complexité
Examen

Nous nous occupons du problème de multiplication de *n* matrices M_1, M_2, M_n : $M = M_1*M_2*...*M_n$.

La valeur calculée est la même quel que soit l'ordre dans lequel les multiplications sont effectuées, en revanche le nombre d'opérations peut être différent suivant le parenthésage du produit . Exemple :

soient M_1 de 2 lignes, 4 colonnes
M_2 de 4 lignes, 100 colonnes
M_3 de 100 lignes, 10 colonnes

Le calcul du produit peut se faire de deux façon différentes :

$(M_1 * M_2)*M_3$ ou $M_1*(M_2 * M_3)$

Si on utilise l'algorithme de multiplication classique, nous faisons 2*4*100=800 multiplications pour $(M_1 * M_2)$ plus 2*100*10= 2000 pour multiplier le résultat par M_3.

Pour l'autre calcul, nous faisons 4*100*10 = 4000 multiplications pour $(M_2 * M_3)$, puis 2*4*10=80 pour multiplier le résultat par M_1.

Nous constatons que le nombre d'opérations varie (2800 ou 4080) suivant le parenthésage adopté pour faire le calcul.

Avant de se lancer dans un tel calcul, il est bon de connaître le meilleur parenthésage, et pour connaître le meilleur parenthésage, on peut tous les énumérer et rechercher le minimum parmi ceux-ci.

Question 1.

Démontrer que le nombre de parenthésages d'un produit de *n* (>0) matrices satisfait la relation de récurrence suivante :

$$P_n = T_{n-1}$$

et

$$\begin{cases} T_0 = 1 \\ T_n = \sum_{k=1}^{n} T_{k-1} \times T_{n-k} \quad \text{pour } n > 0 \end{cases}$$

Le nombre T_n est appelé nombre de Catalan.

Question 2.

Calculer la fonction génératrice $T(x) = \sum_{n\geq 0} T_n \times x^n$. Pour celà on calculera $x \times T(x)$, puis le produit $x \times T(x) \times T(x)$

Question 3.

Le terme général de la suite associée à la fonction génératrice précédente est : $\frac{1}{n+1}\binom{2\times n}{n}$, démontrer que ce terme général satisfait la relation de récurrence suivante :

$$\begin{cases} T_0 = 1 \\ T_n = \frac{2\times(2\times n-1)}{n+1}\times T_{n-1} \text{ pour } n>0 \end{cases}$$

Question 4.

Nous programmons le calcul du nombre de Catalan des 3 façons suivantes :

```
int c( int n, int k){
   if(n==k) return 1;
   else if( k==0) return 1;
        else return c( n-1, k) + c(n-1, k-1);
}

int Catalan1( int n){
   return c(2*n, n)/(n+1);
}
```

```
int Catalan2( int n){
   if(n==1) return 1;
   else return Catalan2(n-1)*2*(2*n-1)/(n+1);
}
```

```
int T( int n){
  if( n==1) return 1;
  else{
       int s=0;
       for(int k = 1; i <= n; ++i)
          s = s + T(k-1)*T(n-K);
       return s;
  }
}

int Catalan3( int n){
  return T(n+1)
}
```

Evaluer le nombre d'appels de fonctions générés par les 3 calculs différents du nombre de Catalan.