DEVOIR D'ALGORITHMIQUE

1- Proposer un algorithme de DPR qui implémente l'équation :

mij = $\begin{cases} 0 & si\ i = j \\ \min(mik + mkj + di - 1 * di * dj\ , & i < k < j \end{cases}$

Dans le cadre de la multiplication de chaînes de matrices.

Algorithme : <u>chaine _mat_recursif</u> (d, i, j) :

SI i = j alors

Retourner 0 ;

mij ← +∞

Pour k=1 à k=j-1 faire

q ← chaine _mat_recursif (d, i, k) + chaine _mat_recursif (d, k+1, j) +di-1*di*dj

Si q < mij alors

mij ←q

fsi

Fin pour

Retourner mij

Déterminons la complexité de cet algorithme.

Nous avons la relation suivante :

T(n) = $\begin{cases} 1 & \text{si n} = 1 \\ 1\ +\ \sum_{k=1}^{n-1}\ (\text{T(k)}\ +\text{T(n} - \text{k)}\ + 1) & \text{pour n} > 1 \end{cases}$

Finalement on a T(n) = 2 $\sum_{i=1}^{n-1} T(i) + n$;

Par conséquent T(n) ≥ 2T(n-1) et T(n) = O (2^n).

2 – Ecrivons l'algorithme glouton de rendu monnaie, évaluons sa complexité et trouvons un contre-exemple qui montre qu'il n'est pas optimal.

```
Algorithme : rendu_monnaie( x : entier , p : tableau)
   Début :
   r : tableau de meme dimension que p
    Trier le tableau p de pièces dans l'ordre décroissant ;
    Pour i = 1 à n faire :
     si  x >p[i] alors
       r[i] ← x div p[i]
        x ← x - r[i]* p[i]
     sinon
       r[i] ←0
     finsi
   finpour
 Retourner r
fin
```

S'agissant de la complexité, on a : O(nlogn) pour le tri du tableau de pièces. Par la suite on calcule le nombre de pièces nécessaires pour chaque étape en mettant à jour le montant. Ces opérations ont un temps O(n) ; ce qui donne finalement un ordre de grandeur de O(nlogn) au problème de rendu monnaie.

Contre-exemple : supposons un système où nous avons n=3 pièces de valeurs respectives 1f, 4f, 6f et que nous avons un montant de 8f à rembourser ;

Pour i = 1, on a 8>6 donc r (1) = 8 /6 = 1 et x = 8 – 1*6 =2

Pour i =2, 2<4, r [2] = 0 ;

Pour i = 3, 2>1 donc r [3] = 2/1 = 2 et x = 2 – 1*2=0

On retourne ici 1 pièce de 6f et 2 pièces de 1f pourtant l'idéal serait de retourner 2 pièces de 4f.