**暨南大学本科实验报告专用纸**

课程名称 算法分析与设计实验 成绩评定

实验项目名称 整数线性规划问题 指导教师 李展

实验项目编号 实验项目类型 综合型

学生姓名 张瑞鹏 学号 2020101124

学院 信息科学技术学院 系 计算机 专业 计科

实验时间 2022 年 月 日 午～ 月 日 下 午

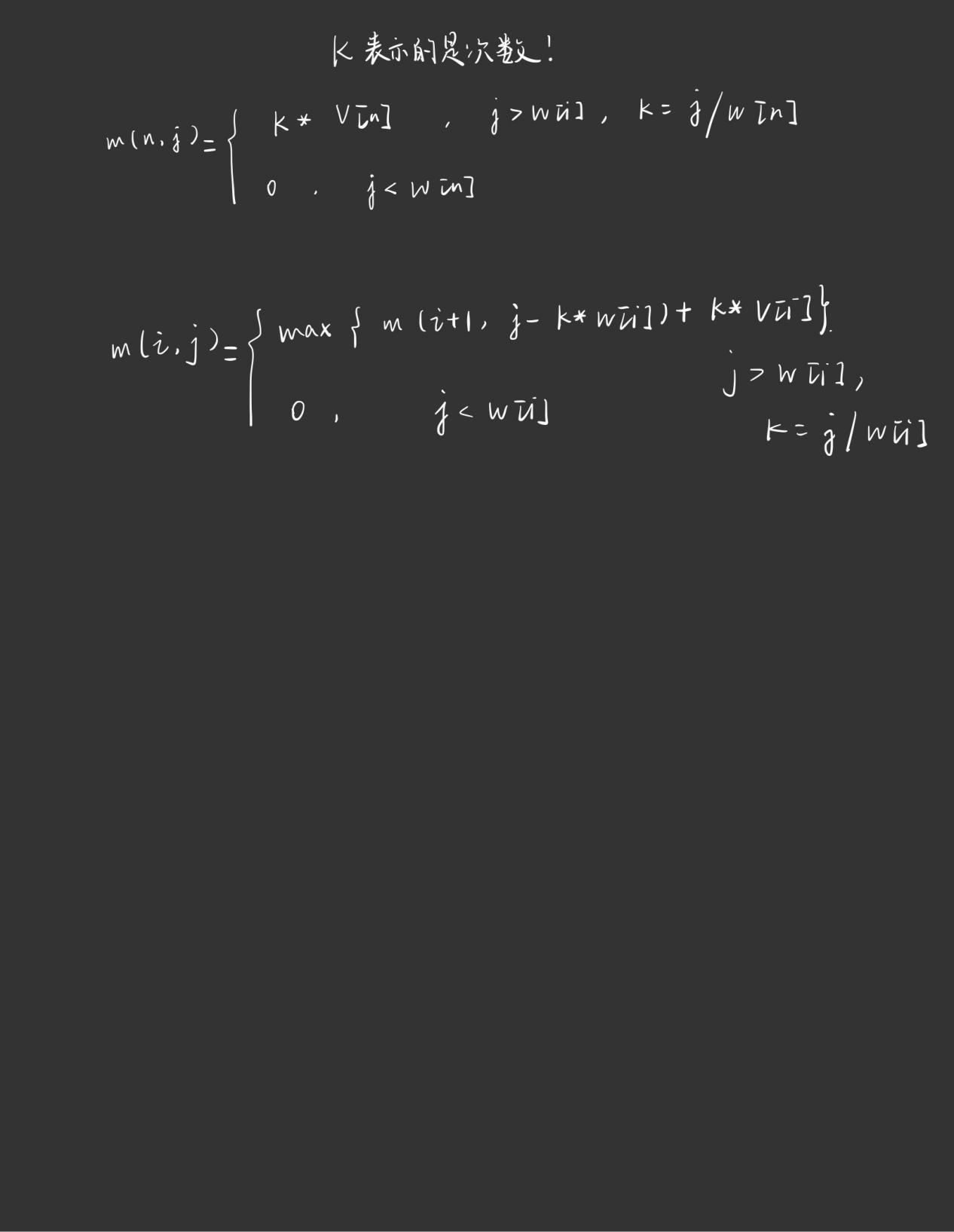
**（一）问题描述**

**整数线性规划，也就是完全背包问题，与0-1背包问题不同的是，物品种类取多少件不再是零和一，而是任意的数字，当然这是建立在不能超出质量的范围之下的。**

**（二）算法思路**（用文字简单说明）

**我们先假设m（i，j）表示的是目前所考虑的是包括i直到n所有物品，还有目前的背包剩余的容量为j，然后m（i，j）表示上述情况所代表的价值，在这里k表示最多可放入的次数。**

**我们列出动规方程：**

****

**V数组表示记载了value，也就是价值**

**W数组记载的是质量，也就是weight**

**k表示的是最多可以放入的次数，用剩余空间除质量取整数计算得出。**

**根据动规方程，也就是说我们从最后一行（每一行从左往右填写，每一个空要分别比较k次），就能填满整张m（i，j）的二维表，我们要的答案就是m（1，c），在二维表的最右侧**

**代码细节:**

**1.我是先直接赋值m(i,j) = m(i+1,j),然后计算出k的大小，比较所有0到k的取值，然后贪心选取较大值更新m(i,j)。**

**2.为了检查哪里出现错误，我把整张二维表打印出来，因为要打印整张二维表，于是我把第一行也填满了，ppt上的算法是没有填写第一行的。**

**3.为了回溯，于是建立了一个remember二维表，对应dp二维表记录放入次数，这样子方便统计最终结果。**

**4.为了老师检查方便，为了自己可以更方便找bug，于是把两张二维表打印出来。**

**（三）算法实施步骤和流程**（伪代码/流程图等方式描述）

**int Knapsack(int\* v, int\* w, int n, int c,int\* tb) {//w为代价array v为价值数组 n为物品的数目 w为背包的容量**

**int dp[n+1][c+1];//打算0单元不使用**

**int remember[n+1][c+1]; int time;**

**for (int j = 0; j <= c;j++) {**

**time = j/w[n-1];//表示能放下多少个就放下多少个**

**dp[n][j] = time\*v[n-1];**

**remember[n][j] = time;**

**}**

**for (int i = n - 1; i >= 1; i--) {**

**dp[i][0] = 0;remember[i][0] = 0;**

**for (int j = 1; j <= c; j++) {**

**dp[i][j] = dp[i + 1][j];//初始化为假设不放入**

**time = j/w[i-1];//计算最多能放进去的个数**

**remember[i][j] = 0;**

**for(int k = 1;k<=time;k++){//一层一层的贪心**

**if ((v[i-1]\*k + dp[i + 1][j - k\*w[i-1]]) > dp[i][j]) {**

**dp[i][j] = v[i-1]\*k + dp[i + 1][j - k\*w[i-1]];//改值**

**remember[i][j] = k;**

**}**

**}**

**}**

**}**

**int i = 1,j = c;**

**for(;i<=n;i++){**

**tb[i] = remember[i][j];**

**j -= remember[i][j]\*w[i-1];**

**}**

**return dp[1][c];**

**}**

**（四）源代码**（通过了编译运行的正确程序）

**#include<stdio.h>**

**#include<iostream>**

**using namespace std;**

**int Knapsack(int\* v, int\* w, int n, int c,int\* tb) {//w为代价array v为价值数组 n为物品的数目 w为背包的容量**

**int dp[n+1][c+1];//打算0单元不使用**

**int remember[n+1][c+1];**

**int time;**

**for (int j = 0; j <= c;j++) {**

**time = j/w[n-1];//表示能放下多少个就放下多少个**

**dp[n][j] = time\*v[n-1];//初始化全部置**

**remember[n][j] = time;**

**}**

**for (int i = n - 1; i >= 1; i--) {**

**dp[i][0] = 0;remember[i][0] = 0;**

**for (int j = 1; j <= c; j++) {**

**dp[i][j] = dp[i + 1][j];//初始化为假设不放入**

**time = j/w[i-1];//计算最多能放进去的个数**

**remember[i][j] = 0;**

**for(int k = 1;k<=time;k++){//一层一层的贪心**

**if ((v[i-1]\*k + dp[i + 1][j - k\*w[i-1]]) > dp[i][j]) {**

**dp[i][j] = v[i-1]\*k + dp[i + 1][j - k\*w[i-1]];//改值**

**remember[i][j] = k;**

**}**

**}**

**}**

**}**

**printf("为了让大家看着更加方便，打印二维表：\n");**

**for(int j = 0;j<=c;j++){**

**printf("%-3d",j);**

**}**

**cout<<"（表示背包剩余容量）";**

**printf("\n\n");**

**for(int i = 1;i<=n;i++){**

**for(int j = 0;j<=c;j++){**

**printf("%-3d",dp[i][j]);**

**}**

**printf("\n");**

**}//输出表格**

**printf("\n\n");**

**for(int i = 1;i<=n;i++){**

**for(int j = 0;j<=c;j++){**

**printf("%-3d",remember[i][j]);**

**}**

**printf("\n");**

**}//输出表格**

**int i = 1,j = c;**

**for(;i<=n;i++){**

**tb[i] = remember[i][j];**

**j -= remember[i][j]\*w[i-1];**

**printf("第%d个物品放进去%d个",i,tb[i]);**

**printf("\n");**

**}**

**printf("\n\n");**

**return dp[1][c];**

**}**

**int main(void)**

**{**

**int v[6] = { 2,5,5,1,10,6 };**

**int w[6] = { 2,3,4,1,5,6 };**

**int tb[7] = {0,0,0,0,0,0,0};**

**int n = 6;//物品数量**

**int c = 23;//容量**

**int result = Knapsack(v, w, n, c,tb);**

**printf("结果是%d", result);**

**printf("\n选择结果是：");**

**for(int i = 1;i<=n;i++){**

**printf("%d ",tb[i]);**

**}**

**return 0;**

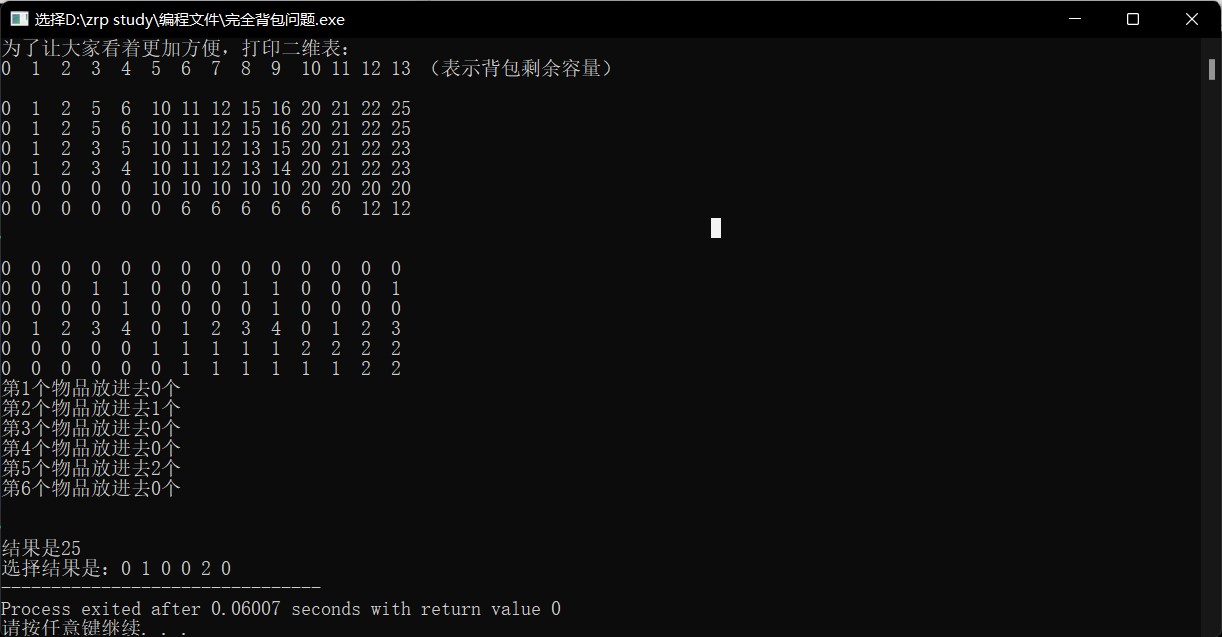
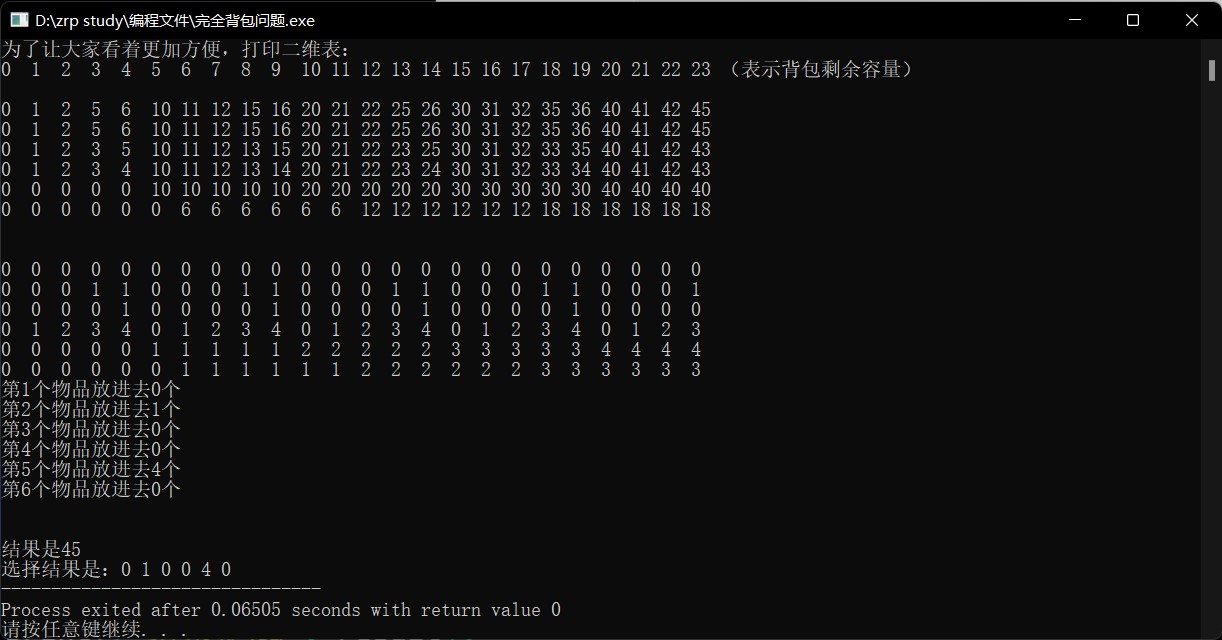
**}**

**（五）测试结果**（至少有两个以上算例及程序运行结果，截图贴进实验报告）

**对输出做一定的解释：**

1.输出的第一个矩阵是m（i，j）矩阵，也就是当可以放入i到n号物品，且空间仅剩j时的最优解

2.输出的第二个矩阵的意义在于，由于完全背包问题不像0-1背包问题，放入了就是放入了，还涉及到放入多少个的问题，因此，如果用一张二维表顺便记录最佳放入个数，就更加方便大家查看，方便运行失败时找原因。

****

**（六）实验总结**（至少三句话，可以写复杂度分析、遇到问题、可能的改进措施、心得体会等）

1.时间复杂度是O（nc^2）。N表示的是遍历每一行，c表示每一个行共有c个单元，每个单元需要比较最多c次。

2.相较与0-1背包问题，多了几次比较，一开始以为要使用三维数组，后来经过思考确定了不需要三维数组。

3.关于数组v和w，与i的关系的问题。i是从1开始的，但是上述两个数组都不是从1开始的，于是一开始出现了很大的问题。下次编写尽量做到对齐，不然容易出错。

**暨南大学本科实验报告专用纸(附页)**