ASD2 & Complexité

La Complexité

EXERCICE 1 :

Soit la suite U_n définit par :

$$\begin{cases} U_n = U_{n-1} \times U_{n-2} + U_{n-3} \\ U_0 = U_1 = U_2 = 1 \end{cases}$$

Question :

- Donner un algorithme récursif qui calcule U_n
- Évaluer sa complexité.

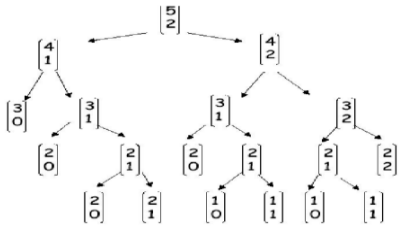
EXERCICE 2 : TRIANGLE DE PASCAL

On veut calculer les coefficients binomiaux $C_n^k = \binom{n}{k} = \frac{n!}{k!(n-k)!}$. Rappelons les propriétés suivantes :

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k} \; pour \; 0 < k < n$$
$$\binom{n}{n} = 1 \; et \; \binom{n}{0} = 1$$

Question :

- Donner un algorithme récursif qui calcul $\binom{n}{k}$
- Évaluer sa complexité.



	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

EXERCICE 3 :

Les nombres de Fibonacci sont définis par la récurrence :

- $F_0 = 1$
- $F_1 = 1$
- $F_n = F_{n-1} + F_{n-2}$ pour $n \geq 2$

1. Écrire une fonction récursive permettant de calculer le n-ième terme de cette suite
2. Évaluer sa complexité.
3. Écrire une fonction itérative permettant de calculer le n-ième terme de cette suite
4. Évaluer sa complexité.
5. la fonction récursive suivante permet de calculer le n-ième terme de cette suite
6. Évaluer sa complexité.

```
int Fib_rec(int N, int j, int i) {
        if (N <2)
                return j;

        return Fib_rec (N - 1, j + i, j);
}
```

EXERCICE 4 :

1. Évaluer la complexité de chacune des fonctions suivantes.

```
int F1 (int N) {
   int count = 0;
   for (int i = 1; i * i <= N; i++) {
      count++;
   }
   return count;
}
```

```
int F2 (int N) {
   int count = 0;
   for (int i = N/2; i <= N; i++) {
      for (int j = 1; j <= N; j *= 2) {
         for (int k = 1; k <= N; k *= 2) {
            count++;
         }
      }
   }
   return count;
}
```

```
void F3 (int T[], int n) {
   for (int i = 1; i < n; i++) {
      int a = T[i];
      int j = i - 1;
      while (j >= 0 && a < T[j]) {
         T[j + 1] = T[j];
         j--;
      }
      T[j + 1] = a;
   }
}
```

```
void F4 (int T[], int n) {
   for (int i = 0; i < n - 1; i++) {
      int minj = i;
      int mina = T[i];
      for (int j = i + 1; j < n; j++) {
         if (T[j] < mina) {
            minj = j;
            mina = T[j];
         }
      }
```

EXERCICE 5:

Évaluer la complexité de chacune des fonctions suivantes. ? Commencez par écrire la relation de récurrence puis trouvez sa solution.

```
void FRec1 (int N) {
   if (N <= 1) {
      return;
   }
   for (int i = 1; i <= N; i++) {
      for (int j = 1; j <= N; j++) {
         printf("*");
      }
      printf("\n");
   }
   FRec1 (N - 3);
}
```

```
void FRec2 (int N) {
   if (N <= 1) {
      return;
   }
   for (int i = 1; i <= 3; i++) {
      FRec2 (N - 1);
   }
}
```

```
int FRec3 (int T[], int i, int j, int a) {
 int m = (i + (j - i) / 2);
 if (j < i)
      return -1;
 if (T[m] < a)
      return FRec3 (T, m + 1, j, a);
 else if (T[m] > a)
      return FRec3 (T, i, m - 1, a);
 else
      return m;
}
```

```
void f1 (int T [ ], int g, int m, int d); //O(1)
void FRec4 (int T [ ], int g, int d){
          int m;
          if (g<d){
                    m=(g+d)/2;
                    FRec4 (T, g, m) ;
                    FRec4 (T, m+1, d) ;
                   for (int i = g; i <= d; i++)
                         f1(T, g, m, d) ;
          }
}
```