Série d'exercices #$3\frac{1}{2}$

IFT-2035

September 21, 2023

Structure de données fonctionelle

Soit le type suivant en Haskell qui définit un arbre binaire que l'on peut utiliser pour représenter une table associative (qui associe des *clés* de type `Int` à des valeurs de type β):

```
data TreeMap b = Empty | Node Int b (TreeMap b) (TreeMap b)
```

L'exercice est de définir les opérations typiques sur une telle structure de donnée. Bien sûr, pour être utile l'arbre doit être maintenu dans l'ordre: toutes les clés dans la branche de gauche d'un `Node` doivent être plus petites que la clé du noeud, et vice versa pour la branche de droite.

Il y a trois opérations:

- *tmLookup*: rechercher la valeur associée à une clé passée en paramètre.
- *tmInsert*: ajouter une entrée (donnée sous la forme d'une clé et de sa valeur) dans la table.
- *tmRemove*: enlever une entrée (dont la clé est passée en paramètre).

Ces fonctions doivent être totales (elles terminent toujours et ne doivent jamais signaler d'erreur).

1. Donner un type acceptable pour chacune de ces trois fonctions.
2. Donner le code des deux premières fonctions (points de karma en bonus pour *tmRemove* qui est plus pénible et moins souvent nécessaire).
3. Pour un arbre de profondeur 10 contenant $\sim$1000 éléments, quel est le coût approximatif de chacune de ces fonctions.

Pour rendre l'exercice plus utile, il est important de faire ces étapes dans l'ordre: i.e. ne pas écrire le code avant d'avoir décidé du type des fonctions.