]
16         return(k,Obj)
```

Question 11 : On commence par créer et calculer le tableau des cumuls (lignes 3-4) puis on crée et on initialise le tableau des sélections (ligne 5). On vérifie que l'objectif est bien réalisable (lignes 6 et 7). On repère le plus petit des k tel que $t[k] \geqslant Obj$ (boucle des lignes 9 et 10). On sélectionne l'entreprise k (ligne 11), on diminue l'objectif à atteindre du chiffre d'affaire de l'entreprise k (ligne 12) puis on complète notre sélection pour atteindre notre nouvel objectif (ligne 13). On affiche la première alliance stable (ligne 14). La boucle des lignes 15 à 19, permet de calculer et d'afficher les sélections successives en suivant l'algorithme décrit par l'énoncé.

```
1  def imprimerAStables1(c, Obj):
2      global n
3      t = n*[0]
4      cumuls(c,t)
5      a = n*[False]
6      if t[n-1] < Obj :
7          return("Erreur : objectif irréalisable")
8      k = 0;
9      while t[k] < Obj :
10         k = k+1
11     a[k] = True
12     Obj = Obj-c[k]
13     (i,Obj) = completer(t,c,a,Obj,k)
14     imprimer(a)
15     while k < n :
16         (k,Obj) = prelude(t,c,a,Obj,i)
17         if k < n :
18             (i,Obj) = completer(t,c,a,Obj,k)
19             imprimer(a)
```