**Mesures de temps d’exécution d’opérations sur différentes structures de données avec Python**

*BONNAIRE Lucas & RENAULT Rémi*

*16/12/2024*

Toutes les mesures ont été prises sur la base de scripts Python exécutés sur un MacBook Pro M4 Pro avec 24Go de RAM.

Le code source utilisé pour ces mesures est disponible sur ce repository GitHub :

<https://github.com/CORT1N/esgi-algorithmic-works>

Toutes les mesures ont été moyennées sur la base de 100 exécutions.

Table des matières

[1. Le tas 3](#_Toc185287893)

[2. La file de priorité 4](#_Toc185287894)

[3. La pile 6](#_Toc185287895)

[4. La file 7](#_Toc185287896)

[5. La liste chaînée 8](#_Toc185287897)

[Interprétation des résultats 10](#_Toc185287898)

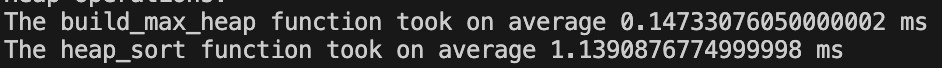
# Le tas

Un tas (ou heap) est une structure de données arborescente qui respecte la propriété d’un ordre partiel, où pour un tas max, chaque parent est supérieur ou égal à ses enfants, et pour un tas min, chaque parent est inférieur ou égal à ses enfants.

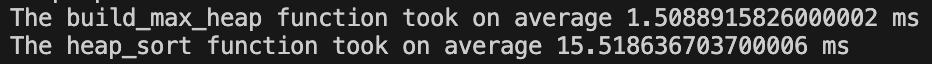
Pour 100 valeurs entre 0 et 9 :



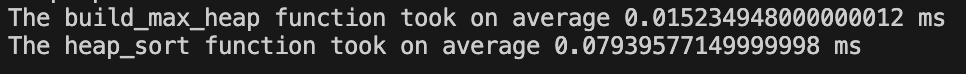
Pour 1000 valeurs entre 0et 9 :



Pour 10000 valeurs entre 0 et 9 :



Pour 100 valeurs entre 0 et 99 :



Pour 1000 valeurs entre 0 et 99 :



Pour 10000 valeurs entre 0 et 99 :



Pour 100 valeurs entre 0 et 999 :



Pour 1000 valeurs entre 0 et 999 :



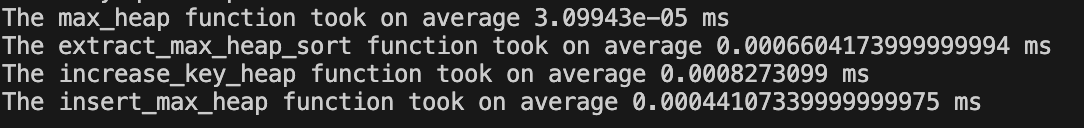
Pour 10000 valeurs entre 0 et 999 :



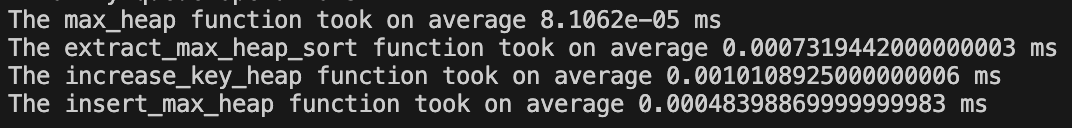
# La file de priorité

Une file de priorité est une structure de données dans laquelle chaque élément possède une priorité, et les éléments sont extraits en fonction de leur priorité (le plus élevé en priorité est extrait en premier), généralement implémentée à l’aide d’un tas.

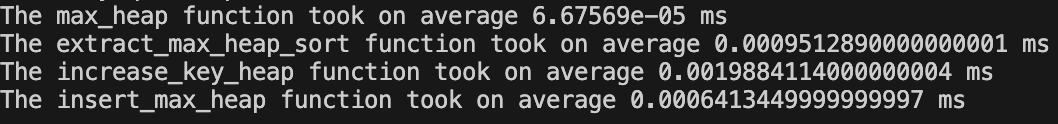
Pour 100 valeurs entre 0 et 9 :



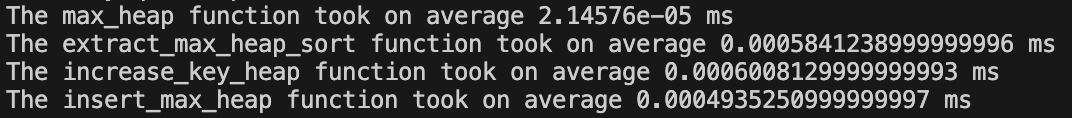
Pour 1000 valeurs entre 0et 9 :



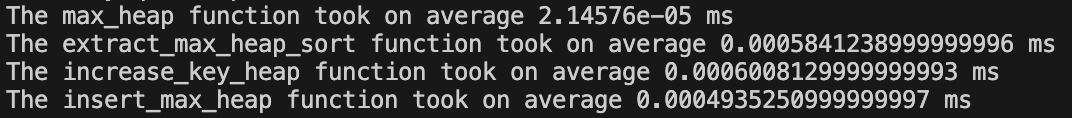
Pour 10000 valeurs entre 0 et 9 :



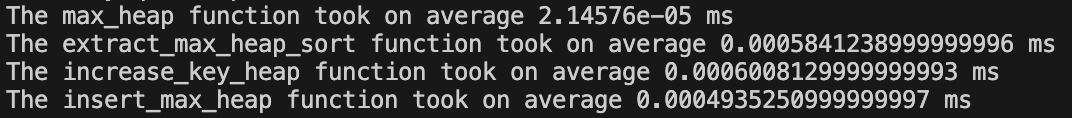
Pour 100 valeurs entre 0 et 99 :



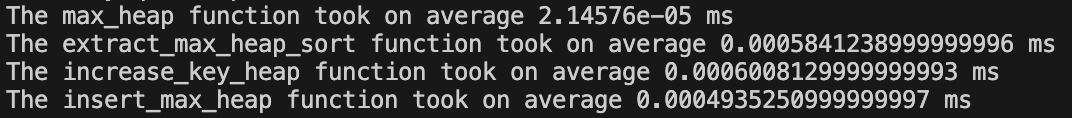
Pour 1000 valeurs entre 0 et 99 :



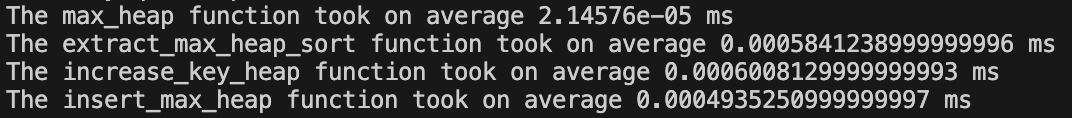
Pour 10000 valeurs entre 0 et 99 :



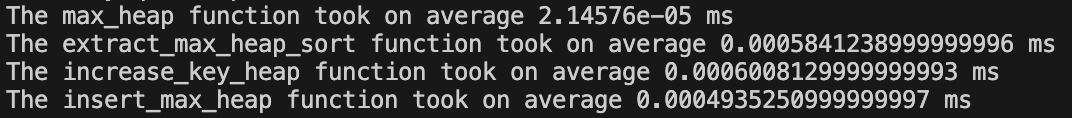
Pour 100 valeurs entre 0 et 999 :



Pour 1000 valeurs entre 0 et 999 :



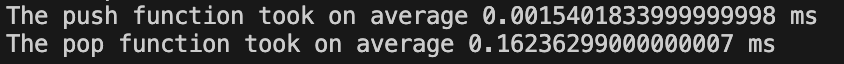
Pour 10000 valeurs entre 0 et 999 :



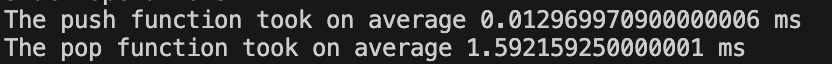
# La pile

Une pile (ou stack) est une structure de données qui suit le principe “Last In, First Out” (LIFO), où le dernier élément ajouté est le premier à être retiré.

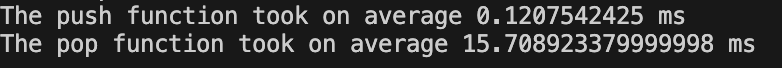
Pour 100 valeurs entre 0 et 9 :



Pour 1000 valeurs entre 0et 9 :



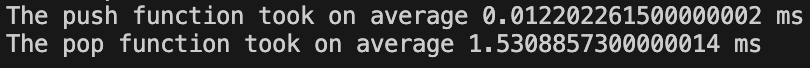
Pour 10000 valeurs entre 0 et 9 :



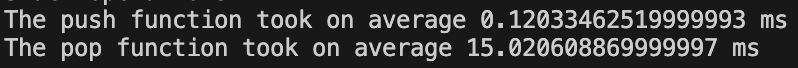
Pour 100 valeurs entre 0 et 99 :



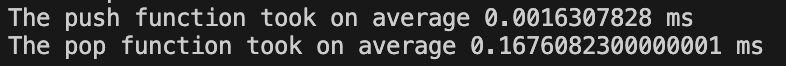
Pour 1000 valeurs entre 0 et 99 :



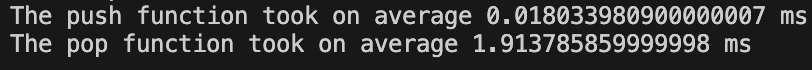
Pour 10000 valeurs entre 0 et 99 :



Pour 100 valeurs entre 0 et 999 :



Pour 1000 valeurs entre 0 et 999 :



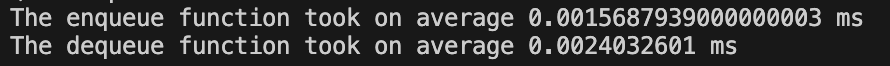
Pour 10000 valeurs entre 0 et 999 :



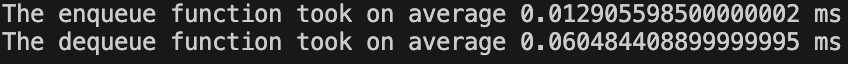
# La file

Une file (ou queue) est une structure de données qui suit le principe “First In, First Out” (FIFO), où le premier élément ajouté est le premier à être retiré.

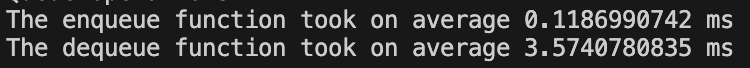
Pour 100 valeurs entre 0 et 9 :



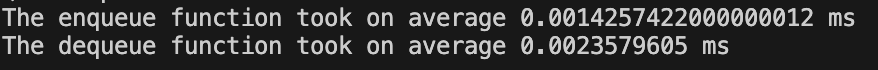
Pour 1000 valeurs entre 0et 9 :



Pour 10000 valeurs entre 0 et 9 :



Pour 100 valeurs entre 0 et 99 :



Pour 1000 valeurs entre 0 et 99 :



Pour 10000 valeurs entre 0 et 99 :



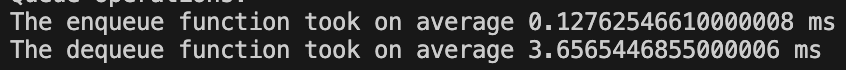
Pour 100 valeurs entre 0 et 999 :



Pour 1000 valeurs entre 0 et 999 :



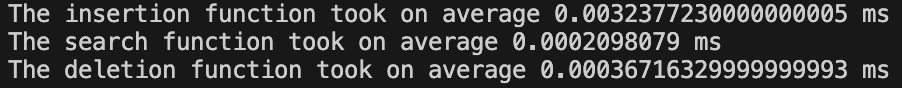
Pour 10000 valeurs entre 0 et 999 :



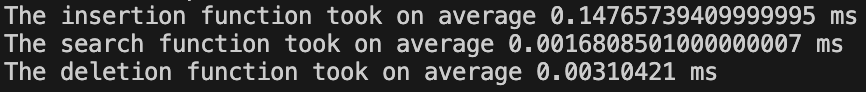
# La liste chaînée

Une liste chaînée est une structure de données composée d’une séquence d’éléments, appelés “nœuds”, où chaque nœud contient une valeur et une référence (ou lien) vers le nœud suivant, permettant une gestion dynamique de la mémoire.

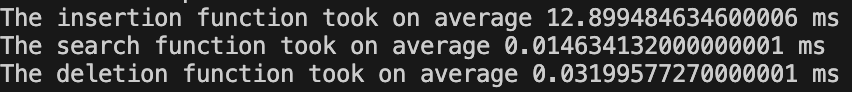
Pour 100 valeurs entre 0 et 9 :



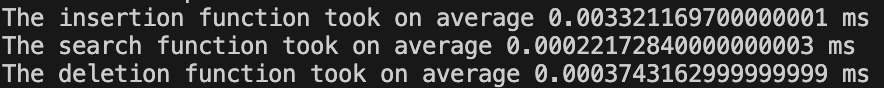
Pour 1000 valeurs entre 0et 9 :



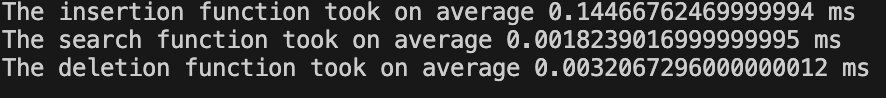
Pour 10000 valeurs entre 0 et 9 :



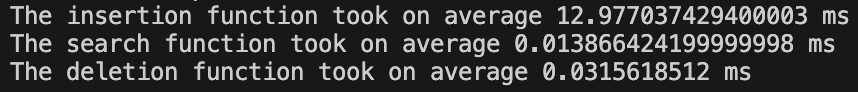
Pour 100 valeurs entre 0 et 99 :



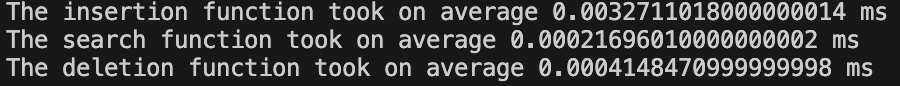
Pour 1000 valeurs entre 0 et 99 :



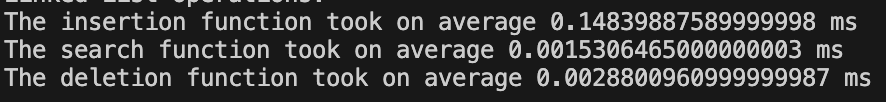
Pour 10000 valeurs entre 0 et 99 :



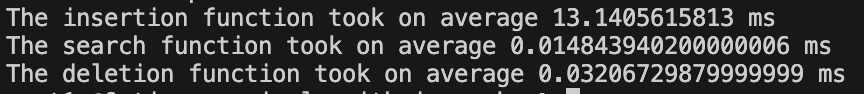
Pour 100 valeurs entre 0 et 999 :



Pour 1000 valeurs entre 0 et 999 :



Pour 10000 valeurs entre 0 et 999 :



# Interprétation des résultats

En commençant par le moins probant, le tas paraît très affecté durant son tri lorsqu’on atteint les 10000 valeurs.

Ensuite, la file de priorité semble résister à n’importe quelle quantité de valeurs.

La pile quant à elle ne semble être affectée qu’en suppression, dépassant les 10000 valeurs.

Au contraire, la file n’a pas l’air de souffrir des mêmes symptômes.

Enfin, pour la même quantité de valeurs, la liste chaînée devient plus lente en insertion.

Pour conclure, nous pourrions affirmer qu’une structure de données adaptée à tout serait la file ou la file de priorité, tandis que si la suppression n’est pas quelque chose de récurrent, la pile devient envisageable. Et enfin, si cette fois l’insertion n’est pas quelque chose de récurrent, c’est la liste chaînée qui le devient.