Travaux pratiques

option informatique

Tri d'une liste

Durant cette séance nous allons programmer trois algorithmes de tri classiques sous la forme de fonctions opérant sur des listes.

Exercice 1. tri par insertion

a) Écrire une fonction **`insertion`** qui insère un élément dans une liste dont les éléments sont supposés *rangés par ordre croissant*.

```
insertion : 'a -> 'a list -> 'a list
```

b) Le tri par insertion consiste à extraire le premier élément de la liste, à trier récursivement le reste de la liste, puis à insérer l'élément isolé à l'aide de la fonction précédente.
Écrire une fonction **`insertion_sort`** qui réalise le tri par insertion d'une liste.

```
insertion_sort : 'a list -> 'a list
```

c) Justifier que le coût de cette fonction est un $O(n^2)$ où n désigne la taille de la liste à trier.

Exercice 2. tri fusion

a) Écrire une fonction **`split`** qui prend une liste ℓ en argument et renvoie une paire de listes (ℓ_1, ℓ_2) telle que les éléments de ℓ sont exactement ceux de ℓ_1 réunis à ceux de ℓ_2 et telle que les longueurs de ℓ_1 et de ℓ_2 diffèrent d'au plus une unité.

```
split : 'a list -> 'a list * 'a list
```

b) Ecrire une fonction **`merge`** qui prend en arguments deux listes supposées triées par ordre croissant et les fusionne en une unique liste triée.

```
merge : 'a list -> 'a list -> 'a list
```

c) Le tri fusion consiste à couper la liste en deux parties à l'aide de la fonction **`split`**, à trier récursivement chacune des deux listes obtenues puis à les fusionner à l'aide de la fonction **`merge`**.
Ecrire une fonction **`mergesort`** qui réalise le tri fusion d'une liste.

```
mergesort : 'a list -> 'a list
```

Nous démontrerons plus tard dans l'année que le coût de cette fonction est un $O(n \log n)$ où n désigne la taille de la liste à trier.

Exercice 3. tri rapide

a) Ecrire une fonction **`partition`** qui prend en arguments un élément x et une liste ℓ et qui renvoie une paire de listes (ℓ_1, ℓ_2) telle que ℓ_1 contient les éléments de ℓ qui sont inférieurs ou égaux à x et ℓ_2 ceux qui lui sont strictement supérieurs.

```
partition : 'a -> 'a list -> 'a list * 'a list
```

b) Le tri rapide consiste à isoler un élément particulier de la liste (par exemple le premier), à partitionner les éléments restants à l'aide de la fonction précédente, à trier récursivement les deux listes obtenues, puis enfin à « recoller » les morceaux (par concaténation).
Écrire une fonction **`quicksort`** qui réalise le tri rapide d'une liste.

```
quicksort : 'a list -> 'a list
```

Nous démontrerons plus tard que le coût de cette fonction est dans le pire des cas un $O(n^2)$ mais qu'en moyenne son coût est un $O(n \log n)$.