Algorithmique des structures de données arborescentes

Feuille d'exercices 3

1 Preuves de propriétés par récurrence

Exercice 3.1 Un arbre binaire t est appelé un *AVL* si, pour ***tout*** nœud x de l'arbre t, la différence entre la hauteur du sous-arbre gauche de x et la hauteur du sous-arbre droit de x est comprise entre -1 et 1.

Le terme AVL provient des initiales des chercheurs ayant introduit ces arbres, Georgii Adelson-Velsky et Evguenii Landis. Pour un entier $h \geqslant 0$, on note $\mathsf{m}(h)$ le nombre ***minimal*** de nœuds dans un AVL de hauteur h.

1. Calculer $\mathsf{m}(0)$, $\mathsf{m}(1)$, $\mathsf{m}(2)$ et $\mathsf{m}(3)$.
2. Trouver une relation de récurrence liant $\mathsf{m}(h+2)$, $\mathsf{m}(h+1)$ et $\mathsf{m}(h)$.
3. On définit la suite $(u_h)_{h\in\mathbb{N}}$ par $u_h = 1 + \mathsf{m}(h)$. Quelle relation de récurrence la suite u_h satisfait-elle ?
4. En déduire qu'il existe une constante $\alpha > 0$ telle que $\mathsf{m}(h) > \alpha(\sqrt{2})^h$.
5. (**Facultatif**) Montrer qu'il existe une constante $\beta > 0$ telle que $\mathsf{m}(h) > \beta \cdot \varphi^h$, où $\varphi = \dfrac{1+\sqrt{5}}{2} \approx 1\,.618$.
6. En déduire qu'il existe une constante $\gamma > 0$ telle que tout AVL à n nœuds a une hauteur au maximum $\gamma \cdot \log_2(n)$.

Cette dernière propriété est importante pour obtenir des algorithmes très efficaces de recherche/insertion/suppression d'une valeur dans un ensemble.

2 TD machine

Exercice 3.2 Dans cet exercice, on utilise le type `'a tree` suivant :

```
type 'a tree = Empty | Bin of 'a * 'a tree * 'a tree
```

Si `p : int -> int -> bool` est un prédicat qui associe un booléen à deux entiers, on dit qu'un arbre `t` est `p`-équilibré si, pour tout nœud `x` de `t` , on a `p h_left h_right` , en notant `h_left` la hauteur du sous-arbre gauche de `x` et `h_right` la hauteur de son sous-arbre droit.

1. Écrire une fonction

   ```
   is_balanced : (int -> int -> bool) -> 'a tree -> bool
   ```

   telle que `is_balanced p t` retourne `true` si `t` est `p`-équilibré et `false` sinon.
2. Utiliser la fonction `is_balanced` pour écrire une fonction `is_perfect : 'a tree -> bool` qui teste si un arbre est parfait.
3. En utilisant la fonction `is_balanced` , écrire une fonction `is_avl : 'a tree -> bool` qui teste si un arbre binaire est un AVL.