SAÉ S1.02

Comparaison d'approches algorithmiques

Berne Léa, Malosse Audran

Table des matières

[Description des tâches réalisées 1](#_Toc124084777)

[Tâche 1 : Vérifier 1](#_Toc124084778)

[Tâche 2 : Décrire 1](#_Toc124084779)

[Tâche 3 : Comparer 1](#_Toc124084780)

[Contribution individuelle au projet 2](#_Toc124084781)

# Description des tâches réalisées

## 1 : Vérifier

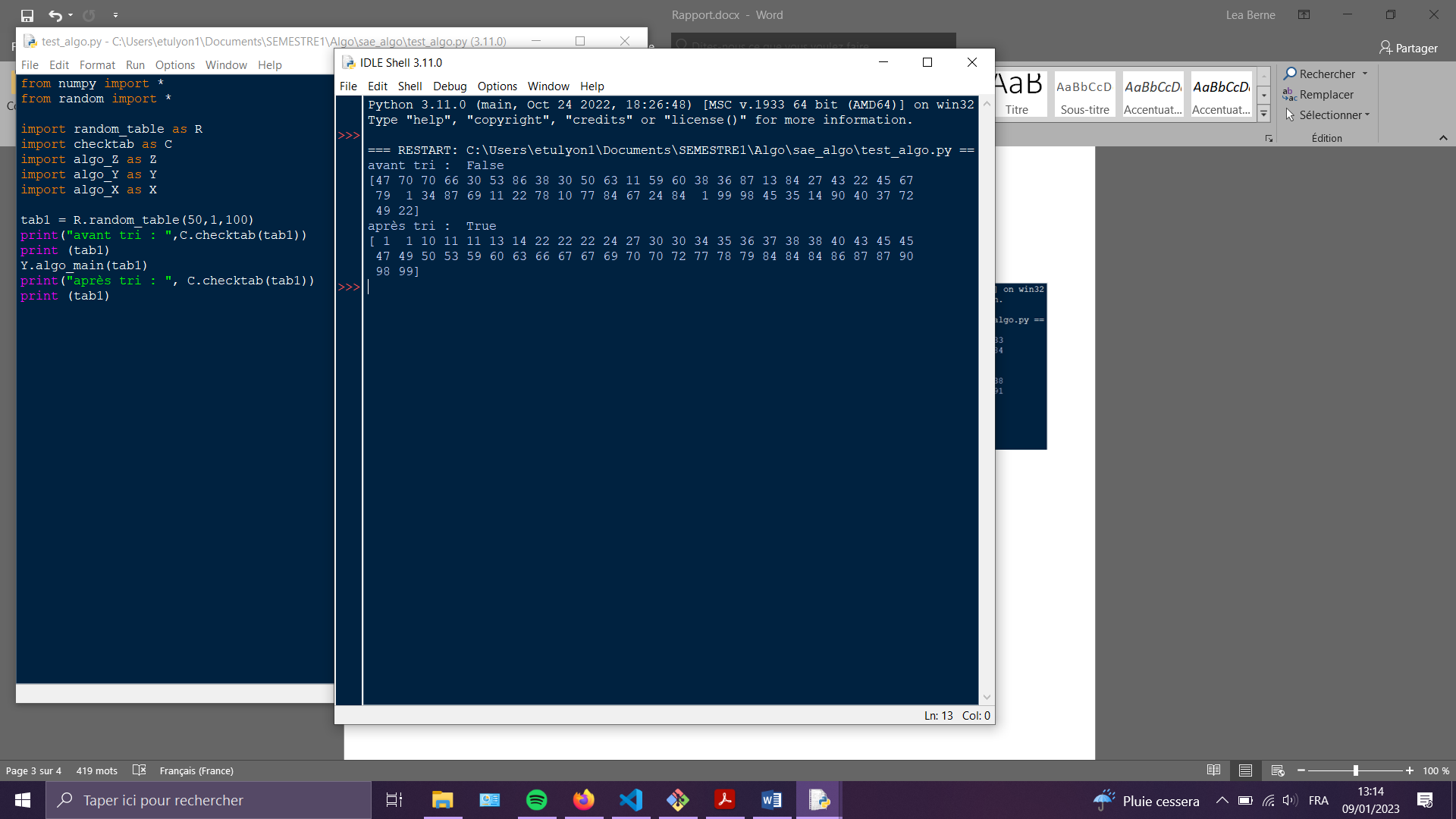
* 1. ALGO X

L’algorithme X respecte la post-condition (trie un tableau par ordre croissant).

### Analyse théorique

Trace + explications de comment il fonctionne + test algo avec capture d’écran

### Analyse pratique

* 1. ALGO Y

L’algorithme Y respecte la post-condition (trie un tableau par ordre croissant).

### Analyse théorique

### Analyse pratique

* 1. ALGO Z

L’algorithme Z ne respecte la post-condition (trie un tableau par ordre croissant).

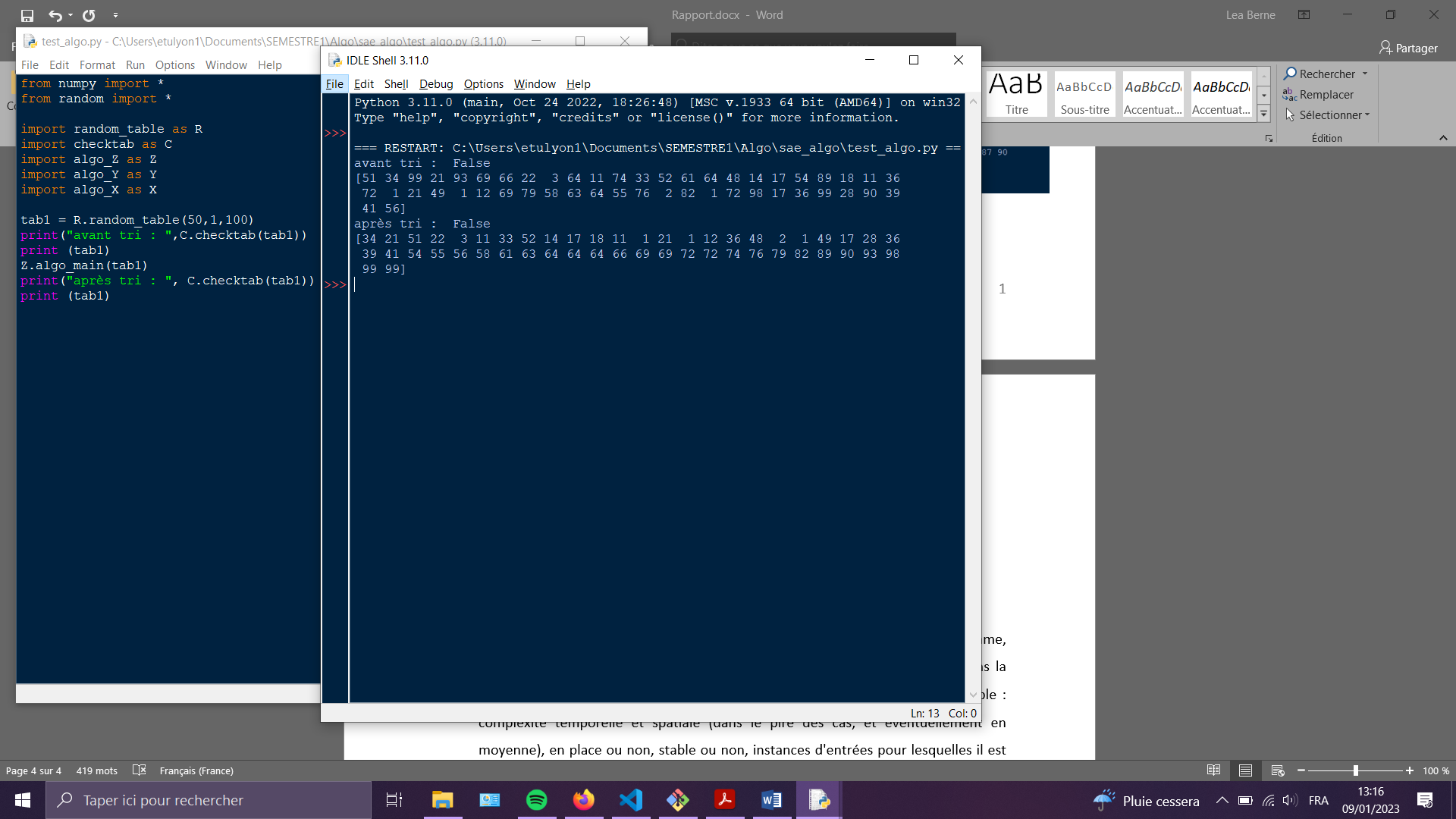
### Analyse théorique

La valeur d'indice 0 ne sera comparée à son voisin qu'une fois, celle d'indice 2 que deux fois etc. or l'algorithme fait remonter vers la droite la valeur la plus grande quand il compare deux valeurs.

A droite, il y aura la valeur la plus grosse trouvée dans la partie non triée, ce qui explique que le deuxième while s'arrête à len(tab)-i-1, car, avant qu'on incrémente i, on aura placé à droite la plus grosse valeur trouvée de toute la partie non triée du tableau.

Correction : mettre j à 0 pour reparcourir toute la partie du tableau qui n'est pas triée (la partie de gauche).

### Analyse pratique



## Tâche 2 : Décrire

* **Décrire** (avec des mots simples et/ou des schémas) le principe de chaque algorithme, en le rapprochant éventuellement d'un algorithme vu en cours ou trouvé dans la littérature. Vous devez également donner les propriétés de chacun. Par exemple : complexité temporelle et spatiale (dans le pire des cas, et éventuellement en moyenne), en place ou non, stable ou non, instances d'entrées pour lesquelles il est plus ou moins performant, etc.

## Tâche 3 : Comparer

* Comparer la performance pratique de ces trois algorithmes entre eux, en ce qui concerne le temps d'exécution. Vous pourrez par exemple évaluer celle-ci en fonction de la taille et/ou de la nature des entrées, et présenter les résultats sous forme de tableaux ou de graphiques que vous commenterez. Vous ferez également le lien avec vos conclusions de la tâche précédente concernant le temps d'exécution théorique.

## Tâche 4 : Ecrire ?

* Écrire votre propre algorithme de tri. Vous donnerez sa description, et le comparerez aux trois algorithmes reçus. Vous pourrez par exemple vous inspirer de l'un des algorithmes que vous avez reçu en essayant de l'améliorer, ou bien d'un autre algorithme : vu en cours, ou trouvé dans la littérature. Vous l'écrirez en Python, et vous n'utiliserez pas un algorithme de tri déjà fourni par une librairie. L'évaluation de cette tâche ne repose pas sur le fait de produire un algorithme "meilleur" que ceux que vous aurez reçus, mais surtout sur l'explication de votre solution et ses différences par rapport aux autres algorithmes.

# Contribution individuelle au projet