

TD - Le tri fusion

Introduction

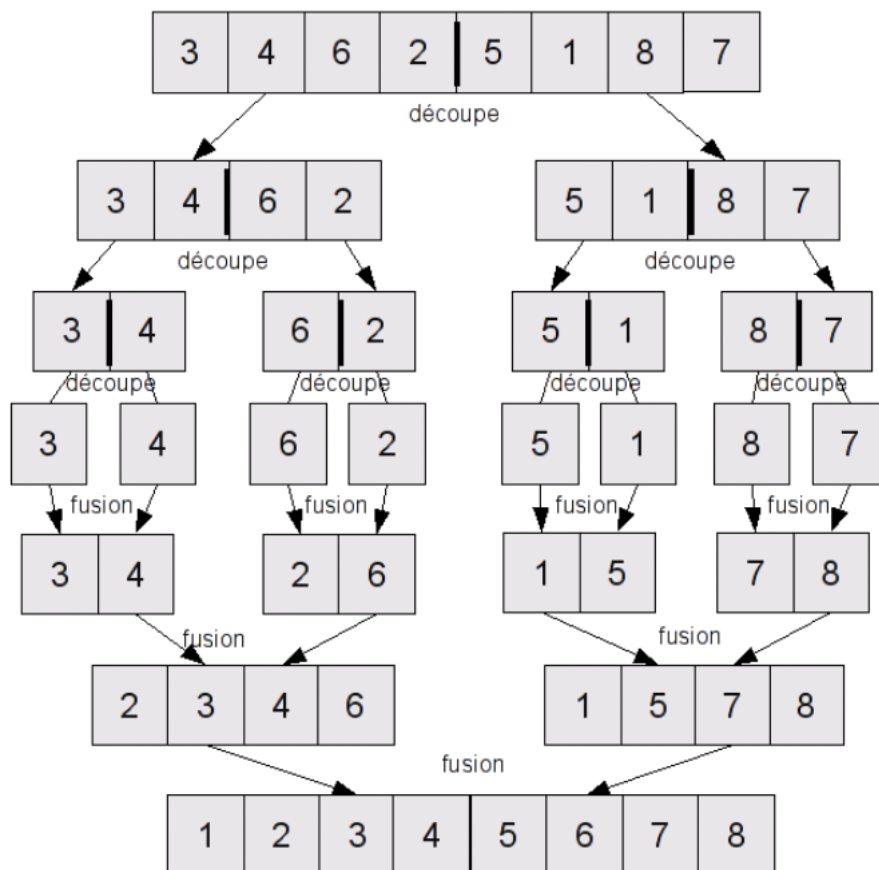
Le **tri fusion** consiste à trier **récurivement** les **deux moitiés** de la liste, **puis** à **fusionner** ces **deux sous-listes triées en une seule**. La condition d'arrêt à la récursivité sera l'obtention d'une liste à un seul élément, car une telle liste est évidemment déjà triée.

Voici donc les trois étapes (**diviser, régner et combiner**) de cet algorithme :

Une vidéo pour bien comprendre: <https://www.youtube.com/watch?v=OEmVnH3aUg>

- **Diviser** la liste en **deux sous-listes** de même taille (à un élément près) en la "coupant" par la moitié.
- **Trier récursivement** chacune de **ces deux sous-listes**. Arrêter la récursion lorsque les listes n'ont plus qu'un seul élément.
- **Fusionner** les **deux sous-listes triées en une seule**.

Ce schéma illustre le processus:



D'une manière générale, la **complexité du tri par fusion est de l'ordre de $n \log(n)$** tout en comptant le temps de calcul nécessaire pour réaliser les différents découpages. Néanmoins, **malgré cette rapidité d'exécution, cet algorithme n'est que très rarement utilisé**.

En effet, en raison des découpages répétés, l'algorithme nécessite de l'espace mémoire supplémentaire qui peut être un facteur critique lorsque le jeu de données à trier est conséquent.

Implémentation "Pythonnesque" du tri fusion...

Voici une version très "pythoneuse" du tri fusion, vous trouverez par ailleurs de nombreuses autres versions.

L'algorithme du tri fusion :

<https://www.lumni.fr/video/la-methode-laquo-diviser-pour-regner-raquo>

fonction trifusion(T):

- Si la taille du tableau T est inférieure ou égale à 1:
 - On renvoie le tableau (un seul élément)
- $T1 \leftarrow$ la première moitié de T
- $T2 \leftarrow$ la seconde moitié de T
- On renvoie la fusion(trifusion(T1),trifusion(T2))

Voici l'algorithme de fusion

fonction fusion(T1,T2):

- Si T1 est vide:
 - On renvoie T2
- Si T2 est vide:
 - On renvoie T1
- Si $T1[0] < T2[0]$:
 - On renvoie $[T1[0]] + \text{fusion}(T1[1:],T2)$
- Sinon:
 - On renvoie $[T2[0]] + \text{fusion}(T1,T2[1:])$

Exercice 1:

Expliquer l'algorithme de fusion

Exercice 2:

Implémenter cet algorithme et tester votre programme.

Rappel - Le tri par insertion

Principe

Le principe du tri par insertion consiste à **insérer un élément dans une liste d'éléments déjà triés** (par exemple, par ordre croissant).

Imaginez un joueur de cartes qui dispose de cartes numérotées. Il a des cartes triées de la plus petite à la plus grande dans sa main gauche, et une carte dans la main droite. Où placer cette carte dans la main gauche de façon à ce qu'elle reste triée ? Il faut la placer après les cartes plus petites, et avant les cartes plus grandes.

Par exemple, si la main gauche est (1 3 6 8), et que j'ai la carte 5 dans la main droite, il faut la placer après (1 3) et avant (6 8). Si l'on fait ça, on se retrouve avec la main gauche (1 3 5 6 8), qui est encore triée.

Pour trier entièrement un ensemble de cartes dans le désordre, il suffit alors de placer toutes ses cartes dans la main droite (la main gauche est donc vide), et d'insérer les cartes une à une dans la main gauche, en suivant la procédure ci-dessus.

Au départ, la main gauche est vide, donc elle bien triée.

À chaque fois que l'on insère une carte depuis la main droite vers la main gauche, la main gauche reste triée, et la main droite (l'ensemble des cartes non triées) perd une carte. Ainsi, si la main droite comprenait au départ N cartes, en N insertions, on se retrouve avec O carte dans la main droite, et N cartes, triées, dans la main gauche : on a bien trié notre ensemble de cartes.

Une vidéo pour bien comprendre: https://www.youtube.com/watch?v=IqVZANS_B4A

Implémentation "Pythonnesque" du tri par insertion...

```
1 def tri_insertion(tableau):
2     for i in range(1,len(tableau)):
3         en_cours = tableau[i]
4         j = i
5         #décalage des éléments du tableau }
6         while j>0 and tableau[j-1]>en_cours:
7             tableau[j]=tableau[j-1]
8             j = j-1
9         #on insère l'élément à sa place
10        tableau[j]=en_cours
11
12 tableau = [64,34,2,63,72,69,32,60,54,19]
13 tri_insertion(tableau)
14 print(tableau)
```

Ci dessus une implémentation du **tri par insertion** que vous testerez dans votre éditeur préféré et que vous essayerez de retenir pour l'examen. Ajouter à ce code les instructions nécessaires pour tester le temps d'exécution du programme en vue de le comparer avec les autres algorithmes de tri

Principe

Le tri par sélection est l'un des tris les plus instinctifs. Le principe est que **pour classer n valeurs, il faut rechercher la plus grande valeur et la placer en fin de liste, puis la plus grande valeur dans les valeurs restante et la placer en avant dernière position et ainsi de suite...**

Considérons un tableau à n éléments. Pour effectuer le tri par sélection, il faut rechercher dans ce tableau la position du plus grand élément. Le plus grand élément est alors échangé avec le dernier élément du tableau. Ensuite, on réitère l'algorithme sur le tableau constitué par les (n-p) premiers éléments où p est le nombre de fois où l'algorithme a été itéré. L'algorithme se termine quand $p=(n-1)$, c'est à dire quand il n'y a plus qu'une valeur à sélectionner ; celle ci est alors la plus petite valeur du tableau.

Une vidéo pour bien comprendre :

https://www.youtube.com/watch?v=tW_UDltFIjM

Ci dessous , le **pseudo-code** du tri par selection

```
procédure tri_selection(tableau t)
  n ← longueur(t)
  pour i de 0 à n - 2
    min ← i
    pour j de i + 1 à n - 1
      si t[j] < t[min], alors min ← j
    fin pour
    si min ≠ i, alors échanger t[i] et t[min]
  fin pour
fin procédure
```

Implémenter le programme correspondant à ce pseudo code en python et le tester

Quelques liens vers des explications interessantes sur les algorithmes de tri :

<http://lwh.free.fr/pages/algo/tri/tri.htm>

<https://waytolearnx.com/2019/04/tri-par-selection-en-python.html>

<https://waytolearnx.com/2019/04/tri-par-insertion-en-python.html>

<https://openclassrooms.com/fr/courses/1467201-algorithmique-pour-lapprenti-programmeur/1467940-introduction-au-probleme-du-tri>