Option Informatique

DM $n°1$ – Tris.

Consignes : le sujet doit être rendu par e-mail au format .ml au plus tard le 25/02/2021 à 23h59.

N'hésitez pas à demander des conseils par e-mail.

Tris par sélection et par insertion.

1. En reprenant le travail effectué pour le TD $n°02$, écrire une fonction `tri_selection_t : 'a array → unit` ne retournant rien mais ayant pour effet de bord de trier un tableau donné en argument en utilisant l'algorithme de tri par sélection. La ou les fonctions auxiliaires utilisées devront être définies à l'extérieur de la fonction principale.

2. En reprenant le travail effectué pour le TD $n°02$, écrire une fonction `tri_selection_l : 'a list → 'a list` retournant une liste triée ayant les mêmes éléments que la liste donnée en argument, en utilisant l'algorithme de tri par sélection. La ou les fonctions auxiliaires utilisées devront être définies à l'extérieur de la fonction principale.

3. En reprenant le travail effectué pour le TD $n°02$, écrire une fonction `tri_insertion_l : 'a list → 'a list` retournant une liste triée ayant les mêmes éléments que la liste donnée en argument, en utilisant l'algorithme de tri par insertion.

4. En reprenant le travail effectué pour le TD $n°02$, écrire une fonction `tri_insertion_t : 'a array → unit` ne retournant rien mais ayant pour effet de bord de trier un tableau donné en argument en utilisant l'algorithme de tri par insertion.

Tris par bulle sur les tableaux.

On présente dans cette section un autre algorithme de tri, le tri à bulles. Le tri à bulles consiste, pour trier un tableau, à :

— Parcourir le tableau en permutant deux éléments consécutifs lorsqu'ils ne sont pas dans le bon ordre ;

— Recommencer sur les n-1 premiers éléments du tableau ; etc.

Par exemple, si l'on veut trier le tableau $[|3;0;2;6;1|]$:

• Lors de la première passe, on permute d'abord 3 et 0 car ils ne sont pas dans le bon ordre, ce qui donne $[|0;3;2;6;1|]$; puis on permute 3 et 2 qui ne sont pas dans le bon ordre, ce qui donne $[|0;2;3;6;1|]$; on ne permute pas 3 et 6 qui sont dans le bon ordre, on conserve donc $[|0;2;3;6;1|]$; enfin on permute 6 et 1 qui ne sont pas dans le bon ordre, ce qui donne $[|0;2;3;1;6|]$ et le plus grand élément est à sa place (mais pas tous les autres).

• Le plus grand élément étant à sa place, la seconde passe se fait en considérant uniquement les $n-1$ premiers éléments, etc. Il suffit de s'arrêter après la $(n-1)^{\text{ième}}$ passe.

5. Écrire une fonction `tri_bulles_t : 'a array → unit` implémentant cet algorithme en Caml.

Tris par pivot sur les listes.

Le *tri par pivot*, également appelé *tri rapide* (« *quicksort* » en V.O.) est également un algorithme de tri important du programme de l'option. Cet algorithme de tri procède comme suit : on prend le premier élément de la liste à trier qu'on appelle le pivot, on parcourt le reste de la liste pour déterminer s'ils sont plus petits ou plus grands que le pivot, on trie récursivement les éléments plus petits et les éléments plus grands, et enfin on raccorde les trois morceaux dans l'ordre.

6. Écrire une fonction `partition : 'a → 'a list → 'a list * 'a list`, telle que `partition k l` retourne un couple (l_1, l_2) tel que les éléments de l_1 sont les éléments de l strictement inférieurs à k et les éléments de l_2 sont les éléments de l supérieurs à k.

7. Écrire une fonction `tri_pivot_l : 'a list → 'a list` implémentant l'algorithme de tri par pivot sur les listes. On pourra sans scrupules utiliser l'opérateur de concaténation `@` qui permet de concaténer deux listes.

Bonus

Ce n'est pas demandé, mais à titre d'exercice bonus vous pouvez réfléchir à des adaptations du tri à bulles pour les listes ou du tri par pivot pour les tableaux.