TP 5 – Insertion, identification et inversions

1. Tri par insertion

Le tri par insertion est un algorithme de tri très intuitif : c'est celui qui est couramment utlisé lors d'un tri de jeu de cartes. Il consiste en la création d'un "nouveau" jeu, en insérant successivement chaque carte à sa place dans le nouveau jeu trié.

(1) Ecrire une fonction permettant l'insertion d'un élément à la bonne place dans une liste déjà triée.

(2) L'utiliser afin d'écrire un algorithme de tri par insertion.

(3) Déterminer la complexité en temps de cet algorithme (dans le meilleur cas, dans le pire des cas et dans un cas moyen).

2. Tri par sélection

Le tri par sélection est un autre algorithme de tri très intuitif, lui aussi couramment utlisé par certains lors d'un tri de jeu de cartes. Il consiste en la recherche de l'élément minimal (ou maximal) de la liste pour pouvoir le placer au bon endroit dans une nouvelle liste.

(4) Ecrire un algorithme de tri par sélection.

(5) Déterminer la complexité en temps de cet algorithme (dans le meilleur cas, dans le pire des cas et dans un cas moyen). Commenter.

3. Tri bulle

Les deux algorithmes précédents ont pour avantage principal d'être très intuitifs, mais ils nécessitent la création d'une nouvelle liste, ce qui implique un coût en mémoire qui peut ne pas être négligeable si la liste à trier comporte beaucoup déléments. Pour pallier cet inconvénient, le cahier des charges peut demander un algorithme qui modifie la liste de départ, sans en écrire une nouvelle. Les opérations de placement sont alors des opérations d'inversion de deux éléments de la liste.

(7) Modifier l'algorithme du tri par sélection afin qu'il réponde au cahier des charges.

Un autre algorithme de tri qui respecte cette condition est le tri à bulles : on parcourt la liste une première fois, et si deux éléments consécutifs sont mal rangés, on les inverse.

(8) Qu'obtient t'on lorsque ce premier parcours de la liste est effectué ?

(9) En déduire alors la fin de l'algorithme du tri à bulles.

(10) Quelle est la complexité en temps de cet algorithme (dans le meilleur cas, dans le pire des cas et dans un cas moyen) ?

(11) Comment améliorer cette complexité dans le cas d'une liste déjà triée ?

Dans ce tri, certains éléments sont qualifiés de tortues (ceux qui trouvent leur place en dernier) et d'autres de lièvres (ceux qui la trouvent en premier).

(12) Qui sont les tortues et qui sont les lièvres ? et si on parcourt la liste dans l'autre sens ?

(13) Ecrire alors l'algorithme de tri cocktail (ou tri shaker) dans lequel la liste est parcourue une fois dans un sens puis une fois dans l'autre. Quelle est sa complexité ?