# 题目

给定一个有相同值的二叉搜索树（BST），找出BST中的所有众数（出现频率最高的元素）。

假定BST有如下定义：

结点左子树中所含结点的值小于等于当前结点的值

结点右子树中所含结点的值大于等于当前结点的值

左子树和右子树都是二叉搜索树

例如：

给定BST [1,null,2,2],

1

\

2

/

2

返回[2].

提示：如果众数超过1个，不需考虑输出顺序

进阶：你可以不使用额外的空间吗？（假设由递归产生的隐式调用栈的开销不被计算在内）

# 分析

## 方法一：哈希表

思路：

代码：

class Solution {

private:

unordered\_map<int,int> mp;

int maxNum = 0;

void dfs(TreeNode \*root)

{

if(root->left) dfs(root->left);

mp[root->val]++;

if(mp[root->val] > maxNum)

maxNum = mp[root->val];

if(root->right) dfs(root->right);

}

public:

vector<int> findMode(TreeNode\* root) {

if(nullptr==root) return {};

dfs(root);

vector<int> nums;

for(auto item: mp)

{

if(item.second == maxNum)

{

nums.push\_back(item.first);

}

}

return nums;

}

};

如果该题目做一下修改，即打印所有众数：

class Solution {

private:

unordered\_map<int,int> mp;

int maxNum = 0;

void dfs(TreeNode \*root)

{

if(root->left) dfs(root->left);

mp[root->val]++;

if(mp[root->val] > maxNum)

maxNum = mp[root->val];

if(root->right) dfs(root->right);

}

public:

vector<int> findMode(TreeNode\* root) {

if(nullptr==root) return {};

dfs(root);

vector<int> nums;

for(auto item: mp)

{

if(item.second == maxNum)

{

int k = maxNum;

while(k--)

nums.push\_back(item.first);

}

}

return nums;

}

};