Data Structure and Algorithm Analysis(H)
Southern University of Science and Technology
Mengxuan Wu
12212006

Work Sheet 9

Mengxuan Wu

Question 9.1

1.

Base case:

A complete binary tree with only one node has height 0 and 0 internal nodes. Thus, $h = 0$ and $i = 2^h - 1 = 0$ holds.

Inductive step:

Suppose for a complete binary tree with h height, the number of internal nodes is $i = 2^h - 1$. For a complete binary tree with $h + 1$ height, the number of internal nodes is the internal points of two subtrees plus the root, which is $2^h - 1 + 2^h - 1 + 1 = 2^{h+1} - 1$.

Thus, the statement holds for every complete binary tree.

2.

Base case:

For a full binary tree with only one node, the number of leaves is 1, and the number of internal nodes is 0. Thus, $l = i + 1$ holds.

Inductive step:

Suppose for a full binary tree with l leaves, the number of internal nodes is i, and $l = i + 1$ holds. To keep the binary tree full, every time we add nodes to the tree, we add at two nodes to a leaf node. Then, $l' = l - 1 + 2 = l + 1$ and $i' = i + 1$. Thus, $l' = i' + 1$ holds.

Thus, the statement holds for every full non-empty binary tree.

3.

Base case:

For a binary tree with only one node, the number of edges is 0, and the number of vertices is 1. Thus, $|V| + |E| + 1$ holds.

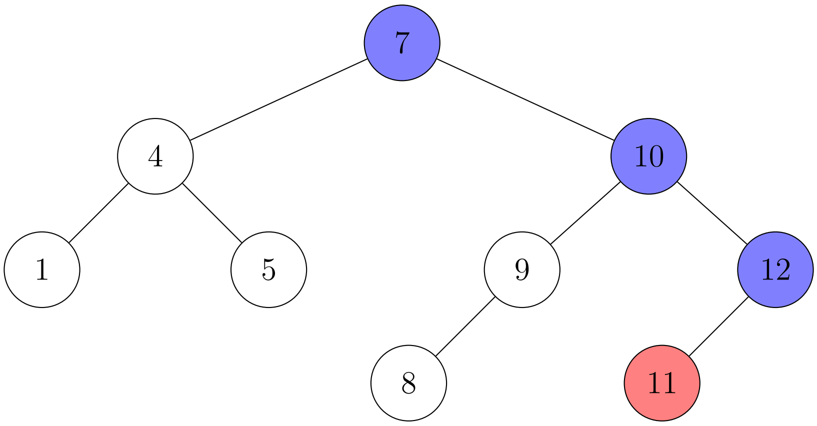
Inductive step:

Every time we add a node to the tree, we add one edge and one vertex. Thus, $|V| + |E| + 1$ holds.

Thus, the statement holds for all non-empty binary trees.

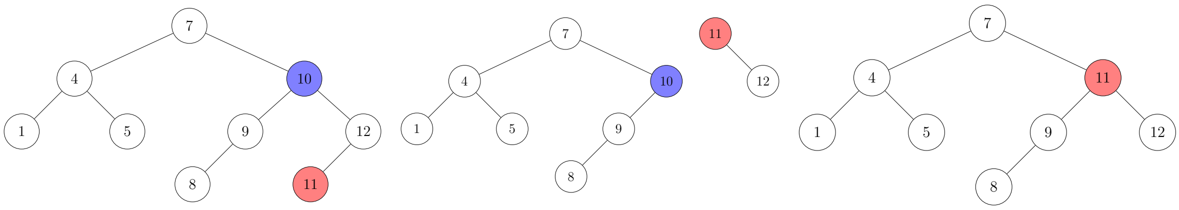
Question 9.2

1.



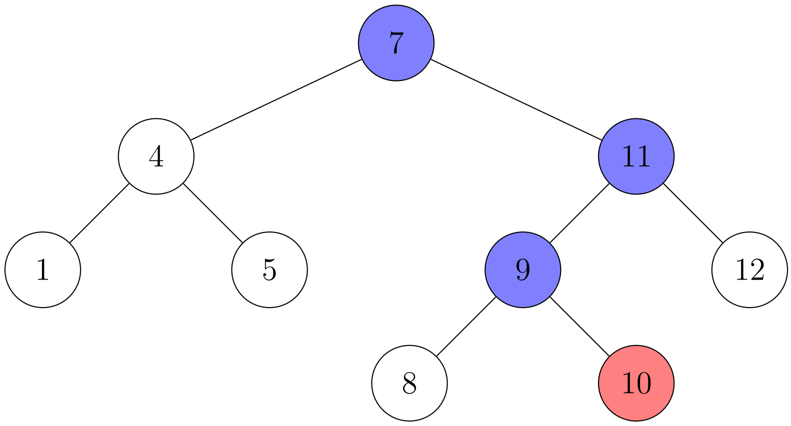
- Step 1: Compare 7 and 11 and find 11 is bigger, go to the right subtree.
- Step 2: Compare 10 and 11 and find 11 is bigger, go to the right subtree.
- Step 3: Compare 12 and 11 and find 11 is smaller, go to the left subtree.
- Step 4: End up at NIL and insert 11.

2.



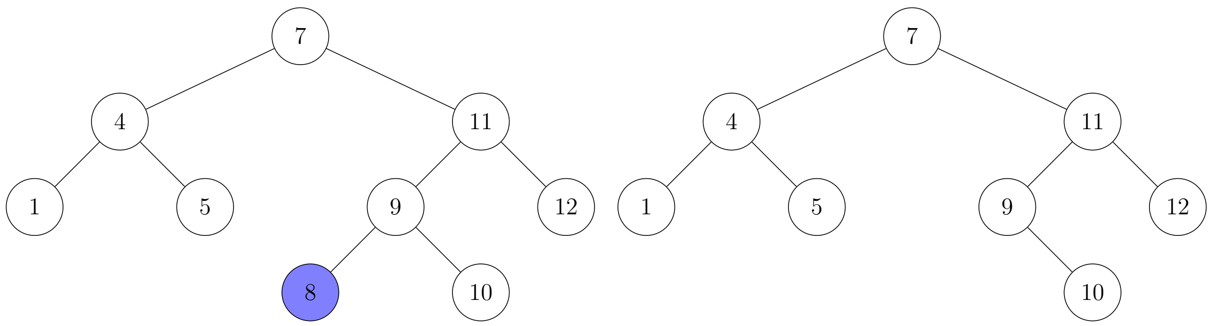
- Step 1: Find the left subtree of 10 is not empty.
- Step 2: Find the right subtree of 10 is not empty.
- Step 3: Find the successor of 10 is 11.
- Step 4: Find the right subtree of 11 is empty, no need to transplant.
- Step 5: Transplant the right subtree of 10 to the right subtree of 11.
- Step 6: Remove 10 and replace it with 11.

3.



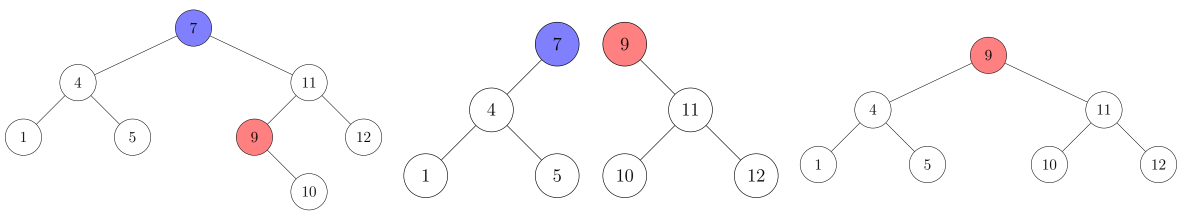
- Step 1: Compare 7 and 10 and find 10 is bigger, go to the right subtree.
- Step 2: Compare 11 and 10 and find 10 is smaller, go to the left subtree.
- Step 3: Compare 9 and 10 and find 10 is bigger, go to the right subtree.
- Step 4: End up at NIL and insert 10.

4.



- Step 1: Find the left subtree of 8 is empty.
- Step 2: The right subtree of 8 is empty, no need to transplant.
- Step 3: Remove 8.

5.

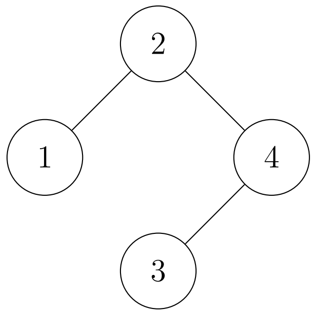


- Step 1: Find the left subtree of 7 is not empty.

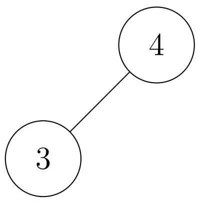
- Step 2: Find the right subtree of 7 is not empty.
- Step 3: Find the successor of 7 is 9.
- Step 4: Find the right subtree of 9 is not empty, transplant the right subtree of 9 to the left subtree of father of 9.
- Step 5: Transplant the right subtree of 7 to the right subtree of 9.
- Step 6: Remove 7 and replace it with 9.

Question 9.3

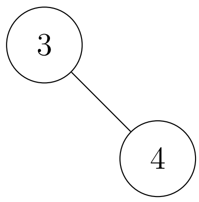
Yes. The result could be different. Here is an example.
For a binary search tree as below:



The order of deleting node 1 and 2 does matter and the results are different.



Delete 1 first



Delete 2 first

Question 9.4

```
TREE-PREDECESSOR(x)
if x.left ≠ NIL
        return TREE-MAXIMUM(x.left)
else
        y = x.p
        while y ≠ NIL and x == y.left
                x = y
                y = y.p
        return y
```

Question 9.5

Base case:

The best case is when the tree is balanced.

For each node x in the tree, it takes $n = x.depth$ times of comparison to insert it. Therefore, the total number of comparison is:

$$\begin{aligned}\sum_{x \in T} n &= \sum_{x \in T} x.depth \\ &= \sum_{i=0}^{\log n} i \cdot 2^i \\ &= (2 \log e)\ n \log n - 2n + 2 \\ &= \Theta(n \log n)\end{aligned}$$

Worst case:

The worst case is when the tree is a linked list.

The total number of comparison is:

$$\begin{aligned}\sum_{x \in T} n &= \sum_{x \in T} x.depth \\ &= \sum_{i=0}^{n} i \\ &= \frac{n(n+1)}{2} \\ &= \Theta(n^2)\end{aligned}$$