COMP9024 19T0

第06a周问题集

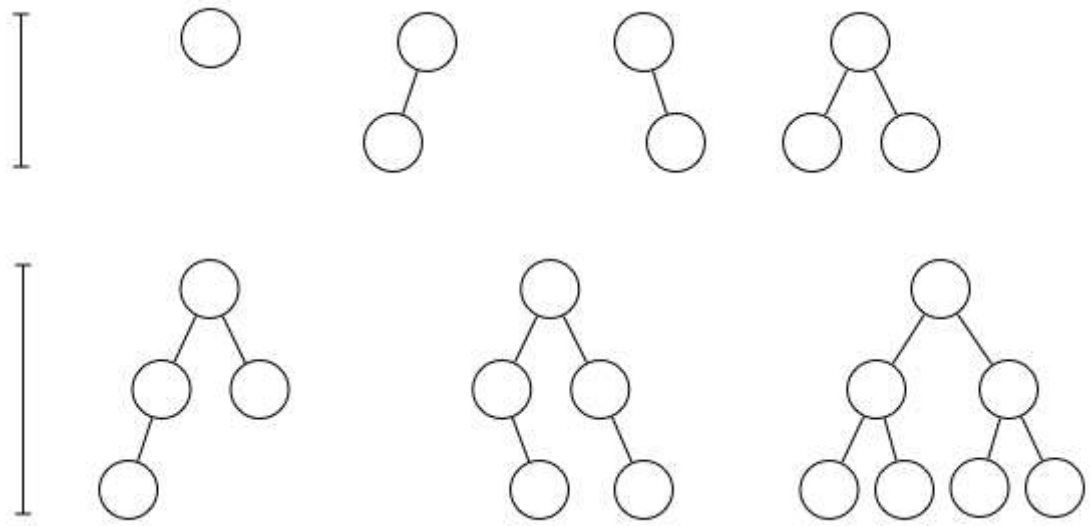
数据结构和算法

搜索树数据结构和算法

[显示没有答案] [显示所有答案]

1. （树属性）

   a. 导出包含*n*个节点的二叉搜索树（BST）的最小高度的公式。回想一下，高度定义为从根到叶子的最长路径上的边数。您可能会发现通过考虑具有最小高度的树的特征开始是有用的。以下图表可能有所帮助：



   b. 在二进制搜索树ADT（BSTree.h，BSTree.c）的讲座中，实现了以下功能：

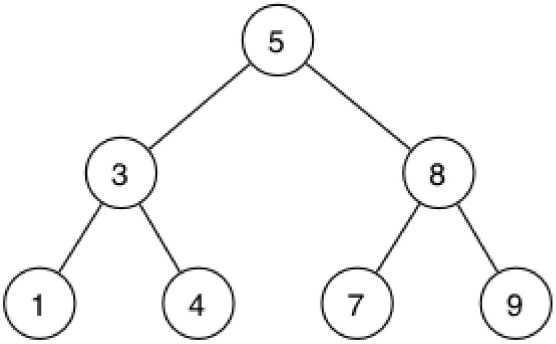
```
int TreeHeight (Tree t) {...}
```

   计算树的高度。

[显示答案]

2. （树遍历）

   考虑以下树及其节点以不同的输出顺序显示：



Infix Order	1 3 4 5 7 8 9
Prefix Order	5 3 1 4 8 7 9
Postfix Order	1 4 3 7 9 8 5
Level Order	5 3 8 1 4 7 9

   a. 什么样的树具有其中缀输出与其前缀输出相同的属性？是否有任何种类的树木，所有四个输出订单都是相同的？

b. 设计二叉搜索树的前缀，中缀和后缀顺序遍历的递归算法。使用伪代码，并定义单个函数 TreeTraversal（树，样式），其中样式可以是“NLR”，“LNR”或“LRN”中的任何一个。

[显示答案]

3. （插入和删除）

在没有讲座的treeLab程序的帮助下回答以下问题。

a. Show the BST that results from inserting the following values into an empty tree in the order given:

```
6 2 4 10 12 8 1
```

Assume "at leaf" insertion.

b. Let `t` be your answer to question a., and consider executing the following sequence of operations:

```
TreeDelete(t,12);
TreeDelete(t,6);
TreeDelete(t,2);
TreeDelete(t,4);
```

Assume that deletion is handled by joining the two subtrees of the deleted node if it has two child nodes. Show the tree after each delete operation.

[show answer]

4. (Insertion at root)

a. Consider an initially empty BST and the sequence of values

```
1 2 3 4 5 6
```

- Show the tree resulting from inserting these values "at leaf". What is its height?
- Show the tree resulting from inserting these values "at root". What is its height?
- Show the tree resulting from alternating between at-leaf-insertion and at-root-insertion. What is its height?

b. Complete the Binary Search Tree ADT (BSTree.h, BSTree.c) from the lecture by an implementation of the function:

```
Tree insertAtRoot(Tree t, Item it) { ... }
```

We have created a script that can automatically test your program. To run this test you can execute the `dryrun` *program that corresponds to the problem set and week. It expects to find a file named* `BSTree.c` *in the current directory. You can use dryrun as follows:*

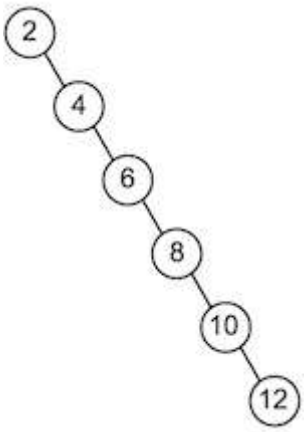
```
prompt$ ~cs9024/bin/dryrun prob06a
```

Note: The `dryrun` *script expects your program to include your implementation for Exercise 1.b.*

[show answer]

5. (Rebalancing)

   Trace the execution of `rebalance(t)` on the following tree. Show the tree after each rotate operation.



   [show answer]

6. **Challenge Exercise**

   The function `showTree()` from the lecture displays a given BST sideways. A more attractive output would be to print a tree properly from the root down to the leaves. Design and implement a new function `showTree(Tree t)` for the Binary Search Tree ADT (`BSTree.h`, `BSTree.c`) to achieve this.

   Please email me your solution. The best solution will be added to our BST ADT implementation and used in the next lecture (week 11). Both the attractiveness of the visualisation and the simplicity of the code will be judged.

   [show answer]