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 3](#_Toc124084779)

[Tâche 3 : Comparer 3](#_Toc124084780)

[Contribution individuelle au projet 4](#_Toc124084781)

# Description des tâches réalisées

## 1 : Vérifier

* 1. ALGO X

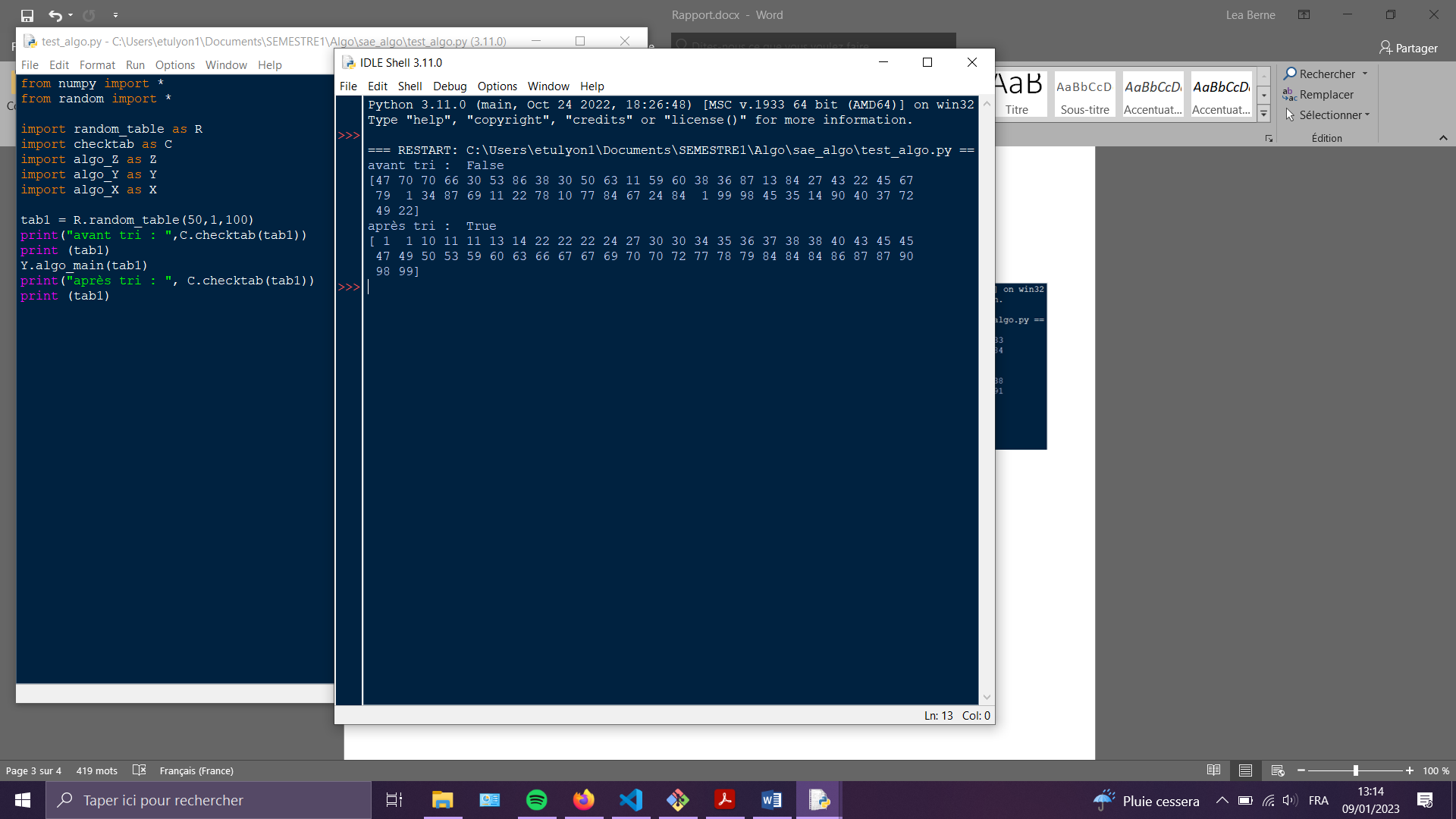
### Analyse théorique

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

La variable i est initialisé à 1. On va donc parcourir le tableau à partir de la seconde valeur de celui-ci. On affecte à la variable j la valeur de i-1, soit l’indice dans le tableau précédent celui de i. Dans la première boucle, on affecte la valeur à l’indice i dans une variable tmp. Puis on affecte la valeur du tableau à l’indice j à tab[j+1] tant que la valeur à indice j est supérieur à la variable tmp (précédemment affecté à la valeur d’indice i) et que j est supérieur ou égale à 0. Et on décrémenter j de 1. Cela a pour effet de faire descendre la valeur à trier dans le tableau pour qu’elle se retrouve à sa place. En sortie de boucle, il faut donc replacer la valeur stockée dans tmp à sa nouvelle place qui est l’indice j+1. On incrémente i et on recommence.

Il s’agit d’un algorithme de type tri insertion.

### Analyse pratique

* 1. ALGO Y

### Analyse théorique

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

Il s’agit en fait d’un tri

### Analyse pratique

* 1. ALGO Z

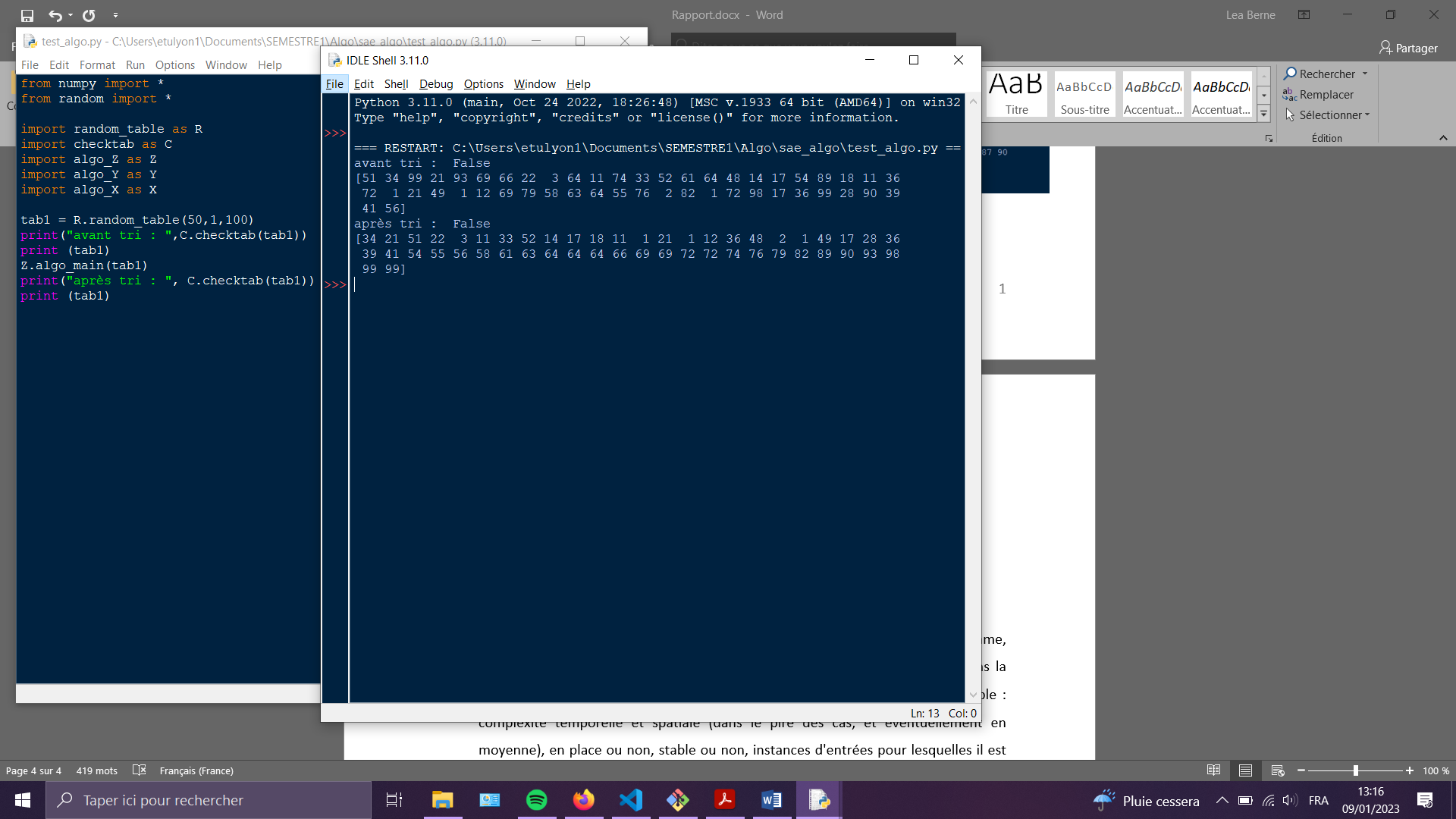
### Analyse théorique

L’algorithme Z ne respecte la post-condition (trie un tableau par ordre croissant). En réalité, l’algorithme compare deux valeurs voisines dans le tableau (celle d’indice j et celle d’indice j+1) et les inverse si la valeur de tab[j] est plus grande que celle de tab[j+1]. Ainsi, à la fin du premier tour de la première boucle while, la plus grande valeur du tableau parcouru aura remonté à droite, et donc la partie de droite sera la partie triée du tableau, ce qui explique que le deuxième while s’arrête à len(tab)-1-i.

Le problème, c’est qu’à la fin du premier tour de la deuxième boucle while, i est incrémenté, or la valeur de j dépend de celle de i. Comme j est initialisé à i, la valeur d’indice 0 ne sera comparée à son voisin de droite qu'une seule fois au court de l’exécution, celle d'indice 1 que deux fois etc. Pour généraliser, la valeur d’indice n ne sera comparée à son voisin de droite que n+1 fois, et s’il se trouve que la valeur d’indice n+1 est plus grande que lui mais que la suivante (n+2) est plus petite, alors la valeur d’indice n restera où elle est malgré tout. La partie de gauche ne sera donc pas triée.

Pour corriger cela, il faut initialiser j à 0 pour reparcourir toute la partie gauche du tableau, puisqu’elle n'est pas triée. Il s’agit en fait d’un tri à bulle.

### 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.

/ !\ 🡪 penser à ajouter des captures pour temps min & temps max + moyenne

## 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