Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université des Sciences et de la Technologie Houari Boumediene
Faculté d'électronique et d'informatique
Département d'informatique

1ère Année Master Informatique, Semestre 1

Option : Systèmes informatiques Intelligents (SII)

Module : Conception et Complexité des Algorithme

Rapport de TP N°3

Mesure du temps d'exécution d'un programme

- OUHOCINE Sarah G3

Professeur :
- Mme m.DJEFFAL

Année universitaire 2018/2019

TP 1 COMPLEXITE

Mesure du temps d'exécution d'un programme

Partie 1 : Développement de l'algorithme et du programme de problème de la somme de n premiers nombres entiers naturels :

➔

Dans ce TP, J'ai développé 4 fonctions pour le problème de la somme de n premiers nombres entiers naturels :

***3 fonctions itératives** : la 1ère avec la boucle ''pour…faire'' ➔ `SommeFor(n)`.*
la 2ème avec la boucle ''répéter…jusqu'à'' ➔ `SommeDoWhile(n)`.
la 3ème avec la boucle '' tant que…faire'' ➔ `SommeWhile(n).`

***1 fonction ''récursive''** : `SommeRecursive(n)`*

Puis j'ai appelé ces 4 fonctions (dans l'ordre) dans le programme principal pour calculer la somme des différents n premiers nombres entiers naturels des nombres demandés ainsi que le temps d'exécution pour chaque nombre demandé dans l'énoncé du tp.

1- LES PROGRAMMES ITTERATIFS (fonctions)

Question 1:

Ci-dessous les algorithmes itératifs:

```
//------------a) La Fonction Somme avec la Boucle "pour…faire" ---------------------//
```

```
reel SommeFor(n :entier)

debut

//déclaration
i, res: int;
debut,fin: clock_t;
tps: reel;


//code
Res=0 ;
debut=clock();
   pour (i=1 à n)
      faire
            res = res+i;
      fait ;
fin=clock();

   tps= (fin-debut)/CLOCKS_PER_SEC;
   écrire("le temps d'exécution = ", tps);
fin
```

//----------------b) La Fonction Somme avec la Boucle "répéter… jusqu'à"------------//

```
reel SommeDoWhile(n :entier)

debut

// déclaration
i, res: entier;
debut,fin : clock_t;
tps : reel;

// code
i=1 ; res=0 ;

  debut=clock();
    répéter  {
            res=res+i;
            i=i+1;
             }
    jusqu'à (i<=n);
  fin=clock();

   tps= (fin-debut)/CLOCKS_PER_SEC;
   écrire("le temps d'execution = ", tps);
fin.
```

//---------------- c) La Fonction Somme avec la Boucle "tant que … faire"----------//

```
reel SommeWhile(n :entier)
debut

// déclaration
i, res : entier;
debut,fin : clock_t;
tps : reel;
```

```
// code
  i=1; res=0;
  debut=clock();

     tant que  (i<=n)
     faire
           res=res+i;
           i=i+1;
     fait ;
  fin=clock();

    tps=(fin-debut)/CLOCKS_PER_SEC;
    écrire("le temps d'execution = ", tps);
fin.
```

Question 2:

Ci-dessous les programmes itératifs (commentés) avec le langage C :

```
//----------------------------Les Bibliothèques-------------------------------------//

#include <stdio.h>
#include <stdlib.h>
#include <time.h>    //bibliothèque pour le temps
#include <math.h>    //bibliothèque pour les différentes fonctions mathématiques.

//------------a) La Fonction Somme avec la Boucle "for" ---------------------//
```

```
 double SommeFor(long int n){

 //déclaration
 long int i,res=0;
 clock_t  debut,fin;
 double tps;


 //code
debut=clock();
    for (i=1;i<n;i++){
           res=res+i;
    }
 fin=clock();

    tps=(double)(fin-debut)/CLOCKS_PER_SEC;
    printf("le temps d'execution = %Lf ", tps);
 }
```

```
//----------------b) La Fonction Somme avec la Boucle "do while"------------//
```

```
double SommeDoWhile(long int n){

 // déclaration
 long int i=1,res=0;
 clock_t  debut,fin;
 double tps;

 // code
 debut=clock();
    do  {
```

```
            res=res+i;
            i=i+1;
    } while (i<=n);
 fin=clock();

    tps=(double)(fin-debut)/CLOCKS_PER_SEC;
    printf("le temps d'execution = %Lf \n", tps);
 }
```

```
//---------------- c) La Fonction Somme avec la Boucle "while"-----------------//
```

```
double SommeWhile(long int n){

 // déclaration
 long int i=1,res=0;
 clock_t  debut,fin;
 double tps;

// code
 debut=clock();
     while (i<=n){
            res=res+i;
            i=i+1;
    }
 fin=clock();

    tps=(double)(fin-debut)/CLOCKS_PER_SEC;
    printf("le temps d'execution = %Lf \n", tps);
 }
```

```
//------------------------------Le Main---------------------------------------//
```

```
int main()
{
       printf("-----------------------------------------\n");
       printf("Temps d'exécution pour la boucle FOR    \n");
       printf("-----------------------------------------\n");

       SommeFor(1000000);       printf(": 10^6  \n");
       SommeFor(2000000);       printf(": 2 * 10^6  \n");
       SommeFor(10000000);      printf(": 10^7  \n");
       SommeFor(20000000);      printf(": 2 * 10^7  \n");
       SommeFor(100000000);     printf(": 10^8  \n");
       SommeFor(200000000);     printf(": 2 * 10^8  \n");
       SommeFor(1000000000);    printf(": 10^9  \n");
       SommeFor(2000000000);    printf(": 2 * 10^9  \n");
       SommeFor(10000000000);   printf(": 10^10  \n");
       SommeFor(20000000000);   printf(": 2 * 10^10  \n");
       SommeFor(100000000000); printf(": 10^11 \n ");
       SommeFor(200000000000); printf(": 2 * 10^11  \n");
       SommeFor(1000000000000);printf(": 10^12  \n");
       SommeFor(2000000000000);printf("2 * 10^12 : \n");

       printf("-------------------------------------------------\n");
       printf("Temps d'exécution pour la boucle DO....WHILE   \n");
       printf("-------------------------------------------------\n");

       SommeDoWhile(1000000);        printf(": 10^6  \n");
       SommeDoWhile(2000000);        printf(": 2 * 10^6  \n");
       SommeDoWhile(10000000);       printf(": 10^7  \n");
       SommeDoWhile(20000000);       printf(": 2 * 10^7  \n");
       SommeDoWhile(100000000);      printf(": 10^8  \n");
```

```
        SommeDoWhile(200000000);     printf(": 2 * 10^8  \n");
        SommeDoWhile(1000000000);    printf(": 10^9  \n");
        SommeDoWhile(2000000000);    printf(": 2 * 10^9  \n");
        SommeDoWhile(10000000000);   printf(": 10^10  \n");
        SommeDoWhile(20000000000);   printf(": 2 * 10^10 \n ");
        SommeDoWhile(100000000000);  printf(": 10^11 \n ");
        SommeDoWhile(200000000000);  printf(": 2 * 10^11  \n");
        SommeDoWhile(1000000000000); printf(": 10^12  \n");
        SommeDoWhile(2000000000000); printf("2 * 10^12 : \n");

        printf("-----------------------------------------\n");
       printf("Temps d'exécution pour la boucle WHILE)   \n");
       printf("-----------------------------------------\n");

        SommeWhile(1000000);         printf(": 10^6  \n");
        SommeWhile(2000000);         printf(": 2 * 10^6  \n");
        SommeWhile(10000000);        printf(": 10^7  \n");
        SommeWhile(20000000);        printf(": 2 * 10^7  \n");
        SommeWhile(100000000);       printf(": 10^8  \n");
        SommeWhile(200000000);       printf(": 2 * 10^8  \n");
        SommeWhile(1000000000);      printf(": 10^9  \n");
        SommeWhile(2000000000);      printf(": 2 * 10^9  \n");
        SommeWhile(10000000000);     printf(": 10^10  \n");
        SommeWhile(20000000000);     printf(": 2 * 10^10 \n ");
        SommeWhile(100000000000);    printf(": 10^11 \n ");
        SommeWhile(200000000000);    printf(": 2 * 10^11  \n");
        SommeWhile(1000000000000);   printf(": 10^12  \n");
        SommeWhile(2000000000000);   printf("2 * 10^12 : \n");

  return 0;
}
```

2- LE PROGRAMME RECURSIF

Question 3 :

Ci-dessous l'algorithme récursif:

```
reel SommeRecursive(n :entier){
debut
 // déclaration
 res : entier;
 debut,fin : clock_t;
 tps : reel;

 // code
 res=0 ;
   debut=clock();

          si(n>0)  alors {res=res+n;
                          SommeRecursive(n-1);}

          Fin_si ;
```

```
   fin=clock();

    tps=(fin-debut)/CLOCKS_PER_SEC;
    écrire("le temps d'execution = ", tps);

fin.
```

Question 4 :

Ci-dessous le programme récursif avec le langage C :

```
//----------------------------Les Bibliothèques-------------------------------------//

#include <stdio.h>
#include <stdlib.h>
#include <time.h>    //bibliothèque pour le temps
#include <math.h>    //bibliothèque pour les différentes fonctions mathématiques.

//-----------------La Fonction Somme avec la "RECURSIVITE"-----------------------//
```

```
 double SommeRecursive(long int n){

 // déclaration
 long int res=0;
 clock_t  debut,fin;
 double tps;

 // code
 debut=clock();

        if(n>0) { res=res+n;
                  SommeRecursive(n-1);}


 fin=clock();

    tps=(double)(fin-debut)/CLOCKS_PER_SEC;
    printf("le temps d'execution = %Lf \n", tps);

}
```

```
//--------------------------------Le Main-------------------------------------//
```

```
int main()
{
    printf("--------------------------------- -------\n");
    printf("Temps d'exécution pour la récursivité    \n");
    printf("-----------------------------------------\n");

    SommeRecursive (1000000);
    SommeRecursive (2000000);
    SommeRecursive (10000000);
    SommeRecursive (20000000);
    SommeRecursive (100000000);
    SommeRecursive (200000000);
```

```
    SommeRecursive (1000000000);
    SommeRecursive (2000000000);
    SommeRecursive (10000000000);
    SommeRecursive (20000000000);
    SommeRecursive (100000000000);
    SommeRecursive (200000000000);
    SommeRecursive (1000000000000);
    SommeRecursive (2000000000000);

    Return 0;}
```

Partie 2 : Mesure du temps d'exécution :

Dans cette partie on note le temps d'exécution pris par chaque programme pour les différentes valeurs de n, créer le graphe correspondant puis étudier la complexité.

1) Mesurer le temps d'exécution des programmes itératifs :

1-SOUS WINDOWS

Question 5

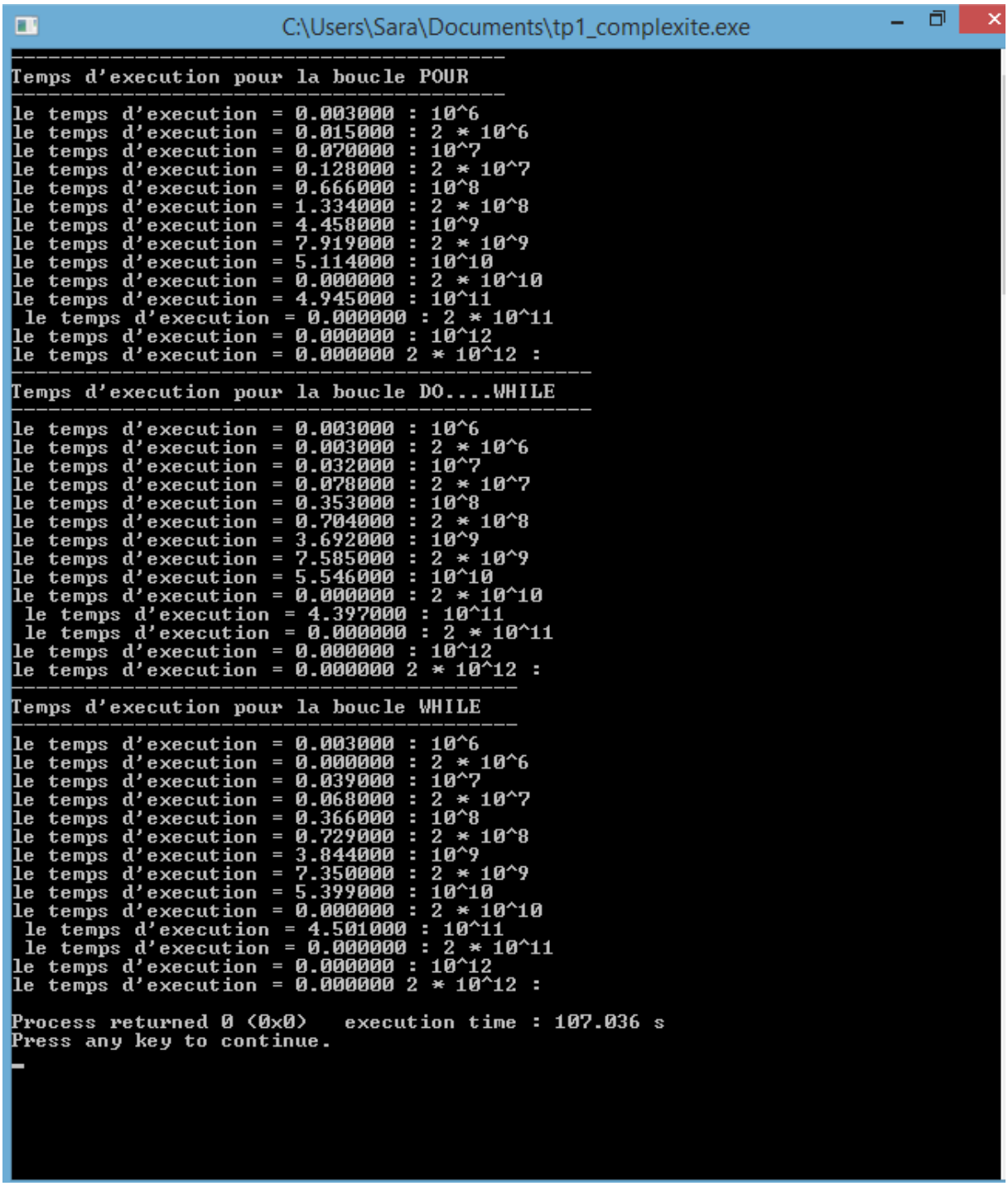
T_1 : Temps d'exécution avec la boucle ''pour…faire''.

T_2 : Temps d'exécution avec la boucle ''répéter…jusqu'à''.

T_3 : Temps d'exécution avec la boucle ''tant que…faire''.

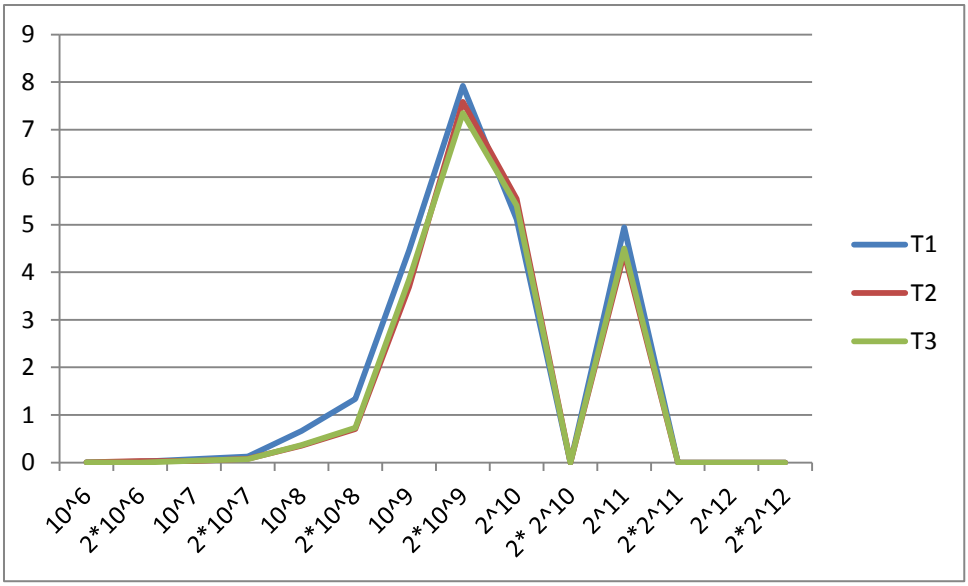
N	10^6	$2*10^6$	10^7	$2*10^7$	10^8	$2*10^8$	10^9	$2*10^9$	10^{10}	$2*10^{10}$	10^{11}	$2*10^{11}$	10^{12}	$2*10^{12}$
T_1	0.003	0.015	0.07	0.128	0.666	1.334	4.458	7.919	5.114	0	4.945	0	0	0
T_2	0.003	0.03	0.032	0.078	0.353	0.704	3.692	7.585	5.546	0	4.397	0	0	0
T_3	0.003	0	0.039	0.068	0.366	0.729	3.844	7.35	5.399	0	4.501	0	0	0

Ci-dessous l'imprime écran de l'exécution des programmes itératifs sous Windows:



Question 6

Ci-dessous le graphe des variations de temps mesuré T1, T2, T3 en fonction de la variable n :



***REMARQUE** : D'après le tableau el le graphe, on remarque bien que sous Windows les temps d'exécution se croit d'une façon presque linéaire jusqu'à un nombre donné (ici 10^{10}), puis au-delà de ce nombre le temps d'exécution se décroit à 0 directement puis il se recroit puis se redécroit… on constate que Windows a une façon de gestion des programmes particulière (propre à lui) lorsque la taille des données est très grande ; du coup on ne peut rien dire sur le graphe*

Malgré ça on peut remarquer que le temps d'exécution des trois boucles sont très proches mais le temps d'exécution des deux boucles 'répéter…jusqu'à'' et ''tant que…faire'' est plus petit que celui de la boucle ''pour…faire''.

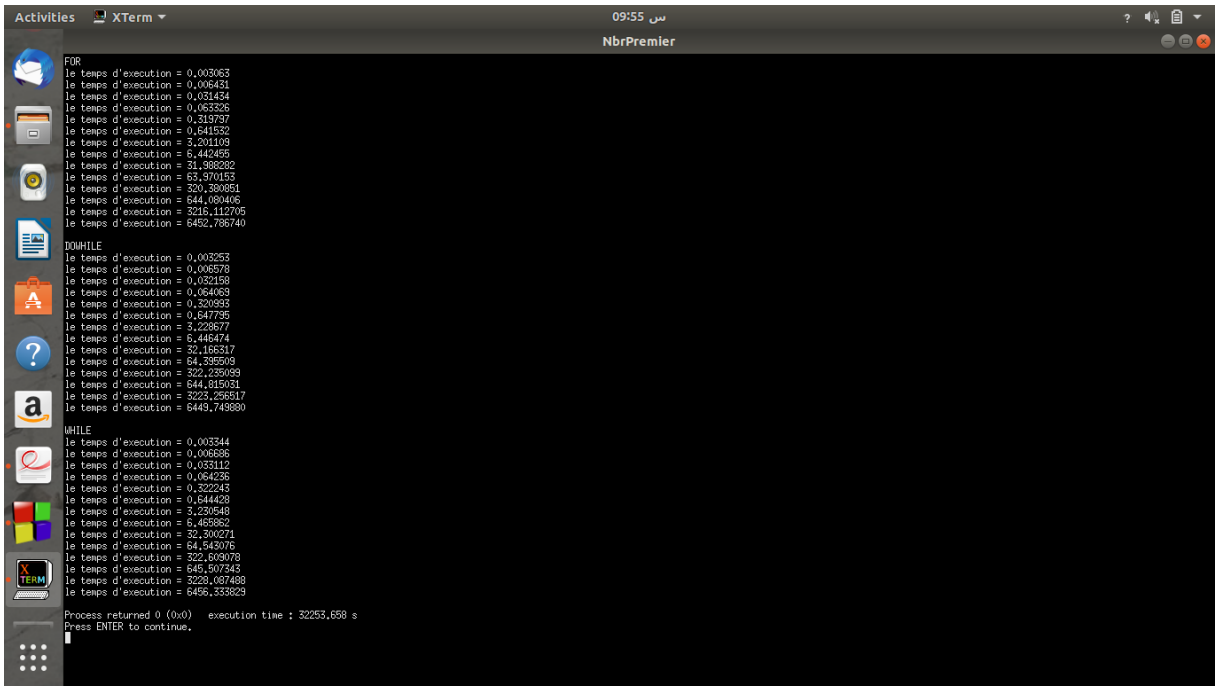
Question 7 :

On sait que la complexité théorique des 3 programmes itératifs est : O(n)=n car cette fonction doit être linéaire, essayons maintenant d'exécuter le programme sous linux pour voir que-est-ce que ça va donner.

2- SOUS LINUX

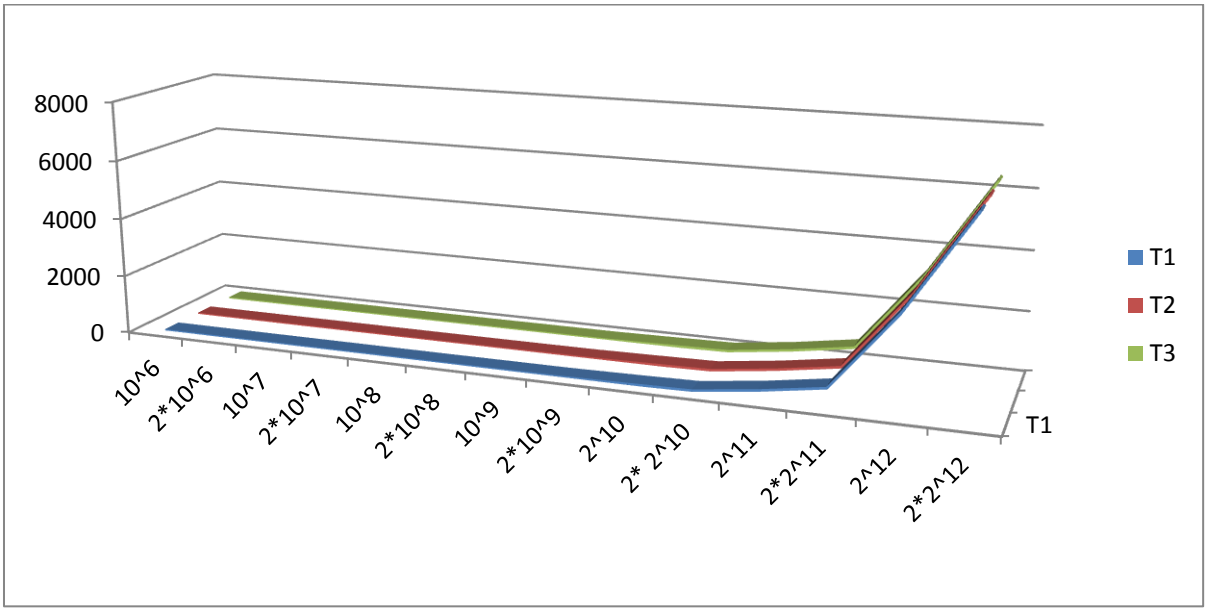
Question 5

N	10^6	$2*10^6$	10^7	$2*10^7$	10^8	$2*10^8$	10^9	$2*10^9$	10^{10}	$2*10^{10}$	10^{11}	$2*10^{11}$	10^{12}	$2*10^{12}$
T1	0,003063	0,006431	0,031434	0,063326	0,319797	0,641532	3,201109	6,442455	31,988282	63,970153	320,380851	644,080406	3216,112705	6452,786774
T2	0,003253	0,006578	0,032158	0,064069	0,320993	0,647795	3,228677	6,446474	32,166317	64,395509	322,235099	644,815031	3223,25652	6449,749888
T3	0,003344	0,006686	0,033112	0,064236	0,322243	0,644428	3,230548	6,465862	32,300271	64,543076	322,609078	645,507343	3228,08749	6456,333883



Question 6

Ci-dessous le graphe des variations de temps mesuré T1, T2, T3 en fonction de la variable n :



***REMARQUE** : On remarque bien maintenant sous Linux, que le temps d'exécution se croit d'une façon on peut dire qu'elle est linéaire jusqu'au bout, mais cela a pris beaucoup de temps pour s'exécuter contrairement à Windows qui n'a pas pris beaucoup de temps (juste*

quelques seconde) mais s'est comporté très bizarrement au bout d'un certain nombre donné.

Question 7 :

La complexité théorique des 3 programmes itératifs est claire maintenant qu'elle est linéaire ➔ $O(n)=n$

2) Mesurer le temps d'exécution du programme récursif :

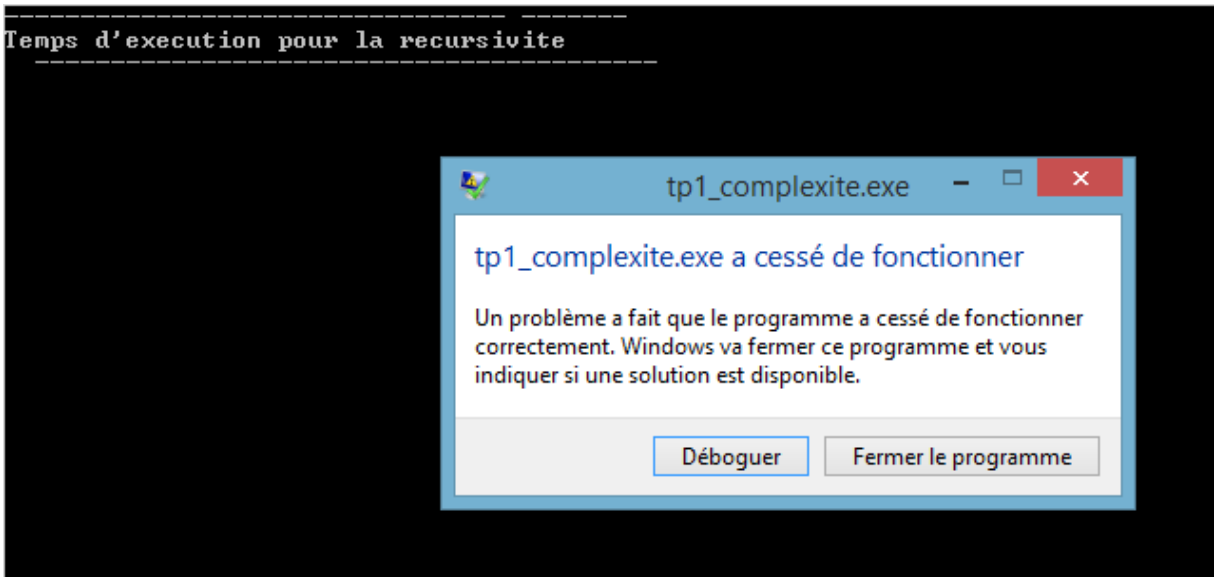
Question 5

1-SOUS WINDOWS

T : Temps d'exécution avec la récursivité.

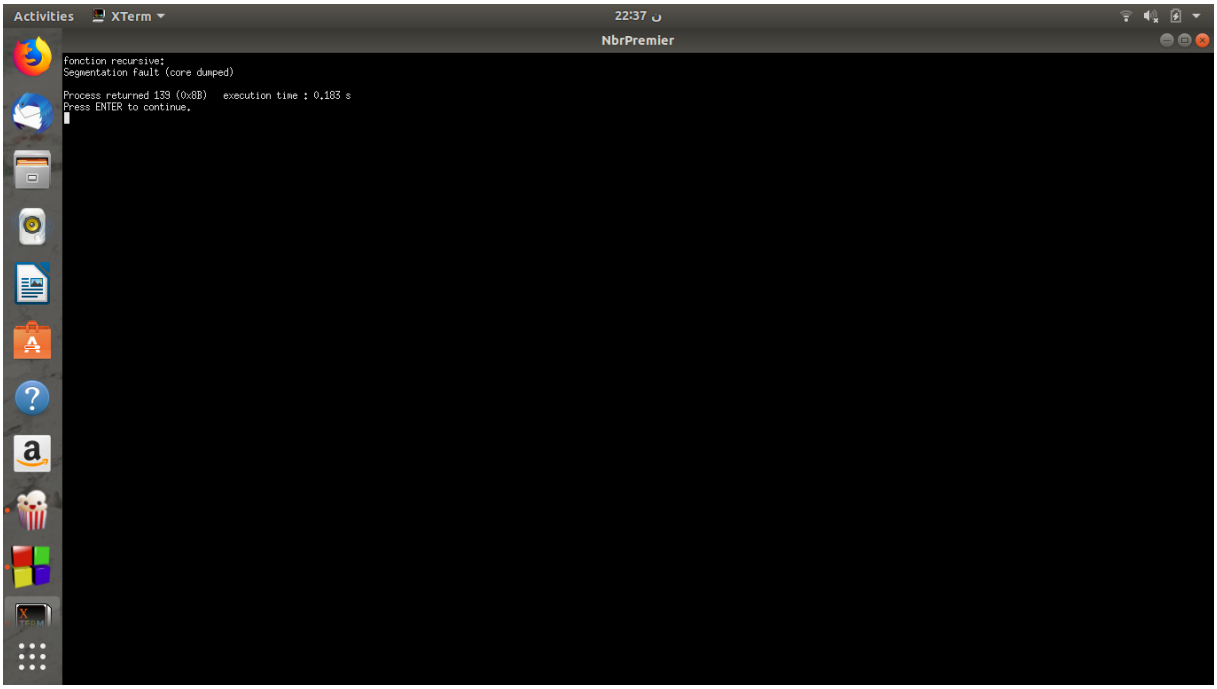
N	10^6	$2*10^6$	10^7	$2*10^7$	10^8	$2*10^8$	10^9	$2*10^9$	10^{10}	$2*10^{10}$	10^{11}	$2*10^{11}$	10^{12}	$2*10^{12}$
T	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Ci-dessous l'imprime écran de l'exécution du programme récursif sous windows:



Essayons maintenant le même programme sous linux.

2-SOUS LINUX



REMARQUE : ***On remarque bien que le système cessera de fonctionner lors de l'exécution du programme récursif que ce soit sous WINDOWS ou LINUX (souffrance du hardware) et ceci revient au débordement de la pile,***

Et donc pour éviter la souffrance du hardware inutilement, les deux OS ont arrêté le programme.

SOLUTION PROPOSEE : ***On peut étendre la pile sous LINUX jusqu'à (8Go), (mais sous Windows, il sera un peu compliqué de l'étendre).***

Question 6

Suite au problème décrit juste avant (question 5) la représentation des variations de temps mesuré T en fonction de la variable n sera impossible.

Question 7

On note que la complexité théorique d'un programme récursif doit être exponentielle c.-à-d. $O(n) = (\text{nbr\_instruction\_du\_prgm\_récursif})^n$.

CONCLUSION:

Comme conclusion de ce TP ;
Lorsqu'on a des données de très grande taille il est préférable d'éviter les fonctions exponentielles et d'utiliser des fonctions linéaires par exemple.
Le temps d'exécution a dépendu : ***1-la fonction utilisée dans le programme (linéaire, expo,……).*** ***2- le système d'exploitation (Windows, Linux…).*** ***3- La performance de la machine utilisée (RAM :4Go, 8Go…) ...***
Les valeurs prohibitives des complexités exponentielles de grandes tailles ne peuvent pas donc constituer une solution pour contourner le problème de l'explosion combinatoire.

FIN

PC UTILISE : Lenovo-PC

PRECESSEUR : INTEL(R) CELERON(R) CPU N2840 @2.16 GHz 2.16GHz

RAM : 4 Go

OS 64 bits, **PROCESSEUR** *64