Sujet 1 (06/11/2024)

contact : jean-baptiste.doderlein@ens-rennes.fr

Question de cours

1. Rappeler la définition d'un arbre binaire de recherche. Dégfinir un type OCaml `'a abr`.

2. Écrire une fonction `recherche : 'a abr -> 'a -> bool`. En quoi cette fonction diffère d'une fonction de recherche dans un arbre binaire ? Est-elle plus rapide dans tous les cas ?

Exercice 1 : Vérification de propriétés d'un arbre binaire

1. Écrire une fonction `est_parfait : 'a arbre -> bool` qui vérifie si l'arbre est parfait (tous les niveaux sont complets).

2. Écrire une fonction `est_symetrique : 'a abr -> bool` qui vérifie si l'arbre est symétrique par rapport à son axe vertical (l'arbre est son propre miroir).

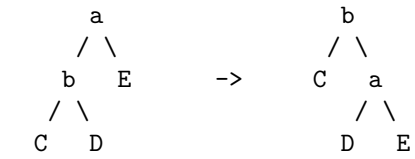
Exercice 2 : AVL

1. Écrire une fonction `inserer_abr : 'a abr -> 'a -> 'a abr` qui insère un élément dans un ABR.

Pour permettre un accès plus rapide des éléments d'un ABR, on cherche à *équilibrer l'arbre*. Ici, on utilise le critère suivant : Un arbre binaire est équilibré si la hauteur du sous arbre gauche et celle du sous arbre droit diffèrent au plus de un.

2. Donner un exemple d'un ABR équilibré. **3.** Écrire la fonction `hauteur : 'a abr -> int`.

La rotation d'un arbre binaire de recherche consiste à faire remonter un nœud dans l'arbre et à en faire redescendre un autre, tout en conservant la propriété d'ordre des éléments de l'ABR. Par exemple une rotation à droite donne :



4. Écrire les fonctions `rotation_droite : 'a abr -> 'a abr` et `rotation_gauche : 'a abr -> 'a abr`.

On souhaite réaliser une insertion dans un ABR équilibré. Pour cela on va utiliser la fonction `inserer_abr` puis rééquilibrer avec des rotations.

5. Est-il toujours possible de rééquilibrer un ABR après une insertion avec une rotation ? deux rotations ? Donner un exemple où deux rotations sont nécessaires.

6. Écrire `equilibre : 'a abr -> 'a abr` qui équilibre un ABR après une insertion.

Sujet 2 (06/11/2024)

contact : jean-baptiste.doderlein@ens-rennes.fr

Question de cours

Donner le type OCaml d'un arbre binaire. Écrire la fonction `dfs : 'a arbre -> 'a list` qui effectue un parcours en profondeur postfixe d'un arbre.

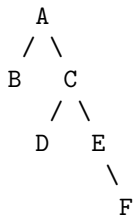
Exercice 1 : k plus grands éléments d'un ABR

1. Rappeler la définition d'un ABR.

2. Écrire une fonction `kpg : 'a arbre -> int -> 'a list` qui renvoie les k plus grands éléments d'un ABR.

Exercice 2 : Plus court chemin entre feuilles

On cherche à établir dans un arbre binaire le plus court chemin entre deux feuilles.



Par exemple dans cet arbre, les nœuds D et F sont à 3 de distance par le chemin `[D;C;E;F]`.

On définit l'ancêtre commun le plus proche x (ACPP) de deux nœuds u et v comme le nœud de profondeur maximale tel que u et v sont dans les sous-arbres de x.

1. Montrer que pour l'ACPP de deux feuilles u et v, u et v n'appartiennent pas au même sous-arbre.

2. Montrer que le chemin entre deux feuilles passe nécessairement par leur ACPP.

3. Écrire `acpp : 'a arbre_binaire -> 'a -> 'a -> 'a` qui renvoie l'ACPP de deux feuilles.

4. Écrire une fonction `pcc : 'a arbre_binaire -> 'a -> 'a -> int` qui calcule la distance entre deux feuilles, en supposant que les feuilles soient bien dans l'arbre.

Exercice 3 : Bijection entre arbres

Proposer une transformation entre arbres généraux et arbres binaires et inversement. Illustrer cet transformation sur un exemple. Implémenter:

- `general_to_binaire : 'a arbre_generaux -> 'a arbre_binaire`
- `binaire_to_general : 'a arbre_binaire -> 'a arbre_binaire`

Si `arbre` est un arbre général, on doit avoir `arbre = binaire_to_general (general_to_binaire arbre)`.

Sujet 3 (06/11/2024)

contact : jean-baptiste.doderlein@ens-rennes.fr

Question de cours

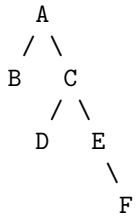
Donner le principe de parcours en profondeur, et les 3 variantes de celui-ci. Illustrer un parcours en profondeur infixe sur un arbre d'au moins 6 nœuds.

Exercice 1 : Diamètre d'un arbre

Le diamètre d'un arbre est défini comme la plus grande distance entre deux nœuds de l'arbre. Autrement dit, c'est la longueur du chemin le plus long entre deux nœuds quelconques de l'arbre.

1. Rappeler le type OCaml d'un arbre binaire.

2. Donner le diamètre de l'arbre suivant :



3. Écrire une fonction `diametre : 'a arbre_binaire -> int` qui calcule le diamètre d'un arbre binaire.

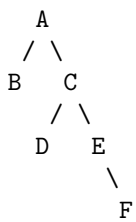
Exercice 2 : Sérialisation d'arbre binaire

On cherche à retrouver la forme d'un arbre binaire à partir de ses parcours en profondeur préfixe et infixe.

1. Donner un type `'a arbre` d'arbre binaire en OCaml

2. Écrire `dfs_prefixe : 'a arbre -> 'a list`, le parcours en profondeur préfixe d'un arbre.

3. Donner les parcours en profondeur préfixe et infixe de



4. Écrire une fonction `cut : 'a list -> 'a -> 'a list * 'a list`, qui pour une liste `l` et un élément `x` dans `l` renvoie la liste des éléments avant `x` dans `l` et la liste des éléments après `x` dans `l`. Par exemple `cut [1;2;3;4] 3` renvoie `[1;2], [4]`.

5. Écrire une fonction `first : 'a list -> int -> 'a list * 'a list` qui à partir d'une liste `l` et d'un entier `n`, renvoie une liste contenant les `n` premiers éléments et une liste contenant les éléments restant.

6. Écrire `reconstruire : 'a list -> 'a list -> 'a arbre` qui à partir du parcours en profondeurs préfixe et infixes reconstruit l'arbre.