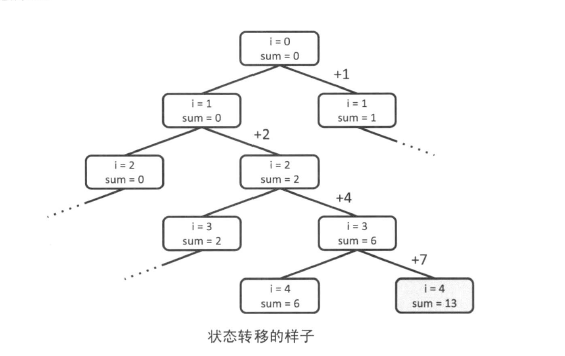
我觉得在解决问题的时候，不能够仅仅

对于部分和问题，非常显然，它的解空间是一个二叉树，如下：



在学习《数据结构》的时候，我们有重点学习过二叉树的遍历的递归写法：

1. **void** preorder(bintree t){
2. **if**(t){
3. printf("%c ",t->data);
4. preorder(t->lchild);
5. preorder(t->rchild);
6. }
7. }

因此，解部分和问题就转变成了对这个问题的解空间进行遍历的问题了。

因此我得出了求部分和问题的程序：

//已经从前i项得到了和sum，然后对于i项之后的进行分支

bool dfs(int i,int sum){

//如果前n项都计算过了，则返回sum是否和k相等

if(i==n)return sum==k;//如果if条件成立，则函数就会执行return语句，下面的语句就不会被执行，它就会返回上一层的递归函数

//不加上a[i]的情况

if(dfs(i+1,sum))return true;//如果在这个分支中找到了解，则函数就会退出而不执行下面的语句

//加上a[i]的情况

if(dfs(i+1,sum+a[i]))return true;

//无论是否加上a[i]都不能够凑成k就返回false

return false;

}

显然，部分和问题的原型是二叉树的遍历。

从问题的解空间入手，我能够很快地写出求解该问题的程序，而不是浮在问题的表面。

函数执行过程的理解：

函数的执行过程是在栈中进行的，递归函数也是这样的。

二叉树式递归是多叉递归的基础。

二叉树式递归函数，它有两个分支。如果第一个分支没有找到解，即返回值为false，那么它就不会执行return true这条语句，这种情况下，函数是不会返回而退出执行的，就会接着执行第二个分支。

经过上面反复的推演，我可以下这样的一个结论：

对于二叉树式递归问题，画出问题的二叉递归调用树，然后采用二叉树的先根遍历来进行遍历，就可以得到函数的调用执行过程。