Travaux pratiques

option informatique

Arbres rouge-noir

Un arbre *bicolore* est un arbre binaire de recherche comportant une information supplémentaire par nœud : sa couleur, qui peut être rouge ou noire. Ceci nous conduit à définir les types :

```
type couleur = Rouge | Noir ;;

type bicolore = Nil | Noeud of couleur * bicolore * int * bicolore ;;
```

Un arbre bicolore est un arbre *rouge-noir* s'il satisfait aux propriétés suivantes :

- chaque nœud est soit rouge, soit noir ;
- la racine est noire ;
- si un nœud est rouge, ses enfants sont noirs ;
- pour un nœud donné, tous les chemins le reliant à une feuille **`Nil`** contiennent le même nombre de nœuds noirs.

Les arbres rouge-noir garantissent qu'aucun des chemins n'est plus de deux fois plus long que n'importe quel autre, ce qui rend l'arbre équilibré (voir exercice 6 pour les détails). Une conséquence immédiate est que les opérations relatives aux ABR peuvent être implémentées en $O(\log n)$ lorsque $n = |A|$ si les opérations d'insertion et de suppression préservent la structure d'arbre rouge-noir.

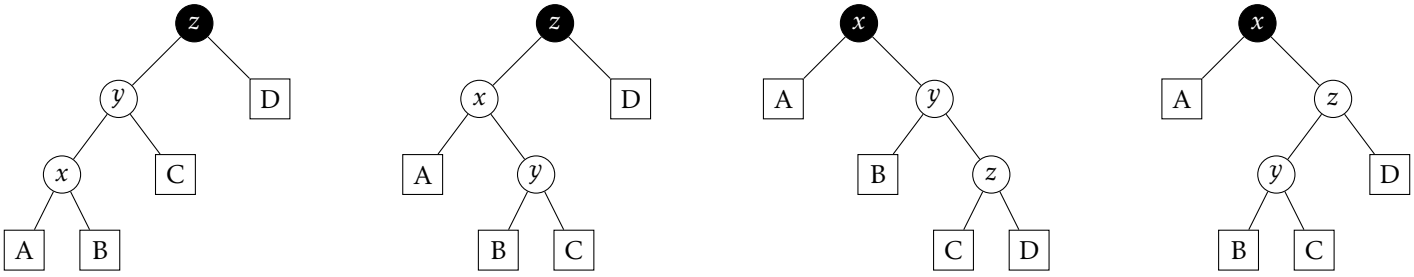
Question 1. On appelle *hauteur noire* d'un arbre rouge-noir A le nombre de nœuds noirs d'un chemin partant de la racine vers une feuille **`Nil`**, et on la note $b(A)$.
Rédiger une fonction **`hauteurnoire`** de type *bicolore -> int* qui calcule la hauteur noire d'un arbre rouge-noir.

Question 2. Rédiger une fonction **`rougenoir`** de type *bicolore -> bool* qui détermine si un arbre bicolore est un arbre rouge-noir (en un seul parcours de l'arbre).

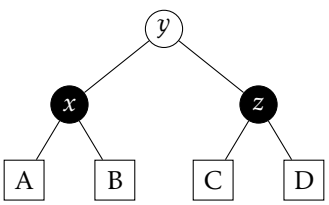
Question 3.

1. Rédiger une fonction **`insereABR`** de type *int -> bicolore -> bicolore* qui insère au niveau des feuilles un nouvel élément dans un arbre rouge-noir. Cette fonction devra préserver la structure d'ABR mais sans se soucier de la structure rouge-noir. On attribuera arbitrairement la couleur rouge au nouveau nœud.
2. Lors de l'ajout via la fonction **`insereABR`**, quelle(s) propriété(s) des arbres rouge-noir peuvent être violées ?

On souhaite rééquilibrer l'arbre afin qu'il redevienne rouge-noir. Pour cela, toute configuration de trois nœuds d'un des quatre types suivants [1] :



est transformée en :



Une telle transformation sera appelée *correction rouge* par la suite.

1. Sur ces dessins les nœuds rouges sont coloriés en blanc.

Question 4. Rédiger une fonction **`correction_rouge`** de type *arbre –> arbre* qui applique une correction rouge à un arbre de l'une des quatre formes présentées ci-dessus, et qui renvoie l'arbre inchangé dans les autres cas.

Question 5. En déduire une nouvelle fonction d'insertion de type *int –> arbre –> arbre* baptisée **`insere`** qui insère un élément dans un arbre rouge-noir tout en préservant la structure rouge-noir.

Question 6. Rédiger enfin une fonction **`test`** de type *int –> bicolore* qui crée un arbre rougenoir en insérant successivement les entiers de 1 à n à l'aide de la fonction **`insere`**.
Vérifier à l'aide de la fonction **`rougenoir`** que l'arbre obtenu pour $n = 32768$ est bien un arbre rouge-noir. Quelle est sa hauteur noire ?