Binary Search Trees (Arbres binaires de Recherche)

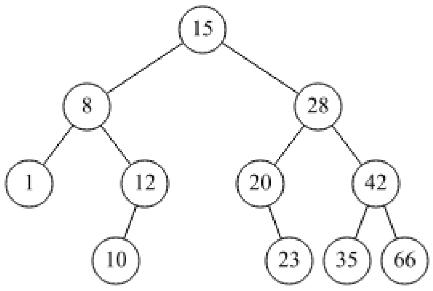

Figure 1: Binary Search Tree (BST) `B1`

1 Tools

Exercise 1.1 (List ↔ BST)

1. Write a function that builds a sorted list with the elements of a binary search tree.
2. Write a function that builds a *balanced* binary search tree using a sorted list.

Exercise 1.2 (Test)

Write a function that tests whether a binary tree is a search tree or not.

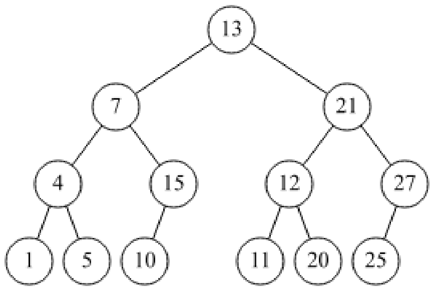

Figure 2: ABR ?

2 Classics

Exercise 2.1 (Researches)

1. (a) Where are the maximum and the minimum values in a non-empty binary search tree?
   (b) Write the two functions `minBST(`B`)` (recursive) and `maxBST(`B`)` (iterative), where B is a non empty BST.
2. Write a function that searches a value x in a binary search tree. It returns the tree whose root contains x if found, the value `None` otherwise.
   Two versions for this function: recursive and iterative!

Exercise 2.2 (Insertion at the leaf)

1. Use the insertion at the leaf principle to create the binary search tree obtained, from an empty tree, by successive insertions of the following values (in that order):
$$13, 20, 5, 1, 15, 10, 18, 25, 4, 21, 27, 7, 12$$
2. Write a recursive function that adds an element to a binary search tree.
   Bonus: write an iterative version of the insertion function.

Exercise 2.3 (Deletion)

Write a recursive function that deletes an element from a binary search tree.

Exercise 2.4 (Root insertion)

1. Use the root insertion principle to create the binary search tree obtained, from an empty tree, by successive insertions of the following values (in that order):
$$13, 20, 5, 1, 15, 10, 18, 25, 4, 21, 27, 7, 12$$
2. Write the function that inserts an element in a binary search tree as a new root.

TU VOIS, IDÉFIX ! CETTE FOIS, J'AI PENSÉ À TOI, JE N'AI PAS DÉRACINÉ L'ARBRE !
373