TD3

Exercice 1 : Suite de Fibonacci

Soit une suite de Fibonacci.

Fib(0) = 0
Fib(1) = 1
Fib(n) = Fib(n-1) + Fib(n-2)

1°) Calculer Fib(5) :

Fib(2) = 1
Fib(3) = 2
Fib(4) = 3

Fib(5) = Fib(4) + Fib(3)
Fib(4) = Fib(3) + Fib(2)
Fib(3) = Fib(2) + Fib(1)
Fib(2) = Fib(1) + Fib(0)
Fib(1) = 1
Fib(0) = 0

Fib(0) = 0
Fib(1) = 1
Fib(2) = 1 + 0 = 1
Fib(3) = 1 + 1 = 2
Fib(4) = 2 + 1 = 3
Fib(5) = 3 + 2 = 5

2°) Ecrire une fonction sans appel récursif en utilisant un vecteur F[0…n] pour stocker les éléments de calculs au fur et à mesure. Imprimer F

Algorithme de Fibonacci

```
Entier : n, i, F[100] ;

Début :
        Afficher  (« Entrez la taille du tableau : ») ;
        Lire(n) ;

        F[0] <= 0 ;
        Afficher(F[0]);
        F[1] <= 1 ;
        Afficher(F[1]);

        Pour i allant de 2 à n Faire :
                F[i] <= F[i-1] + F[i-2] ;
        Afficher(F[i]);
        Fin Pour
Fin
```

3°) Calculer la complexité de cet algorithme portant sue les opérations : addition et l'affectation.

Complexité sur l'affectation :
- **2+(n-1) = n+1**

Complexité sur l'addition :
- **n-1**

4°) Ecrire une deuxième fonction pour le calcul de la suite en utilisant 3 variables, et imprimer la suite.

```
Algorithme de Fibonacci
Entier : n, nb1, nb2, nb3, i ;

Début :
        n <= 0 ;
        Afficher (« Entrez une valeur : ») ;
        Lire(n) ;

        nb1 <= 0 ;
        Afficher (nb1) ;
        nb2 <= 1;
        Afficher (nb2) ;

        Pour i allant de 2 à n Faire:
                nb3 <= nb1 + nb2 ;
                nb1 <= nb2 ;
                nb2 <= nb3 ;
                Afficher (nb3) ;
        Fin Pour
Fin
```

Complexité sur l'affectation :
- **3*(n-1) + 2 = 3n+1**

Complexité sur l'addition :
- **n-1**

Complexité sur l'écriture :
- **(n-1) + 2 = n+1**

Cet algorithme est en taux de 1.

Exercice 2 : Suite de Fibonacci

Apres avoir calculer le temps de calcul de Fib(5), généraliser le calcul de la complexité pour Fib(n).

Sachant T(Fib(0)) = T(Fib(1)) = 0

T(Fib(5))= T(Fib(4)) + T(Fib(3)) + 1
T(Fib(4))= T(Fib(3)) + T(Fib(2)) + 1
T(Fib(3))= T(Fib(2)) + T(Fib(1)) + 1
T(Fib(2))= T(Fib(1)) + T(Fib(0)) + 1
T(Fib(1)) = 0
T(Fib(0)) = 0

T(Fib(2)) = 0 + 0 + 1 = 1
T(Fib(3)) = 1 + 0 + 1 = 2
T(Fib(4)) = 2 + 1 + 1 = 4
T(Fib(5)) = 4 + 2 + 1 = 7

Trouver un encadrement qui déterminera les bornes inferieures et supérieures de la fonction temps de calcul.

Exercice 3 : Fusion de tableaux d'entiers

Soient T1, et T2 deux tableaux déjà triés, trier ces 2 tableaux dans un ordre croissant et stocker ces valeurs dans T3.

Algorithme trie et fusion:

```
Variables :

Définir : MAX = 10
Entier : tab[MAX], tab2[MAX], tabFinal[MAX], i, j, tmp, sizeTab1, sizeTab2, sizeTab3

Début :
        Tant que sizeTab1<0 ou sizeTab 1> MAX Faire :
            Afficher(''Taille du second tableau %d'' MAX:) ;
            Lire(''%d'' &sizeTab1);
        Fin Tant que


        Tant que sizeTab2<0 ou sizeTab 2> MAX Faire :
            Afficher(''Taille du second tableau %d'' MAX:) ;
            Lire(''%d'' &sizeTab2);
        Fin Tant que


        Afficher(''Saisie du premier tableau : ) ;
        Pour i allant de 0 à sizeTab1 Faire :
            Afficher(''tab[%d'', i]) ;
            Lire(''%d'', &tab[i]) ;
        Fin Pour


        Afficher(''Saisie du second tableau : ) ;
        Pour i allant de 0 à sizeTab2 Faire :
            Afficher(''tab2[%d'', i]) ;
            Lire(''%d'', &tab[2i]) ;
        Fin Pour


        Pour i allant de 0 à sizeTab1 Faire :
            tabFinal[i]← tab[i] ;
        Fin Pour

        sizeTab3←sizeTab1+sizeTab2 ;

        Pour j allant de sizeTab1 à sizeTab3 && i allant de 0 à sizeTab2 Faire ;
            tabFinal[j]←tab2[i]
        Fin Pour

        Pour j allant de 1 à sizeTab3 Faire :
            Pour i allant de 0 à sizeTab3 Faire :
                Si tabFinal[i] > tabFinal[i+1] Alors :
                    tmp←tabFinal[i] ;
                    tabFinal[i]=tabFinal[i+1] ;
```

```
                tabFinal[i+1]=tmp ;
            Fin Si
        Fin Pour
    Fin Pour

    Afficher(''Tableau trié et fusionné : '') ;
    Pour i allant de 0 à siezTab3 Faire ;
        Afficher(''[%d]'', tabFinal[i]) ;
    Fin Pour
    Fin
```

Exercice 4 : Recherche d'une majorité dans un tableau trié et non trié.

1	2	2	2	5	5	5	5	5	6	6	6	10

5	10	6	2	5	6	5	10	5	2	6	6	10

Construire un algorithme qui permet de trouver l'élément le plus présent dans ce tableau, et dire s'il constitue la majorité.

Algorithme majorité :

```
Variables :

Définir :  MAX = 10
Entier : i, j, nb, max = 0, pos = -1, size, tab[MAX];

Début :
        Tant que size < 0 OU size > MAX Faire :
                Afficher(« Entrez une taille <%d », MAX) ;
                Lire(« %d », &size) ;
        Fin tant que

        Pour i allant de 0 à size Faire :
                Afficher(« tab[i] :  », i) ;
                Lire(« %d », & tab[i]) ;
        Fin pour

        Pour i allant de 0 à size Faire :
                nb <= 0;
                Pour j allant de 0 à size Faire :
                        Si tab[i] = tab[j] Alors :
                                nb <= nb + 1 ;
                        Fin si
                Fin pour

                Si nb>max Alors :
                        max <= nb ;
                        pos <= i ;
                Fin si
        Fin pour

        Si pos = -1 Alors :
                Afficher(« Pas de majorité ») ;
```

```
        Sinon :
                Afficher(« Majorité : %d », tab[pos]) ;
        Fin si

Fin
```