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

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

Rapport de TP N° COMPLEXITÉ :

**BOUDOUR Mehdi / 201500008386/ TP:**



*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 *rechElets\_TabNonTriés* :

# Soit un tableau de n (n>=2) valeurs entières non triées :

# Écrire une fonction rechElets\_TabNonTriés permettant de vérifier l’existence d’une valeur x donnée.

# Algorithme :

**FONCTION RECHELETS\_TABNONTRIES(E/ T ; TABLEAU[N] D' ENTIER ,**

**N,X :ENTIER) : ENTIER**

**I:ENTIER**

**DEBUT**

**POUR I=0 JUSQU'A N FAIRE**

**SI (T[I]=X) ALORS**

**RETOURNER I;**

**FIN SI;**

**RETOURNER -1;**

**FIN;**

**FONCTION RECHELETS\_TABNTRIES(E/ T:TABLEAU[N] D'ENTIER,**

**E/ N,X:ENTIER ):ENTIER**

**I:ENTIER**

**DEBUT**

**POUR I=0 JUSQU'A N FAIRE**

**SI (T[I]=X) ALORS**

**RETOURNER I;**

**SINON**

**SI(T[I]>X) ALORS**

**RETOURNER -1;**

**FIN SI;**

**FIN SI;**

**FIN POUR;**

**RETOURNER -1;**

**FIN;**

**FONCTION RECHELETS\_DICHO(E/ T:TABLEAU[N] D'ENTIER,**

**E/ N,X:ENTIER) : ENTIER**

**D,F,M:ENTIER;**

**DEBUT**

**SI (T[0]=X) ALORS**

**REOUTNER 0;**

**SINON**

**SI(T[N-1]=X) ALORS**

**RETOURNER N-1;**

**SINON**

**SI(X<T[0] OU X>T[N-1]) ALORS**

**RETOURNER -1;**

**FIN SI;**

**FIN SI:**

**FIN SI;**

**D=1; F=N-1;**

**TANT QUE (D<F) FAIRE**

**FM=(D+F)/2**

**SI (T[M]=X) ALORS**

**RETOURNER M;**

**SINON**

**SI(X<T[M]) ALORS**

**F=M;**

**SINON**

**D=M;**

**FIN SI;**

**FIN SI;**

**FIN TANT QUE;**

**RETOURNER -1;**

**FIN;**

**PROCEDURE MINMAXA(E/ T:TABLEAU[N] D' ENTIER,**

**E/ N,MIN,MAX:ENTIER)**

**I:ENTIER;**

**DEBUT**

**MIN=T[0]; MAX=T[0];**

**POUR I=1 JUSQU'A N FAIRE**

**SI (T[I]<MIN) ALORS**

**MIN=T[I];**

**SINON**

**SI(T[I]>MAX) ALORS**

**MAX=T[I];**

**FIN SI;**

**FIN SI;**

**FIN POUR;**

**FIN;**

**FONCTION ALGORITHME1(N:ENTIER) : BOOLÉEN**

**I: ENTIER;**

**DEBUT**

**POUR I = 2 JUSQU'A N-1 FAIRE**

**SI ( N MOD I = 0 ) ALORS**

**RETOURNER (FAUX);**

**FIN SI;**

**FIN POUR;**

1

**RETOURNER (VRAI);**

**FIN;**

# Complexité :

**Au pire des cas :** Le nombre ***N*** est premier ainsi la boucle s’itérera jusqu’à ***i=N-1*** car le test ne trouvera aucun nombre ***i*** ∈ { , 𝑥∈ ℕ / ***2*** ≤ 𝑥 ≤ ***N-1***} qui soit diviseur de ***N***.

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

# Implémentation : En langage C

**int Algorithme1(int N)**

**{**

**int i;**

**for(i=2 ; i<= N-1 ; i++)**

**{**

**if(N%i == 0) return 0;**

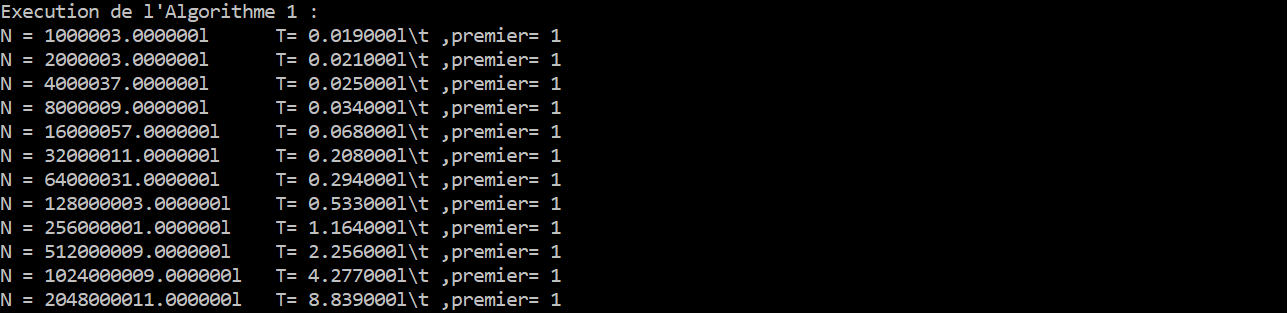
**}**

**return 1;**

**}**

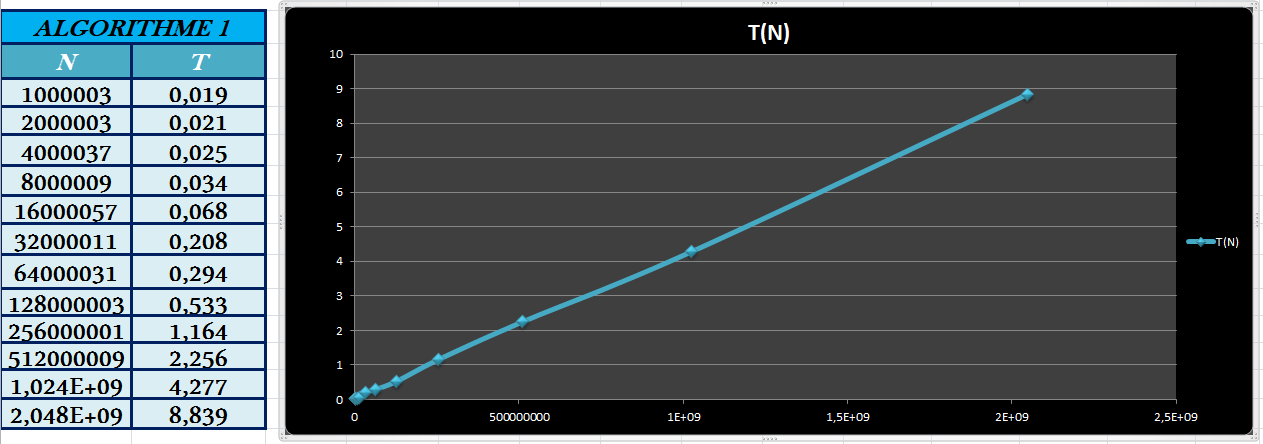
# Exécution :

# *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 Algorithme1 premier = Retour de Algorithme1).*

****

# Représentation Graphique :

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

****