1.1) P1069712

1.2) IFT2015 H15 Major TP1.

1.3) Date de remise : Dimanche 8 mars 23h55
1.4) Date de remise : Dimanche 8 mars 23h00

2) Résumé

2.1) Nous avons implanté les fonctions genereADDS1, genereADDS2, genereADDS3, genereADDS1var et genereADDS2var, puis, on les a utilisées pour générer les fonctions ADDSX et ADDSXVARX, mais nous n'avons pas pu générer ADDS3var.

2.2) Nous avons observé des mesures et des graphiques qui nous ont on permis de faire des analyses empiriques et théoriques de nos algorithmes.

2.3) Le temps d'exécution de ADDS3 était très proche de celui de ADDS2 ce qui nous a surpris.

3) Résultats expérimentaux

3.1) Graphique donnant le temps d'exécution en fonction de la longueur des nombres

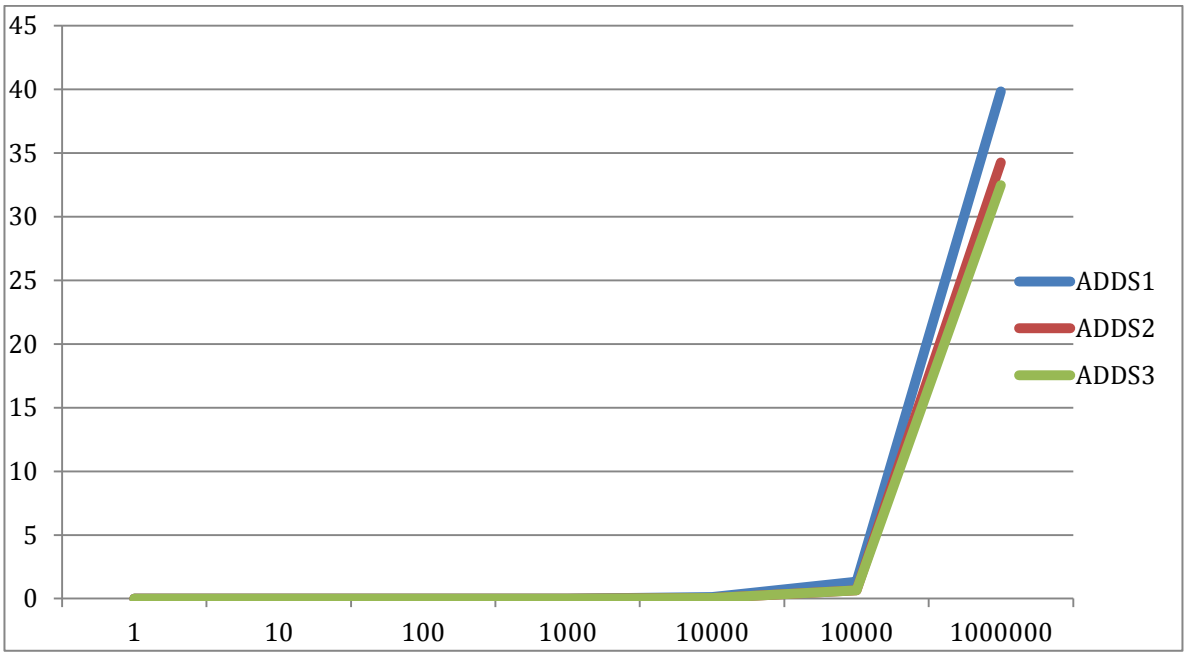

Figure1.Temps d'exécution en fonction de la longueur des nombres

3.2) Tableau

	ADDS1 ( )	ADDS2 ( )	ADDS3 ( )
Temps d'exécution	2.431838187010726	0.6315139260113938	0.5515116

Tableau1. Temps d'exécution de tfunc1, tfunc2 et Tfunc3

3.3) Graphique donnant le temps d'exécution en fonction de la valeur de la base pour les 3 algorithmes

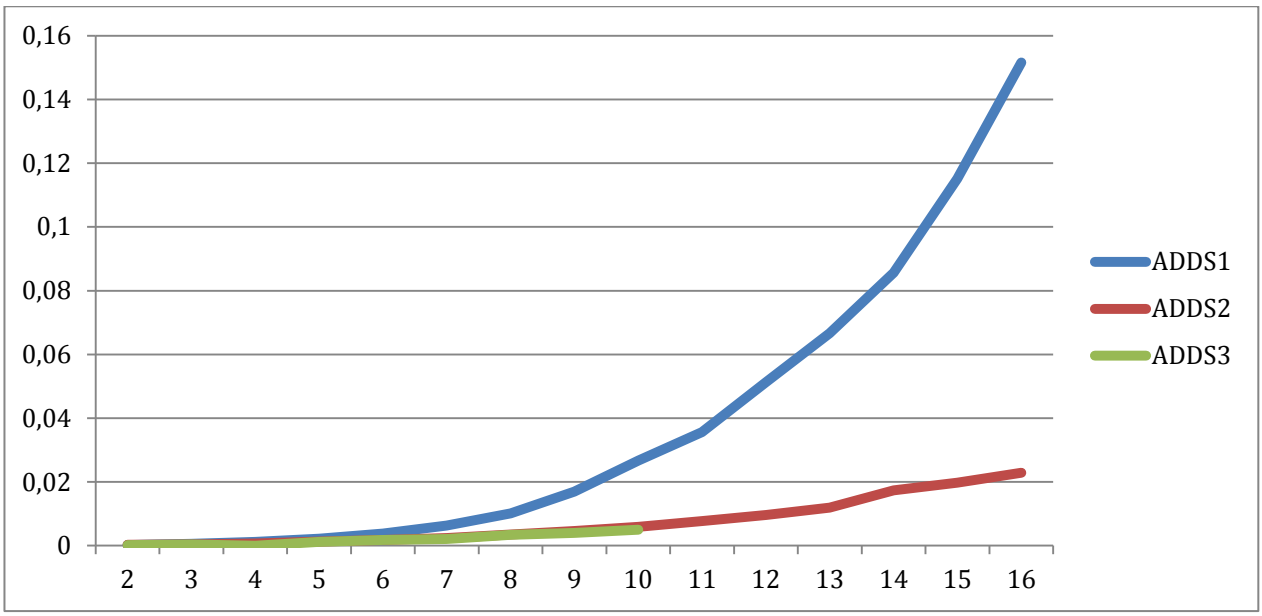

Figure2.Temps d'exécution en fonction de la valeur de la base

4) Les réponses aux questions :

4.1) Analyse théorique grand-O de ADDS1

- Meilleur cas : l'algorithme fait le nombre minimal des comparaisons qui est 2. C'est le cas si ((a=='0') & (b=='0'))
  Le coût d'évaluation de cette branche est 2, et le coût du retour est 1,alors cet algorithme est O (1)
- Pire cas : l'algorithme fait le nombre maximal des comparaisons qui est 81, alors l'algorithme est O (n^2).

4.2) Analyse théorique grand-O de ADDS2

- Meilleur cas : L'algorithme va faire deux comparaisons alors cet algorithme est O (1)
- Pire cas : l'algorithme fait le nombre maximal des comparaisons qui est 18, donc il est O (2n) alors cet algorithme est O (n)

4.3) Analyse théorique grand-O de ADDS3

- Meilleur cas : L'algorithme est O (1)
- Pire cas : l'algorithme est O (log (n))

4.4) ADDS4 en O (1)

Oui c'est possible, en mettant les valeurs dans une table de hachage, l'ordre de l'accès à l'élément se fait à l'ordre de 1, alors cet algorithme fait l'addition à l'ordre de 1.

4.5) Comportement des Tadd avec les 3 algorithmes

L'application tADD va être rapide avec ADDS1, mais, pas aussi rapide qu'avec ADDS2 et pas aussi rapide qu'avec ADDS3 car le résultat sera obtenu d'une manière dichotomique alors le tADD devrait être évidemment plus rapide.

4.6) Discussion concernant les résultats obtenus dans les graphiques

En observant la figure 1 ci dessus, nous remarquons que pour les 3 algorithmes le temps d'exécution augmente quand le nombre de chiffres augmente.

Ainsi, nous remarquons que pour n=15, ADDS2<ADDS1 alors ADDS2=O (ADDS1), et pour n=25, ADDS3<ADDS1 alors ADDS3=O (ADDS2).

- Conclusion : ADDS3 =O (ADDS2) =O (ADDS1).

En observant le graphique 2 ci dessus, nous remarquons que la courbe de ADDS1 est une courbe de la fonction n^2 ,ce qui confirme nos analyses théoriques, de même qu'avec la figure 1 nous pouvons conclure que ADDS3 =O (ADDS2) =O (ADDS1).

4.7) **Les résultats obtenus avec tfunc et Tadd**

Les résultats obtenus avec les fonctions tfunc et tadd sont consistants entre eux dans le fait que ADDS1 a le plus grand temps d'exécution.

Mais, il ne sont pas consistants entre eux dans le faite que le temps d'exécution obtenu avec **Tadd** des algorithmes ADDS1 et ADSS2 sont très proches alors qu'avec **Tfunc** il y'a un gros décalage, ADDS1 fait une tour de boucle par 0.02431838 ns alors que ADSS2 fait une tour de boucle par 0, 00631514, ADDS2 est 4 fois plus rapide que ADDS1. (Voir tableau ci dessus)

4.8) Assignation de classe de complexité basée sur les observations expérimentaux

Les observations expérimentales ont des limites et dépendent des machines alors on peut pas assigner a chaque algorithme une classe de complexité.

Et puisque les analyses théoriques et les observations expérimentales sont complémentaires alors en ajoutant a nos analyses empiriques des analyses théoriques on peut arriver a assigner a chaque algorithme une classe de complexité.

4.9) **Les résultats théoriques s'accordent bien avec vos résultats expérimentaux**

Mes résultats théoriques ne s'accordent pas bien avec mes résultats empiriques car empiriquement ADDS2 et ADDS3 ont presque le même temps d'exécution, alors, qu'avec nos analyses théoriques ADDS3 est meilleur que ADDS2.

4.10) **Les différences entre le temps d'exécution de ADDS1 et ADDS2 ?**

Le temps d'exécution de ADDS1 et ADDS2 sont différents car ADDS1 fait 4,5 fois plus de comparaisons qu'ADDS2

Les différences entre le temps d'exécution de ADDS2 et ADDS3 ?

Le temps d'exécution de ADDS2 et ADDS3 ne sont pas très différents.

Les différences entre le temps d'exécution de ADDS1 et ADDS3 ?

Le temps d'exécution de ADDS1 et ADDS3 sont différents, car, le premier fait des comparaisons bêtes, alors, que le deuxième fait une recherche dichotomique qui divise le nombre de comparaisons par deux.

4.11) Est-ce que le fait que le code soit plus sophistiqué (plus complexe) mène nécessairement à une application (tADD) plus rapide?

Logiquement, oui mais pas nécessairement, car ADDS3 est plus sophistiqué qu'ADDS2, mais l'application Tadd n'est pas plus rapide.

L'inverse est vrai car le code du ADDS2 n'est pas complexe mais tADD est rapide.

4.12) Le nombre de chiffres dans les nombres additionnés soit une valeur utile pour l'analyse des algorithmes

Oui, le nombre de chiffres dans les nombres additionnés est une valeur utile mais pas suffisante pour l'analyse des algorithmes.

5) Source de mes informations

5.1) Irina Tatur et Samuel Huppé m'ont aidé.

5.2) Sources internet

Principe de la fouille binaire.
Disponible sur :**<**http://code18.blogspot.ca/2009/01/arbre-binaire-de-recherche.html>

Apprenez a programmez en python
Disponible sur :<http://openclassrooms.com/courses/apprenez-a-programmer-en-python>

Livres

Michael T. Goodrich, Roberto Tamassia & Michael H. Goldwasser (2013) Data Structures & Algorithms in Python, John Wiley & Sons, Inc (ISBN 978-1-118-29027-9)