```
/*
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

来源：力扣（LeetCode）
链接：https://leetcode-cn.com/problems/binary-tree-paths
著作权归领扣网络所有。商业转载请联系官方授权，非商业转载请注明出处。
*/
```

分析:

- 递归法:
  - 从非NULL点根节点开始遍历,若当前节点不为叶子节点,就把已经访问过的节点值暂时存在string变量中
  - 若遇到叶子节点,就把叶子节点的值保存到string变量中,然后将string变量压入容器中

方法一:C++_递归法

```
/**
 * 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
{

    private:
        vector<string> ret_val;
        void __pathRoot2Leaves(TreeNode* node,string& s)
        {
            string s1(s);
            s1 += to_string(node->val);
```

```
            if(  (node->left==NULL)
               &&(node->right==NULL)
            )
            {
                ret_val.push_back(s1);
            }

            s1 += "->";
            if(node->left)
            {
                __pathRoot2Leaves(node->left,s1);
            }

            if(node->right)
            {
                __pathRoot2Leaves(node->right,s1);
            }
        }

    public:
        vector<string> binaryTreePaths(TreeNode* root)
        {
            ret_val.clear();
            string s;
            if(root!=NULL)
            {
                __pathRoot2Leaves(root,s);
            }
            return ret_val;
        }
};


/*
执行结果:
通过
显示详情
执行用时 :4 ms, 在所有 C++ 提交中击败了93.07% 的用户
内存消耗 :11.7 MB, 在所有 C++ 提交中击败了80.95%的用户
*/
```

AlimyBreak
2019.09.16