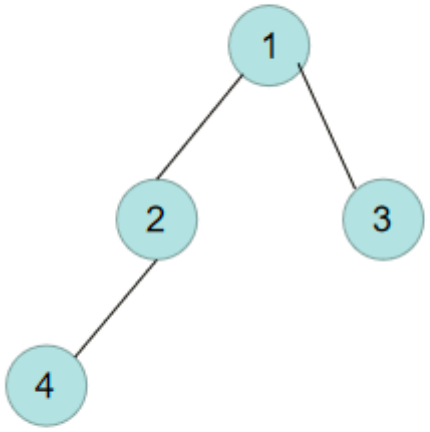
根据二叉树创建字符串

题目描述：

需要采用前序遍历的方式，将二叉树转换为一个由括号和整数组成的字符串。空间点则用一对空括号“()”表示。而且需省略所有不影响字符串与原始二叉树之间的一对一映射关系的空括号对。

实例1：

输入：二叉树：[1,2,3,4]

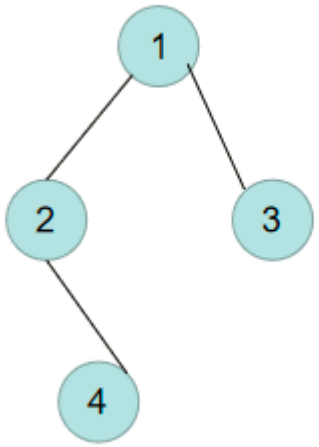


输出："1(2(4))(3)"

解释：原本是“1(2(4))(3())”,在省略所有不必要的空括号对之后，将变为"1(2(4))(3)"

示例2：

输入：二叉树：[1,2,3,null,4]



输出："1(2()(4))(3)"

解题思路：

通过理解题意和对比示例，可以发现二叉树根节点值正常输出，从第二层开始输出数值前需在值前面加"("，且当一直有左子树时递归输出，直到叶子节点才输出")"，右子树同理。但值得注意的是：当一个节点只有左子树没有右子树时，右子树的"()"省略即可，但当一个节点只有右子树而没有左子树时，在遍历输出右子树之前需先输出"()"。

leetcode代码实现：

```
//当前代码实现的是普通二叉树
public class TreeNode{
 int val;
 TreeNode left;
 TreeNode right;
 TreeNode(int x){val=x;}
}
class Solution{
 public String tree2str(TreeNode t){
  StringBuilder res=new StringBuilder();
  tree2str(t,res);
  return res.toString();
 }
private void tree2str(TreeNode node,StringBuilder res){
```

```
15   if(node==null) return;
16   res.append(node.val);
17   if(node.left!=null){
18   res.append("(");
19   tree2str(node.left,res);
20   res.append(")");
21   }
22   else if(node.left==null && node.right!=null){
23   res.append("()");
24   }
25   if(node.right!=null){
26   res.append("(");
27   tree2str(node.right,res);
28   res.append(")");
29   }
30  }
31  }
```

完整代码实现：

```
1  //当前代码实现的是二叉搜索树
2  import java.lang.*;
3  //泛型，继承内部比较接口原因是使其二叉树值具有天然的比较能力
4  //二叉搜索树的值特点(根节点大于左子树的所有数值，小于右子树的所有数值)
5  class Solution<E extends Comparable>{
6   private class TreeNode{
7   E val;
8   TreeNode left;
9   TreeNode right;
10   public TreeNode(E x){
11   val=x;
12   }
13   }
14   private TreeNode root;
15   //向二叉树中添加元素
16   public void add(E e){
17   root=add(root,e);
```

```
18  }
19  //该方法作用为添加完元素后返回根节点(因每添加一个元素，树的结构都发生了变化，返回树结构变化后的根节点，若能得知根节点即可访问所有节点)
20  private TreeNode add(TreeNode node,E e){
21  //当根节点为null时，创建新节点返回
22  if(node==null){
23  TreeNode newNode=new TreeNode(e);
24  return newNode;
25  }
26  //若根节点不为空，当插入元素大于节点元素时，遍历根节点的右子树寻找插入点插入，最后根节点的右子树(node.right)=改变后的右子树
27  if(e.compareTo(node.val)>0){
28  node.right=add(node.right,e);
29  }
30  //左子树同理右子树
31  else if(e.compareTo(node.val)<0){
32  node.left=add(node.left,e);
33  }
34  //最终返回根节点即可
35  return node;
36  }
37  //覆写toString()方法使二叉树能以指定形式输出
38  @Override
39  public String toString() {
40  StringBuilder sb=new StringBuilder();
41  tree2str(root);
42  return sb.toString();
43  }
44  public String tree2str(TreeNode t){
45  StringBuilder res=new StringBuilder();
46  tree2str(t,res);
47  return res.toString();
48  }
49  //实现题目要求的输出形式
50  private void tree2str(TreeNode node,StringBuilder res){
```

```
51   //当根节点为空时直接返回
52   if(node==null) return;
53   //前序遍历需先输出值
54   res.append(node.val);
55   //当左子树不为空时，先输出“(”，再一直递归直到叶子结点，最后再输出")"
56   if(node.left!=null){
57   res.append("(");
58   tree2str(node.left,res);
59   res.append(")");
60   }
61   //当左子树为空且右子树不为空时，左子树部分要输出"()"才可输出右子树
62   else if(node.left==null && node.right!=null){
63   res.append("()");
64   }
65   //当右子树不为空时，同理左子树不为空；但一个节点只有左子树而没有右子树时，右子树处的"()"省略
66   if(node.right!=null){
67   res.append("(");
68   tree2str(node.right,res);
69   res.append(")");
70   }
71   }
72   //将最终输出结果以字符串形式打印即可
73   public void fun(){
74   System.out.println(tree2str(root));
75   }
76  }
77  public class Main{
78   public static void main(String[] args){
79   Solution<Integer> s=new Solution<>();
80   int[] data=new int[]{28,16,30,13,22,29,42,15};
81   for(int i=0;i<data.length;i++){
82   s.add(data[i]);
83   }
84   s.fun();
```

```
85   }
86 }
87 //28(16(13()(15))(22))(30(29)(42));
```