Evaluation de NSI

Programmation dynamique (Corrigé)

Exercice : récursivité et programmation dynamique

Question 1 : Donner la liste de toutes les possibilités pour l'entreprise lors de la vente d'une **barre de 4 mètres**. Quel est le **revenu optimal** ?

1 barre de 4 mètres : 9 euros

1 barre de 3 mètres et une de 1 mètre : 8 + 1 = 9 euros

*2 barres de 2 mètres : 5 * 2 = **10 euros** (solution la plus rentable)*

*1 barre de 2 mètres et deux barres de 1 mètre : 1*2 + 5 = 7 euros*

*4 barres de 1 mètre : 4*1 = 4 euros*

Question 2 : En s'aidant des informations ci-dessus, **compléter** (sur la copie) la fonction *revenu_barre(n)*.

```
def revenu_barre(n):
    # Les indices de prix représentent la longueur
    # de la barre.
    prix = [0,1,5,8,9,10,17,17,20,24,30]

    # Cas d'arrêt
    if n == 0:
        return 0

    # Cas général
    else :
        revenu = 0
        for i in range(1, n+1) :
            revenu = max(revenu, prix[i] + revenu_barre(n-i))

        return revenu
```

Question 3 : L'entreprise a modifié ses tarifs :

Longueur (m)	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Prix (euros)	1	6	9	11	12	19	20	23	24	26

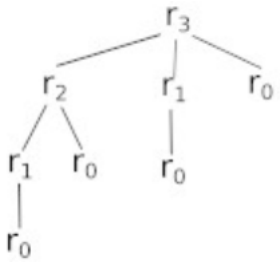
Que doit-on **changer** dans le programme ci-dessus pour qu'il corresponde aux nouveaux tarifs ?

On modifie simplement la liste prix avec les nouveaux tarifs :

Prix = [1,6,9,11,12,19,20,23,24,26]

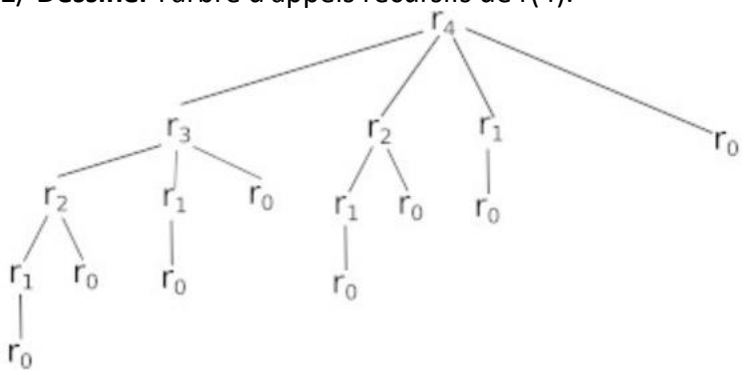
Question 4 :

Voici l'arbre d'appel récursif du calcul de r(3) :



1/ **Combien de fois** est calculé r1 ? Quel type de problèmes cela pourra-t-il poser ?
Deux calculs de r1 : de nombreuses valeurs sont recalculées (complexité exponentielle) ce qui limite la calculabilité d'un tel algorithme.

2/ **Dessiner** l'arbre d'appels récursifs de r(4).



Question 5 : **Proposer** alors une version de la fonction précédente (on gardera la programmation dynamique descendante) pour corriger les problèmes précédents : on recopiera et complètera le programme sur la copie.

```
def revenu_barre(n):
    # Les indices de prix représentent la longueur
    # de la barre.
    prix = [0,1,5,8,9,10,17,17,20,24,30]
    memo = {}

    # Cas d'arrêt
    if n == 0:
        return 0

    # Cas général
    else :
        if n not in memo :
            memo[n] = 0
            for i in range(1, n+1) :
                memo[n] = max(memo[n], prix[i] + revenu_barre(n-i))
        return memo[n]
```

Une approche ascendante …

Question 6 : **Compléter** le programme suivant, version non récursive de la précédente :

```
def revenu_barre(n):
    # Les indices de prix représentent la longueur
    # de la barre.
    prix = [0,1,5,8,9,10,17,17,20,24,30]
    revenu = [0]*11

    for i in range(1, n+1) :
        for j in range(i+1) :
            revenu[i] = max(revenu[i], prix[j] + revenu[i-j])

    return revenu[n]
```