Path Sum

Question Solution

Given a binary tree and a sum, determine if the tree has a root-to-leaf path such that adding up all the values along the path equals the given sum.

For example:  
Given the below binary tree and sum = 22,

5

/ \

4 8

/ / \

11 13 4

/ \ \

7 2 1

return true, as there exist a root-to-leaf path 5->4->11->2 which sum is 22.

首先，这个问题的意思要弄清楚，它指的是能否找到“根”到“叶”的和等于sum的路径。

一开始我理解成了从根往下找，能否找到一条路径使得各个节点的和等于sum。这两个理解是完全不一样的。

OK， 所以这个问题关键就是要求出所有从根到叶子节点和。

分析：

很明显可以用递归来读取二叉树，可以从根开始往下找，能有一条路径的和等于sum，那么返回TRUE；

退出条件：

* 如果根是NULL，那么很明显，无法再往下查找，返回false
* 如果当前的节点值等于sum，并且这个节点是叶子节点，那么返回TRUE；
* 其他情况，递归查找左边的叶子和右边叶子

这个问题是一个“深度优先搜索”（DFS）。DFS的基本含义是找出每一个有效的路径，然后转而向上求下一个有效路径。深度优先搜索所遵循的策略就是尽可能“深”的在图中进行搜索，对于图中某一个顶点V，如果它还有相邻的顶点（在有向图中就是还有以V为起点的边）且未被访问，则访问此顶点。如果找不到，则返回到上一个顶点。这一过程一直进行直到所有的顶点都被访问为止。  DFS可以搜索出从某一个顶点到另外的一个顶点的所有路径。  由于要进行返回的操作，我们采用的是递归的方法。

与之相对的是广度优先搜索（BFS）：

BFS的思想：  
   从一个图的某一个顶点V0出发，首先访问和V0相邻的且未被访问过的顶点V1、V2、……Vn，然后依次访问与V1、V2……Vn相邻且未被访问的顶点。如此继续，找到所要找的顶点或者遍历完整个图。  
   由此可以看出，用BFS进行搜索所搜索的顶点都是按深度进行扩展的，先找到到V0距离为1的所有顶点，然后找到距离V0为2的顶点……所以BFS所搜索到的都是最短的路径。  
   由于要将距离V0为d(d>0)的且未被方位的点都记录起来，我们采用队列这种数据结构。队列的特点是先进先出(FIFO)，从某个顶点出发，记此顶点已访问标记，然后依次搜索和此顶点相邻的且未被访问的顶点，将其加入队列，并置已访问标记，重复此步骤，直到找到需要搜索的顶点或者所有的顶点都被访问为止。

广度搜索的判断重复如果直接判断十分耗时，我们一般借助[哈希表](http://baike.baidu.com/view/329976.htm)来优化[时间复杂度](http://baike.baidu.com/view/104946.htm)。