Algorithmes de Tri

Introduction

Les *algorithmes de tri* sont des algorithmes omniprésents en informatique : ils sont utilisés comme étape intermédiaire de nombreux calculs, mais aussi dès lors qu'une application souhaite présenter des données dans un ordre précis. Par exemple :

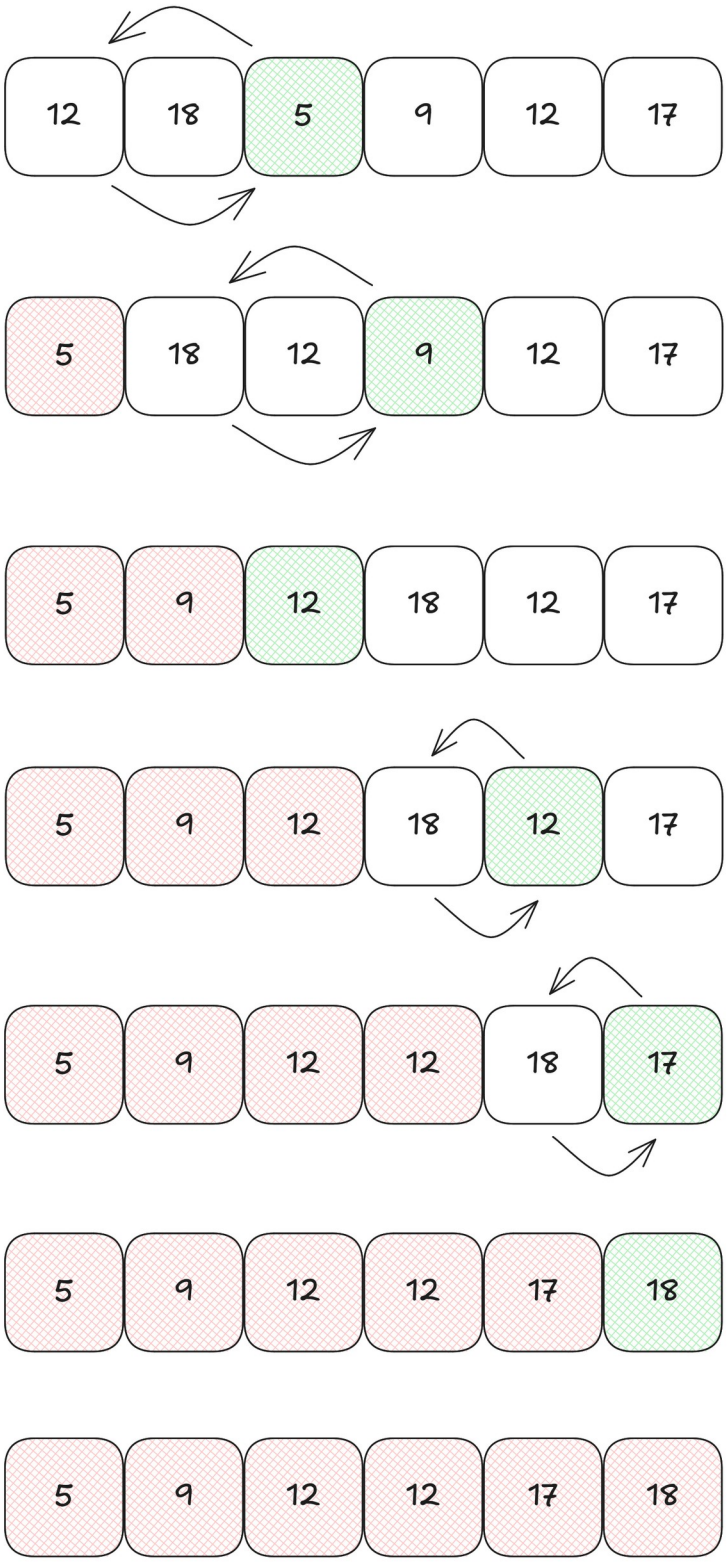
- Un tableur qui présente des élèves dans l'ordre alphabétique ;
- Un fil d'abonnement qui présente des publications dans l'ordre chronologique ;
- Un jeu de cartes qui les présente dans l'ordre de leur valeur ;
- ...

1. Vous disposez de cartes numérotées. En n'en regardant que 2 à tout moment, triez-les et essayez de réfléchir à la manière dont vous y prenez. Sur une feuille, écrivez un algorithme permettant de refaire cette opération.

2. Échangez votre feuille avec un autre groupe, et suivez leurs instructions. Sont-elles suffisamment précises ? Une fois que leur algorithme est fini, vos cartes sont-elles triées ?

Tri par Sélection

On va maintenant trier des listes d'entiers avec Python. Le concept du *Tri par Sélection* est simple : à tout moment, on cherche le plus petit élément de la liste qu'on n'a pas encore traîté, et on le place au début de la liste.



3. Reproduire ce concept à la main sur la liste suivante, en précisant les étapes intermédiaires : `[59, 75, 35, 6, 13, 23]`.

4. Écrire une fonction `echange(li, n, p)` qui échange les éléments d'indice `n` et `p` de la liste `li`.

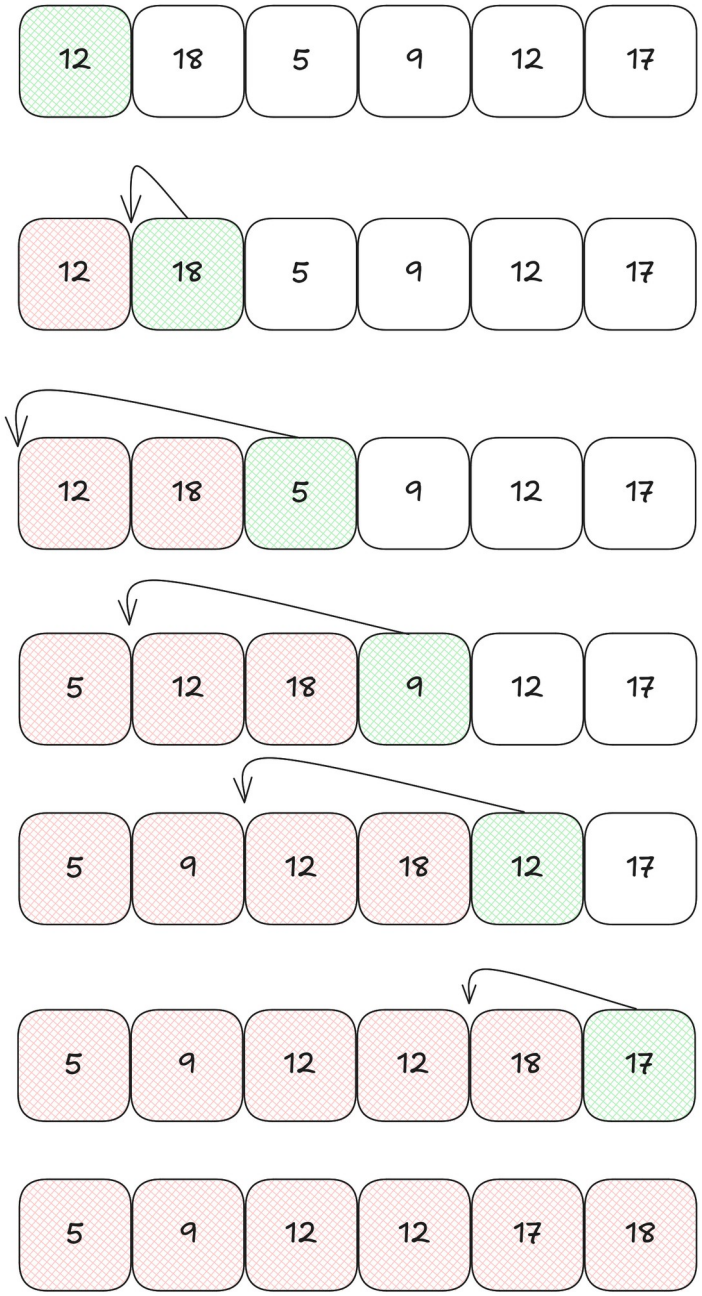
5. Écrire une fonction `minimum_apres_n(li, n)` qui renvoie l'indice du plus petit élément de la liste `li` situé après l'indice `n`.

6. Écrire une fonction `tri_selection(li)` qui trie la liste `li` dans l'ordre croissant en utilisant l'algorithme du Tri par Sélection.

7. Quelle est la complexité de cette fonction ?

Tri par Insertion

Le *Tri par Insertion* est un peu différent du Tri par Sélection : plutôt que de sélectionner les éléments dans l'ordre croissant pour les mettre à leur place, on va les prendre dans l'ordre où ils viennent et on va les insérer à une place cohérente par rapport à tous ceux qui ont été traîtés jusqu'à présent.



8. Reproduire ce concept à la main sur la liste suivante, en précisant les étapes intermédiaires : `[59, 75, 35, 6, 13, 23]`.

9. Traduire le pseudo-code suivant en Python (n'hésitez pas à réutiliser la fonction `echange`).

```
VARIABLES :
t : tableau d'entiers
i : nombre entier
ind : nombre entier
DEBUT
POUR i allant de 2 à taille(t)
  ind = i
  TANT QUE ind > 1 et t[ind] > t[ind - 1]
    echanger t[ind] et t[ind - 1]
    ind = ind - 1
FIN
```

10. Quelle est la complexité de cette fonction ?