Épreuve d'informatique de l'X - 2007 - PSI

Découpe de tissu

Question 1 :

Pour $L = 14$, on trouve $M = 8$ ($1 \times 6 + 1 \times 8 = 14$ et $1 \times 3 + 1 \times 5 = 8$)
Pour $L = 16$, on trouve $M = 10$ ($0 \times 6 + 2 \times 8 = 16$ et $0 \times 3 + 2 \times 5 = 10$)
Pour $L = 18$, on trouve $M = 10$ ($0 \times 6 + 2 \times 8 = 16 \leqslant 18$ et $0 \times 3 + 2 \times 5 = 10$)
Pour $L = 24$, on trouve $M = 15$ ($0 \times 6 + 3 \times 8 = 24$ et $0 \times 3 + 3 \times 5 = 15$)

Question 2 :

Pour $L = 14$, on trouve $M = 9$ ($1 \times 6 + 1 \times 8 = 14$ et $1 \times 4 + 1 \times 5 = 9$)
Pour $L = 16$, on trouve $M = 10$ ($0 \times 6 + 2 \times 8 = 16$ et $0 \times 4 + 2 \times 5 = 10$)
Pour $L = 18$, on trouve $M = 12$ ($3 \times 6 + 0 \times 8 = 18$ et $3 \times 4 + 0 \times 5 = 12$)
Pour $L = 24$, on trouve $M = 16$ ($4 \times 6 + 0 \times 8 = 24$ et $4 \times 4 + 0 \times 5 = 16$)

Question 3 : Avec la boucle (ligne 3 à 7), on calcule tous les prix de vente obtenus en coupant 0, 1, 2, ..., ou $L \div a_0$ fois le premier produit et le reste du tissu avec le deuxième produit. La ligne 4 permet de calculer j, le nombre de deuxièmes produits que l'on peut découper dans le reste du tissu. On calcule le prix de vente correspondant (ligne 5) puis on modifie le maximum si nécessaire (ligne 7).

```
1 def coutRuban2Produits(a0,v0,a1,v1,L) :
2     M = 0
3     for i in range(L/a0 + 1) :
4         j =  (L -i*a0)/a1
5         cout = i*v0+j*v1
6         if cout > M :
7             M = cout
8     return M
```

Question 4 : On reprend le même principe qu'à la question précédente avec cette fois-ci deux boucles imbriquées pour tenir compte des trois produits. La première boucle permet de prendre en compte les différents nombres de découpages possibles du premier produit, la deuxième boucle des différents découpages possibles du reste du tissu. La ligne 5 permet de calculer k, le nombre de troisièmes produits que l'on peut découper dans le reste du tissu.

```
1 def coutRuban3Produits(a0,v0,a1,v1,a2,v2,L) :
2     M = 0
3     for i in range(L/a0) :
4         for j in range((L -i*a0)/a1 + 1) :
5             k = (L-i*a0-j*a1)/a2
6             cout = i*v0+j*v1+k*v2
7             if cout > M :
8                 M = cout
9     return M
```

Question 5 : La boucle des lignes 3-9 sert à remplir le tableau M crée à la ligne 2. La boucle interne des lignes 5-8 permet de calculer $M(x)$ selon la formule donnée dans l'énoncé par le calcul d'un maximum. Le coût de notre algorithme est en $O(n.L)$ ce qui est bien inférieur au coût en $O(L^n)$ de la solution précédente.

```
1 def coutRuban(a, v, n, L) :
2     M = allouer(L+1)
3     for x in range(L+1) :
4         m = 0
5         for i in range(n) :
6             k = x - a[i]
7             if k >= 0 and M[k]+v[i] > m :
8                 m = M[k]+v[i]
9         M[x] = m;
10    return M[L]
```

Question 6 : On reprend le principe de la question précédente pour remplir les tableaux M et D. À la ligne 11, on rajoute la mise à jour de $D(x)$ en indiquant quel morceau de tissu a été découpé pour obtenir le coût maximum pour une longueur x de tissu. Le coût de cette fonction est identique au coût de la fonction précédente soit en $\underline{O(L.n)}$.

```
1  def decoupageRuban(a,v,n,L) :
2      M = allouer(L+1)
3      D = allouer(L+1)
4      for x in range(L+1) :
5          m = 0
6          D[x] = -1
7          for i in range(n) :
8              k = x-a[i]
9              if k >= 0 and M[k]+v[i] > m :
10                 m = M[k]+v[i]
11                 D[x] = i
12         M[x] = m
13     long = L
14     print("on coupe pour L = %2d :" % L)
15     while D[long] <> -1 :
16         print("%3d" % D[long])
17         long = long - a[D[long]]
```

Question 7 : Si la longueur de tissu est inférieure à la longueur de la pièce à découper (cas $x < a(i-1)$) alors $M(i,x) = M(i-1,x)$, sinon $M(i,x)$ sera égal au maximum de $M(i-1,x)$, coût maximum avec un découpage n'utilisant pas la i-ème pièce, et $v(i-1)+M(i-1,x-a(i-1))$ qui est le coût maximum obtenu en découpant la i-ème pièce dans le tissu de longueur x. On obtient ainsi la formule de récurrence suivante :

$$M(i,x) = \begin{cases} 0 & i = 0 \\ M(i-1,x) & \text{si } x < a(i-1) \\ \max(M(i-1,x), v(i-1) + M(i-1,x-a(i-1))) & \text{sinon} \end{cases}$$

Question 8 : La boucle des lignes 3-4 correspondant à la première ligne de notre formule de récurrence et les lignes 7 à 10 correspondent aux deux autres cas. L'ordre de grandeur du temps d'exécution est en $\underline{O(L.n)}$ dû aux deux boucles imbriquées des lignes 5 à 10.

```
1  def coutRubanSansRepetitions(a,v,n,L) :
2      M = make_matrice(n+1,L+1)
3      for x in range(L+1) :
4          M[0][x] = 0
5      for i in xrange(1,n+1) :
6          for x in xrange(L+1) :
7              if x < a[i-1] :
8                  M[i][x] = M[i-1][x]
9              else :
10                 M[i][x] = max(M[i-1][x], v[i-1]+M[i-1][x-a[i-1]])
11     return(M[n][L])
```

Question 9 : On peut voir que pour remplir la i-ème ligne du tableau `M` dans la fonction précédente, il suffit de connaître la $(i-1)$-ème ligne. Dans la boucle des lignes 6 à 13, on calcule dans `M` ce qui correspond à la i-ème ligne à partir de la $(i-1)$-ème ligne qui est stockée dans le tableau temporaire `temp`.

```
1  def coutRubanSansRepetitions(a,v,n,L) :
2      temp = allouer(L+1)
3      M = allouer(L+1)
4      for x in range(L+1) :
5          temp[x] = 0
6      for i in xrange(1,n+1) :
7          for x in range(L+1) :
8              if x < a[i-1] :
9                  M[x] = temp[x]
10             else :
11                 M[x] = max(temp[x], v[i-1]+temp[x-a[i-1]])
12         for x in range(L+1) :
13             temp[x] = M[x]
14     return(M[L])
```