Exercice 2, Algo et complexité

1 Enoncé

Une liste L est triée avec la méthode suivante: si L est vide ou ne contient qu'un élément, rendre L; sinon, soit k la valeur du premier élément de L; les éléments de L inférieurs à k sont collectés dans une première liste L1, les éléments égaux à k dans une deuxième liste L2, les éléments supérieurs à k dans une troisième liste L3. L1, L2, L3 sont triées par trois appels récursifs; les trois listes résultantes sont concaténées dans une liste L' qui est retournée. L'élément qui permet de partitionner L s'appelle le pivot de L.

1. Cette méthode contient une erreur. Laquelle? La suite suppose que cette erreur a été corrigée.

2. Proposez une variante pour choisir l'élément k. Justifier .

3. Quel est le nom de cette méthode, aussi utilisable pour des tableaux?

4. Quelle est la complexité espérée de cette méthode, sur un tableau ? Quelle est la complexité dans le pire des cas (pour les utilisateurs très malchanceux), sur un tableau ?

5. Même question, quand des listes sont utilisées. (2 lignes)

6. Pour choisir un pivot aléatoirement, on peut compter le nombre d'éléments dans la liste courante, et choisir un élément dont le rang est un nombre pseudo aléatoire, entre 1 et n, où n est la longueur de la liste. Ceci modifie-t-il la complexité de la méthode ?

7. Quelle est l'ordre de grandeur de l'espace mémoire utilisé par cette méthode, utilisée sur une liste ?

8. Existe t - il des méthodes plus efficaces ? Lesquelles ?

2 Correction

1. La méthode boucle lorsque la liste L2 (qui contient les éléments égaux à la clef) a plus d'un élément. Pour corriger, il suffit de ne pas trier L2.

2. Prendre pour pivot le médian des trois premiers éléments dans la liste. Ou bien prendre pour pivot le médian de trois éléments de rang (pseudo-) aléatoire. Ces pivots sont souvent plus proches du médian.

3. Tri rapide, ou quicksort. Ceci a été vu en TP et en TD.

4. Sauf malchance insigne, la complexité attendue est $O(n \log n)$. Si le pivot est systématiquement loin du médian, $O(n^2)$. Vu en TD.

5. Même complexité avec des listes ou des tableaux (si programmé correctement, bien sûr...). Vu en TP et en TD.

6. Non.

7. $O(n \log n)$. (Un algorithme en $O(n \log n)$ ne peut pas affecter plus de $O(n \log n)$ données)

8. Le tri par tas (heapsort), le tri par fusion (mergesort) sont toujours en $O(n \log n)$. Vu en TP et en TD.