Les algorithmes de tri

1. Introduction

Qu'est-ce qu'un algorithme de tri ?

Un algorithme de tri est un algorithme qui permet de trier une liste d'éléments. Par exemple, on peut trier une liste de nombres, une liste de mots, une liste de personnes, etc.

Pourquoi trier ?

Le tri permet de retrouver plus facilement un élément dans la liste et de gagner énormément de temps de recherche.

Comment trier ?

Il existe de nombreux algorithmes de tri. Chaque algorithme a ses avantages et ses inconvénients. Certains algorithmes sont plus adaptés pour trier des listes de nombres, d'autres pour trier des listes de mots, etc.

2. Activité

1. Prenez un jeu de cartes et triez-le en utilisant la méthode de votre choix.
2. Comparez votre méthode avec celle des autres groupes.
3. Remélangez le paquet et triez le selon l'algorithme suivant (du tri par selection):
    - On recherche le plus petit élément de la liste
    - On inverse sa place avec l'élement juste à la suite du dernier élément trié de la liste
    - On recommence jusqu'à ce que la fin de la liste soit atteinte

3. *Le tri par sélection*

Le tri par sélection est un algorithme de tri qui consiste à parcourir la liste à trier et à placer les éléments dans l'ordre croissant (ou décroissant).

- On recherche le premier élément de le liste (ici le plus petit)
- On inverse sa place avec le premier élément non-trié de la liste
- On recommence jusqu'à ce que la fin de la liste

> Exemple : On souhaite trier la liste suivante : ['e', 'b', 'd', 'c', 'a'] On recherche le plus petit élément de la liste : 'a' On le place en position 0 en l'inversant avec 'e' On recommence avec la liste ['a', 'b', 'd', 'c', 'e']

Etape	Liste à trier	Partie triée
0	[**'e', 'b', 'd', 'c', 'a'**]	[-, -, -, -, -]
1	['a', **'b', 'd', 'c', 'e'**]	['a', -, -, -, -]
2	['a', 'b', **'d', 'c', 'e'**]	['a', 'b', -, -, -]

Etape	Liste à trier	Partie triée
3	['a', 'b', 'c', **'d', 'e**']	['a', 'b', 'c', -, -]
4	['a', 'b', 'c', 'd', **'e**']	['a', 'b', 'c', 'd', -]
5	['a', 'b', 'c', 'd', 'e']	['a', 'b', 'c', 'd', 'e']

Cet algorithme est simple à comprendre et à mettre en oeuvre. Cependant, il est très lent. En effet, pour trier une liste de 10000 éléments, il faut effectuer 10000 recherches qui chacune demande de parcourir toute la liste. C'est beaucoup trop long !

Question 1

```
Quelle est la complexité de cet algorithme ?
```

Question 2

```
On va chercher à implémenter cette algorithme en Python. Pour cela on va utiliser
une fonction echanger qui permet d'échanger deux éléments d'une liste.
```

Question 3

```
Implémentez une fonction qui permette de trouver l'indice du plus petit élément
d'une liste à partir d'un indice donné.
```

Question 4

```
En utilisant les fonctions précédentes, implémentez l'algorithme de tri par
sélection. Vous pouvez vous appuyer sur l'algorithme écrit précédemment

et sur le code suivant :
```

```python
def tri_selection(liste):
    for i in range(len(liste)):
        # On cherche le plus petit élément de la liste
        # On l'échange avec le premier élément non-trié
```

Question 5

```
Testez votre algorithme sur une liste de 10000 éléments. Combien de temps met-il
pour trier la liste ?
```

4. *Le tri par insertion*

Le tri par insertion est un algorithme de tri qui consiste à parcourir la liste et à insérer chaque élément à sa place dans une nouvelle liste. Pour insérer un élément à sa place dans la nouvelle liste, on le compare avec les éléments déjà présents dans la nouvelle liste. On insère l'élément à sa place dès qu'on trouve un élément plus grand que lui.

- On parcourt la liste de ce qui n'est pas encore trié jusqu'à trouver un élément mal placé par rapport au précédent (ici un élément plus petit)
- Tant que l'élément est mal placé (plus petit que le précédent), on l'inverse avec le précédent
- On recommence jusqu'à la fin de la liste

> Exemple : On souhaite trier la liste suivante : [5, 2, 4, 3, 1]

Etape	Liste à trier	Liste triée
0	[**5, 2, 4, 3, 1**]	[5]
1	[5, **2, 4, 3, 1**]	[2, 5]
2	[2, 5, **4, 3, 1**]	[2, 4, 5]
3	[2, 4, 5, **3, 1**]	[2, 3, 4, 5]
4	[2, 3, 4, 5, **1**]	[1, 2, 3, 4, 5]

Question 6

```
Quelle est la complexité de cet algorithme ?
```

Question 7

```
Implémentez cet algorithme en Python.
```

5. *Autres algorithmes de tri*

Il existe de nombreux autres algorithmes de tri, le tri-fusion et le tri-rapide sont les plus connus. Ces algorithmes sont plus complexes à comprendre et à mettre en oeuvre. Cependant, ils sont beaucoup plus rapides que les algorithmes précédents. Vous les verrez plus en détail l'année prochaine.

6. Pour aller plus loin

Ces algorithmes fonctionnent pour trier des listes de nombres ou de mot car on sais comment les comparer. Mais comment trier une liste d'objets ? Par exemple, comment trier une liste de personnes ? Pour cela, il faut définir une fonction de comparaison qui permet de comparer deux personnes. On peut alors utiliser cette fonction de comparaison pour trier la liste de personnes.

Question bonus

```
Créez une classe Personne avec un attribut age et créez une liste de personnes que
vous remplissez avec des personnes de différents ages.

Implémentez une fonction de comparaison pour trier une liste de personnes par age.
Et adaptez un de vos algorithmes de tri pour trier cette liste de personnes.
```