# 题目

给定一个二叉树，返回所有从根节点到叶子节点的路径。

说明: 叶子节点是指没有子节点的节点。

**示例:**

**输入:**

1

/ \

2 3

\

5

输出: ["1->2->5", "1->3"]

解释: 所有根节点到叶子节点的路径为: 1->2->5, 1->3

# 分析

## 方法一：递归法

**思路：**

使用两个全局变量 vec与path。

vec作为最终返回的vector值，path字符串则表示单条路径。

递归终止条件为节点是叶子节点，或者节点为空。

每次递归将当前节点的值记录到path中，并且设置局部变量path2来得到当前路径的副本

如果节点是叶子节点，则说明有一条路径走到了尽头，此时将叶子节点的值追加到path中，再将path放在vec的最后面即完成一条路径的记录。

然后需要通过之前设置的局部变量path2 作为副本 恢复path的值，使其与当前节点路径保持一致

递归循环即可完成

**代码：**

/\*\*

\* Definition for a binary tree node.

\* struct TreeNode {

\* int val;

\* TreeNode \*left;

\* TreeNode \*right;

\* TreeNode(int x) : val(x), left(NULL), right(NULL) {}

\* };

\*/

class Solution {

public:

vector<string> vec;

string path;

vector<string> binaryTreePaths(TreeNode\* root) {

if(!root)

{

return vec;

}

if(!root->left && !root->right)

{

path += to\_string(root->val);

vec.push\_back(path);

return vec;

}

path += to\_string(root->val) +"->";

string path2 = path;

binaryTreePaths(root->left);

path = path2;

binaryTreePaths(root->right);

path = path2;

return vec;

}

};

或：

## 方法二：迭代法/栈