

## Feuille 12 : Tris

**Exercice 12.1** Réviser le principe du *tri rapide*.

**Exercice 12.2** Écrire une fonction `partition` de type `'a -> 'a list -> 'a list * 'a list` qui prend en paramètre un pivot  $p$ , une liste  $l$  et qui retourne un couple de listes  $(l_1, l_2)$  telles que  $l_1$  contient les éléments de  $l$  qui sont  $< p$  et  $l_2$  les autres. Exemples :

```
# 1;;
- : int list = [1; 5; 3; 6; 8; 3; 9; 10; 2]
# partition 5 1;;
- : int list * int list = ([2; 3; 3; 1], [10; 9; 8; 6; 5])
# partition 3 1;;
- : int list * int list = ([2; 1], [10; 9; 3; 8; 6; 3; 5])
```

**Exercice 12.3** Écrire la fonction `quick_sort` de type `'a list -> 'a list` qui prend en paramètre une liste  $l$  et qui retourne la liste triée selon le prédicat  $<$  des éléments de  $l$  en suivant l'algorithme récursif du tri rapide. Comme pivot, on utilisera le premier élément de la liste. Exemples :

```
# quick_sort [];;
- : 'a list = []
# quick_sort [1; 5; 3; 6; 8; 3; 9; 10; 2];;
- : int list = [1; 2; 3; 3; 5; 6; 8; 9; 10]
```

**Exercice 12.4** Réviser le principe du *tri fusion*.

**Exercice 12.5** Écrire une fonction `split` de type `'a list -> 'a list * 'a list` qui prend en paramètre une liste  $l$  et qui retourne un couple de liste  $(l_1, l_2)$  résultant du découpage de la liste  $l$  en deux sous-listes dont la différence des longueurs est  $\leq 1$ . Exemples :

```
# split [1; 5; 3; 6; 8; 3; 9; 10; 2];;
- : int list * int list = ([1; 5; 3; 6], [8; 3; 9; 10; 2])
# split [1; 5; 3; 6; 8; 3; 9; 10; 2; 7];;
- : int list * int list = ([1; 5; 3; 6; 8], [3; 9; 10; 2; 7])
```

**Exercice 12.6** Écrire la fonction `merge` de type `'a list -> 'a list -> 'a list` qui prend en paramètre deux listes triées  $l_1$  et  $l_2$  et qui retourne la liste triée éléments de  $l_1$  et  $l_2$  obtenue par fusion des deux listes. Exemples :

```
# merge [1;3;5;9] [4;6;7];;
- : int list = [1; 3; 4; 5; 6; 7; 9]
```

**Exercice 12.7** En utilisant les fonctions `split` et `merge`, écrire la fonction `merge_sort` de type `'a list -> 'a list` qui prend en paramètre une liste  $l$  et qui retourne la liste triée des éléments de  $l$  en suivant l'algorithme récursif du tri fusion. Exemples :

```
# merge_sort [1; 5; 3; 6; 8; 3; 9; 10; 2];;  
- : int list = [1; 2; 3; 3; 5; 6; 8; 9; 10]
```