### 回溯法解决MKP问题的思路分析

**代码思路分析：**

首先要了解了测试数据的结构，

测试数据结构：

6 10 3800 //n物品个数 m约束条件个数（背包个数） best最优解决方案值

100 600 1200 2400 500 2000 //价值数组

8 12 13 64 22 41 //m行n列维度（或者说质量）的二维数组

8 12 13 75 22 41

3 6 4 18 6 4

5 10 8 32 6 12

5 13 8 42 6 20

5 13 8 48 6 20

0 0 0 0 8 0

3 0 4 0 8 0

3 2 4 0 8 4

3 2 4 8 8 4

80 96 20 36 44 48 10 18 22 24 //每一行（每一维度）的背包上限（约束条件）

在确定了解了数据的结构后，从根结点出发，

以深度优先的方式往下搜索所有节点，即遍历每一个物品

从某节点开始，即当前的扩展节点，先用约束函数在该节点处剪除不满足约束的子树；

约束函数条件：

然后我会判断每一组属性加起来后的总和是否会超出与之对应的背包上限，如果超出那就说明没有必要继续往下搜索，即往下搜索不会得到最有解决方案，那接下来就逐层回溯。

接着进入子树，继续按照深度优先的策略进行搜索。

1. 满足约束条件，即该物品放入背包，那就往下搜索左子树：
2. 物品不放入背包，那就搜索右子树：
3. 当t=Point[b].n时，说明深度搜索已经搜索到了最底层，即已经搜索到叶子节点，又或者说遍历到最后一个物品。搜索结束了，在这些一系列的解中比较并保存最大价值以及它的路径

**时间复杂度分析：**

这个MKP问题的解空间很明显，就是2的n次方

因为我们写的是0-1背包问题，就是物品的放与不放，所有的方法都是在穷举法的基础上减少时间复杂度的，所以最坏肯定是O(2^n)，但由于有m个属性，所以最坏打算为O(m\*2^n)