TD 3

Tris

Exercice 1 : tri à bulles

L'algorithme de tri à bulles consiste à parcourir le tableau à trier et à comparer chaque élément avec son voisin de droite. Lorsque deux éléments consécutifs ne sont pas dans l'ordre croissant, ils sont échangés. Après un parcours complet du tableau, on recommence l'opération. Lorsqu'un parcours n'a donné lieu à aucun échange, cela signifie que le tableau est trié : l'algorithme s'arrête.

- 1) (TD) Trier le tableau suivant selon l'algorithme du tri à bulles en donnant toutes les étapes intermédiaires réalisées.

1	4	3	5	2	0	6	8	7
---	---	---	---	---	---	---	---	---

- 2) (TD) Donner en pseudo-code l'implémentation de la fonction `tri_bulles(tableau)` qui prend en paramètre un tableau et le trie selon l'algorithme du tri à bulles.
- 3) (TD) Donner la complexité de cet algorithme.
- 4) (TP) L'implémenter en Java.

Exercice 2 : tri par sélection

Il consiste en la recherche du plus grand élément du tableau qui va être placé à sa position définitive c'est-à-dire en dernière position. L'algorithme continue en recherchant le plus grand élément du tableau privé de son dernier élément, qui va être placé en avant dernière position. Et ainsi de suite pour tous les éléments du tableau, qui sera finalement trié.

- 1) (TD) Trier le tableau suivant selon l'algorithme du tri par sélection en donnant toutes les étapes intermédiaires réalisées.

1	4	3	5	2	0	6	8	7
---	---	---	---	---	---	---	---	---

- 2) (TD) Donner en pseudo-code l'implémentation de la fonction `tri_selection(tableau)` qui prend en paramètre un tableau et le trie selon l'algorithme du tri par sélection.
- 3) (TD) Donner la complexité de cet algorithme.
- 4) (TP) L'implémenter en Java.

Exercice 3 : tri par insertion

Le tri par insertion consiste à maintenir un tableau trié au fur et à mesure de sa création. Pour ce faire, à chaque ajout d'élément, le tableau est parcouru de manière à trouver la bonne place du nouvel élément. Il ne reste ensuite plus qu'à décaler à droite les éléments qui se trouvent après la position d'insertion.

- 1) (TD) Utiliser le tri par insertion pour obtenir un tableau trié en insérant successivement les valeurs 1, 4, 3, 5, 2, 0, 6, 8, 7.

- 2) (TD) Donner en pseudo-code l'implémentation de la fonction `insérer(tableau, e)` qui prend en paramètre un tableau trié et un élément, et insère l'élément dans le tableau de manière à ce qu'il reste trié.
- 3) (TD) Donner la complexité de cet algorithme.
- 4) (TP) L'implémenter en Java.

Exercice 4 : tri rapide

La méthode consiste à placer un élément du tableau (appelé *pivot*) à sa place définitive, en permutant tous les éléments de telle sorte que tous ceux qui lui sont inférieurs soient à sa gauche et que tous ceux qui lui sont supérieurs soient à sa droite. Cette opération s'appelle le partitionnement. Pour chacun des sous tableaux situés à droite et à gauche du pivot, on définit un nouveau pivot et on répète l'opération de partitionnement. Ce processus est répété récursivement, jusqu'à ce que l'ensemble des éléments soit trié. Généralement, on choisit comme pivot le premier élément du tableau.

- 1) (TD) Trier le tableau suivant selon l'algorithme du tri rapide en donnant toutes les étapes intermédiaires réalisées.

1	4	3	5	2	0	6	8	7
---	---	---	---	---	---	---	---	---

- 2) (TD) Donner en pseudo-code l'implémentation de la fonction `tri_rapide(tableau)` qui prend en paramètre un tableau et le trie selon l'algorithme rapide.
- 3) (TD) Donner la complexité de cet algorithme.
- 4) (TP) L'implémenter en Java.

Exercice 5 : tri fusion

L'algorithme consiste à effectuer de manière récursive les opération suivantes :

- (i) le tableau est découpé en deux parties de longueur équivalentes ;
- (ii) les deux moitiés du tableau sont triées par l'algorithme de manière récursive ;
- (iii) les deux moitiés maintenant triées sont fusionnées pour obtenir finalement la version triée du tableau initial.

L'intérêt de l'algorithme repose sur le fait que la fusion de deux tableaux triés est réalisable en temps linéaire.

- 1) (TD) Trier le tableau suivant selon l'algorithme du tri fusion en donnant toutes les étapes intermédiaires réalisées.

1	4	3	5	2	0	6	8	7
---	---	---	---	---	---	---	---	---

- 2) (TD) Donner en pseudo-code l'implémentation de la fonction `tri_fusion(tableau)` qui prend en paramètre un tableau et le trie selon l'algorithme fusion. Commencer par écrire la fonction `fusion(tableau1, tableau2)` qui fusionne les deux tableaux en un tableau trié.
- 3) (TD) Donner la complexité de cet algorithme.
- 4) (TP) L'implémenter en Java.

Références

Les algorithmes présentés dans ce TD/TP sont tirés des pages de wikipedia consacrées aux tris.