OPTION INFORMATIQUE

TP $n°10$: on révise les tris

I Des tris très classiques

I.1 Tri par insertion

On rappelle la version enveloppée du tri par insertion vue avant les vacances. Pour trier une liste ℓ :

- Si ℓ est vide, alors elle est triée.
- Sinon, ℓ est de la forme $h :: t$. On trie récursivement t et on insère h à sa place dans la liste triée.

Pour implémenter cet algorithme en OCaml, on a commencé par écrire une fonction auxiliaire (locale ou pas) `insere : 'a →'a list→'a list` telle que, si `l` est une liste triée alors `insere x l` retourne une liste **triée** ayant les mêmes éléments de `l`, plus l'élément `x` (à sa place).

1. Vérifions qu'on sait encore le faire.

On reste sur le tri par insertion, mais on se propose d'en écrire une version *récursive terminale*. Pour trier une liste ℓ :

- On considère deux listes : la liste ℓ_1 des éléments qui sont encore à tirer, initialement ℓ, et la liste ℓ_2 des élément déjà triés, initialement [].
- On prend la tête de ℓ_1 et on l'insère à sa place dans ℓ_2 ;
- On recommence jusqu'à que ce ℓ_1 soit vide.

Bien sûr, dans la description précédente, il n'y a pas de réelle modification des deux listes, simplement des appels récursifs sur des listes modifiées.

2. On le code.

I.2 Tri par fusion

Le tri par fusion consiste, pour trier une liste, à :

- Diviser la liste en deux moitiés ;
- Trier récursivement chacune des deux moitiés ;
- Fusionner les deux moitiés triées pour reconstituer la liste triée.

Le cas de base étant celui des listes de longueur ≤ 1.

3. On le code.

I.3 Tri par pivot

On rappelle le principe du tri par pivot : on prend le premier élément du tableau ou de la liste à trier qu'on appelle le pivot, on parcourt le reste du tableau ou de la liste pour déterminer s'ils sont plus petits ou plus grands que le pivot, on trie récursivement la liste des éléments plus petits et celle des éléments plus grands, et enfin on raccorde les trois morceaux dans l'ordre.

4. Allez, hop, on le code.

II D'autres tris sympatiques

II.1 Tri par tiroirs

On se limite au cas où la liste à trier est une liste d'entiers naturels. On commence par déterminer l'entier maximal m apparaissant dans la liste. Puis on crée un tableau d'entiers initialement nuls indexé de 0 à m et on parcourt la liste en incrémentant la $i^{\text{ième}}$ case du tableau pour chaque entier i rencontré dans la liste. Enfin, on reconstitue la liste triée en parcourant le tableau.

5. Bon, on le code.

II.2 Tri à bulles

Le tri à bulles consiste, pour trier une liste, à procéder comme suit :

- On parcourt la liste ou en permutant deux éléments consécutifs à chaque fois qu'ils ne sont pas dans le bon ordre ;
- Le maximum est maintenant à sa place, on recommence sur les $n-1$ premiers éléments de la liste ;
- Les deux plus grands éléments sont maintenant à leur place, on recommence sur les $n-2$ premiers éléments ;
- On s'arrête lorsqu'il n'y a plus qu'un seul élément à traiter **ou** lorsqu'aucune permutation n'a été faite.

6. On... le code ?

II.3 Tri par séquences croissantes

Le principe du *tri par séquences croissantes* sur les listes est le suivant : on découpe la liste à tirer en liste de listes triées, en découpant la liste initiale à chaque observation d'une décroissance. Par exemple, le découpage de la liste $[1;2;3;4;5;2;5;7;9;3;6;7;0;1]$ doit donner la liste de listes $[[1;2;3;4;5];[2;5;7;9];[3;6;7];[0;1]]$. Ceci doit être fait en un seul parcours de la liste afin de garantir que cette étape soit de complexité linéaire.

Dans un second temps, on fusionne les listes triées obtenues.

7. Écrire une fonction `decouper : 'a list → 'a list list` réalisant le découpage décrit ci-dessus.

Pour fusionner les listes obtenues par découpage, on aura besoin de fonctions `fusionner : 'a list → 'a list → 'a list` permettant de fusionner deux listes triées en une liste triée, et `scinder : 'b list -> 'b list * 'b list` permettant de scinder une liste en deux listes de longueurs égales à une unité près. Normalement, on les a déjà codées pour le tri par fusion. Les récupérer.

8. Écrire une fonction `megafusionner : 'a list list → 'a list` telle que, si `l` est une liste de listes triées, alors `megafusionner l` fusionne toutes les listes de `l` en utilisant la stratégie "diviser pour régner" suivante : on découpe la liste de listes en deux (disons les $\lfloor \frac{m}{2} \rfloor$ premières et les $\lceil \frac{m}{2} \rceil$ suivantes, ou l'inverse), on réalise récursivement la fusion des $\lfloor \frac{m}{2} \rfloor$ premières et la fusion des $\lceil \frac{m}{2} \rceil$ suivantes, puis on réalise la fusion des deux listes obtenues à l'aide de la fonction `fusionner`.

9. Finalement, en utilisant les questions précédentes, écrire une fonction `tri_sc : 'a list → 'a list` permettant de trier une liste à l'aide de l'algorithme de tri par séquences croissantes.

II.4 Tri par tas

Bon c'est long le tri par tas et on en a déjà beaucoup parlé en TD avec le tri lisse donc j'ai fait l'impasse. Mais si vous y tenez, on peut le faire en question bonus.