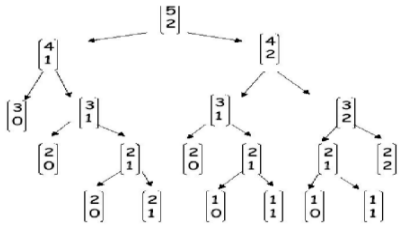
Correction TD 4 : Prog Dynamique

Exercice 1 :

- Solution 1

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

➔Complexité de la solution 1: exponontielle $O(2^n)$



	0	1	2	3	⋯	$n-1$	n
0	1						
1	1	1					
2	1	2	1				
3	1	3	3	1			
⋮	⋮	⋮	⋮		⋱		
$n-1$	1	$n-1$	$\binom{n-1}{2}$	$\binom{n-1}{3}$	⋯	1	
n	1	n	$\binom{n}{2}$	$\binom{n}{3}$	⋯	n	1

- Solution 2

Idée: Pour éviter de calculer plusieurs fois un nombre, on calcule tous les nombres de petites tailles, ensuite, de tailles de plus en plus grandes avant d'arriver au nombre désiré.

➔ Complexité de la solution 2: O(N*k) ➔$O(N^2)$

En Java

```
public int Combinaison(int n, int k) {
    int[][] B = new int[n + 1][k + 1];
    for (int i = 0; i <= n; i++) {
        for (int j = 0; j <= Math.min(i, k); j++) {
            if (i == 0 || j == i)
                B[i][j] = 1;
            else
                B[i][j] = B[i - 1][j - 1] + B[i - 1][j];
        }
    }
    return B[n][k];
}
```

En C

```
int min (int a, int b) {
    return (a < b) ? a : b;
}
int Combinaison(int n, int k) {
    int B[n + 1][k + 1];
    for (int i = 0; i <= n; i++) {
        for (int j = 0; j <= min(i, k); j++) {
            if (i == 0 || j == i)
                B[i][j] = 1;
            else
                B[i][j] = B[i - 1][j - 1] + B[i - 1][j];
        }
    }
    return B[n][k];
}
```

- Solution 3

➔ Complexité de la solution 3: O(N*k) ➔$O(N^2)$

En Java	En C

public int Combinaison(int n, int k) {
int[] t = new int[n + 1];
t[0] = 1;
for (int i = 1; i <= n; i++) {
t[i] = 1;
for (int j = i - 1; j >= 1; j--)
t[j] = t[j] + t[j - 1];
}
return t[k];
}
	int Combinaison(int n, int k) {
	int t[n + 1];
	t[0] = 1;
	for (int i = 1; i <= n; i++) {
	t[i] = 1;
	for (int j = i - 1; j >= 1; j--)
	t[j] = t[j] + t[j - 1];
	}
	return t[k];
	}

Exercice 2 :

Solution itérative simple O(n)

Solution récursive simple O(n)

Solution par Tabulation en Java O(n)

```
public int puissanceDynamique(int a, int n) {
    int[] T = new int [n + 1];
    T[0] = 1;
    for (int i = 1; i <= n; i++) {
        T[i]= T[i-1]*a ;
    }

    return T[n];
}
// sol non compatible avec la description
```

Solution par Memoisation en Java O(log(n))

```
public int puissanceDynamique(int a, int n) {
    int[] T = new int[n + 1];
    if (n == 0)
        return 1;
    if (n == 1)
        return a;
    T[n/2] = puissanceDynamique(a, n/2);
    if (n % 2 == 0)
        T[n] = T[n/ 2] * T[n / 2];
    else
        T[n] = T[n / 2] * T[n/ 2] * a;
    // sans if else
// T[n] = T[n/2] * T[n/2] * (n%2 == 1 ? a : 1);
    return T[n];

}
```

```
// sol compatible avec la description
public int puissanceDynamique(int a, int n) {
    int[] T = new int[n + 1];
    T[0] = 1;
    for (int i = 1; i <= n; i++) {
        if (i % 2 == 0)
            T[i] = T[i / 2] * T[i / 2];
        else
            T[i] = T[i / 2] * T[i / 2] * a;
// sans if else
// T[i] = T[i / 2] * T[i / 2] * (n%2 == 1 ? a : 1);
    }

    return T[n];
}
```

Exercice 3 :

1. le produit de matrices utilisant cet ordre ((AB)C)D nécessite 30800 produits scalaires, c'est-à-dire,

AB : 20 × 5 × 100 = 10000
((AB)C) : 20 × 100 × 8 = 16000
((AB)C)D : 20 × 8 × 30 = 4800

2. le produit de matrices utilisant cet ordre (A(BC))D nécessite 9600 produits scalaires, c'est-à-dire,

BC : 5 × 100 × 8 = 4000
((A(BC)) : 20 × 5 × 8 = 800
((AB)C)D : 20 × 8 × 30 = 4800

Question 5.2 Ecrire une formule de récurrence pour calculer $c(i,j)$.

Correction

Supposons que la meilleure façon de parenthéser $M_i \dots M_j$ soit $(M_i..M_k)(M_{k+1} \dots M_j)$. La matrice $M_i \dots M_k$ est une matrice $d_{i-1} \times d_k$ et $M_{k+1} \dots M_j$ est une matrice $d_k \times d_j$. Le produit de ces deux matrices necessite $d_{i-1}d_kd_j$ produits scalaires. Au total, le nombre total de produits scalaires pour calculer $M_iM_{i+1} \dots M_{j-1}M_j$ est $c(i,k) + c(k+1,j) + d_{i-1}d_kd_j$.

On obtient

$$c(i,j) = \begin{cases} \min_{i \le k < j}(c(i,k) + c(k+1,j) + d_{i-1}d_kd_j) & \text{si } i < j. \\ 0 & \text{si } i = j. \end{cases}$$

□

Question 5.3 Ecrire un algorithme utilisant la programmation dynamique

Correction

Donnée : une suite de matrices $M_1 \dots M_n$ avec la matrice M_i de dimention $d_{i-1} \times d_i$.

1. initialiser tous les éléments de la matrice à $\infty+$
2. pour i à 1 allant n faire $c(i,i) = 0$;
3. pour ℓ à 1 allant n faire
   (a) pour i à 1 allant $n - \ell$ faire
      i. pour k à 1 allant ℓ faire
         $c(i, i+\ell) = \min(c(i, i+\ell), c(i, i+k) + c(i+k+1, i+\ell) + d_{i-1}d_{i+k}d_{i+\ell})$;

□