Université des Sciences et de la Technologie HOUARI BOUMEDIENE
Faculté d'Electronique et d'Informatique
Départements d'Informatique

RAPPORT DE TPS :

« Complexité »

Nom : HAMICI

Prénom : Ryadh

Matricule : 201500009082

Spécialité : MIV

Année universitaire : 2018/2019

Algorithmes sur les ensembles :

A) Génération de sous-ensembles :

1- Algorithme :

La complexité des algorithmes (itératif et récursif) est Téta(2^n), l'idée est de commencer par générer 2 listes de sous-ensembles représentant 2 éléments du tableau en entrée (se sera donc 2 singletons), puis créer la liste des sous-ensembles obtenus par la concaténation de chaque sous ensemble de la première liste avec les sous-ensembles de la deuxième liste et fusionner les 3 listes pour obtenir une seule liste contenant tous les sous-ensembles possibles avec les 2 éléments de départs, puis passer au prochain élément, générer son singleton, générer les sous-ensembles issus de sa concaténation aux sous-ensembles déjà existants, et fusionner toutes les listes et répéter l'opération pour tous les éléments du tableau.

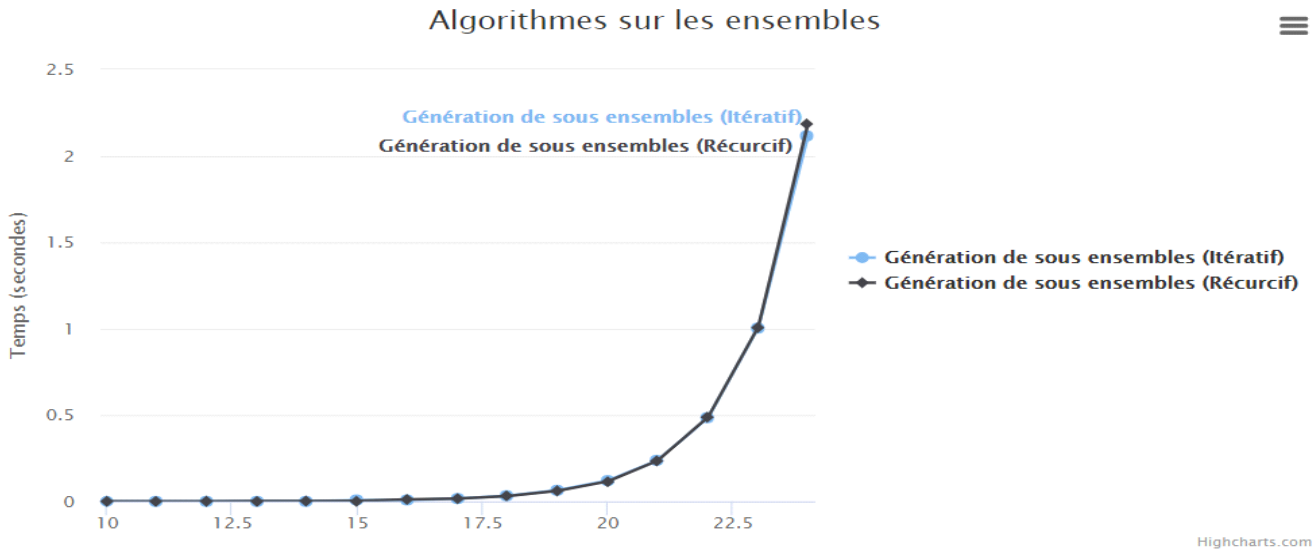
Pour l'algorithme récursif, le principe de "diviser pour mieux régner" a été appliqué, l'idée est la même que dite précédemment, à la différence que le tableau est divisé en sous tableaux pour les quels tous les sous-ensembles possibles seront générés, et que la génération des sous-ensembles du tableau principal se fera par la fusion des listes des sous-ensembles de chaque sous tableau et de la liste résultante de la concaténation de chaque élément d'une liste aux éléments de l'autre liste (fusion de liste avec liste pour l'algorithme récursif et fusion de liste avec un singleton pour l'algorithme itératif).

2- Code source :

Fichier "sous_ensembles.c".

3- Résultats :

Fichier "sous_ensembles.htm".



Algorithmes sur les ensembles

Category	Génération de sous ensembles (Itératif)	Génération de sous ensembles (Récurcif)
10	0	0
11	0	0
12	0	0
13	0.0015	0.0016
14	0.0016	0.0016
15	0.0063	0.0031
16	0.0062	0.0109
17	0.0156	0.0157
18	0.0328	0.0312
19	0.0642	0.061
20	0.1202	0.1156
21	0.2375	0.236
22	0.4842	0.4873
23	1.003	1.0061
24	2.1164	2.1871

B) Recherche d'un sous ensemble dont la somme est égale à une valeur donnée :

1- Algorithme :

La complexité de l'algorithme est O(2^n), l'idée est la même que pour l'algorithme précédent de génération des sous-ensembles sauf que :

-Il s'arrête au premier sous ensemble dont la somme des éléments équivaut à la valeur donnée en paramètre.

-Si un élément à une valeur supérieure à la valeur recherchée, il est ignoré (n'est pas pris en compte pour la génération des sous-ensembles).

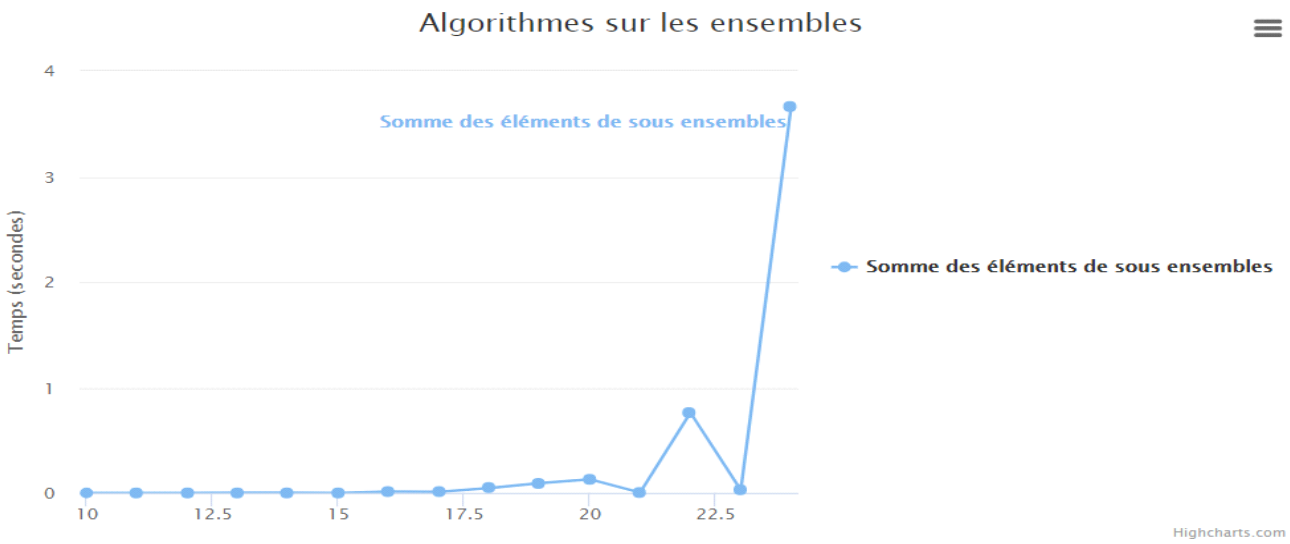
-Si la somme des éléments d'un sous ensemble est supérieure à la valeur recherchée, il est ignoré (n'est pas pris en compte pour la génération des futurs sous-ensembles).

2- Code source :

Fichier "somme_sous_ensembles.c".

3- Résultats :

Fichier "somme_sous_ensembles.htm".



Algorithmes sur les ensembles

Category	Somme des éléments de sous ensembles
10	0
11	0
12	0.0012
13	0.0024
14	0.0028
15	0.0008
16	0.0124
17	0.0104
18	0.0476
19	0.0936
20	0.1301
21	0.004
22	0.7615
23	0.0304
24	3.6644

C) Calcul du Delta de 2 ensembles :

1- Algorithme :

Pour 2 ensembles dont la taille est n et m, on va supposer que n > m. La complexité de l'algorithme est O(n log(n)), on commence par utiliser l'algorithme de tri fusion (de complexité n log(n)) sur les 2 ensembles qui vont être utilisés, puis on effectue un parcours sur les 2 ensembles en même temps, soient i et j les indexes des ensembles 1 et 2 :

-Si ensemble1[i] < ensemble2[j] : On ajoute ensemble1[i] au delta et on incrémente l'index de ensemble1 (i++).

-Si ensemble2[i] < ensemble1[j] : On ajoute ensemble2[j] au delta et on incrémente l'index de ensemble1 (j++).

-Si ensemble1[i] == ensemble2[j] : L'élément n'appartient pas au delta, on incrémente les indexes de ensemble1 et ensemble2 (i++ j++).

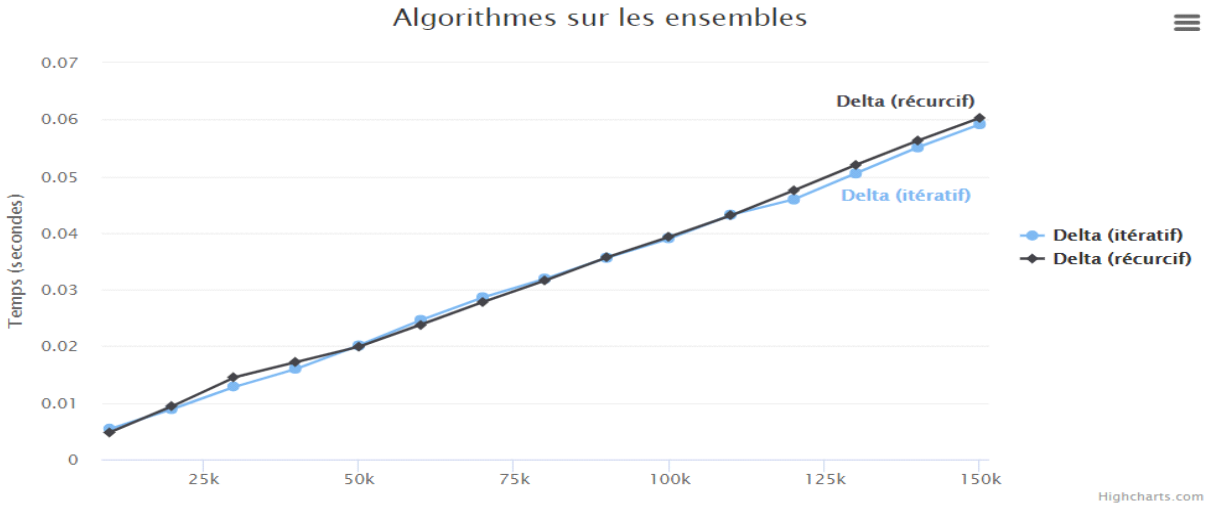
Une fois tous les éléments d'un ensemble parcourus (index == taille de l'ensemble) tous les éléments restants de l'ensemble restant sont ajouté au delta.

2- Code source :

Fichier "delta.c".

3- Résultats :

Fichier "delta.htm".



Algorithmes sur les ensembles

Category	Delta (itératif)	Delta (récurcif)
10000	0.0054	0.0048
20000	0.0089	0.0094
30000	0.0128	0.0145
40000	0.016	0.0172
50000	0.0201	0.0199
60000	0.0246	0.0238
70000	0.0286	0.0278
80000	0.0319	0.0316
90000	0.0356	0.0357
100000	0.039	0.0393
110000	0.0432	0.0431
120000	0.0459	0.0475
130000	0.0505	0.052
140000	0.0551	0.0563
150000	0.0592	0.0603

Algorithmes de Tris :

1- Algorithmes :

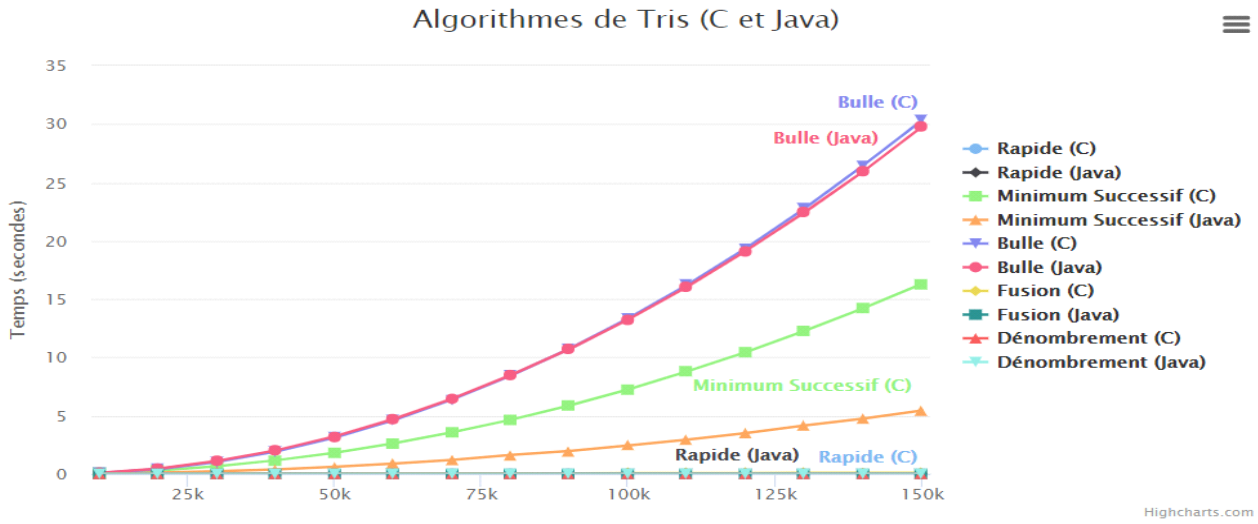
Tri Bulle, Minimum Successif, Rapide, Fusion et Dénombrement.

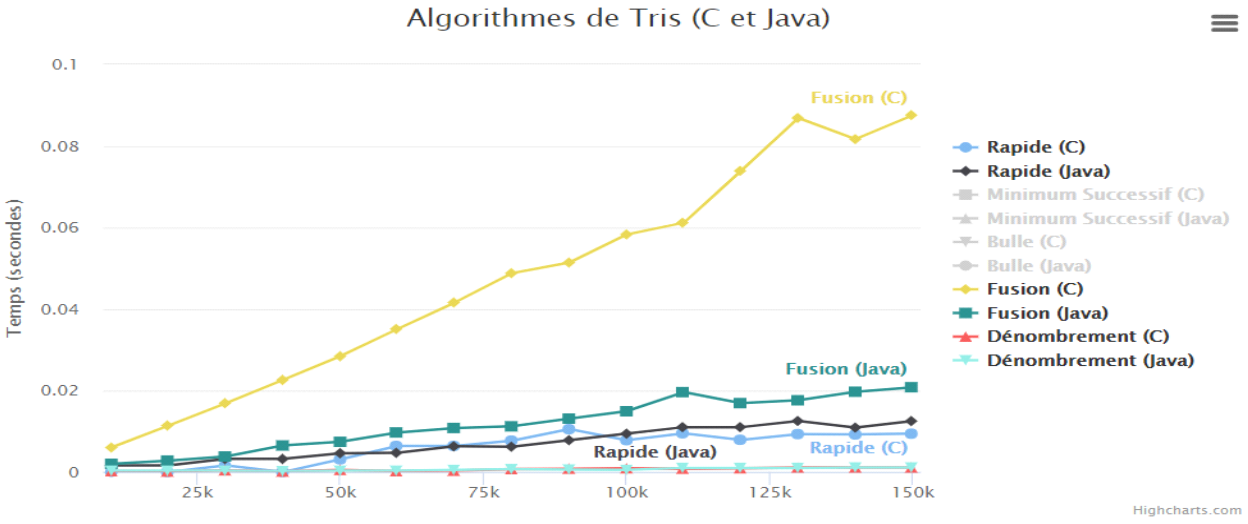
2- Code source :

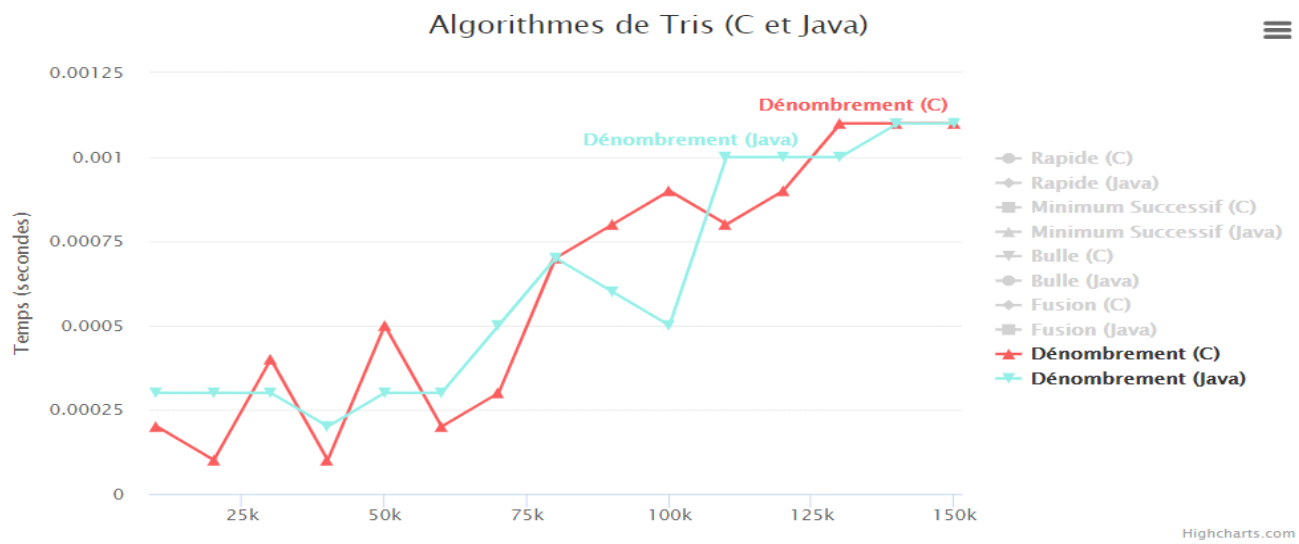
Fichier "tris.c".

3- Résultats :

Fichier "tris.htm".







Category	Rapide (C)	Rapide (Java)	Minimum Successif (C)	Minimum Successif (Java)	Bulle (C)	Bulle (Java)	Fusion (C)	Fusion (Java)	Dénombrement (C)	Dénombrement (Java)
10000	0	0.0016	0.0828	0.0265	0.0749	0.09059	0.006	0.002	0.0002	0.0011
20000	0	0.0016	0.2919	0.1031	0.4057	0.4545	0.0114	0.0028	0.0001	0.0003
30000	0.0016	0.0032	0.6561	0.2343	1.0249	1.1006	0.0169	0.0038	0.0004	0.0003
40000	0	0.0032	1.1623	0.3921	1.9214	2.0248	0.0226	0.0065	0.0001	0.0002
50000	0.0031	0.0046	1.8136	0.6077	3.1097	3.2174	0.0284	0.0074	0.0005	0.0003
60000	0.0064	0.0047	2.6117	0.8779	4.6028	4.6964	0.0351	0.009699	0.0002	0.0003
70000	0.0064	0.0063	3.5651	1.1903	6.3771	6.4448	0.0416	0.010799	0.0003	0.0005
80000	0.0077	0.0062	4.6406	1.6262	8.4376	8.47919	0.0488	0.011199	0.0007	0.0007
90000	0.0105	0.0078	5.8714	1.9713	10.734	10.7188	0.0514	0.013099	0.0008	0.0006
100000	0.0078	0.0094	7.2518	2.44219	13.3355	13.2511	0.0583	0.0149	0.0009	0.0005
110000	0.0095	0.011	8.7882	2.9377	16.2141	16.05019	0.0612	0.0196	0.0008	0.001
120000	0.0079	0.011	10.4416	3.4954	19.3507	19.1361	0.0739	0.0169	0.0009	0.001
130000	0.0093	0.0125	12.2645	4.1499	22.776	22.44359	0.0869	0.0176	0.0011	0.001
140000	0.0092	0.0109	14.2148	4.7498	26.471	25.9604	0.0817	0.0197	0.0011	0.001099
150000	0.0094	0.0125	16.3059	5.4417	30.3496	29.7915	0.0876	0.020799	0.0011	0.0011

Autres Algorithmses :

A) Calcul de 2^(2^n)

1- Algorithmes :

Trois Algorithmes :

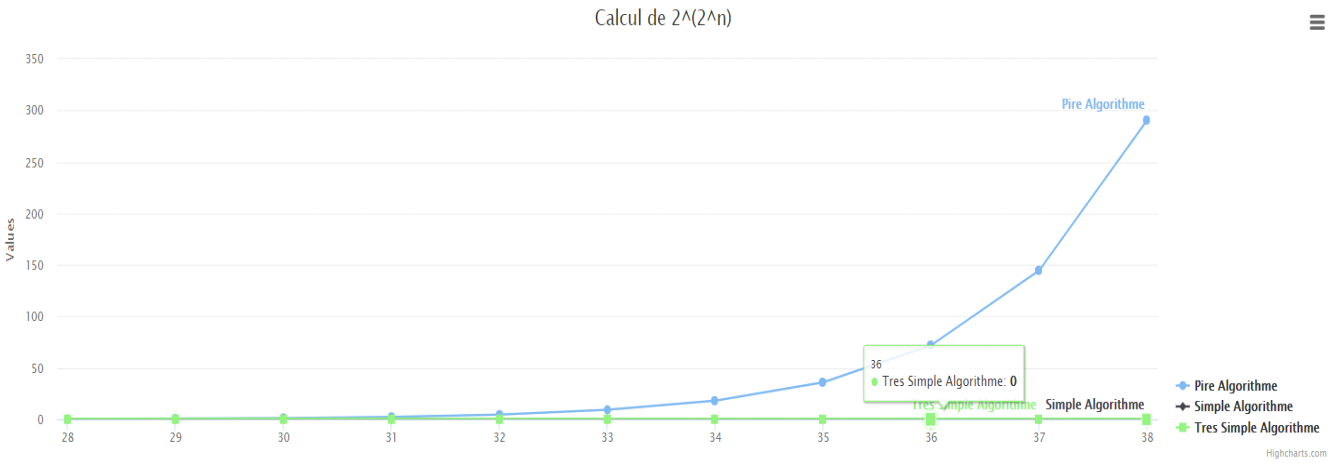
-Récursif de complexité O(2^n)
-Récursif de complexité O(n)
-Itératif de complexité O(n)

2- Code source :

Fichier "2P(2Pn).c".

3- Résultats :

Fichier "2P(2Pn).htm".



Calcul de 2^(2^n)

Category	Pire Algorithme	Simple Algorithme	Tres Simple Algorithme
28	0.3	0	0
29	0.59	0	0
30	1.24	0	0
31	2.31	0	0
32	4.52	0	0
33	9.15	0	0
34	18.03	0	0
35	35.97	0	0
36	72.04	0	0
37	144.61	0	0
38	290.56	0	0

B) Multiplication de M matrices de taille NxN

1- Algorithme :

Algorithme itératif de complexité O(m*(n^3)).

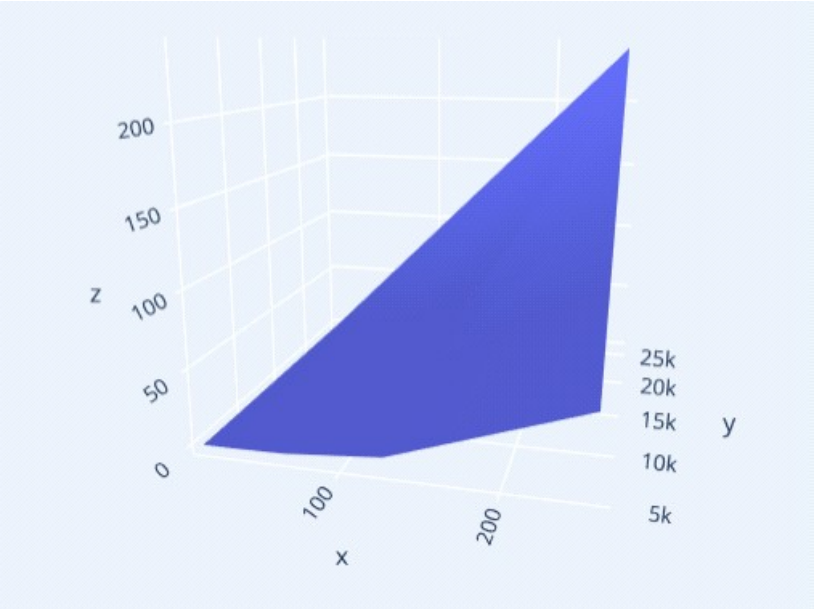
2- Code source :

Fichier "matrices.c".

3- Résultats :

Fichier "matrices.htm".

Avec x = N (taille de la matrice), y = M (nombre de matrices à multiplier) et z = temps d'exécution.



8	5000	0.003100
8	10000	0.004700
8	15000	0.009400
8	20000	0.011000
8	25000	0.011000
16	5000	0.015600
16	10000	0.029700
16	15000	0.040600
16	20000	0.053100
16	25000	0.076600
32	5000	0.096900
32	10000	0.189900
32	15000	0.281100
32	20000	0.371800
32	25000	0.496800
64	5000	0.751400
64	10000	1.456000
64	15000	2.158500
64	20000	2.886500
64	25000	3.994300
128	5000	6.061000
128	10000	12.128700
128	15000	18.170400
128	20000	24.222000
128	25000	30.049800
256	5000	48.878700
256	10000	98.748300
256	15000	146.586700
256	20000	213.073300
256	25000	240.449800

C) Calcul du Nème élément de la suite de Fibonacci

1- Algorithmes :

2 Algorithmes :

-Récurcif de complexité O(2^n).

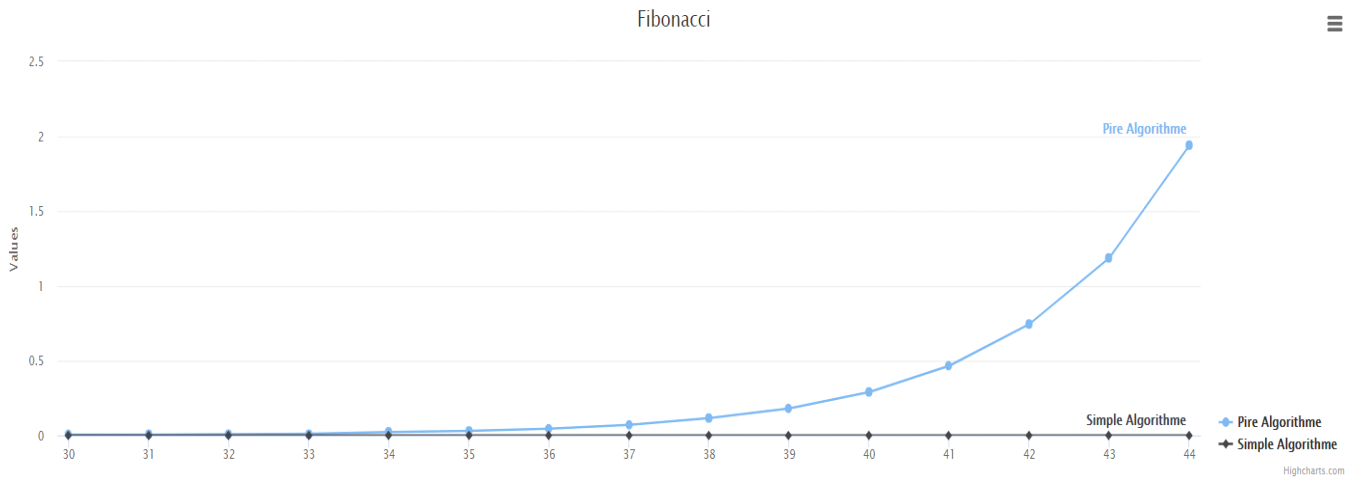
-Itératif de complexité O(n).

2- Code source :

Fichier "fibonacci.c".

3- Résultats :

Fichier "fibonacci.htm".



Fibonacci

Category	Pire Algorithme	Simple Algorithme
30	0.0044	0
31	0.0052	0
32	0.0078	0
33	0.0112	0
34	0.022	0
35	0.03	0
36	0.0456	0
37	0.0708	0
38	0.1164	0
39	0.181	0
40	0.2909	0
41	0.4662	0
42	0.7451	0
43	1.1872	0
44	1.9411	0