**算法设计与分析——多维背包问题回溯算法说明文档**

玉林师范学院 计算机171班 邓辉 201706401113

**1.问题描述**

给定一个具有m个约束条件（c1,c2,……，cm）的背包，n个物品，rij表示物品j对背包的约束i的消耗，物品价值P=[p1,p2,……,pn]，xj表示物品j是否放入背包，xj=1时表示物品放入背包，当xj=0时表示物品不放入背包，求当各约束条件均满足时，背包内所装入物品的价值最大的解。m个约束条件可以看成是m维限制。

**2.数学模型**

j · xj

i j · xj ci

xj=1 or xj =0

**3. 回溯法**

回溯法（英语：backtracking），又称为试探法，是一个既带有系统性又带有跳跃性的搜索算法。它在问题的解空间树中，按深度优先策略，从根节点出发搜索解空间树。算法搜索至解空间树的任一结点时，先判断该节点是否包含问题的解。如果肯定不包含，则跳过对以该节点为根的子树的搜索，逐层向其祖先节点回溯。否则，进入该子树，继续按深度优先的策略搜索。回溯法求问题的所有解时，要回溯到根，且根节点的所有子树都已经被搜索遍才结束。

回溯法搜索解空间树时，为了提高搜索效率，通常采用剪枝函数避免无效搜索。一种是用约束函数在扩展节点处剪去不满足约束的子树；另一种是用限界函数剪去得不到最优解的子树。

回溯法解题一般包括以下三个步骤：

（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解构；

（3）以深度优先的方式搜索解空间，并在搜索过程中用剪枝函数避免无效搜索。

**4.子集树**

当所给问题是从n个元素的集合S中找出满足某种性质的子集时，相应的解空间是一棵子集树。例如，n个物品的多维背包问题所相应的解空间树就是一棵子集树。这类子集树通常有2n个叶节点，其节点总个数为2n+1-1，由于具有m个维度，所以遍历此子集树的任何算法均需要O (m\*2n)的计算时间。

教材习题5-21子集树探究的是装载问题，由于装载问题是等价值的特殊0-1背包问题，所以本算法是在习题5-21子集树基础上修改得到。

回溯法搜索多维背包问题解的子集树的Java算法描述如下：

**private** **static** **void** backtrack(**int** t){

if(t > *n*)

*record*();

**else**

{

**for**(**int** i=0; i<=1; i++)

{

*x*[t] = i; //是否装入物品t，i的值为0时不装入，值为1时装入

**if**(*constraint*(t) && *bound*(t)) //没有超过上界，且有可能取得最优值

{

*change*(t); //change()方法改变当前各属性的值

*backtrack*(t+1); //递归调用自己，进行深度搜索

*restore*(t); //向上层回溯

}

}

}

}

其中，形参t表示递归深度，即当前扩展节点在解空间树中的深度。n用来控制递归深度，当t>n时，算法已搜索到叶节点。此时，由record()记录得到的可行解。i表示在当前扩展节点处x[t]的可选值（i的取值可为0或1，值为0时表示物品t不装入背包，值为1时表示将物品t装入背包）。constraint(t)和bound(t)表示在当前扩展节点处的约束函数和限界函数。当constraint(t)返回的值为true时，在当前扩展节点处x[1:t]的取值满足问题的约束条件，否则不满足问题的约束条件，可以剪去相应的子树。当bound(t) 返回的值为true时，在当前扩展节点处x[1:t]的取值未使目标函数越界，还需由backtrack(t+1)对其相应的子树做进一步的搜索。否则，在当前扩展节点处x[1:t]的取值使目标函数越界，可以剪去相应的子树。执行了算法的for循环后，已搜索遍当前节点的所有未搜索过的子树。backtrack(t)执行完毕，调用restore(t)方法返回t-1层继续执行，对还没有测试过的x[t-1]的值继续搜索。当t=1时，若已测试完x[t]=1的所有可选值，外层调用就全部结束。显然，这一搜索过程按深度优先方式进行，且调用一次backtrack(1)即可完成整个回溯搜索过程。

**5.数据输入**

在每组数据的第一行有三个数字，分别表示物品个数n，约束条件个数m，以及期望的所装入背包的物品最大价值P。数据第二行是以上n个物品所对应的价值。接下来会有n\*m个数据，每个数据分别对应第i个物品的第j个属性（1≤i≤n，1≤j≤m）；最后一行有m个数据，分别表示各约束条件的上限值。

**6.程序输出**

本程序将输出背包在装入满足各项约束条件的物品时最大价值，以及各个物品此时的装入情况（0表示不装入背包，1表示装入背包），装入背包的物品及物品个数。

**7.运行实例**

**输入：**

6 10 3800

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

**输出：**

装入背包的最大价值为：3800

各物品的装入情况为：0 1 1 0 0 1

即：将第2、3、6、共3个物品装入背包

**8.复杂度分析**

**（1）时间复杂度：**

每个物品只有装入或不装入两种状态总共有n个物品，遍历这些物品的子集树算法时间复杂度为O(2n)，但由于每个物品有m个约束，所以算法中执行次数最多的语句为m\*2n次，即算法的时间复杂性为O(m\*2n)。

**（2）空间复杂度：**

算法开辟了一个二维数组存储n个物品的m种属性，所以算法的空间复杂度为O(n\*m)。

**9.算法改进**

对于此算法，可以按物品价值大小进行排序，使得每次要装入物品时，装入的物品都是当前可装入的具有最大价值的物品，这样可以更好地利用约束函数进行剪枝。但是需要排序的不只是价值，物品的各个属性也需要跟着其所对应的价值更改。因为回溯法的时间复杂度是O(m\*2n)，排序算法的时间复杂度小于O(m\*2n)，所以整体时间复杂度不变，但是需要开辟额外的辅助存储空间。