算法设计与分析

一、问题描述：

有n件物品，每件物品都有各自的价值和各自对每种属性的消耗，vi为每种物品的价值，rij为每种物品的每种属性的消耗，uj为背包对每种属性的限制

要求将前n件物品放入背包，每种属性的总和不超过背包对每种属性的限制，且价值达到预期价值。

二、数据

输入：

第一行三个数分别为n：物品数量；m：约束条件；V：期望最大价值。

第二行分别为n个物品的价值，接下n\*m个数据分别为每个物品对每个约束的消耗。

最后一行为每个约束的上限值。

输出：

结果（最大价值）

各物品放入情况（0表示不放入，1表示放入）

三、算法选择

采用回溯法，回溯法的优势在于避开了很多无用的节点，大大的降低了时间和空间复杂度

四、算法设计

首先建立头节点，令其深度为-1，深度就是指物品序号i，当深度为-1时候，自然就不属于这n个物品。从头节点开始，依次建立子节点，当子节点为左子节点时，代表不放入当前第i个物品，右节点表示放入，当然，放入时需继续是否满足放入条件。每个子节点都有一个记录当前最大价值的变量，它等于父节点的最大值加上当前物品的最大价值，再乘以xi，当其为左子节点时，xi=0，为为子节点时，xi=1。

根据规则，依次建立好二叉树后，开始遍历其叶子节点，记录价值最大的价值的叶子节点，再根据每个子节点记录的父亲节点，依次回溯，还原整个最大过程。

五、算法复杂性分析

建立二叉树左子树不需要计算上界，右子树需要计算上界，一维情况下上界计算的时间复杂度为0（n），多维情况下为O（nm），因为我们需要遍历每一个维度的约束，所以需要m的复杂，最坏的情况需要计算O（2^n)次右子树的上界限，所以综合来说时间复杂度为O（nm2^n)。空间复杂度为最坏情况下子树个数0（n^2).

Traceback时的时间复杂度为0（n），我们只需要顺着父节点找回去即可，空间复杂度为O（n^2)，好的情况下总大于O（n）。

六、测试



