FP-TP1 (sujet)

December 16, 2024

1 Tri par insertion

1. Proposer un programme permettant d'insérer une valeur à la "bonne" place dans une liste (initialement triée).
2. Comment généraliser le programme précédent en permettant d'ajouter une liste de valeurs dans une liste triée ?
3. Discuter du résultat obtenu en insérant une liste de valeurs dans une liste initialement vide.
4. Comment choisir la relation d'ordre (ex. supérieure ou inférieure) dans le programme précédent ?

2 Arbres Binaires de Recherche (ABR)

5. Définir le type `ABR` vu en cours et proposer une fonction pour y injecter une valeur ?

   Note. Ajouter en paramètre la relation d'ordre.
6. Comment utiliser la fonction précédente pour transformer une liste en arbre ?
7. Proposer un programme `flatten` permettant de transformer (sérialiser) un arbre en liste.
8. Proposer un opérateur `add` permettant de combiner 2 arbres.

3 Ensembles (TreeSet)

9. Proposer une fonction `contains` permettant de savoir si une valeur se trouve dans un arbre.
10. En représentant des ensembles de valeurs par des arbres et en utilisant la fonction précédente, définir deux programmes pour calculer l'union et l'intersection de 2 ensembles/arbres.

    On notera que l'union est similaire à la fonction `add` définie plus haut et qu'elle évite simplement d'avoir la même valeur dans le résultat.

4 Dictionnaires (TreeMap)

11. Comment définir le type `TreeMap` qui sont des ABR dont les éléments sont des paires (clef,valeur) et où il existe une relation d'ordre sur les clef ?

Comment définir aussi une fonction `lookup` permettant d'obtenir la valeur associée à une clef ?

5 Expressions Algébriques

12. Une expression algébrique est soit une valeur, soit une variable, soit la somme de 2 expressions, soit le produit de 2 expressions.

Comment définir en `F#` le type `Exp` permettant de définir des expressions algébriques ? Comment représenter `e=1+(x*3)` ?

13. Proposer un programme `tostring` permettant de représenter sous forme de chaîne de caractères une expression.

14. En considérant que les valeurs des variables sont enregistrées dans un `TreeMap` (ex. `x=2` sera représenté par la paire `("x",2)`), proposer un programme `eval` permettant de connaître la valeur d'une expression.

6 Langage de requêtes

15. En s'appuyant sur la définition de la fonction `eval`, proposer une fonction d'`inter`pretation permettant de choisir à quels opérateurs correspondent `Add` et `Mul`.

    Vérifier que `inter` permet bien de retrouver la fonction `eval`.

16. En considérant que les variables correspondent à des ensembles de mots/noms et que les symbols Add/Mul correspondent à l'union/intersection, expliquer comment évaluer `1A*(IR+ASE)` en prenant les valeurs suivantes:

```
IR  = { bob, kate }
ASE = { max, amy }
1A  = { kate, bob }
```