1. **问题描述**

给定n种物品和一背包。物品i的重量是wi，其价值为vi，背包的容量为C。问应如何选择装入背包的物品，使得装入背包中物品的总价值最大。

1. **问题分析**

本次作业共实现了两种针对该问题的算法。

第一种是针对重量及价值为整数的基本算法。

第二种是适用于小数重量及价值的改进算法。

1. **输入输出格式**

输入：n（物品个数）、C（背包容量）

输出：最大价值、物品编号（1~n）。

1. **基本算法**
2. 算法思想

线性规划的递推关系式为

因而构造一个dp[n+1][C]大小的二维矩阵，其中第一行（即下标为0）全置为0。从第二行开始，从capacity=C向capacity=0按照递推关系式计算即可。

1. 测试情况

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| n | C | w | v | maxValue | selection |
| 5 | 10 | {2, 2, 6, 5, 4} | {6, 3, 5, 4, 6} | 15 | {1, 2, 5} |
| 5 | 8 | {3, 5, 1, 2, 2} | {4, 5, 2, 1, 3} | 10 | {2, 3, 5} |
| 5 | 100 | {77, 22, 29, 50, 99} | {92, 22, 87, 46,90} | 133 | {3, 4} |
| 8 | 200 | {79, 58, 86, 11, 28, 62, 15, 68} | {83, 14, 54, 79, 72, 52, 48 62} | 334 | {1, 4, 5, 6, 7} |

1. **改进算法**
2. 算法思想

观察基本算法容易发现，对每一个确定的i (1≤i≤n)，函数m(i,j)是关于变量j的阶梯状单调不减函数。跳跃点是这一类函数的描述特征。在一般情况下，函数m(i,j)由其全部跳跃点唯一确定。

因而改进方法即求出所有的跳跃点，并在跳跃点进行判断计算即可求解。

设第i个物品的跳跃点集为p[i]，则计算方法为：

由表p[i+1]计算表p[i]时，可先由p[i+1]计算出q[i+1]，然后合并表p[i+1]和表q[i+1]，并清除其中的受控跳跃点(c,d)得到表p[i]。

1. 测试情况

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| n | C | w | v | maxValue | selection |
| 5 | 10 | {2, 2, 6, 5, 4} | {6, 3, 5, 4, 6} | 15 | {1, 2, 5} |
| 5 | 8 | {3, 5, 1, 2, 2} | {4, 5, 2, 1, 3} | 10 | {1, 3, 4, 5} |
| 5 | 100 | {77, 22, 29, 50, 99} | {92, 22, 87, 46,90} | 133 | {3, 4} |
| 8 | 200 | {79, 58, 86, 11, 28, 62, 15, 68} | {83, 14, 54, 79, 72, 52, 48 62} | 334 | {1, 4, 5, 6, 7} |

观察到第二组测试的选择方案与基本算法中不同，这是由于对物品编号的判断由大到小，而基本算法中的判断由小到大，这两种方案对于解决问题而言均是正确的。

1. **总结**

通过本次作业，我对利用线性规划解决问题有了更深入的认识。线性规划的重点在于寻找递推关系式，例如本次作业中基本算法中m(i,j)的求解关系式。另一方面，常规的线性规划往往有着较高的空间复杂度，例如基本算法中空间复杂度为O(nC)，但是其中大部分内容主要用于记忆，而非必要，因而通过更进一步的对问题求解重点进行分析，例如改进算法中的跳跃点，使得问题的时间复杂度从O(n2n)降低到min(nC,2n)，空间复杂度虽然不固定，但低于O(nC)。

对于01背包问题的求解，通过改进算法的学习与设计分析，我有了更进一步的理解和认识，对进一步优化线性规划问题有了相当大的启发。

非常感谢老师的作业设计！