**多维背包问题算法与设计分析**

用回溯法来解决多维背包问题（回溯法定义）：回溯法是一个既带有系统性又带有跳跃性的搜索算法。它在包含问题的所有解的解空间树中按照深度优先的策略，从根节点出发搜索解空间树。算法搜索至解空间树的任一节点时，总是先判断该节点是否肯定不包含问题的解。如果肯定不包含，则跳过对以该节点为根的子树的系统搜索，逐层向其原先节点回溯。否则，进入该子树，继续按深度优先的策略进行搜索。

运用回溯法解题通常包含以下三个步骤：

1、针对所给问题，定义问题的解空间；

2、确定易于搜索的解空间结构；

3、以深度优先的方式搜索解空间，并且在搜索过程中用剪枝函数避免无效搜索；

【回溯法】－－多维背包问题算法设计分析

【整体思路】

多维背包问题与01背包问题性质差不多，应该是属于找最优解问题，只不过01背包是关于价值和重量，只要其重量不超过背包容量，就可以放进背包中。而有关于多维背包，放入的物品必须满足其所有约束条件，与01背包问题一样，我们也可以用回溯法需要构造解的子集树。对于每一个物品i，对于该物品只有选与不选2个决策，总共有n个物品，如果满足其所有约束条件，就可以放进背包中，可以顺序依次考虑每个物品，这样也就形成了一棵解空间树：

基本思想就是遍历这棵树，以枚举所有情况，最后进行判断，如果重量不超过背包容量，且价值最大的话，该方案就是最后的答案。多维背包问题使用回溯法的大概思路应该是：把物品看成是一棵树，0、1就是树枝，从一开始一直往下搜索下一个节点，满足其所有约束条件就可以放入背包中，只要其其中一个属性不满足它的约束条件，就不能放入背包中，继续遍历下一层，然后，递归回溯到上一个，比较上一种取或者不取的相对情况，规划出另一种方案，知道把整棵树遍历完，然后把每一种方案的所有解进行比较，最大的解就是其最优的价值了。

      （剪枝函数）在搜索状态空间树时，只要左子节点是可一个可行结点，搜索就进入其左子树。对于右子树时，先计算上界函数，以判断是否将其减去（剪枝）。

多维背包问题应该和01背包问题的性质差不多，在搜索解空间树时，只要左儿子节点是一个可行节点，搜索就进入左子树，在右子树中有可能包含最优解才进入右子树搜索，否则将右子树剪去。

　　上界函数bound():当前价值cw+剩余容量可容纳的最大价值<=当前最优价值bestp。　在运用上界函数剪枝之前，选择先将物品按照其单位重量价值从大到小排序，此后就按照顺序考虑各个物品。本题中，我没有运用到上界函数，在使用上街函数前必须先使用冒泡排序把它的价值属性排好序，而我直接用递归回溯法遍历求出最优解，然后，对于不符合约束条件的物品则用剪枝函数进行剪枝。

第一份数据的运行数值和运行结果如下：

6 10 //6表示物品数量，10表示其数量的限制条件（维度）

100 600 1200 2400 500 2000 //这一行表示物品价值

8 12 13 64 22 41

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 //这一部分则表示属性，约束条件的值

80 96 20 36 44 48 10 18 22 24 //限制条件的约束

3800.0 //运行结果，最优解