|  |  |
| --- | --- |
|  | **COMPLEXITÉ** |
|  | **Master 1 IL**  **Groupe 2**  **2018** |

|  |
| --- |
| **[ Algorithmique avancÉe et complexitÉ ]** |
| E-mail : geronimotoutcourt@gmail.com |

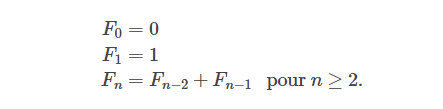
Rapport de TP N°5 COMPLEXITÉ : Algorithmes itératifs vs récursifs

**BOUDOUR Mehdi / 201500008386/ TP: Algorithmes itératifs vs récursifs**



*Ce document présent les solutions en 5 étapes : (1) les algorithmes écris en pseudo-code. (2) le calcul de la complexité au pire des cas. (3) Implémentation de l’algorithme en langage C. (4) capture de l’exécution de l’algorithme. (5) représentation graphique de l’évolution du temps d’exécution en fonction de N. Le programme C complet contenant les détails (affichage, calcul du temps d’exécution,…) d’implémentation est présenté à la fin du document.*

# I. Algorithme *Suite de Fibonacci*:



# Ecrire une fonction Fibo\_Rec qui calcul récursivement le nième terme de la suite de Fibonacci.

# Algorithme : *Réccursif*

**FONCTION FIBO\_REC(E/ N:ENTIER) :ENTIER**

**DEBUT**

**SI F(N=0)ALORS**

**RETOURNER 0;**

**SINON**

**SI (N=1) ALORS**

**RETOURNER 1;**

**SINON**

**SI(N>=2) ALORS**

**RETOURNER FIBO\_REC(N-1)+FIBO\_REC(N-2);**

**FIN SI;**

**FIN SI;**

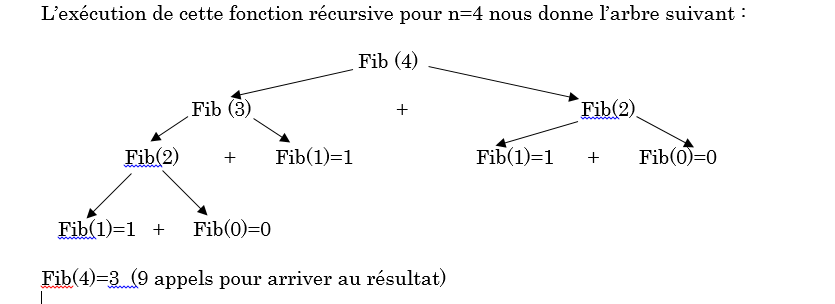
**FIN SI;**

**FIN;**

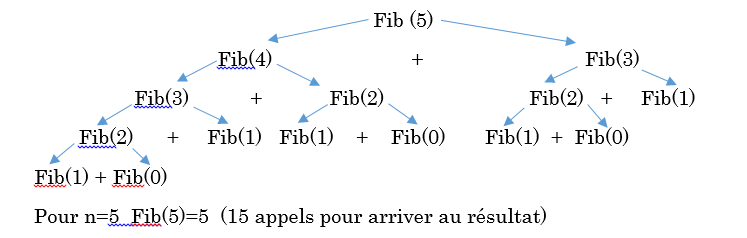
# Complexité :

**Arbre récursif :**

F(4)= ?



F(5) = ?



T(N) ∼ O() Exponentielle

# Implémentation : En langage C

**long Fibo\_Rec(long n)**

**{**

**if(n==0) return 0;**

**else**

**if(n==1) return 1;**

**else**

**if(n>=2)**

**return Fibo\_Rec(n-1)+Fibo\_Rec(n-2);**

**}**

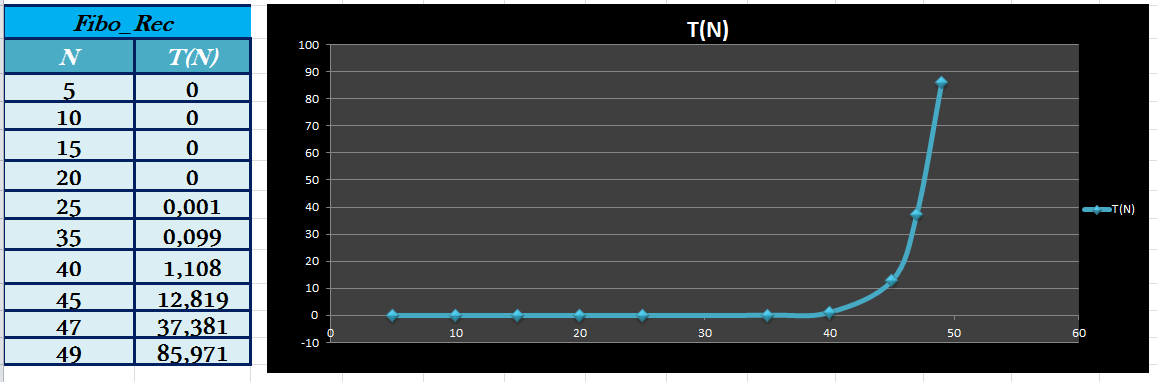
# Exécution :

# *Affichage du temps d’exécution de l’algorithme pour chaque valeur de N (T = le temps d’exécution calculé pour chaque exécution de la fonction Fibo\_Rec ).*

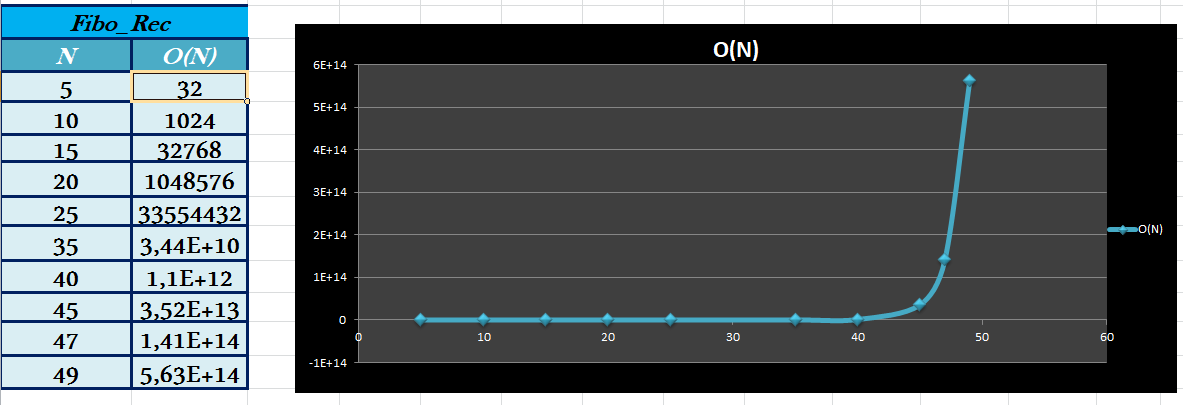
# C:\Users\Moflawer\Desktop\Dol_Gul_Dur\WorkShop_Tree\C\Almost_Done\TP_COMPLEXITÉ\TP5\Screen Shot\exe2.1.PNG

# Représentation Graphique :

# *Graphe du temps d’exécution en fonction de N.*



# *Graphe de la complexité théorique en fonction de N.*



# Algorithme : *Itératif*

**FONCTION FIBO\_ITER(E/ N:ENTIER):ENTIER**

**F1,F2,F:ENTIER;**

**DEBUT**

**SI (N=0) ALORS**

**RETOURNER 0;**

**SINON**

1

**SI(N=1) ALORS**

**RETOURNER 1;**

**FIN SI;**

**FIN SI;**

2

**F1=0,F2=1;**

**TANT QUE (N>=2) FAIRE**

**F = F1 + F2;**

**F1 = F2;**

**F2 = F;**

**N=N-1;**

**FIN TANT QUE;**

1

**RETOURNER F;**

**FIN;**

# Complexité :

T(N)= +1+2+1 = 4(N-2 + 1) +4 = 4N ∼ O(N) Linéaire

# Implémentation : En langage

**long Fibo\_iter(long n)**

**{**

**if(n==0) return 0;**

**else**

**if(n==1) return 1;**

**long F1=0,F2=1,F;**

**for(; n>=2 ; n--)**

**{**

**F = F1 + F2;**

**F1 = F2;**

**F2 = F;**

**}**

**return F;**

**}**

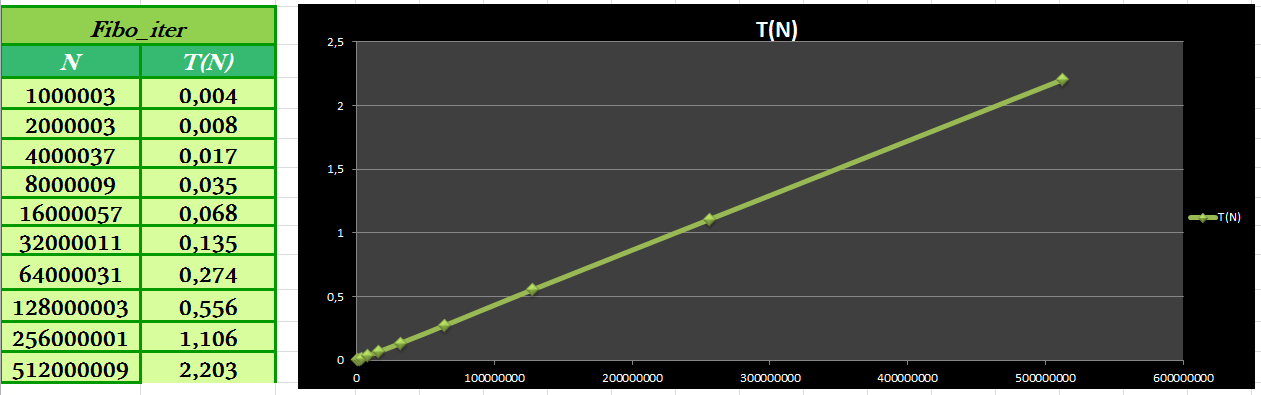
# Exécution :

# *Affichage du temps d’exécution de l’algorithme pour chaque valeur de N (T = le temps d’exécution calculé pour chaque exécution de la fonction Fibo\_iter).*

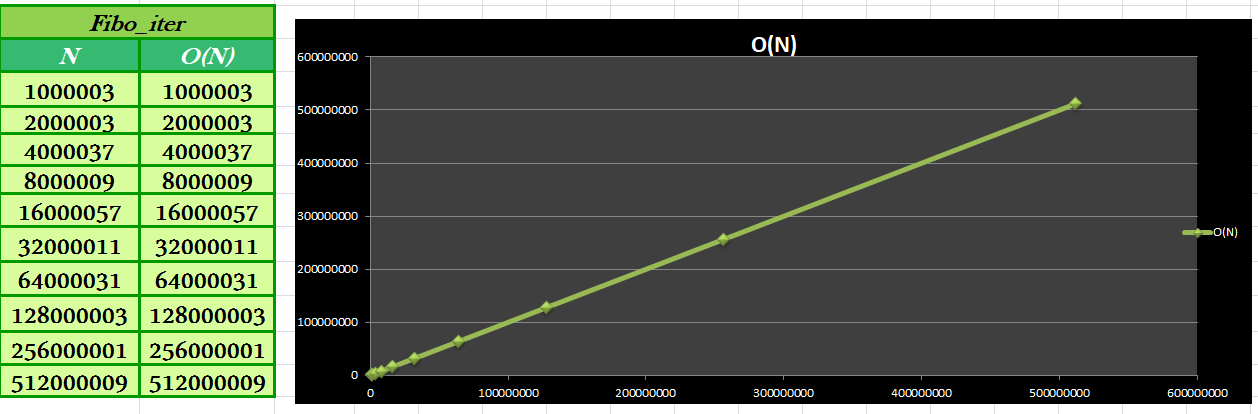
# *C:\Users\Moflawer\Desktop\Dol_Gul_Dur\WorkShop_Tree\C\Almost_Done\TP_COMPLEXITÉ\TP5\Screen Shot\exe2.PNG*

# Représentation Graphique :

# *Graphe du temps d’exécution en fonction de N.*



# *Graphe de la complexité théorique en fonction de N.*



# I. Algorithme *Phi*:

# On peut monter que la suite (𝑈𝑛), 𝑛 ∈ Ν vérifie 𝑈𝑛 ~ 𝜙,

# 𝑜ù 𝜙 est le nombre d’or. Ainsi lim 𝑈𝑛 /Un-1 = 𝜙 quand 𝑛→+∞. Faites une copie de la fonction Fibo\_iter et modifiez-la pour tester cette propriété (utilisez le type double). Qui calcul 𝜙.

# Remarque : 𝜙= 1+√5 ≅ 1,6180339887498948482045868343656

# Algorithme :

**FONCTION PHI(E/ N:ENTIER : REEL**

**F1,F2,F:REEL**

**DEBUT**

**SI (N=0) ALORS**

**RETOURNER 0;**

**SINON**

1

**SI (N=1) ALORS**

**RETOURNER 1;**

**FIN SI;**

**FIN SI;**

2

**F1=0,F2=1;**

**TANT QUE (N>=2) FAIRE**

**F = F1 + F2;**

**F1 = F2;**

**F2 = F;**

**N=N-1;**

**FIN TANT QUE;**

1

**RETOURNER F2/F1;**

**FIN;**

# Complexité :

T(N)= +1+2+1 = 4(N-2 + 1) +4 = 4N ∼ O(N) Linéaire

# Implémentation : En langage C

**double Phi(long n)**

**{**

**if(n==0) return 0;**

**else**

**if(n==1) return 1;**

**double F,F1=0,F2=1;**

**for(; n>=2 ; n--)**

**{**

**F = F1 + F2;**

**F1 = F2;**

**F2 = F;**

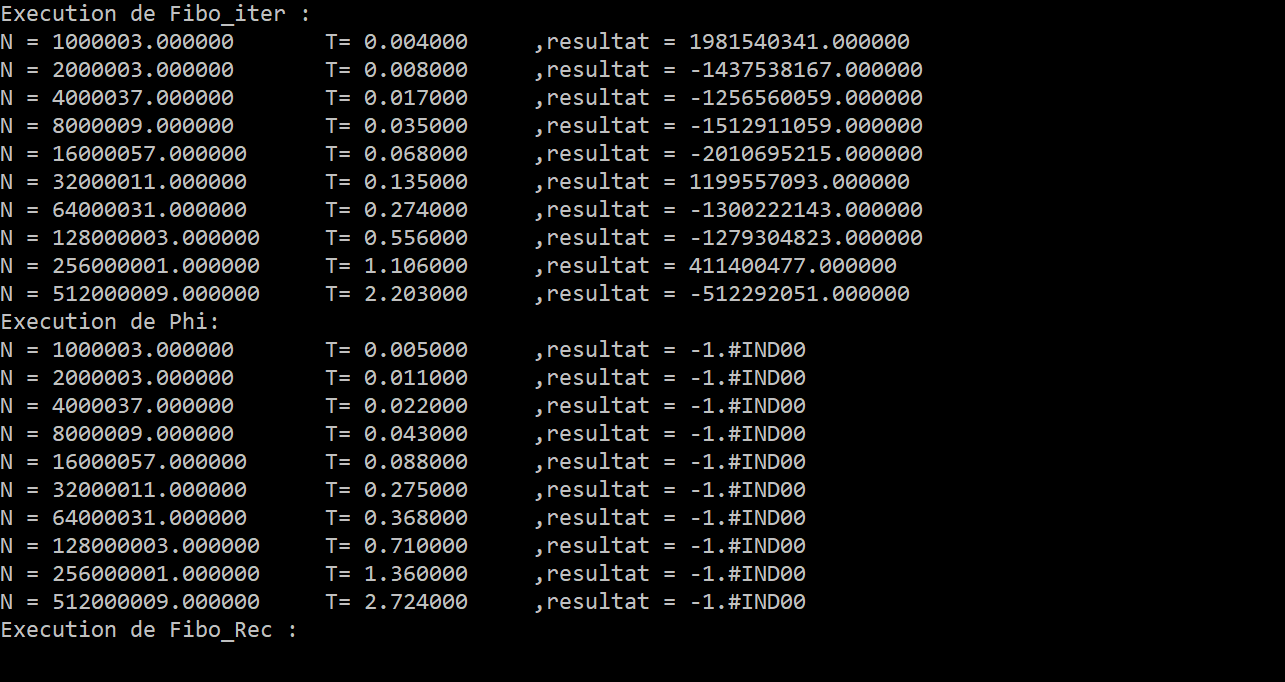
**}**

**return (double) F2/F1;**

**}**

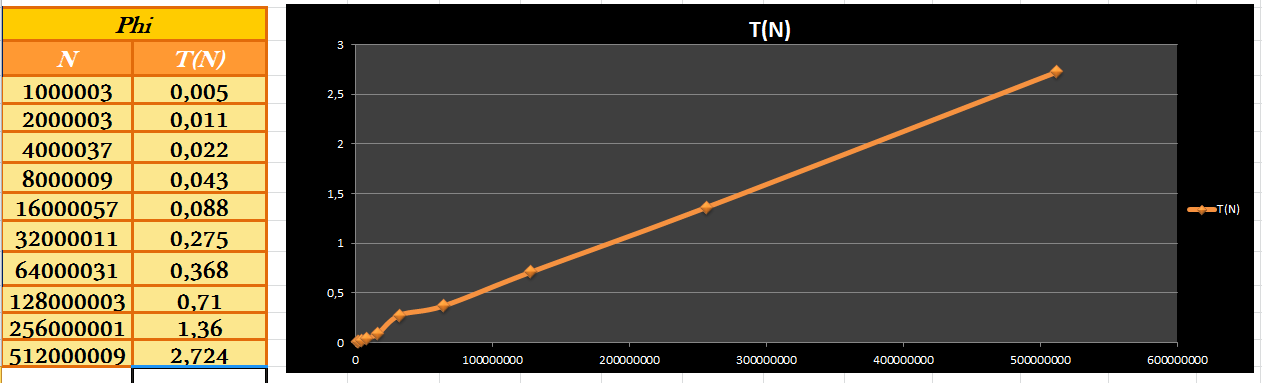
# Exécution :

# *Affichage du temps d’exécution de l’algorithme pour chaque valeur de N (T = le temps d’exécution calculé pour chaque exécution de la fonction Phi).*

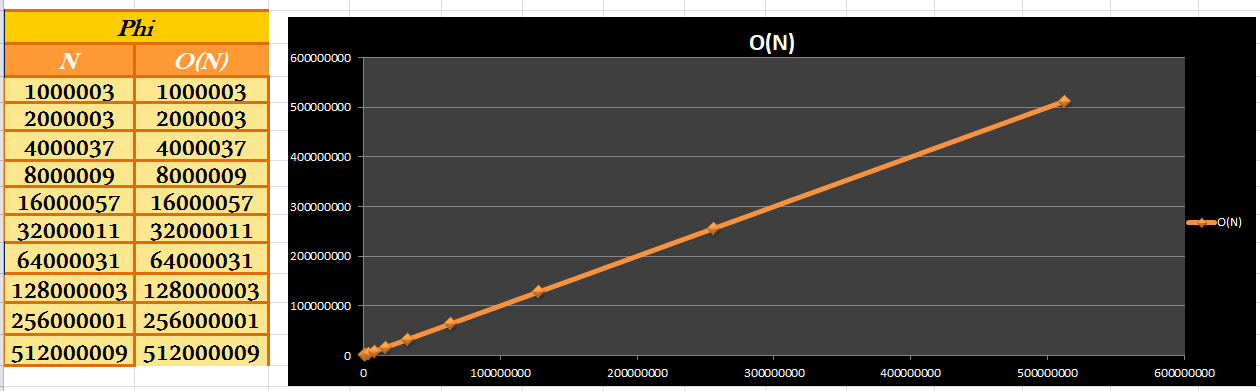


# Représentation Graphique :

# *Graphe du temps d’exécution en fonction de N.*



# *Graphe de la complexité théorique en fonction de N.*



(\*)Code Source du Programme complet :

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <time.h>**

**#include <math.h>**

**long Fibo\_Rec(long n)**

**{**

**if(n==0) return 0;**

**else**

**if(n==1) return 1;**

**else**

**if(n>=2)**

**return Fibo\_Rec(n-1)+Fibo\_Rec(n-2);**

**}**

**long Fibo\_iter(long n)**

**{**

**if(n==0) return 0;**

**else**

**if(n==1) return 1;**

**long F1=0,F2=1,F;**

**for(; n>=2 ; n--)**

**{**

**F = F1 + F2;**

**F1 = F2;**

**F2 = F;**

**}**

**return F;**

**}**

**double Phi(long n)**

**{**

**if(n==0) return 0;**

**else**

**if(n==1) return 1;**

**double F,F1=0,F2=1;**

**for(; n>=2 ; n--)**

**{**

**F = F1 + F2;**

**F1 = F2;**

**F2 = F;**

**}**

**return (double) F2/F1;**

**}**

**double \*\*Calcul\_des\_Temps(double \*\*tab , long algorithme)**

**{**

**long j;**

**double resultat;**

**for(j=0 ; j<11 ; j++)**

**{**

**clock\_t begin = clock();**

**switch(algorithme)**

**{**

**case 1: resultat = Fibo\_Rec(tab[0][j]); break;**

**case 2: resultat = Fibo\_iter(tab[0][j]); break;**

**case 3: resultat = Phi(tab[0][j]); break;**

**}**

**clock\_t end = clock();**

**tab[1][j] = (double)(end - begin) / CLOCKS\_PER\_SEC;**

**tab[2][j] = resultat;**

**}**

**return tab;**

**}**

**double \*\*Tableau\_de\_ValeursFibRec(void)**

**{**

**long i ;**

**double \*\*tab;**

**tab = (double \*\*)malloc(3\*sizeof(double \*));**

**for(i=0 ; i<3 ; i++) tab[i] = (double \*)malloc(11\*sizeof(double));**

**tab[0][0]=5;**

**tab[0][1]=10;**

**tab[0][2]=15;**

**tab[0][3]=20;**

**tab[0][4]=25;**

**tab[0][5]=35;**

**tab[0][6]=40;**

**tab[0][7]=45;**

**tab[0][8]=47;**

**tab[0][9]=49;**

**tab[0][10]=50;**

**for(i=0 ; i<11 ; i++)tab[1][i] = 0 ;**

**return tab;**

**}**

**double \*\*Tableau\_de\_ValeursFibIter(void)**

**{**

**long i ;**

**double \*\*tab;**

**tab = (double \*\*)malloc(3\*sizeof(double \*));**

**for(i=0 ; i<3 ; i++) tab[i] = (double \*)malloc(11\*sizeof(double));**

**tab[0][0]=1000003;**

**tab[0][1]=2000003;**

**tab[0][2]=4000037;**

**tab[0][3]=8000009;**

**tab[0][4]=16000057;**

**tab[0][5]=32000011;**

**tab[0][6]=64000031;**

**tab[0][7]=128000003;**

**tab[0][8]=256000001;**

**tab[0][9]=512000009;**

**tab[0][10]=1024000009;**

**for(i=0 ; i<10 ; i++)tab[1][i] = 0 ;**

**return tab;**

**}**

**void Afficher\_Tableau\_de\_Valeurs(double \*\*tab)**

**{**

**long j;**

**for(j=0 ; j<10 ; j++)**

**{**

**printf("N = %lf \t T= %lf \t ,resultat = %lf \n",tab[0][j],tab[1][j],tab[2][j]);**

**}**

**}**

**int main(int argc, char \*argv[])**

**{**

**printf("Execution de Fibo\_iter :\n");**

**Afficher\_Tableau\_de\_Valeurs(Calcul\_des\_Temps(Tableau\_de\_ValeursFibIter(),2));**

**printf("Execution de Phi:\n");**

**Afficher\_Tableau\_de\_Valeurs(Calcul\_des\_Temps(Tableau\_de\_ValeursFibIter(),3));**

**printf("Execution de Fibo\_Rec :\n");**

**Afficher\_Tableau\_de\_Valeurs(Calcul\_des\_Temps(Tableau\_de\_ValeursFibRec(),1));**

**getchar();**

**return 0;**

**}**