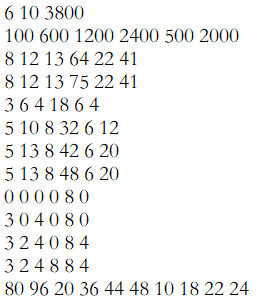
我的算法设计与分析

班别：计算机171班 姓名：杨仕安 学号：201706401106

我们先来分析题目，题目是多维背包问题，与0-1背包问题一样，都会给出一个背包，多个物品，物品有价值和不同的属性，属性会有一个上限，最后在满足所有条件的情况下，装入价值最高的那些物品即可。多维背包问题与0-1背包问题的区别就在于多维，0-1背包问题就只有一个属性作为限制条件，多维背包问题则有很多的属性作为限制条件，相当于在0-1背包的问题的基础上加了多个限制条件或者多个物品。在那七组数据中就有6到50个限制条件，大大加深了循环运算次数。

下面我们来分析数据，以第一组数据为例，第一行，6：表示物品个数，10：表示限制条件个数，3800：表示在这组数据中能得到的最大价值。第二行：表示每个物品的价值。第三行到第十二行：表示每个物品的限制条件的数值。最后一行：表示那十个限制条件的上限。

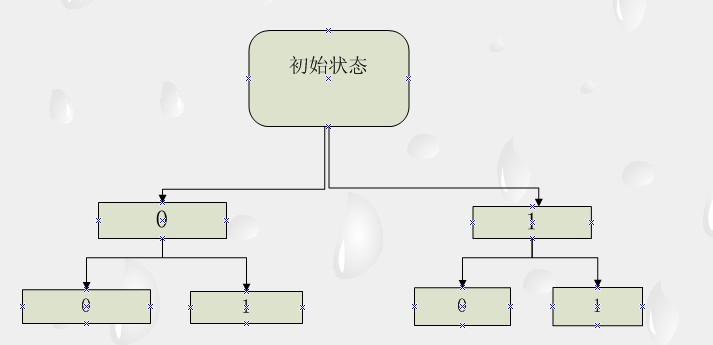


|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | 物品 个数 | 限制条件个数 | 最大  价值 |  | | | 条件  上限 |
| 6 | 10 | 3800 |  |
| **序号** | | 1 | 2 | 3 | 4 | 5 | 6 |
| **物品价值** | | 100 | 600 | 1200 | 2400 | 500 | 2000 |
| 约 束 项 | 1 | 8 | 12 | 13 | 64 | 22 | 41 | 80 |
| 2 | 8 | 12 | 13 | 75 | 22 | 41 | 96 |
| 3 | 3 | 6 | 4 | 18 | 6 | 4 | 20 |
| 4 | 5 | 10 | 8 | 32 | 6 | 12 | 36 |
| 5 | 5 | 13 | 8 | 42 | 6 | 20 | 44 |
| 6 | 5 | 13 | 8 | 48 | 6 | 20 | 48 |
| 7 | 0 | 0 | 0 | 0 | 8 | 0 | 10 |
| 8 | 3 | 0 | 4 | 0 | 8 | 0 | 18 |
| 9 | 3 | 2 | 4 | 0 | 8 | 4 | 22 |
| 10 | 3 | 2 | 4 | 8 | 8 | 4 | 24 |

后面的数据就以此类推。

下面来说一下我写的代码，我有的是Java来编写的，用了算法设计的回溯法。主要用Bag.java文件（主函数）来控制输入输出和循环，用laosi.java文件（类）来构建回溯法的方法。

通过回溯法的需要，定义n、m、shuxing、limite、value、selection等参数然后在构造方法。用布尔型函数判断限定条件：当前背包的条件限制量加上新物品的限制条件是否会超出限制条件的上限。如果超过就不要这个物品，然后不往下执行。如果满足条件就继续并循环，执行backtrack这个类，首先确定第几个物品，然后再确认这个物品符不符合最优解，如果不是最优解就剪枝，还原之前的价值总量。如果属于最优解的部分则继续往下执行。保存最优的selection值，把最优的选择序列保存在best\_selection中，然后不断循环，直至循环结束。如下图大概解释了我的函数算法。



1、如上图碰到一组数据，有两种可能:选或者不选，在树种分别由1，0表示。

2、使用递归，在遍历完n个数的时候，判断最终的数是否比最佳价值大，如果比最佳价值大，就把值赋给bestv。

最后分析我的时间复杂度，以第一题为例，我的最差时间复杂度为640。最优的时间复杂度为80。得到时间复杂度的公式是：m\*2^n。m：表示物品的限制条件的个数，n：表示物品的个数。所以，当物品越多的时候时间复杂度就越大。就是通过上面这个公式，我在运行28 10 12400 的那一个测试数据的时候，我只用了18秒，然而，无在执行39 5 10618 的这个测试数据的时候却用了五个多小时，他们的时间复杂度差了整整2的十次方倍，1024倍。与50 5 16537这个数据差了2048\*1024倍，大约需要10000多个小时，相当于一年的时间。所以最后一个数据就不测试了。下图是我运行了五个小时之后的截图。

