Partie II : Algorithmique et Programmation

SOMMAIRE

Chapitre 1 : Les algorithmes de tri avancés

Les Algorithmes de tri avancés

I. Le tri par fusion

1. Principe de l'algorithme

Le tri fusion est construit suivant la stratégie "diviser pour régner", en anglais "divide and conquer". Le principe de base de la stratégie "diviser pour régner" est que pour résoudre un gros problème, il est souvent plus facile de le diviser en petits problèmes élémentaires. Une fois chaque petit problème résolu, il n'y a plus qu'à combiner les différentes solutions pour résoudre le problème global.

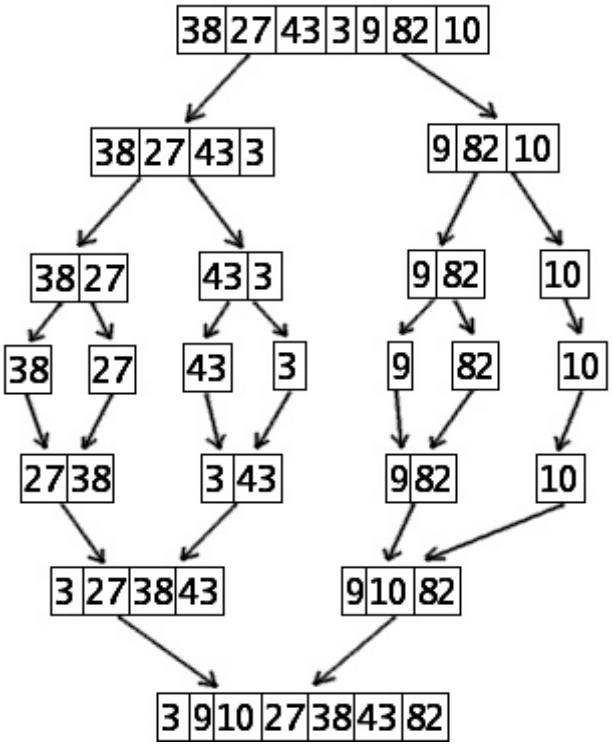
La méthode "diviser pour régner" est tout à fait applicable au problème de tri : plutôt que de trier le tableau complet, il est préférable de trier deux sous tableaux de taille égale, puis de fusionner les résultats.

L'algorithme proposé ici est récursif. En effet, les deux sous tableaux seront eux même triés à l'aide de l'algorithme de tri fusion. Un tableau ne comportant qu'un seul élément sera considéré comme trié : c'est la condition sine qua non sans laquelle l'algorithme n'aurais pas de conditions d'arrêt. Etapes de l'algorithme :

- Division de l'ensemble de valeurs en deux parties
- Tri de chacun des deux ensembles
- Fusion des deux ensembles

Exemple

Grandes étapes de l'évolution du tableau au fil de l'algorithme :



Comme vous l'avez remarqué, nous allons implémenter cet algorithme de façon récursive.
Il existe également une version itérative qui se construit de façon plus ou moins semblable.

2. Implémentation de l'algorithme

L'implémentation de cet algorithme repose essentiellement en 3 fonctions :

def inserer(e, A) :

Cette fonction insère un élément quelconque dans une liste triée par ordre croissant.
La fonction doit retourner une nouvelle liste avec le nouvel élément inséré.
Dans le cas contraire, on parcourt le tableau à la recherche de la position d'insertion.

def fusionner(A, B) :

Cette fonction prend en paramètre deux listes triées dans un ordre croissant et retourne une nouvelle liste triée qui contient les éléments de A et B fusionnés.

def tri_fusion(L) :

On vérifier si la liste L fournie en entrée n'est pas vide ou ne se résume pas un seul élément, auquel cas, il n'y a rien à faire !
Si les tests sont concluants, on lancera la fonction de fusion sur 2 sous séquences récursives du tableau.

II. Le tri rapide (Quick Sort)

1. Principe de l'algorithme

Le tri rapide (quick sort en anglais) est un algorithme de tri par comparaison inventé en 1962 par Hoare, son fonctionnement est plutôt simple à comprendre, il est très utilisé sur de grandes entrées. En effet il a pour complexité moyenne de $O(n \log(n))$ et de $O(n^2)$ dans le pire des cas. Cependant même si cet algorithme est lent dans le pire des cas, il est le plus utilisé en pratique que d'autres tris comme le tri par fusion qui a une complexité dans le pire des cas en $O(N * \log(N))$.

Le principe de cet algorithme est de diviser un ensemble d'éléments en deux parties. Pour faire cette séparation, une valeur pivot est choisie. Les valeurs sont séparées par le pivot suivant qu'elles lui soient supérieures ou inférieures. Ensuite, on fait la même chose pour les deux sous-ensembles. Cet algorithme est donc récursif.
Le choix du pivot est un problème majeur, le mieux serait de pouvoir séparer les ensembles en deux parties égales. Toutefois une recherche s'avèrerait trop coûteuse. C'est pour cela qu'on choisit le premier élément ou le dernier élément.

Exemple

Soit le tableau T à trier suivant :

4	6	8	5	1	0	2	7

Pivot choisit : T[0] = 4

1	0	2	4	6	8	5	7

Pivot du sous ensemble gauche : T[0] = 1

Pivot du sous ensemble droite : T[4] = 6

0	1	2	4	5	6	8	7

Maintenant on 4 sous-ensembles, trois sous-ensembles ne contiennent qu'un seul élément donc trié :

0	1	2	4	5	6	8	7

Pour le quatrième ensemble, on choisit le pivot : T [6]=8

0	1	2	4	5	6	7	8

Il ne reste qu'un élément :

0	1	2	4	5	6	7	8

2. Implémentation de l'algorithme

1. Ecrire la fonction « **def partition(L, debut, fin)** » : qui prend en paramètre une liste L un indice de début et un indice de fin.

   Cette fonction choisit un pivot et réorganise les éléments de la liste autour du pivot, puis renvoie le nouvel indice du pivot.

2. Ecrire la fonction « **tri_rapide(L, debut, fin)** » : : qui prend en paramètre une liste L un indice de début et un indice de fin.

   Cette fonction implémente l'algorithme du tri rapide, à la fin la liste L doit être triée.