

**TD2 (SUITE)****EXERCICE 1 :**

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. la fonction récursive suivante permet de calculer le n-ième terme de cette suite
2. 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 2 :**

1. Évaluer la complexité de chacune des fonctions suivantes.

|  |   |
|--|---|
| <pre>int F1 (int N) {     int count = 0;     for (int i = 1; i * i &lt;= N; i++) {         count++;     }     return count; }</pre>  | <pre>int F2 (int N) {     int count = 0;     for (int i = N/2; i &lt;= N; i++) {         for (int j = 1; j &lt;= N; j *= 2) {             for (int k = 1; k &lt;= N; k *= 2) {                 count++;             }         }     }     return count; }</pre>                                     |
| <pre>void F3 (int T[], int n) {     for (int i = 1; i &lt; n; i++) {         int a = T[i];         int j = i - 1;         while (j &gt;= 0 &amp;&amp; a &lt; T[j]) {             T[j + 1] = T[j];             j--;         }         T[j + 1] = a;     } }</pre> | <pre>void F4 (int T[], int n) {     for (int i = 0; i &lt; n - 1; i++) {         int minj = i;         int mina = T[i];         for (int j = i + 1; j &lt; n; j++) {             if (T[j] &lt; mina) {                 minj = j;                 mina = T[j];             }         }     } }</pre> |

**EXERCICE 3:**

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

|   |   |
|---|---|
| <pre>void FRec1 (int N) {     if (N &lt;= 1) {         return;     }     for (int i = 1; i &lt;= N; i++) {         for (int j = 1; j &lt;= N; j++) {             printf("*");         }         printf("\n");     }     FRec1 (N - 3); }</pre>                                      | <pre>void FRec2 (int N) {     if (N &lt;= 1) {         return;     }     for (int i = 1; i &lt;= 3; i++) {         FRec2 (N - 1);     } }</pre>   |
| <pre>int FRec3 (int T[], int i, int j, int a) {     int m = (i + (j - i) / 2);     if (j &lt; i)         return -1;     if (T[m] &lt; a)         return FRec3 (T, m + 1, j, a);     else if (T[m] &gt; a)         return FRec3 (T, i, m - 1, a);     else         return m; }</pre> | <pre>void f1 (int T [ ], int g, int m, int d); //O(1) void FRec4 (int T [ ], int g, int d){     int m;     if (g&lt;d){         m=(g+d)/2;         FRec4 (T, g, m) ;         FRec4 (T, m+1, d) ;         for (int i = g; i &lt;= d; i++)             f1(T, g, m, d) ;     } }</pre> |

**EXERCICE 4:**

La fonction suivante permet de multiplier deux grands entiers x et y (chacun nécessitant n bits). Évaluez sa complexité en fonction de n.

```
int Multiply (int x, int y) {
    if (y == 0) {
        return 0;
    }
    int z = Multiply (x, y / 2);
    if (y % 2 == 0) {
        return 2 * z;
    } else {
        return x + 2 * z;
    }
}
```