DS 2

Informatique pour tous, deuxième année

Julien Reichert

Durée : 2 heures maximum.

Exercice 1 (question de cours) : Écrire un des algorithmes de tri parmi le tri fusion et le tri rapide, au choix dans sa version en place ou non en place. Prouver sa terminaison ; calculer sa complexité dans le pire des cas par une analyse rigoureuse. Prouver aussi la correction de l'algorithme.

Exercice 2 : Écrire en Python une fonction implémentant le « tri Lyes » : on crée une copie de la liste de la bonne taille et on place chaque élément à une position qui se déduit du nombre d'éléments inférieurs. Attention à bien gérer les cas d'égalité. Donner la complexité de la fonction écrite.

Exercice 3 : Écrire en Python une fonction implémentant le « tri Dubos » : on crée une liste supplémentaire de taille exponentielle en la taille de la liste, on place le premier élément de la liste au milieu de la liste supplémentaire et les suivants à une place libre déterminée par dichotomie. Une fois tous les éléments placés, on parcourt la liste supplémentaire et on récupère tous les éléments de la liste d'origine. Inutile de donner la complexité. . . [1]

Exercice 4 : Écrire en Python une fonction implémentant le « tri Dijkstra » : il s'agit de trier en place une liste dont les valeurs des éléments sont toutes dans un ensemble de taille trois, par exemple les nombres `0`, `1` et `2`. Le principe est de parcourir la liste en faisant des échanges à des indices pertinents, un peu à la manière du tri rapide en place.

Exercice 5 : On considère sur l'exercice entier une table `Resultats` pouvant matérialiser des épreuves quelconques. Les attributs sont limités à une date `Date` de type chaîne de caractères de taille limitée à 20, un identifiant `Id` et à un score `Score`, tous les deux de type entier. Le couple `(Date, Id)` est une clé primaire.

Question 1 : Écrire une requête qui permet de récupérer le score maximum réalisé, la date à laquelle il a été réalisé et l'identifiant de la personne qui l'a réalisé.

Question 2 : Écrire une requête qui permet de récupérer le score maximum réalisé à une date notée `d`.

Question 3 : Écrire une requête qui permet de déterminer le nombre de personnes ayant fait mieux qu'un score `s` donné à une date notée `d`.

Question 4 : Écrire une requête qui permet de déterminer le nombre de personnes ayant fait mieux que le score d'une personne dont l'identifiant `n` est donné à une date notée `d`.

Question 5 : Écrire une requête qui permet de récupérer l'ensemble des couples d'identifiants de personnes telles que le score de la première soit strictement supérieur au score de la seconde, le tout à une date notée `d`.

Question 6 : Déduire des questions qui précèdent une requête qui permet d'obtenir le classement à la date `d` [2] en attribuant la même place à des égalités (l'ordre d'affichage en cas d'égalités n'importe pas). On comprendra que la place est à inclure dans le résultat de la requête.

1. Remarque : Il s'agit simplement du tri par ABR, aussi efficace que les tris qui suivent le paradigme « diviser pour régner » mais en implémentant de manière totalement non optimale la structure d'ABR. Le besoin de connaître la structure d'ABR et l'absence d'avantages sur les tris fusion et rapide fait que ce tri reste assez méconnu.
2. ou le classement général si on préfère