DS 1

Option informatique, deuxième année

Julien Reichert

Exercice 1 : Écrire en Caml une fonction `maximum_ABR` qui prend en entrée un arbre binaire de recherche et qui retourne la valeur maximale y figurant. Donner aussi sa signature. Définir au préalable un type adapté pour les arbres binaires de recherche.

Exercice 2 : Écrire en Caml ou en pseudo-code une fonction `rechercher_ABR` qui prend en entrée un arbre binaire de recherche et une valeur et qui retourne la liste des directions prises pour trouver l'élément en question, le premier élément de la liste étant la direction prise depuis la racine, et les directions étant des booléens : true pour aller à droite et false pour aller à gauche. Si l'élément existe en plusieurs exemplaires, n'importe laquelle des listes possibles convient, et s'il n'est pas dans l'arbre, il faut déclencher une erreur.

À titre d'exemple, pour trouver la valeur `13` dans un arbre binaire de recherche en passant par les nœuds `1`, `42`, `19` et `13`, la liste renvoyée sera `[true; false; false]`.

Exercice 3 : Tri par tas.

Le principe est simple : on dispose d'une liste et on retourne sa version triée en créant un tas-min contenant tous les éléments de la liste en question puis en retirant successivement tous les éléments minimaux du tas en question. Pour éviter de consommer trop de temps, il s'agit de disposer de tas modifiables, donc au choix des tas presque complets, pour lesquels les opérations élémentaires devront maintenir cette structure, plus faciles à implémenter, ou le type enregistrement proposé ci-dessous.

```
type 'a tas_binaire = { mutable vide : bool; mutable etiquette : 'a ;
mutable gauche : 'a tas_binaire; mutable droite : 'a tas-binaire; };;
```

Un tas vide a sa rubrique `vide` à `true`, et on ignore alors les autres rubriques.

Question 1 : Avec l'implémentation au choix, écrire les opérations de base pour les tas.

Question 2 : Écrire alors simplement le tri par tas.

Question 3 : Estimer la complexité en admettant la propriété du cours sur la hauteur des tas (qui est de toute façon vérifiée sur les tas presque complets).

Exercice 4 : Arbres binaires et occurrences.

Il est possible de représenter un arbre binaire sous forme d'occurrences. Dans ce cas, si un nœud a pour occurrence μ, alors son fils gauche aura pour occurrence $\mu 0$ et son fils droit $\mu 1$.

Nous nous proposons donc de représenter un arbre binaire à l'aide d'une chaîne de caractères constituée des diverses occurrences de l'arbre séparée les unes des autres par une virgule. Nous représenterons la racine par E (l'occurrence vide).

Soit l'arbre B défini, sous forme d'occurrences et représenté à l'aide d'une chaîne de caractères de la manière suivante : `B = "E,0,1,00,01,10,11,001,101,1010,1011"`

Question 1 : Comment, sans représenter l'arbre B, peut-on déterminer sa taille, sa hauteur, le fait qu'un de ses nœuds soit une feuille, la somme des hauteurs de ses nœuds et la hauteur moyenne de ses nœuds ?

Question 2 : Représenter graphiquement l'arbre B.

Exercice 5 : Mettre les expressions booléennes suivantes en forme normale disjonctive et éventuellement simplifier.

- $(a \vee \neg b) \wedge (b \vee \neg c) \wedge (c \vee \neg a)$
- $(a \vee b \vee c) \wedge (a \vee \neg b \vee c) \wedge (a \vee \neg b \vee \neg c)$
- $(\neg a \vee b) \wedge (a \vee b \vee c) \wedge \neg c$