Solutions for Exercise Sheet 9

Handout: Nov 14th — Deadline: Nov 28th, 4pm

Question 9.1 (1 mark)

1. Prove by induction that every complete binary tree of height h has $2^h - 1$ internal nodes.

   Solution. Proof by induction on h.

   - Base case: For $h = 0$, the tree has only a root node and $0 = 2^0 - 1$ internal nodes.
   - Induction step: assume that the claim holds for h, and we want to prove it for $h + 1$. A tree of height $h + 1$ has two complete binary subtrees of height h, each of which has $2^h - 1$ internal nodes by the induction hypothesis. Constructing the complete binary tree of height h adds one new internal node (the root). This yields $2 \cdot (2^h - 1) + 1 = 2 \cdot 2^h - 1 = 2^{h+1} - 1$ internal nodes.

2. Prove by induction that in every full nonempty binary tree the number of leaves is one more than the number of internal nodes.

   Solution. Proof by induction on the number n of internal nodes.

   - Base case: If there is no internal node and the tree is nonempty then it must consist of one leaf (the root).
   - Induction step: Suppose the claim holds for all graphs with up to n internal nodes. We must show it for a tree T with $n + 1$ internal nodes. In that case the root is an internal node with precisely two nonempty immediate subtrees (as the tree is full). Suppose the left subtree has n_L internal nodes and the right subtree has n_R internal nodes. Then $n = n_L + n_R$ and we can apply the induction hypothesis to both subtrees. Thus $l_L = n_L + 1$ and $l_R = n_R + 1$, where l_L and l_R stand for the number of leaves in the left and right subtree, respectively. Obviously, $l_L + l_R = l$, which is the number of leaves in T. Hence $l = n_L + n_R + 2 = (n + 1) + 1$, which proves the claim.

3. Prove by induction that every nonempty binary tree satisfies $|V| = |E| + 1$.

   Solution. Proof by induction on $n = |V|$ for a tree T.

   - Base case: For $n = 1$, T has only a root node, hence $|E| = 0$.
   - Induction step: Suppose $|V| = |E| + 1$ for $|V| \leq n$ (Induction hypothesis). We must prove the claim for $|V| = n + 1$.

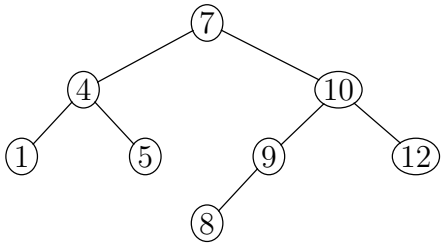
     There are three subcases.

     - The root of T has only a left child. Then this child has n nodes and therefore $n - 1$ edges by the induction hypothesis. Since T adds one single edge from the root to its left child, it has n edges (and $n + 1$ nodes by assumption).
     - The case where the root of T has only a right child is symmetric.

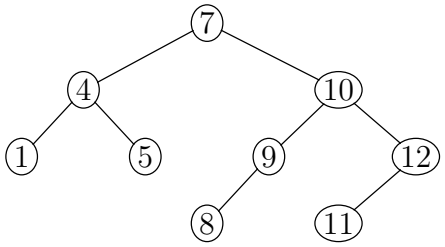
– The root of T has two children. Suppose the left subtree has n_L nodes and the right subtree has n_R nodes. Then $n = n_L + n_R$ and it follows that $n_L < n+1$ and $n_R < n+1$. We can therefore apply the induction hypothesis to both subtrees, which yields $e_L = n_L - 1$ and $e_R = n_R - 1$, writing e_L for the number of edges in the left subtree of T and e_R for the number of edges in its right subtree. Hence, by substitution, $n = e_L + e_R + 2 = |E|$, and therefore $|E| = |V| - 1$.

Question 9.2 (0.25 marks)

1. Insert a node with key 11 into the following binary search tree. Give a step-by-step explanation.

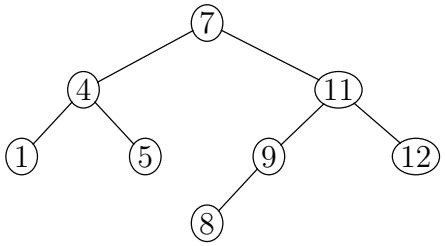


Solution. The algorithm for insertion inspects the path $(7, 10, 12)$ and then inserts 11 as a left child of 12.



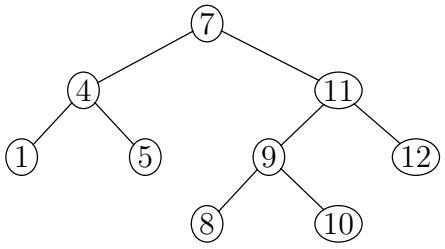
2. Delete the node with key 10 into the resulting binary search tree. Give a step-by-step explanation.

Solution. The algorithm for deletion identifies the successor of 10 in its right subtree which is 11. Then 12's left child points to NIL since 11 had no right child, and 11 replaces 10 and connected to its children.



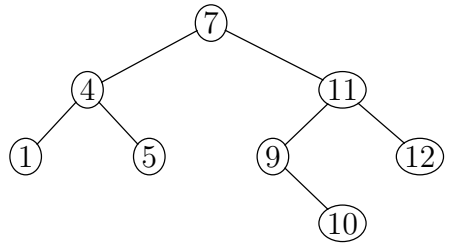
3. Insert a node with key 10 into the resulting binary search tree. Give a step-by-step explanation.

Solution. The algorithm for insertion inspects the path $(7, 11, 9)$ and then inserts 10 as a right child of 9.



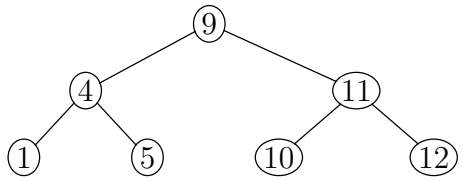
4. Delete the node with key 8 from the resulting binary search tree. Give a step-by-step explanation.

   Solution. The algorithm for deletion simply transplants 8 with an empty tree NIL since 8 was a leaf.



5. Delete the node with key 7 from the resulting binary search tree. Give a step-by-step explanation.

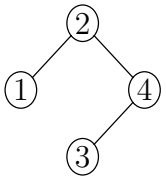
   Solution. The algorithm for deletion identifies 9 as the successor of 7 by inspecting 7's right subtree. Then the right child of 9 becomes the left child of 9's parent (11) and 9 gets transplanted in place of 7 and connected to its children.



Question 9.3 (0.25 marks)

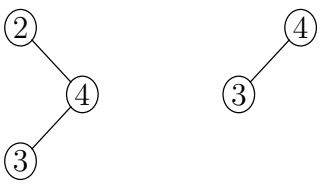
Delete two different nodes in different order from a binary search tree (e.g. first node x and then node y, or alternatively first node y and then node x). Can the resulting trees be different? Explain your answer.

Solution. The resulting trees can be different. The following minimal example proves this claim. To explain that in detail the concrete code in DELETE needs to be inspected. Consider the following BST.



Deleting 2 and then 1 yields the following BSTs.

Deleting 1 and then 2 yields the following BSTs.



Question 9.4 (0.25 marks)

Write the TREE-PREDECESSOR(x) procedure.

Solution.

TREE-PREDECESSOR(x)

```
1: if x.left ≠ NIL then
2:     return TREE-MAXIMUM(x.left)
3: else
4:     y = x.p
5:     while y ≠ NIL and x == y.left do
6:         x = y
7:         y = y.p
8:     return y
```

Question 9.5 (0.25 marks)

You can sort a set of n numbers by the following procedure:

1. Build a binary search tree by inserting each element using TREE-INSERT (n times)
2. Print the numbers in sorted order by an INORDER tree walk.

What are the worst case and best case runtimes of this sorting algorithm?

Solution.

The worst case runtime occurs if the elements are already sorted. Then when inserting element i we need to traverse $i-1$ nodes. This leads to a runtime of $c_0 + \sum_{i=1}^{n-1} c \cdot i = \Theta(n^2)$. Then the INORDER procedure runs in $\Theta(n)$.

The best case is when the tree is balanced such that it's height is $O(\log n)$, leading to an $O(n \log n)$ runtime to insert n elements. Then the INORDER procedure runs in $\Theta(n)$.

Question 9.6 (1 mark)

1. Implement a Binary Tree using a linked list (as explained during the lecture) to encode a mathematical expression with binary operators (+,-,*,/) provided in input in prefix notation i.e., functional programming notation. You should use a stack to keep track of the pointers of the nodes that you will need later to fill in their right child.
2. Implement the procedures INORDER, PREORDER and POSTORDER and print the respective outputs when applied to the resulting tree from Step 1.

Example input: $-+a*bc/de$ which leads to the tree given in the lecture slides.