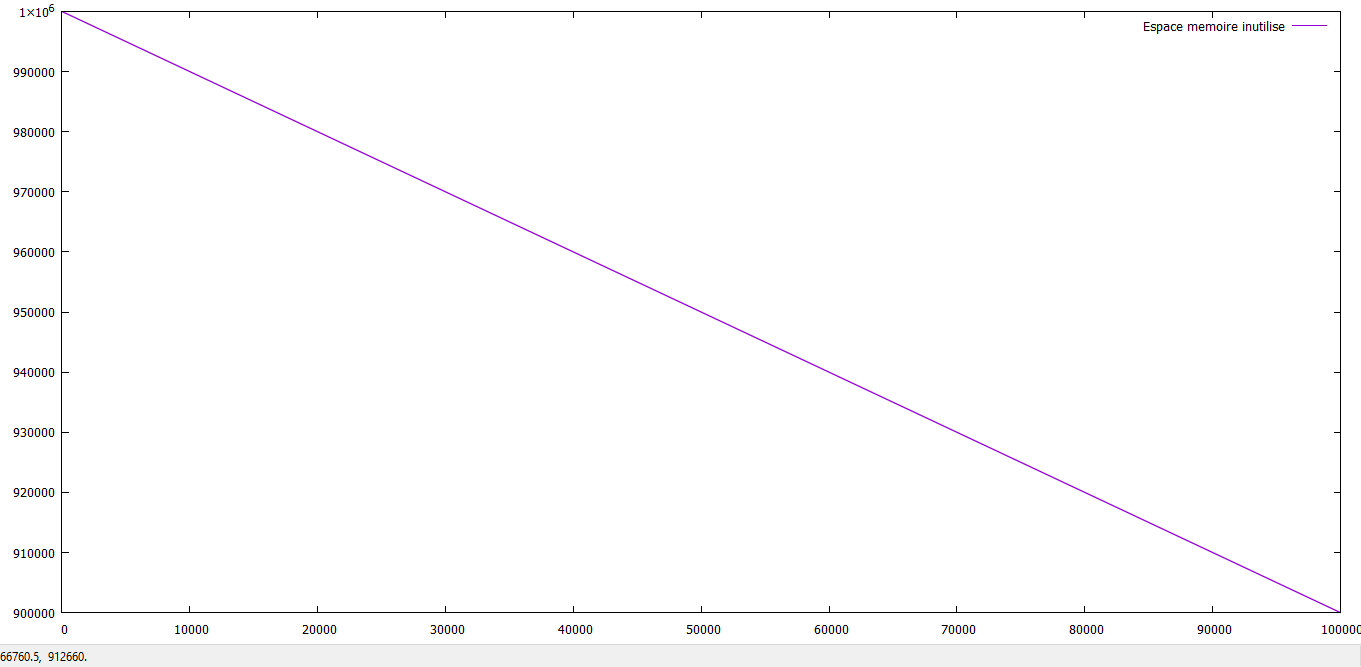
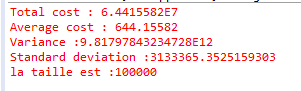
**B-ARBRES :**

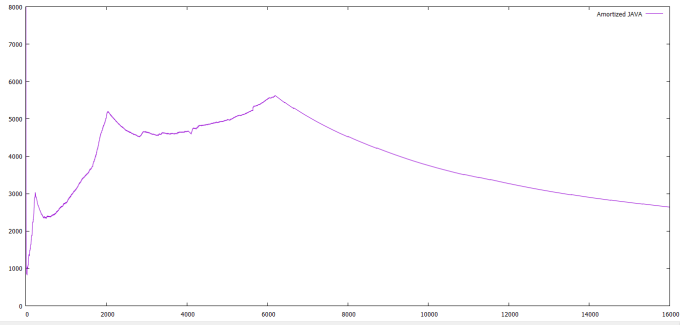
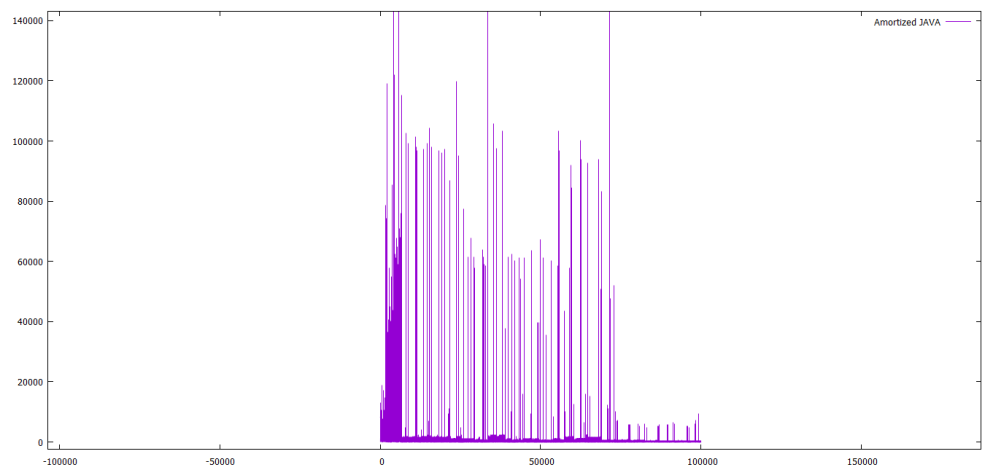
Ajout seulement :

Random :

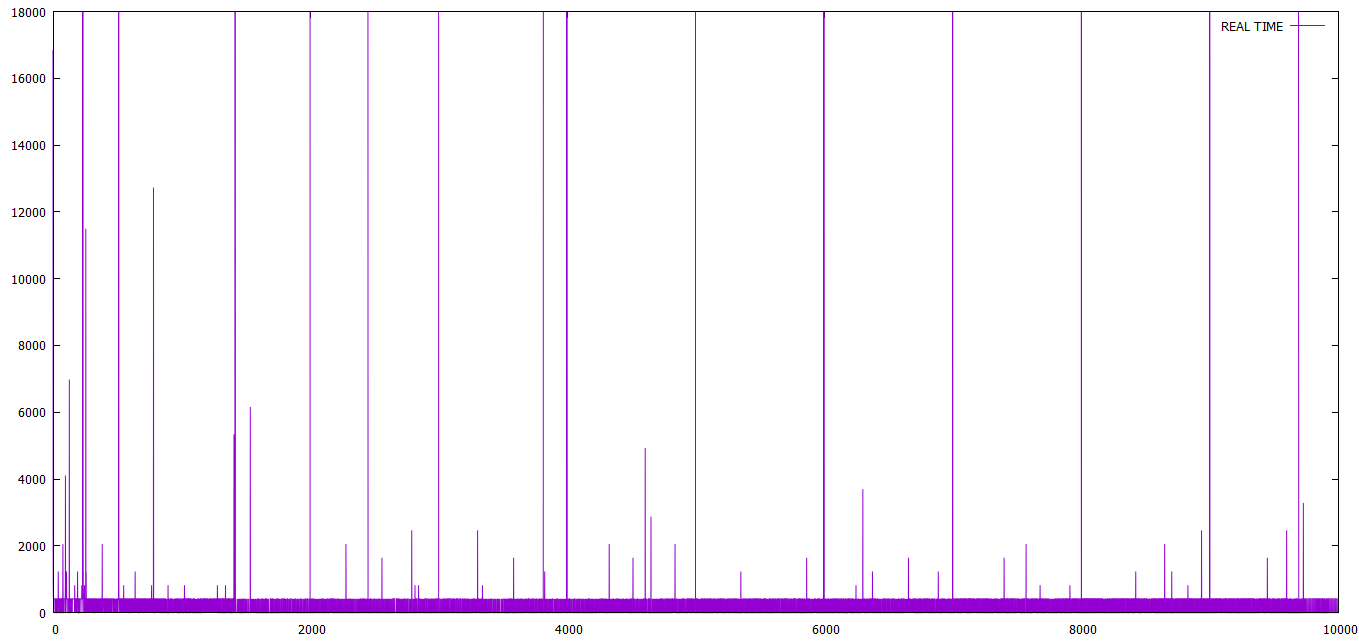
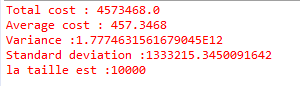
Parfois certains plot sont zoomé car les résultats sont plus claires et parlant comme ça.

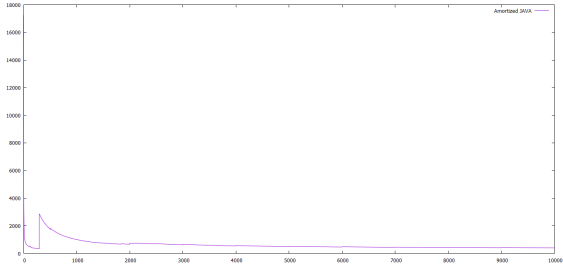






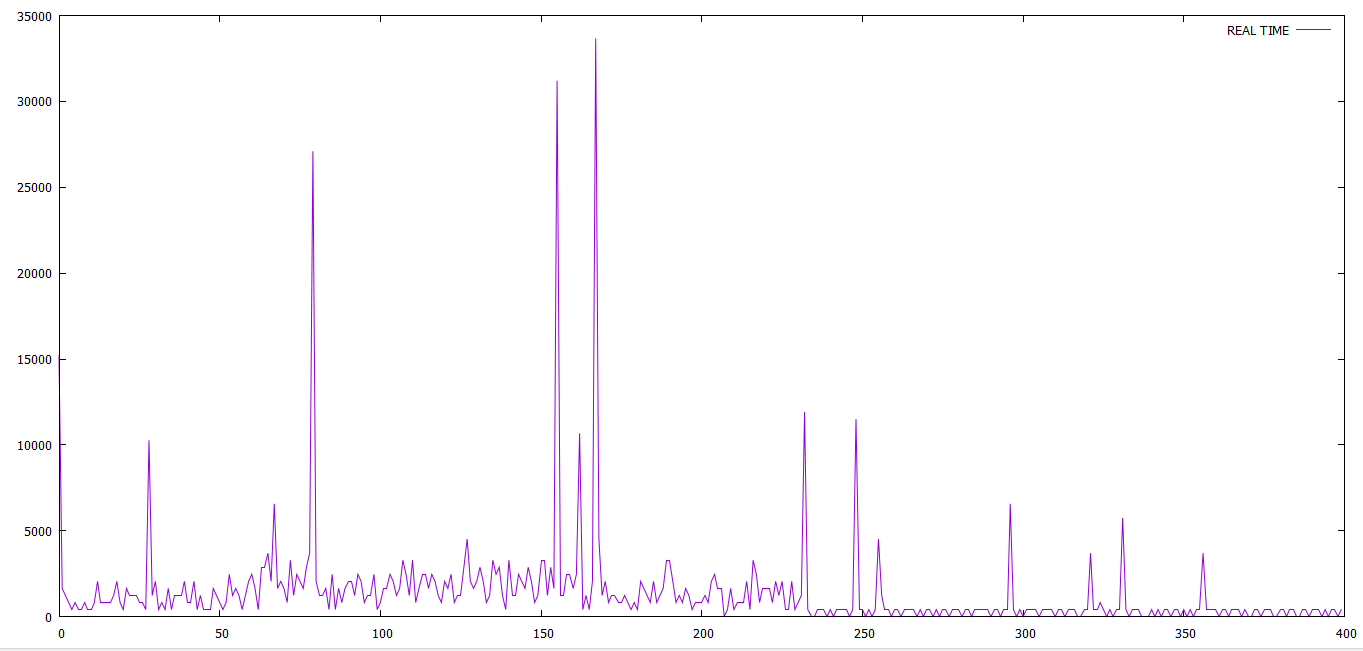
Croissant :



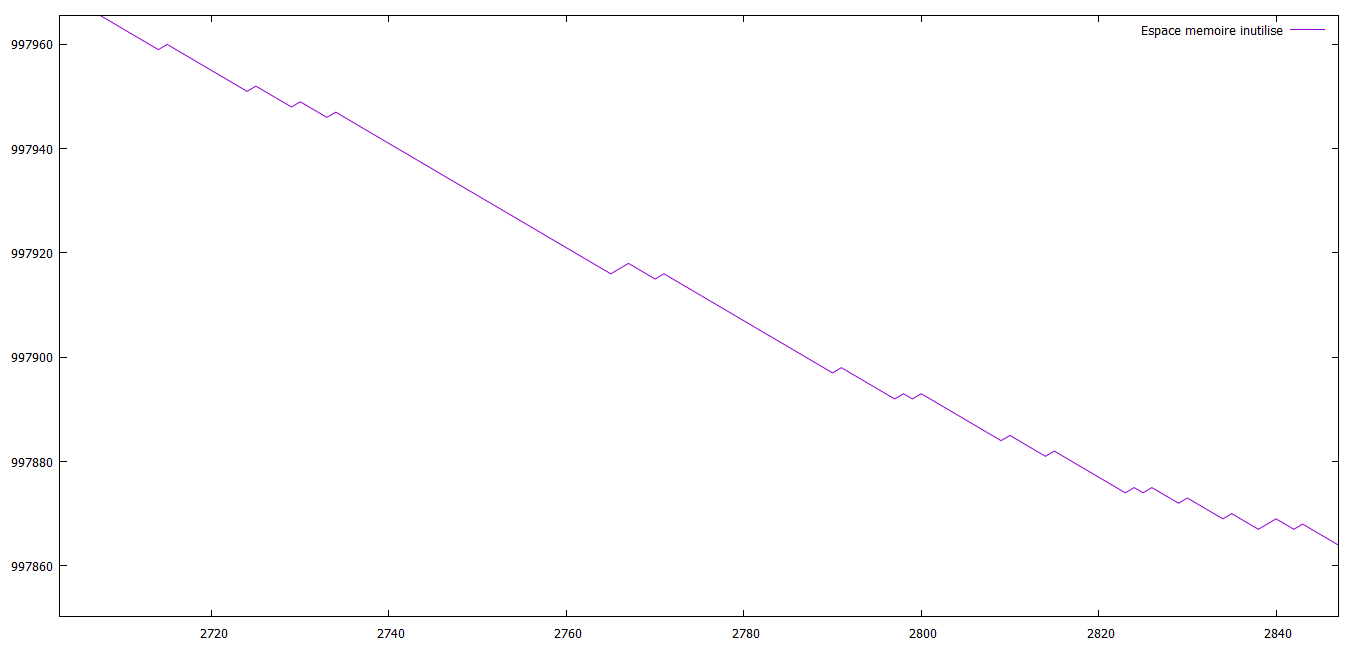
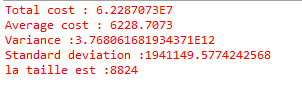
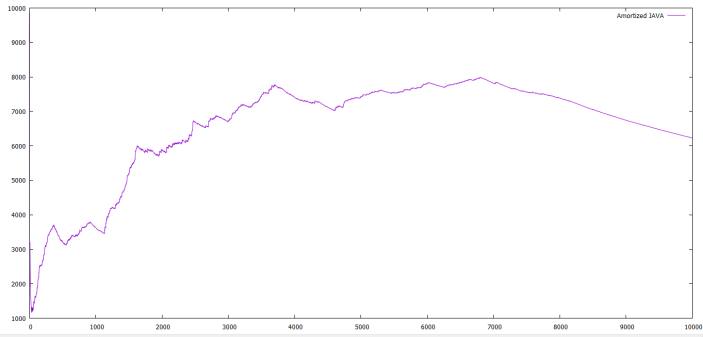


Ajout et suppression

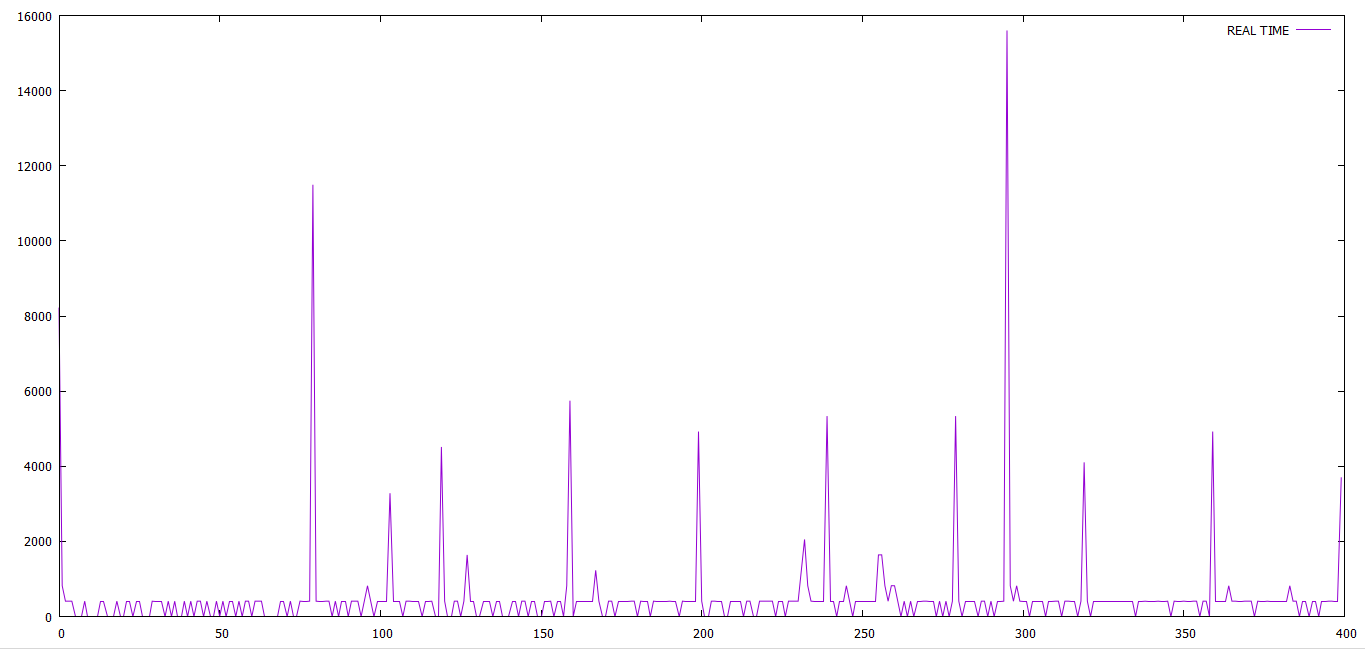
Random :



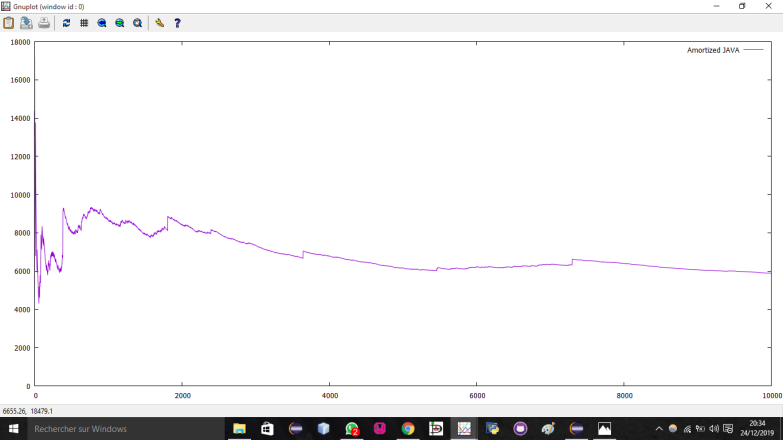
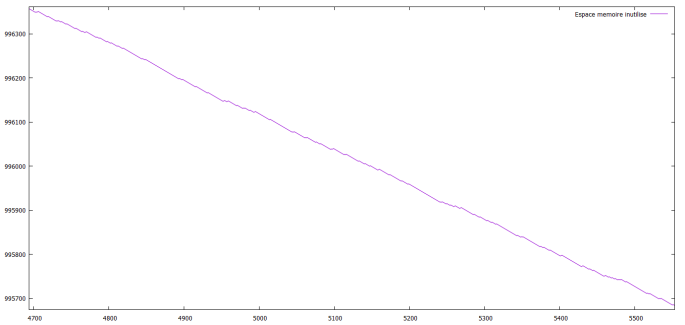
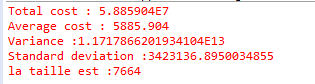
Real time est de plus prés



Croissant :



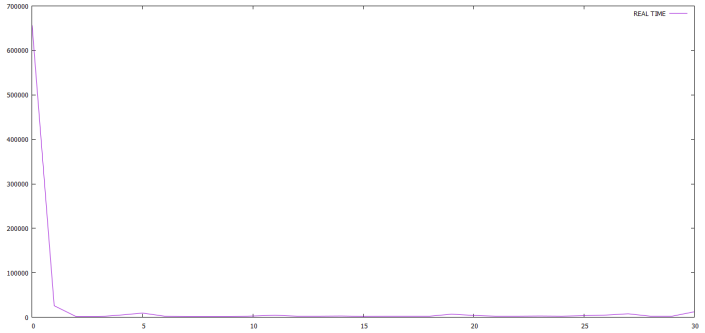
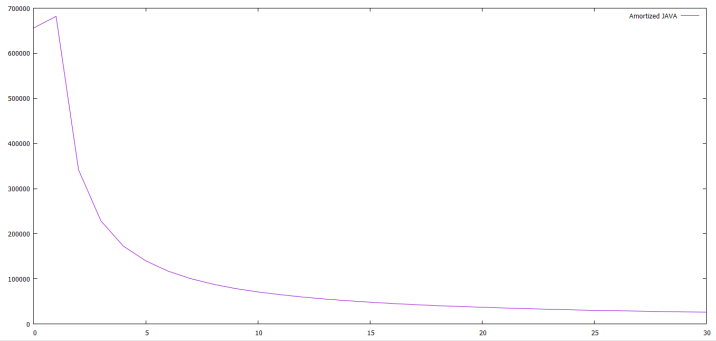
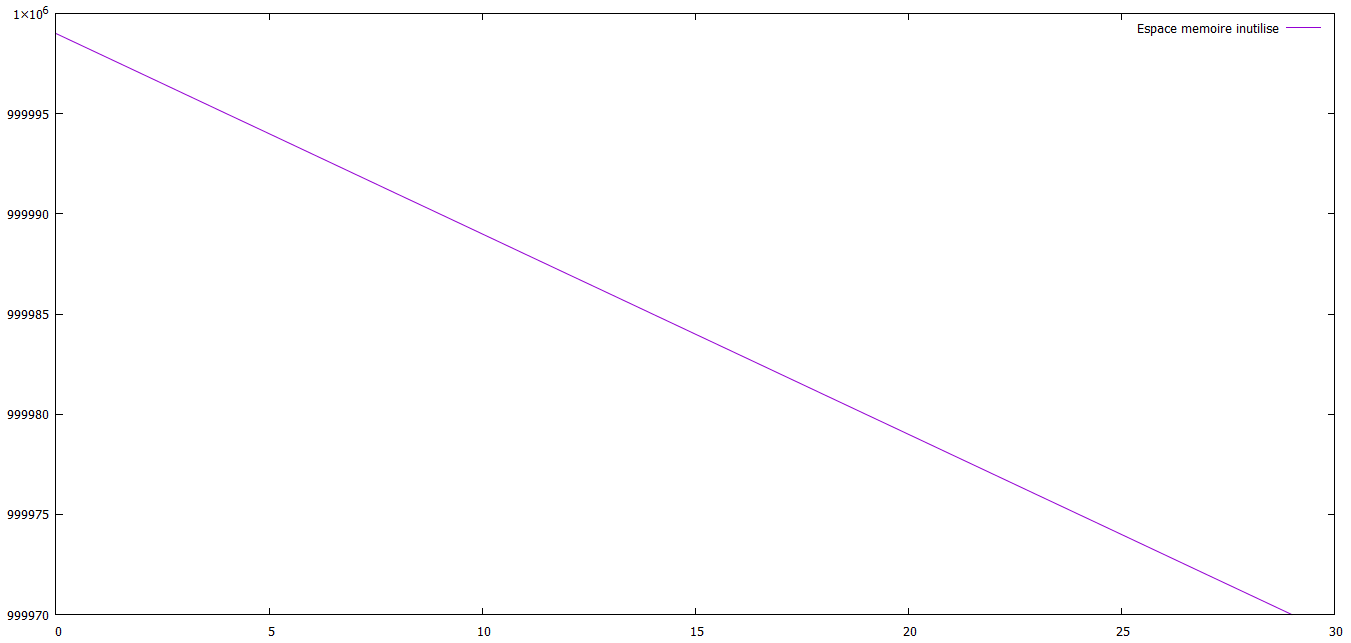
Real time est de plus prés

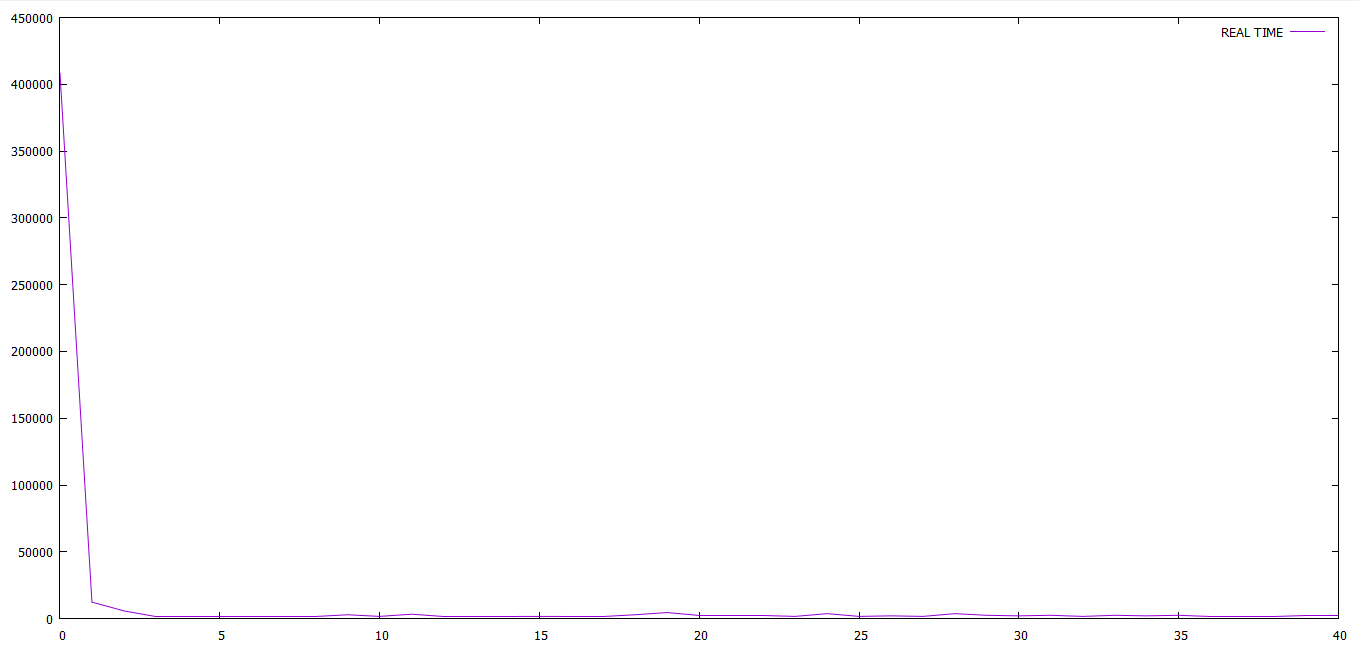
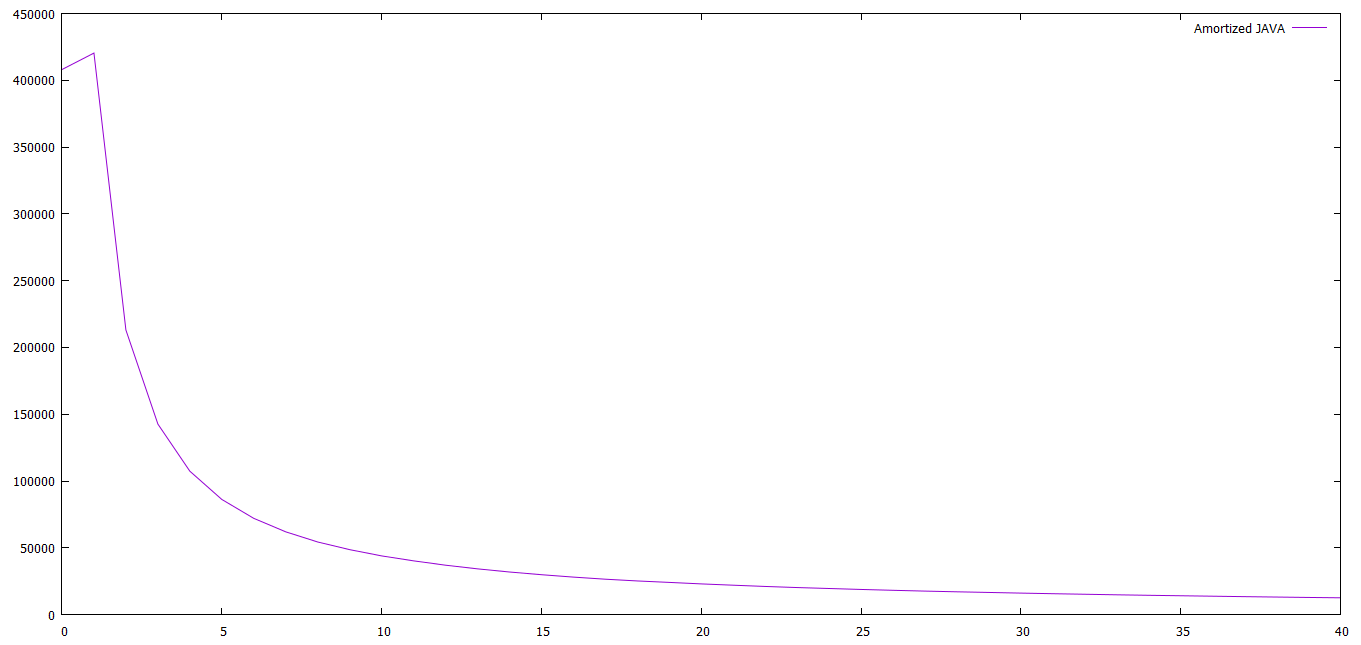


**AVL-ARBRES :**

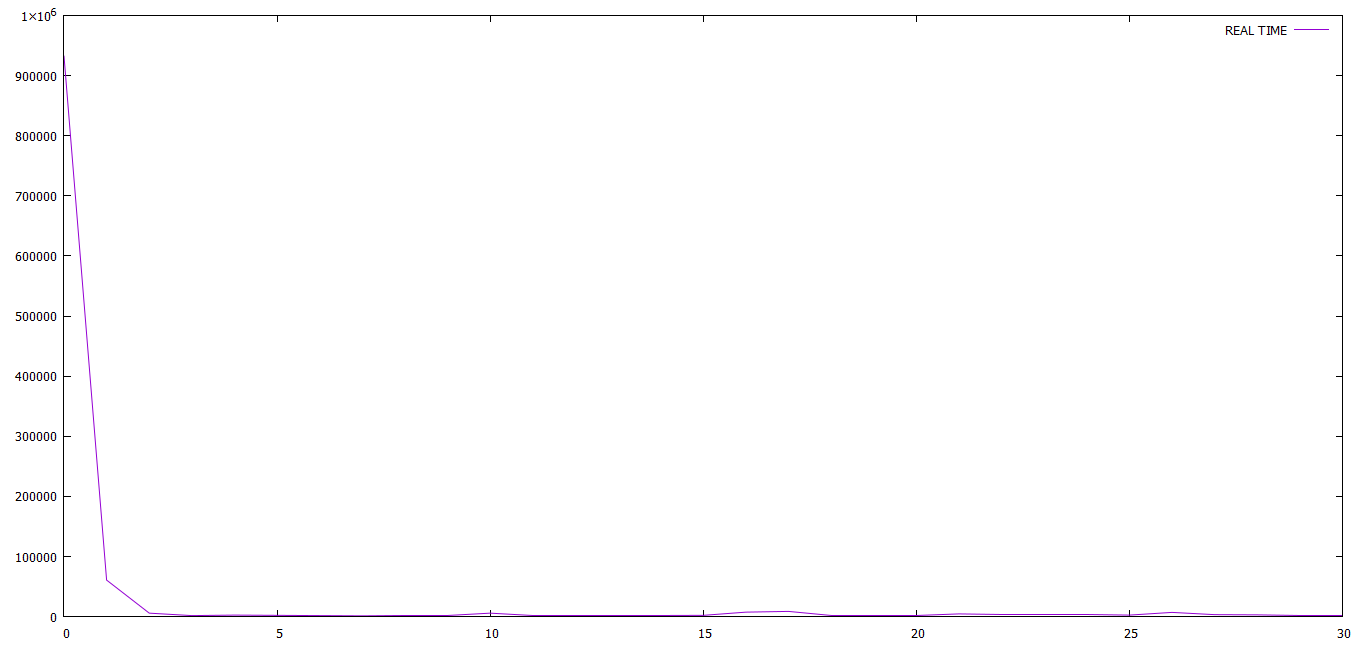
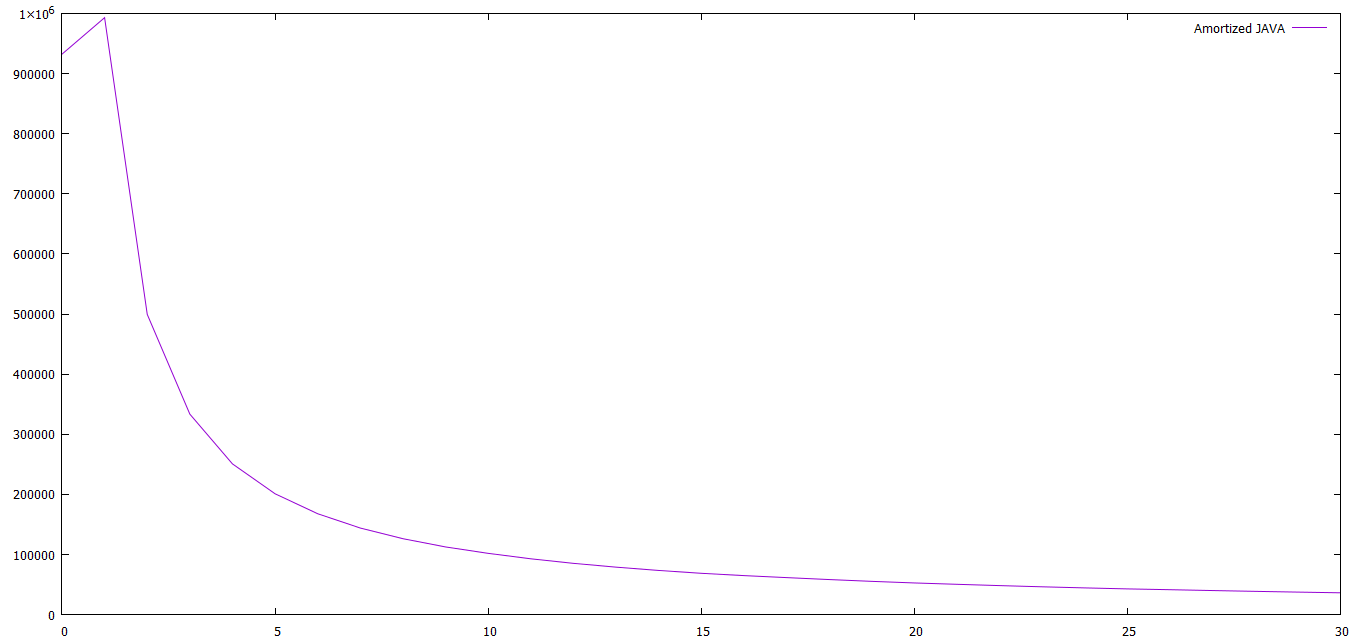
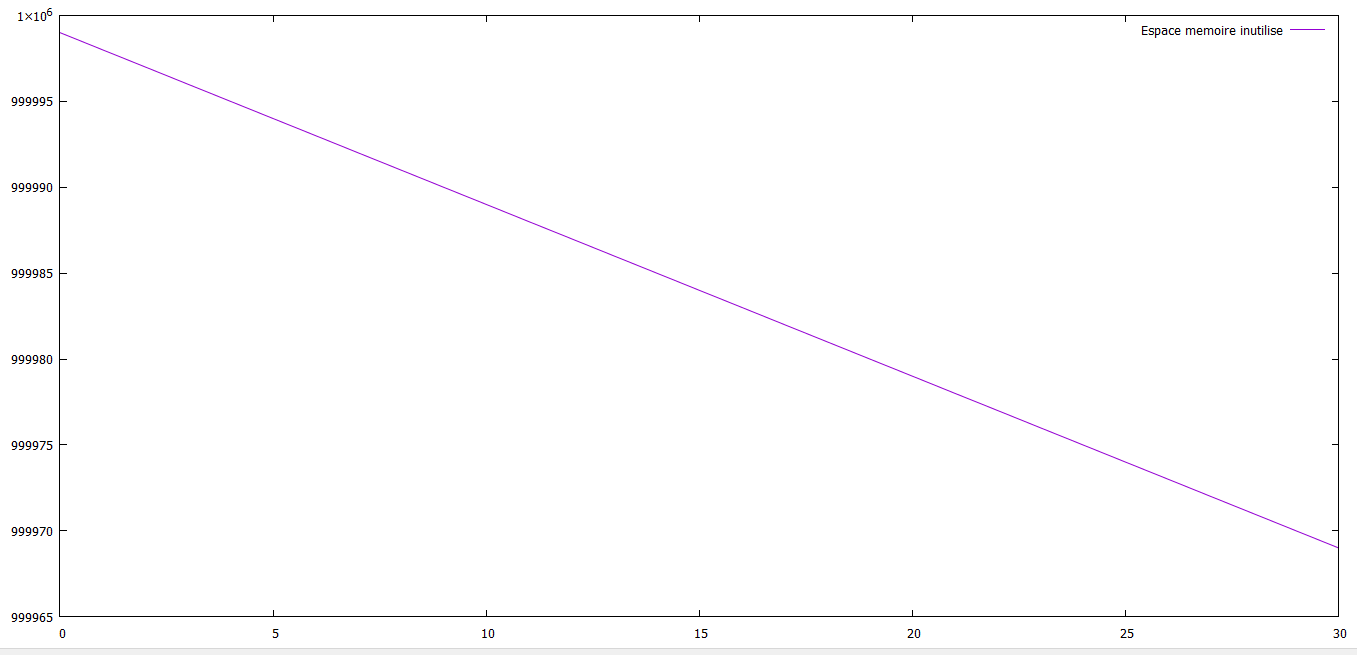
Ajout seulement :

Random :



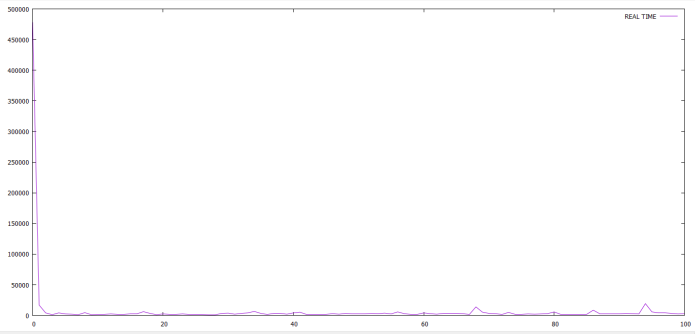
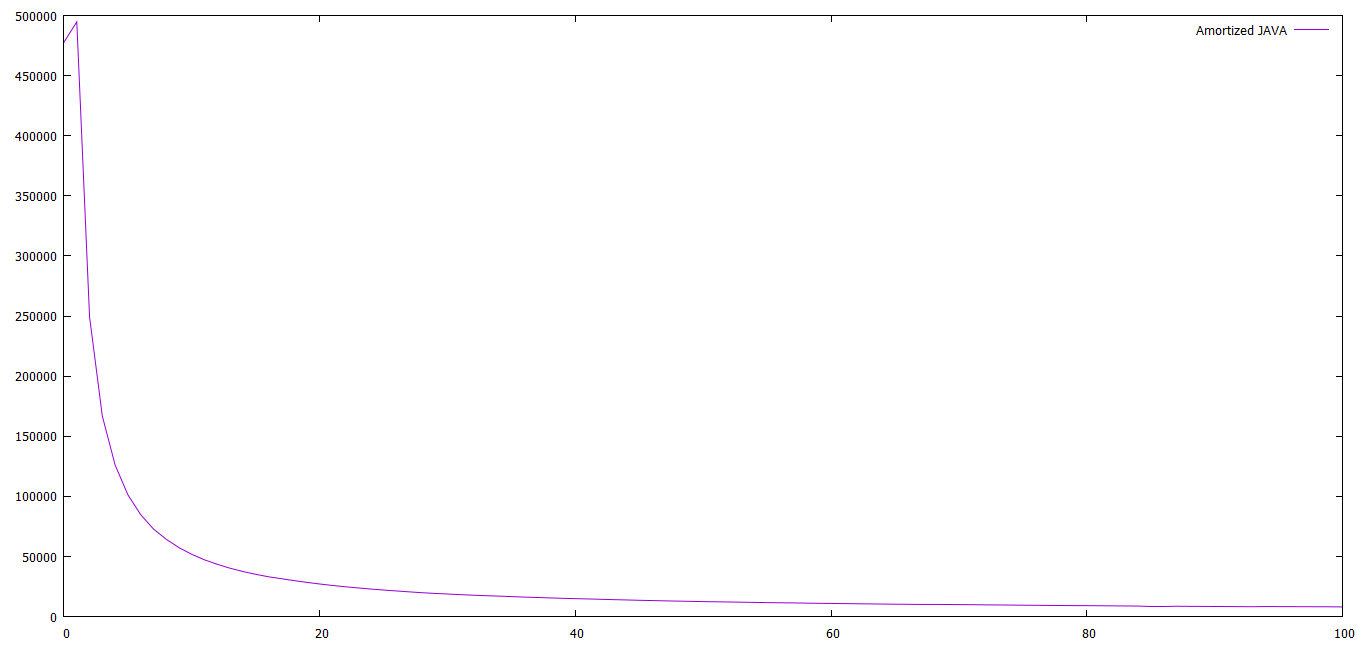
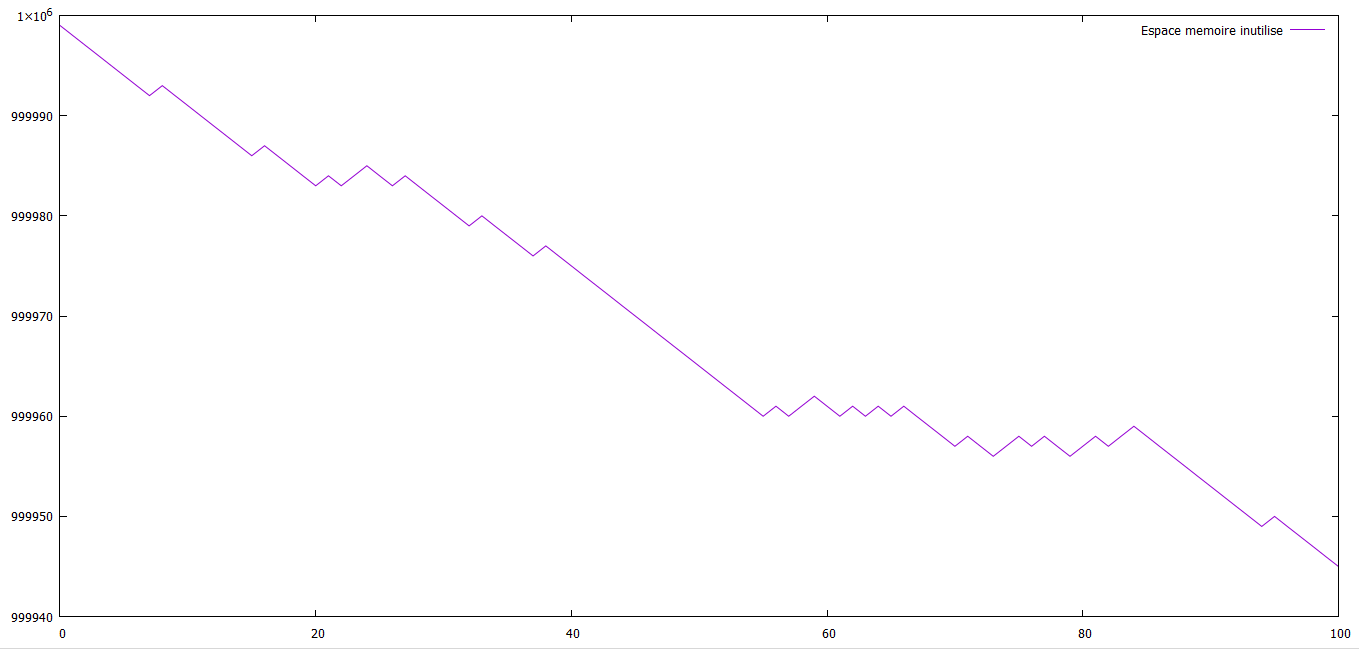


Croissant :

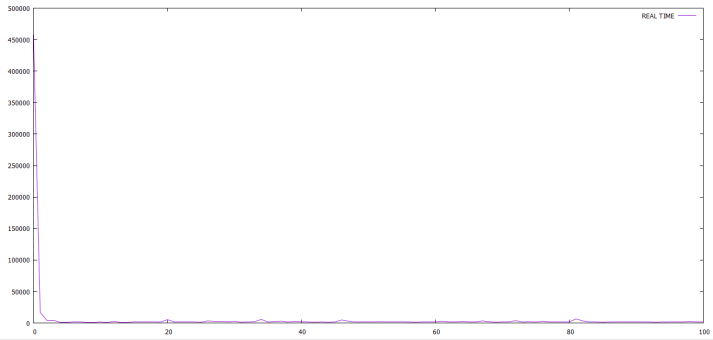
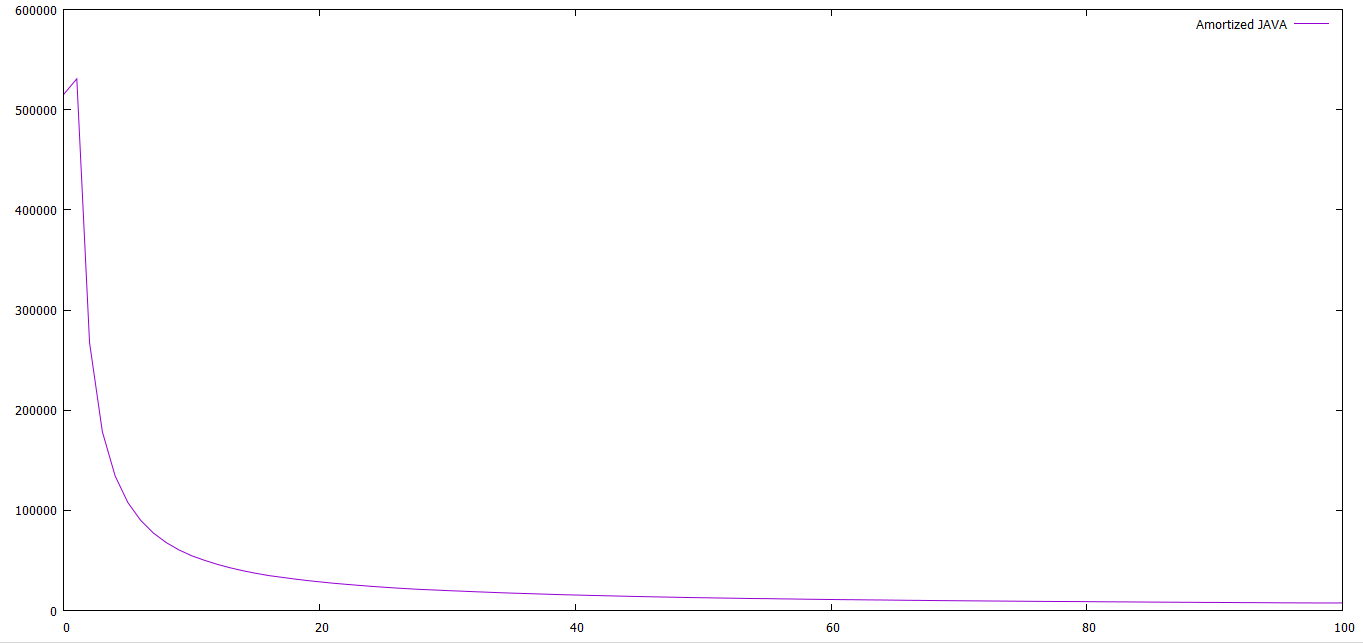
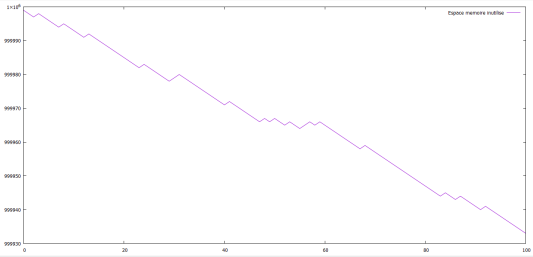


Ajout et suppression :

Random :



Croissant :



Conclusion :

Barbre :

Lors de l’ajout seulement on voit que l’espace mémoire non utilisé et le même partout car on fait que des ajouts il n’ya pas de suppression, l’espace mémoire est strictement décroissant.

Lors de l’ajout et la suppression on voit que l’espace mémoire inutilisé varie beaucoup car les insertions le réduisent et les suppressions l’augmentent.

Lors de l’opération ajout croissant les opérations sont régulières, elles se répètent après chaque laps de temps passé car il ya des ajouts jusqu'à ce que l’ordre t soit atteint puis des split au même moment étant donnée que l’ajout se fait toujours du même coté et le split se produit au même moment.

Lors de ajout random des fois plusieurs split sont fait a des moment proches (car des valeurs proches s’ajoutent au même endroit et donc la taille max du nœud est atteinte) d’autres fois non (on voit qu’il ya a des moment des résultat meilleur dans le random et d’autres fois des résultats moins bon par rapport a l’ajout croissant), je pense que c’est du au split qui est fait différemment a différents endroits.

Dans le plot les valeurs qui montent très haut sont des split qui remontent jusqu’à la racine.

Pour l’ajout et suppression on ne peut pas réellement comparer entre les plots car on les obtient par rapport a la probabilité p qui a chaque exécution est différente et en plus tout dépend du p qu’on a prit.

Ce qu’on peut dire c’est que la suppression et l’opération inverse de l’ajout au lieu d’effectuer et des split on joint les deux parties donc on peut en conclure que le cout lors de l’ajout croissant est moindre par rapport à l’ajout random.

AVLarbre :

On remarque que quand c’est un ajout croissant on fait plus souvent des rotations que quand c’est un ajout random car en ajoutant des valeurs croissantes on ajoute les éléments dans le sous arbre gauche et donc a chaque fois il faut refaire un équilibrage des sous arbres contrairement a si on ajoute des valeurs random l’ajout se fait aléatoirement et donc moins de rotations des sous arbres gauches et droits (contrainte : différence entre sous arbre gauche et droit = -1,0 ou 1)

L’auto équilibrage du b arbre est plus complexe que l’auto équilibrage du avl arbre qui consiste uniquement à faire des rotations, les arbres B utilisent un facteur de remplissage (la quantité du nœud est plein), puis divisent ou fusionnent les nœuds adjacents en conséquence.

Entre AVL et B arbres :

Plus la hauteur de l’arbre est courte plus cela minimise le nombre d’accès au disque qui sont couteux, plus t (ordre) est grand plus large sera l’arbre donc on conclut que les b arbre sont plus adéquat pour les accès en disque car ils sont moins long, les arbres AVL sont meilleurs en accès mémoire et les bases de données où des récupérations plus rapides sont nécessaires.

Les arbres AVL fournissent des recherches plus rapides que les b arbres car ils sont plus strictement équilibrés.

Les b arbres offrent des opérations d'insertion et de suppression plus rapides que les arbres AVL car moins de rotations sont effectuées (dans le cas ou on a de nombreuses insertions et suppressions).

Les arbres AVL sont mieux quand il y a peu d'opérations d'insertion et de suppression + le temps de recherche court est nécessaire + les données d'entrée sont triées ou presque triées.

Arbres AVL ne sont pas conçus pour contenir de grandes quantités de données.

Efficacité des operations selon t(ordre) :

Plus t (orde) est grand plus la recherche est couteuse car on doit parcourir tous les elements de chaque nœud.

Plus t est grand plus insertion+ délétion est meilleure car il y’a moins de split. (+ de split= + de temps).