TP : Algorithmes de tri

Dans ce TP, nous allons voir plusieurs algorithmes de tri, de listes ou de tableaux.

Le prochain cours sera dédié au calcul de complexité ; nous verrons, parmi ces algorithmes, lesquels sont les plus efficaces pour trier des listes.

Pour pouvoir tester nos fonctions, on va écrire des fonctions générant des listes et tableaux d'entiers aléatoires. On pourra utiliser la fonction `Random.int : int -> int` qui renvoie un entier aléatoire entre 0 et son argument.

1 Tri par sélection

Ce tri est le plus naturel : on cherche le minimum de la liste ou du tableau, et on le place en première position, puis on itère le processus.

Implémenter ce tri pour les listes et les tableaux.

Pour les listes, on pourra commencer par écrire une fonction qui renvoie le minimum d'une liste, et le reste de la liste.

2 Tri par insertion

C'est le tri du joueur de cartes. On regarde les éléments uns par uns, qu'on place au bon endroit dans la partie déjà triée.

Implémenter ce tri pour les listes et les tableaux.

3 Tri fusion

Cet algorithme de tri de listes est un algorithme de type "diviser pour régner". Il fonctionne de la façon suivante :

- si la liste est vide ou contient un seul élément, elle est triée
- sinon, on la divise en deux, on tri chacune des deux moitiés, et on les fusionne.

Écrire des fonctions `divise : int list -> int list * int list` et `fusion : int list -> int list -> int list` qui réalisent les opérations de division et de fusion, puis une fonction qui implémente le tri fusion.

4 Tri rapide

Là encore, c'est un algorithme de type "diviser pour régner". Il fonctionne de la façon suivante :

- on choisit un élément de la liste, qu'on appelle *pivot* (on peut par exemple prendre le premier élément de la liste)
- on partitionne la liste suivant ce pivot : on crée deux listes ; la première contient tous les éléments inférieurs au pivot, la seconde ceux supérieurs
- on trie récursivement les deux sous-listes, et on les concatène.

Écrire une fonction `partition : int list -> int -> int list * int list` qui partitionne une liste suivant un pivot, puis implémenter le tri rapide.