TP D'INFORMATIQUE 9
Algorithmes de tri

Trier des données contenues dans une liste est une solution simple et efficace pour permettre d'en avoir un accès rapide, comme on l'a vu avec la recherche dichotomique. Il existe de nombreux algorithmes différents pour trier une liste, ce sujet en présente quelques-unes parmi les plus classiques.

1 Tri par sélection

Le tri par sélection se caractérise par son approche intuitive du problème :
- on cherche l'élément le plus petit et on le met tout à gauche ;
- on cherche l'élément le plus petit parmi les restants et on le met à droite de ceux déjà placés ;
- on répète le procédé jusqu'à ce que tous les éléments soient triés.

1. Écrire une fonction `indice_minimum` qui prend en argument une liste `L` (non supposée triée) et un indice `i`, et qui renvoie l'indice $j \geq i$ tel que `L[j]` soit minimal.

   Par exemple, `indice_minimum([2, 4, 8, 10, 5, 7], 2)` doit renvoyer 4, puisqu'on cherche le minimum à partir de l'indice 2, qui est donc 5, dont l'indice est 4.

2. En déduire une procédure `tri_selection` qui prend en argument une liste et qui la trie.

   On rappelle que l'instruction `L[i],L[j] = L[j],L[i]` permet d'échanger la valeur des cases d'indices i et j.

3. Un tri est dit **comparatif** s'il se base sur des comparaisons des éléments de la liste. Il est dit **en place** s'il travaille directement sur la liste à trier sans créer de nouvelle liste. Il est dit **stable** s'il conserve l'ordre des éléments qui ont la même clé dans l'ordre considéré : par exemple, si on trie relativement à l'ordre alphabétique de noms de famille, deux personnes avec le même nom restent dans le même ordre avant et après l'application d'un tri stable.

   Quelles sont les caractéristiques du tri par sélection ?

4. Évaluer le nombre maximal de comparaisons effectuées pour trier une liste de taille n.

2 Tri par insertion

Le tri par insertion est celui habituellement utilisé pour trier ses cartes. Il consiste à insérer successivement chaque élément dans la partie déjà triée de la liste. On détaille l'algorithme sur l'exemple de la liste `L = [8, 3, 5, 9, 4]` :
- `L = [8, 3, 5, 9, 4]` $\longrightarrow$ `L = [ 3, 8, 5, 9, 4]`;
- `L = [3, 8, 5, 9, 4]` $\longrightarrow$ `L = [ 3, 5, 8, 9, 4]`;
- `L = [3, 5, 8, 9, 4]` $\longrightarrow$ `L = [ 3, 5, 8, 9, 4]`;
- `L = [3, 5, 8, 9, 4]` $\longrightarrow$ `L = [ 3, 4, 5, 8, 9 ]`;

Chaque étape d'insertion nécessite une boucle `while` pour intervertir l'élément à insérer avec son prédécesseur autant de fois que nécessaire.

1. Écrire une fonction `tri_insertion` qui prend en argument une liste `L` et la trie selon la méthode décrite dans l'exemple ci-dessus.

2. Donner les caractéristiques du tri par insertion.

3. Donner le nombre maximal de comparaisons effectuées pour une liste de longueur n.

3 Tri rapide

Le tri rapide est un algorithme récursif basé sur le principe de diviser pour régner :
- si la liste est vide ou admet un seul élément alors on s'arrête : elle est déjà triée ;
- on définit le **pivot** p comme le dernier élément de la liste ;
- on répartit les éléments en 3 listes : ceux strictement inférieurs à p, ceux égaux, et ceux strictement supérieurs ;
- on trie récursivement la première et la dernière liste, et on concatène les trois listes.

1. Écrire une fonction `repartition` prenant en argument une liste et renvoyant les 3 listes décrites ci-dessus.
2. En déduire une implémentation du tri rapide.
3. Déterminer les caractéristiques de ce tri.
4. Avec notre choix de pivot, que se passe-t-il si la liste est triée ? Quel autre choix de pivot serait préférable ?

4 Tri fusion (ou tri par partition-fusion)

Le tri fusion est aussi un algorithme récursif basé sur le principe de diviser pour régner :
- si la liste est vide ou admet un seul élément alors on s'arrête : elle est déjà triée ;
- on divise la liste en deux moitiés, que l'on trie récursivement ;
- on les fusionne en une liste triée en interclassant leurs éléments.

1. Écrire une fonction `fusion` qui prend en argument deux listes `L1` et `L2` **supposées triées** et retourne la fusion triée des deux listes. Cette fonction devra être de coût linéaire en la somme des longueurs des listes. En particulier, on ne pourra pas utiliser une fonction de tri préalablement écrite.

   Indication : Stocker en mémoire les deux indices `i1` et `i2` correspondant au parcours de chaque liste, et à chaque étape ajouter la plus petite des deux valeurs `L1[i1]` et `L2[i2]` au résultat.
2. En déduire une fonction récursive `tri_fusion` implémentant le tri fusion.
3. Déterminer les caractéristiques de ce tri.

5 Tri par comptage

Le tri par comptage est un algorithme de tri spécifique pour une liste d'entiers :
- on forme un dictionnaire indiquant le nombre d'occurrences de chaque entier contenu dans la liste ;
- on calcule le minimum a et le maximum b de la liste ;
- pour chaque entier de a à b inclus, on l'ajoute au résultat autant de fois qu'indiqué dans le dictionnaire.

1. Implémenter cet algorithme de tri.
2. Quelles sont les caractéristiques du tri par comptage ?

6 Tri rapide en place

On souhaite à présent obtenir une version en place du tri rapide.

1. Écrire une fonction `repartition_en_place` prenant en argument une liste `L` ainsi que deux indices `a` et `b`, qui utilise `L[b]` comme pivot et permute les éléments de `L` entre ces indices de façon à ce qu'à gauche du pivot ne se trouve que des éléments qui lui sont inférieurs ou égaux, et à sa droite que des éléments qui lui sont strictement supérieurs. De plus la fonction renverra l'indice final du pivot.
2. Écrire une procédure `tri_sous_liste` prenant en argument une liste `L` et deux indices `a` et `b`, et qui trie en place selon la méthode du tri rapide la partie de `L` entre les indices `a` et `b` inclus.
3. En déduire une procédure `tri_rapide_en_place` prenant en argument une liste `L` et la triant par un tri rapide en place.

7 Comparaison des temps de calcul

1. Importer les bibliothèques `timeit`, `matplotlib.pyplot` et `random`.
2. Écrire une fonction `mesure_temps` prenant en argument 4 entiers k, n, p, d, et, pour chaque i entre 1 et n, détermine le temps de calcul pour chaque fonction de tri écrite pour trier k listes contenant id entiers tirés aléatoirement entre 0 et p. La fonction affichera ces temps de calcul en fonction de la longueur de la liste pour chaque algorithme de tri.

   On pourra utiliser les fonctions `randint` de `random`, `default_timer` de `timeit`, `legend` et `plot` (avec son argument optionnel `label`) de `matplotlib.pyplot`.
3. Tracer ces courbes pour $k = 100, n = 100, p = 1000, d = 2$.