

剑指Offer（二十二）：从上往下打印二叉树

🕒 2017年12月11日 17:08:56 📄 2 🌡 6,543 °C 🖨 编辑



一、前言

本系列文章为《剑指Offer》刷题笔记。

刷题平台：[牛客网](#)

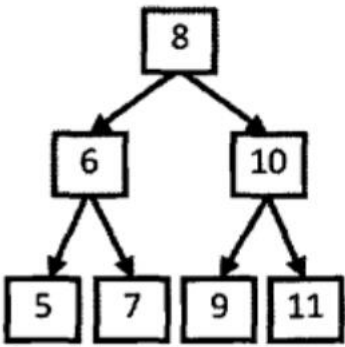
书籍下载：[共享资源](#)

二、题目

从上往下打印出二叉树的每个节点，同层节点从左至右打印。

1、思路

举例说明：



如上图所示，因为按层打印的顺序决定应该先打印根结点，所以我们从树的根结点开始分析。为了接下来能够打印值为8的结点的两个子结点，我们应在遍历该结点时把值为6和10的两个结点保存到一个容器里，现在容器内就有两个结点了。按照从左到右打印的要求，我们先取出为6的结点。打印出值6之后把它的值分别为5和7的两个结点放入数据容器。此时数据容器中有三个结点，值分别为10、5和7。接下来我们从数据容器中取出值为10的结点。注意到为10的结点比值5、7的结点先放入容器，此时又比这两个结点先取出，这就是我们通常说的先入先出，因此不难看出这个数据容器应该是一个队列。由于值为5、7、9、11的结点都没有子结点，因此只要依次打印即可。

整个打印过程如下图所示：

步骤	操作	队列
1	打印结点 8	结点 6、结点 10
2	打印结点 6	结点 10、结点 5、结点 7
3	打印结点 10	结点 5、结点 7、结点 9、结点 11
4	打印结点 5	结点 7、结点 9、结点 11
5	打印结点 7	结点 9、结点 11
6	打印结点 9	结点 11
7	打印结点 11	

通过上面具体例子的分析，我们可以找到从上到下打印二叉树的规律：每一次打印一个结点的时候，如果该结点有子结点，则把该结点的子结点放到一个队列的末尾。接下来到队列的头部取出最早进入队列的结点，重复前面的打印操作，直至队列中所有的结点都打印出来为止。

2、代码

C++:


C++

```
1  /*
2  struct TreeNode {
3      int val;
4      struct TreeNode *left;
5      struct TreeNode *right;
6      TreeNode(int x) :
7          val(x), left(NULL), right(NULL) {
8      }
9  };*/
10 class Solution {
11 public:
12     vector<int> PrintFromTopToBottom(TreeNode* root) {
13         TreeNode* fr;
14         if(root == NULL){
15             return result;
16         }
17         que.push(root);
18         while(!que.empty()){
19             fr = que.front();
20             result.push_back(fr->val);
21             if(fr->left != NULL){
22                 que.push(fr->left);
23             }
24             if(fr->right != NULL){
25                 que.push(fr->right);
26             }
27             que.pop();
28         }
29         return result;
30     }
31 private:
32     vector<int> result;
33     queue<TreeNode*> que;
34 };
```

Python2.7:

Python

```
1  #- coding:utf-8 -*-
2  # class TreeNode:
3  #     def __init__(self, x):
4  #         self.val = x
5  #         self.left = None
6  #         self.right = None
7  class Solution:
8      # 返回从上到下每个节点值列表，例：[1,2,3]
9      def PrintFromTopToBottom(self, root):
10         # write code here
11         if not root:
12             return []
13         result = []
14         tmp = [root]
15         while len(tmp):
16             cur = tmp.pop(0)
17             result.append(cur.val)
18             if cur.left:
19                 tmp.append(cur.left)
20             if cur.right:
21                 tmp.append(cur.right)
22         return result
```



微信公众号

分享技术，乐享生活：微信公众号搜索「JackCui-AI」关注一个在互联网摸爬滚打的潜行者。

生存还是毁灭，这是个问题。--- 莎士比亚