Listes triées, Insertions et Fusions

Cette séance de Travaux dirigée est consacrée à l'étude de deux algorithmes de tri sur les tableaux et listes chaînées. Dans tout les exercices, on travaille avec des listes simplement chaînées d'entiers en mode mutatif avec une fausse tête.

▸ Exercice 1. Insertion dans un tableau

On dispose d'un tableaux d'entiers `tab` de taille `taille`. Pour un entier $0 \leq i <$ `taille`, on suppose que les éléments $\texttt{tab}[1], \ldots, \texttt{tab}[i-1]$ sont triées.

1. Écrire un algorithme qui insère $\texttt{tab}[i]$ à sa place dans le tableau. Quelle est la complexité de cet algorithme ?
2. En déduire un algorithme de tri. Quelle est la complexité de cet algorithme ?

▸ Exercice 2. Insertion dans une liste triée

1. Écrire un algorithme qui, étant donnée une liste triée et un pointeur vers une cellule contenant un entier, insère cet entier à sa place dans la liste. Quelle est la complexité de cet algorithme ?
2. En déduire un algorithme de tri. Quelle est la complexité de cet algorithme ?

▸ Exercice 3. Coupe d'une liste en deux sous-listes

1. Écrire une fonction
   ```
   list cut(list lst, int i);
   ```
   qui coupe une liste en position `i`. La partie coupée est retournée par la fonction.
2. Quelle est la complexité de cette fonction ?
3. Écrire une fonction
   ```
   list cut(list lst);
   ```
   qui coupe une liste au millieu. La partie coupée est retournée par la fonction.

▸ Exercice 4. (Fusion de deux listes triées)

1. Écrire une fonction qui, étant données deux listes triées, fusionne les deux listes pour ne faire qu'une seule liste triées. Quelle est la complexité de cette fonction ?

On peut utiliser la fonction précédente pour obtenir un algorithme de tri :
- Couper la liste en deux sous listes ;
- Par un appel récursif, trier les deux sous listes ;
- Fusionner les deux listes pour avoir une liste triée.

2. Écrire une fonction qui implante cet algorithme de tri.
3. Quelle en est la complexité ?