Haseeb JAVAID
Mathieu JUGI

Rapport TPs SDA

TP1:

1)

$C(n)$ coût pour la $n-$ième opération

$a(n)$ coût amortie pour la $n-$ième opération

$\Phi(n)$ Fonction potentielle

$$a(n) = c(n) + \Phi(n) - \Phi(n-1)$$

$$a(n) \geq 0$$

Si C(n) est bas, $\Phi(n)$ est grand

C(n) est élevé, $\Phi(n)$ est petit

=0

$$\Phi(n) = 2.n - capacité$$

$$a(n) = C(n) + \Phi(n) - \Phi(n-1)$$

$$a(n) = \begin{cases} 1+2\text{-capacité-}2(n-1)+\text{capacité}=3 \\ n+2n-2n-2(n-1)+n=2 \end{cases}$$

2)

On agrandit le tableau d'une constante α avec $\alpha > 1$

$(n \rightarrow \alpha n) \qquad \alpha = \theta(n)$

$$\theta(n) = \frac{\alpha.n - capacité}{\alpha - 1}$$

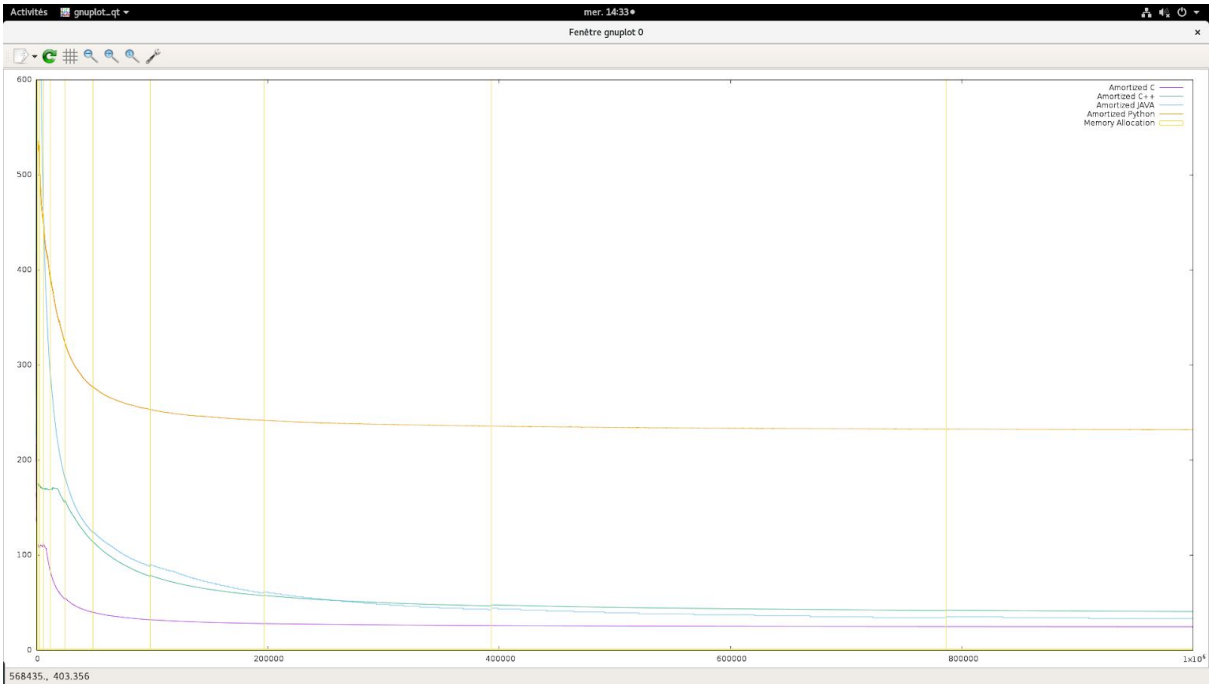
$$a(n) = C(n) + \Phi(n) - \Phi(n-1)$$

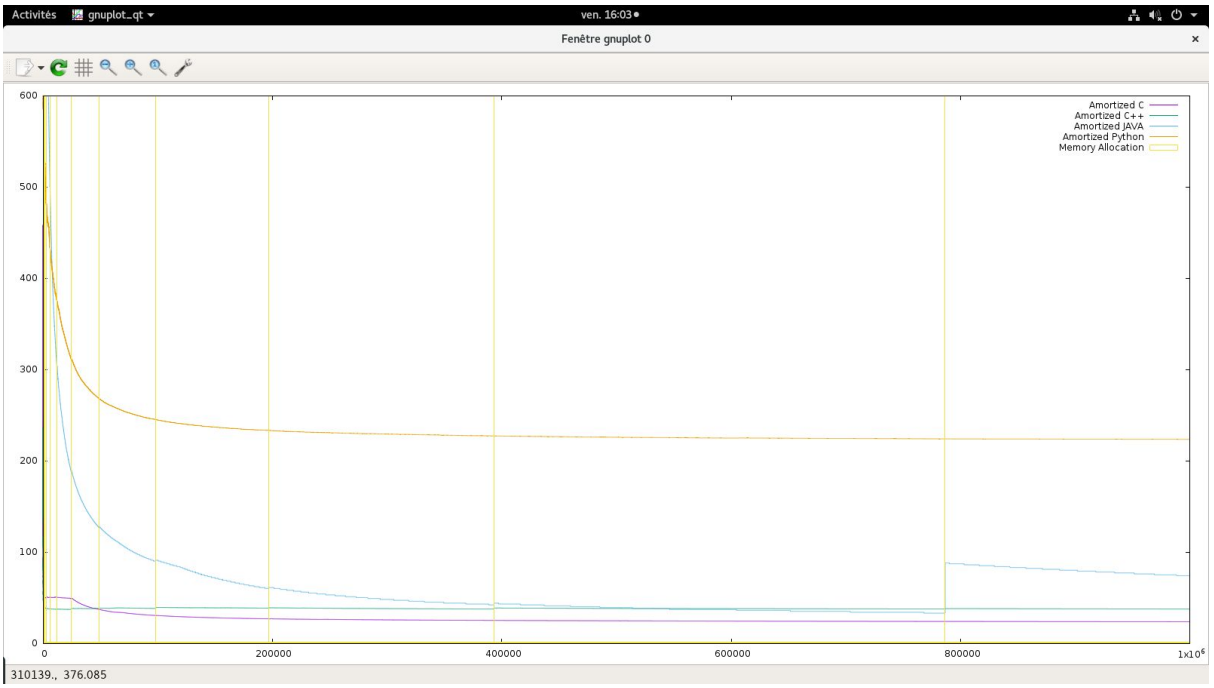
$$a(n) = \begin{cases} 1 + (\alpha.n - \text{capacité}) / (\alpha - 1) - (\alpha.(n - 1) - \text{capacité}) / (\alpha - 1) \\ 1 + (\alpha.n - \text{capacité} - \alpha.(n - 1) + \text{capacité}) / (\alpha - 1) \\ 1 + \alpha / (\alpha - 1) \\ n + (\alpha n - \alpha n) / (\alpha - 1) - (\alpha(n - 1) - n) / (\alpha - 1) = (n(\alpha - 1) - \alpha(n - 1) + n) / (\alpha - 1) \\ (\alpha - n - \alpha n + \alpha + n) / (\alpha - 1) = \alpha / (\alpha - 1) \end{cases}$$

$$a(n) = \begin{cases} 1 + \alpha / (\alpha - 1) = \Theta(1) \\ \alpha / (\alpha - 1) = \Theta(1) \end{cases}$$

3)
Dans https://depot.lipn.univ-paris13.fr/david/sda on effectue les étapes suivantes dans un terminal :
> git clone + adresse
> cd sda
> git checkout tp1
> On copie le contenu du README.md et on attend la fin de l'exécution
> quit
> cd plots
> more plot result
> gnuplot
> On copie les 3 lignes une par une et on affiche le graphe des coûts amorties pour les différents langages de programmation

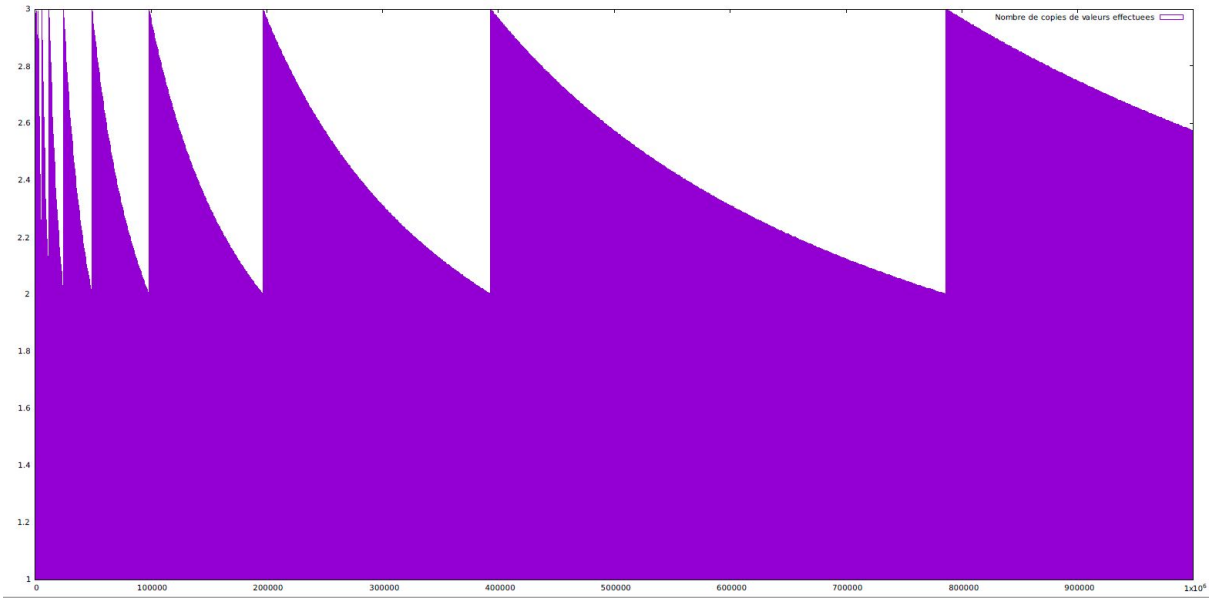
a)





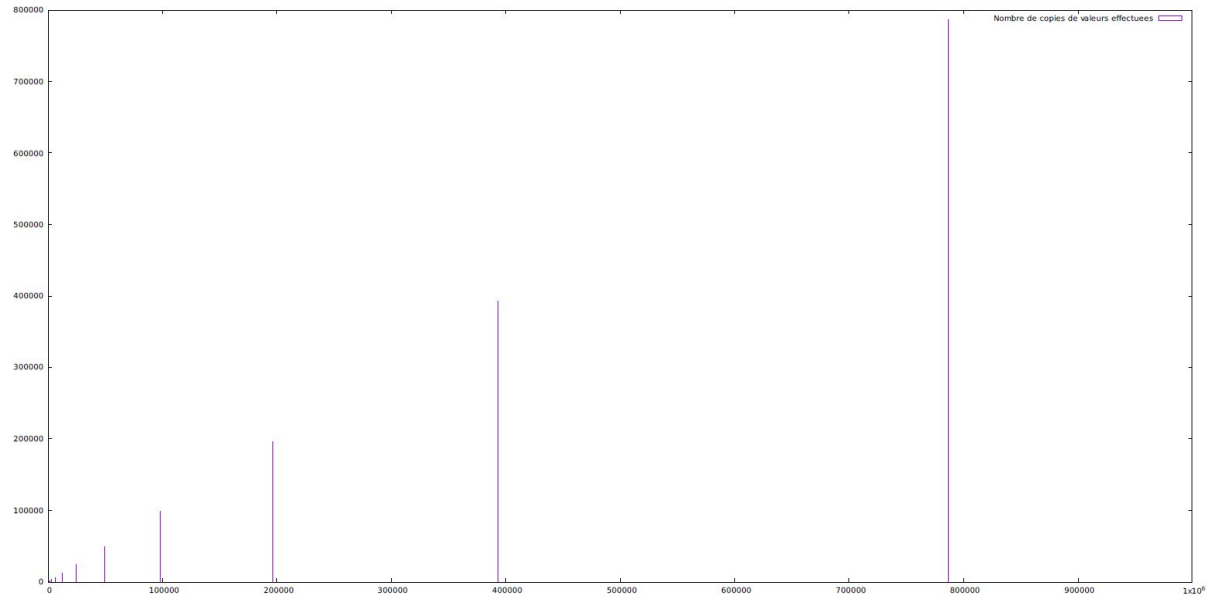
Le code Python semble être le plus lent à s'exécuter malgré le fait que les différentes fonctions ait la même complexités. Cette lenteur s'explique par le fait qu'après avoir enregistrer le résultat sur la ram, on l'enregistre sur le disque dur qui est beaucoup plus lent.

b)



En générale, c'est au moment de l'allocation mémoire que le coût amorti augmente. Mais, il peut aussi augmenter à d'autres moments, et cela est dû aux tâches simultanés effectuées par l'OS(le programme n'est pas le seul à tourner).

c) On remarque que les valeurs fluctuent entre 2 et 3 (la théorie l'avait prédit). On obtient de tel résultat parce qu'ici, on a seulement les copies sans les opérations annexes qui sont normalement réalisées (allocation tableau etc...). La théorie nous permet de prévoir le nombre de fois où une opération va être exécuté mais pour différents temps et différentes opérations.



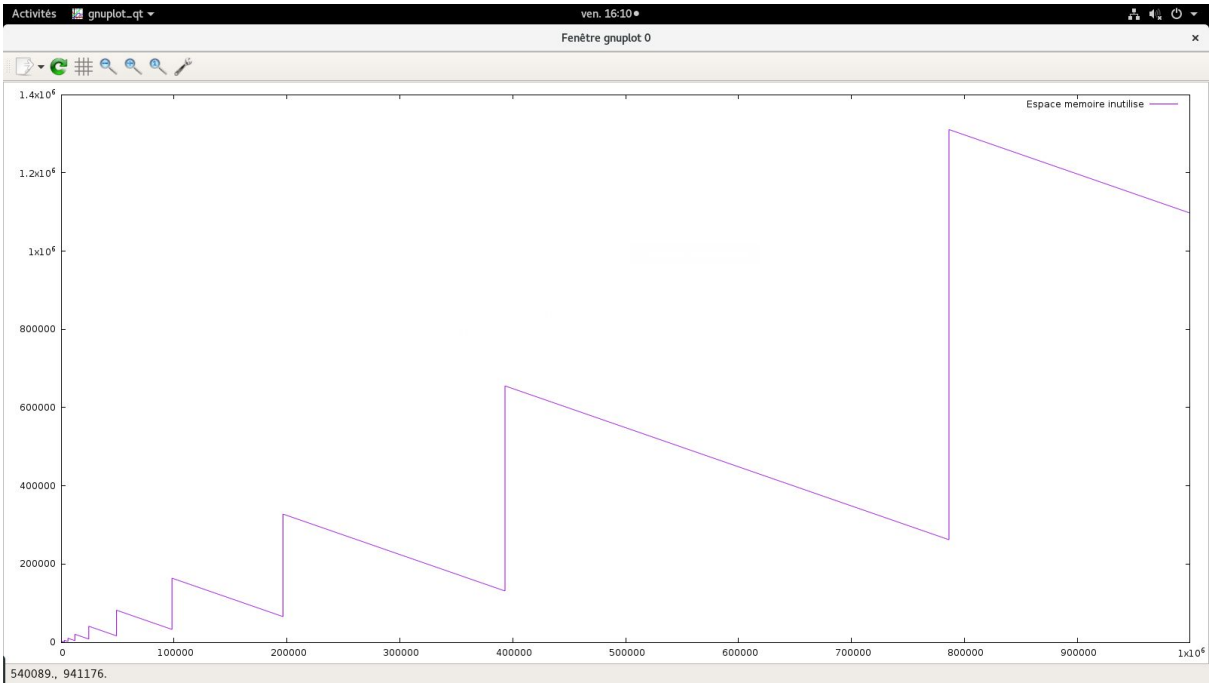
d) Les temps d'exécution changent parce que d'autres programmes tournent en parallèle.

e) Certains langages sont plus rapides que d'autres parce qu'ils y a des différences notables entre ceux-ci :

- En C et C++ sont totalement compilés
- Java et Python sont interprétés et on un garbage collector
- En C++,JAVA et python on a écrit une classe qui en appelle un autre qui fait exactement les même test, ce qui est inutile.

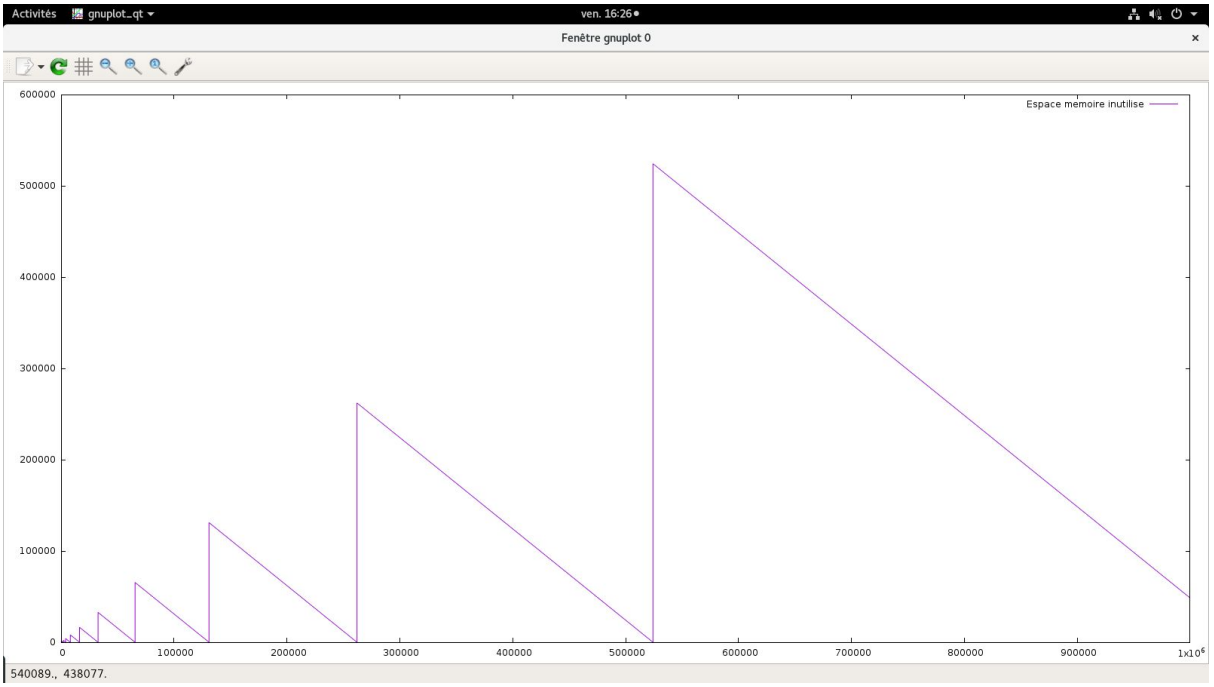
Si Java accélère dans certains cas, c'est qu'il compile les bouts de codes qui sont souvent réutilisés.

f) La mémoire utilisées augmente par pics. Cela est dû au allocations mémoires qui doublent le tableau à chaque insertions. L'agrandissement de la taille du tableau s'effectue lorsqu'il est remplie au ¾, donc ¼ du tableau n'est pas utilisé à chaque fois.On peut imaginer un scénario ou le tableau est copié en boucle.


Activités
gnuplot_qt
ven. 16:10
Fenêtre gnuplot 0
Espace memoire inutilise
1.4x10^6
1.2x10^6
1x10^6
800000
600000
400000
200000
0
0
100000
200000
300000
400000
500000
600000
700000
800000
900000
1x10^6
540089., 941176.

4)

Dans la fonction “do_we_need_to_enlarge_capacity”, il suffit de modifier “(capacity * 3)/4” et de mettre “capacity” tout seul. Ainsi nous n’avons pas de perte.

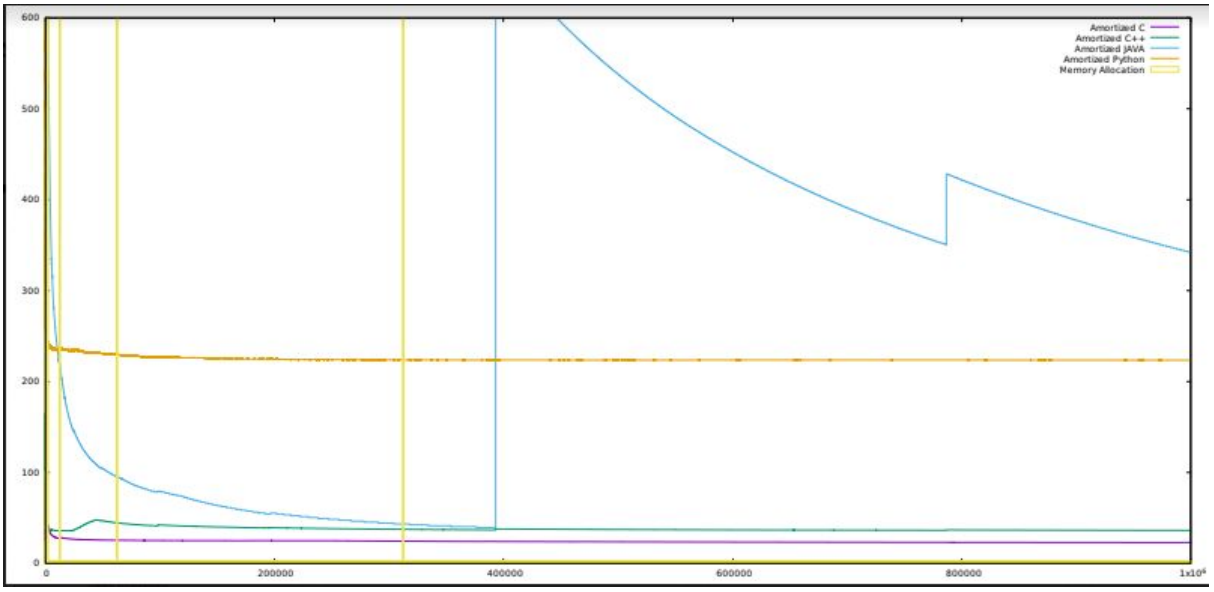


5)

En modifiant le facteur multiplicatif α, on modifie l’agrandissement du tableau.

On remarque que le rapport entre le coût en temps et le coût en espace est multiplié par α.

Quand α=5:



TP2:

1)
$\Phi(i) = |2 * nomi - taillei|$
$\Phi(n) = 2n - capacite$

taillei-1=3.nomi	Ci = 1
taillei=2.nomi	nomi - nomi-1 = -1
Ci=nomi	

$\Phi(n) = |2nomi - taillei|$
$O(n) = C(n) + \Phi(n) - \Phi(n-1)$

$O(n) =$ | 1+ 2.nomi - taillei - 2.nomi-1 + taillei-1
| nomi + 2.nomi - taillei - 2.nomi-1 + taillei-1

$O(n) =$ | 1 + 2.nomi - 2nomi - 2.nomi-1 + 3nomi
| 3nomi - 2nomi - 2nomi-1 + 3nomi

$O(n) =$ | 1 - 2nomi-1 + 3nomi
| 4nomi - 2nomi-1

$O(n) =$ | nomi -1
| 2nomi -2