TP numéro 3

Programmation fonctionnelle, ENSIIE

Semestre 4, 2019–20

Exercice 1 : Interface des dictionnaires

1. Écrire le fichier d'interface des dictionnaires (fichier `map.mli`). Il contient le type abstrait `map` des dictionnaires, et les quatre prototypes des fonctions de manipulation des dictionnaires `create`, `find`, `add` et `remove`.
2. Écrire un fichier `test.ml` qui déroulera le scénario suivant :
   - créer un dictionnaire vide ;
   - ajouter l'association `3` $\mapsto$ `"coucou"` dans le dictionnaire ;
   - vérifier que la recherche de `3` dans le dictionnaire renvoie bien `"coucou"` ;
   - ajouter l'association `1` $\mapsto$ `"toto"` dans le dictionnaire ;
   - ajouter l'association `2` $\mapsto$ `"titi"` dans le dictionnaire ;
   - vérifier que la recherche de `1` dans le dictionnaire renvoie bien `"toto"` ;
   - supprimer `1` dans le dictionnaire ;
   - vérifier que la recherche de `3` dans le dictionnaire renvoie bien `"coucou"`.

Exercice 2 : Listes d'association

3. Dans un fichier `assoc_liste.ml`, implémenter les dictionnaires à l'aide de listes d'association.
4. Utilisez le `Makefile` fourni pour compiler le test avec les listes d'associations : `make test_al`
5. Compiler, tester.

Exercice 3 : Arbres binaires de recherche

6. Dans un fichier `abr.ml`, implémenter les dictionnaires à l'aide d'arbres binaires de recherche.
7. Utilisez le `Makefile` fourni pour compiler le test avec les listes d'associations : `make test_abr`

Exercice 4 : Comparaison de performances

8. Dans `test.ml`, écrire une fonction
   `add_random : int -> (int, string) map -> (int, string) map` qui prend en paramètre un entier `n` et un dictionnaire `d`, et qui retourne un dictionnaire qui correspond à l'ajout dans `d`, de `n` associations entre un entier choisi au hasard entre 0 et `n` (cf. documentation de `Random.int`) et sa conversion en tant que chaîne (`string_of_int`).
9. Écrire une fonction
   `remove_random : int -> int -> (int, string) map -> (int, string) map` qui prend en paramètre deux entiers `n` et `m` ainsi qu'un dictionnaire `d`, et qui retourne un dictionnaire qui correspond à la suppression, dans `d`, de `m` associations dont les clefs sont des entiers choisis au hasard entre 0 et `n`.
10. Écrire une fonction
    `print_assoc: int -> (int, string) map -> unit` qui prend en paramètre un entier `n` et un dictionnaire `d`, et qui affiche, pour chaque entier entre 0 et `n-1`, soit la valeur associée dans le dictionnaire si elle existe, soit un message disant qu'elle n'existe pas.
11. À l'aide de ces fonctions, tester en prenant en argument du programme une valeur pour `n` (grâce à `int_of_string Sys.argv.(1)`), qui ajoute `n` associations au dictionnaire vide, qui en supprime `n/2` et qui affiche les `n` associations potentielles.
12. Recompiler les deux programmes de test, comparer leur temps d'exécution avec la commande shell `time`.