TD6 - Tris et structures de données

Exercice 1 – Tri par insertion dichotomique

Dans ce tri, on ajoute de nouveaux éléments à un tableau déjà trié T. La recherche de la place de l'élément à ajouter est faite par une méthode dichotomique. Si x est l'élément à insérer, on le compare à $T[M]$ où M est l'élément milieu du tableau trié. Si $x < T[M]$, on recommence la recherche dans la partie gauche du tableau. Si $x > T[M]$, on recommence dans la partie droite. La recherche s'arrête lorsque la partie du tableau dans laquelle on recherche est vide (indice min=indice max).

Q 1.1 Programmer de façon récursive, la fonction `rang` qui à partir du tableau T trouve la place de l'élément x. Quelle est la complexité-temps pire cas de cette fonction ?

Q 1.2 Utiliser cette fonction pour créer une tableau trié en insérant les divers éléments au fur et à mesure. Quelle est la complexité-temps pire cas de cet algorithme de tri ?

Q 1.3 Sachant que l'insertion dans une liste chaînée ne nécessite pas de décaler tous les éléments vers la droite, pensez-vous que l'utilisation d'une liste chaînée permettrait d'améliorer la complexité-temps de cet algorithme de tri ?

Exercice 2 – Tri par dénombrement

Dans le tri par dénombrement (aussi appelé tri postal), on ne compare pas les éléments à trier entre eux. On définit un nouveau tableau d'entiers T qui va nous servir comme ensemble de cases de tri (comme les bacs du tri postal). Ce tableau d'entiers T est initialisé avec des zéros. Puis on parcourt le tableau des éléments à trier, et pour chacun d'eux, on incrémente de 1 la case du tableau T correspondante. Enfin, on termine par lire le tableau T (le tableau des cases de tri) pour construire un tableau contenant les éléments triés.

Exemple : Tableau à trier {5,4,6,9,1,5,8,9,1,9}, les valeurs du tableau sont dans l'intervalle $[\![1, 9]\!]$. On crée un tableau de cases de tri composé de 9 cases, et après lecture du tableau à trier, on obtient le tableau de cases de tri suivant :

2	0	0	1	2	1	0	1	3
1	2	3	4	5	6	7	8	9

ce qui permet de trier le tableau initial en relisant le tableau de cases de tri : {1,1,4,5,5,6,8,9,9,9}.

Q 2.1 Écrire une fonction `tri` par dénombrement prenant trois paramètres : le tableau à trier `tab`, et les deux bornes de l'intervalle dans lequel sont définies les valeurs du tableau à trier.

Q 2.2 Donner la complexité-temps et complexité-mémoire d'une telle fonction. Est-ce que cette complexité peut-être atteinte pour un tri de valeurs quelconques ?

Q 2.3 On considère à présent un tableau de n réels avec n très grand. Peut-on utiliser un principe de

hachage au sein de l'algorithme de tri par dénombrement pour conserver une complexité intéressante (sous hypothèse de répartition uniforme des valeurs entre la valeur minimale et la valeur maximale) ?

Exercice 3 – Le tri comme primitive

Q 3.1 On considère un tableau de n entiers, ainsi qu'un entier B. Le but est de savoir s'il existe deux entiers dans le tableau dont la somme égale B.

1. Proposez une méthode simple pour résoudre ce problème. Quelle est la complexité de votre méthode ?
2. Proposez une méthode en $O(n \log n)$ pour résoudre le problème.

Q 3.2 On considère n appartements $(a_1, a_2, \ldots, a_n)$ évalués sur deux critères : le loyer mensuel ℓ_i et le temps de trajet pour venir à l'université t_i. On supposera pour simplifier que les loyers sont tous différents, ainsi que les temps de trajet. On dit qu'un appartement a_i domine un appartement a_j si $\ell_i < \ell_j$ et $t_i < t_j$ (a_i est meilleur sur les deux critères). Comment (et avec quelle complexité) répondriez-vous aux questions suivantes :

1. Déterminer si l'appartement a_1 est dominé (par un autre appartement) ou pas.
2. Trouver un appartement qui n'est dominé par aucun autre.
3. Déterminer s'il existe un appartement qui est dominé par un autre.

Q 3.3 Vous avez n tâches $t_1, \cdots, t_n$ à exécuter séquentiellement sur un processeur ; chaque tâche t_i a une durée $p_i \geq 0$. Vous voulez minimiser la date de fin moyenne des tâches. Le but est de déterminer dans quel ordre les tâches doivent être exécutées par le processeur.

1. On considère 3 tâches de durée respective $p_1 = 8$, $p_2 = 2$ et $p_3 = 5$. Quelle est la date de fin moyenne si on exécute ces tâches dans cet ordre ? Pouvez-vous proposer une solution meilleure ?
2. De manière générale, comment doit-on procéder ? Quelle est sa complexité de votre algorithme ? (question subsidiaire : pouvez-vous prouver que votre algorithme donne effectivement la meilleure solution ?)