**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 二维背包问题 指导教师 李展 实验地点 N116

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**

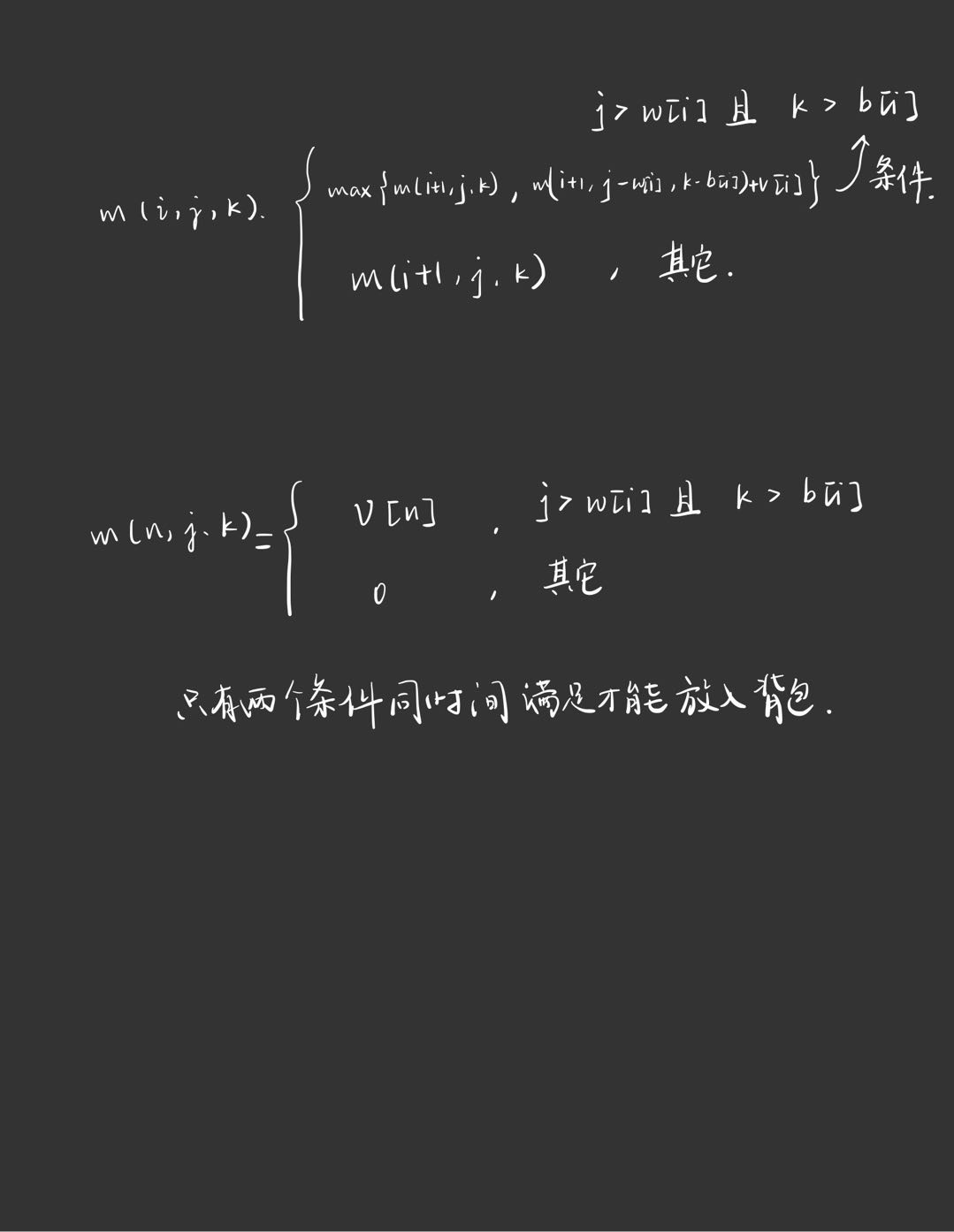
二维背包问题，在3-3的0-1背包问题的基础上增加了空间的限制，其余条件与上一题相同

**（二）算法思路**（用文字简单说明）

**1.根据上一题的提示，这次的算法主要是由上一次的改进得来，由于涉及到两个限制条件，因此联想到三维数组。Dp[i][j][k],i表示的是当前所处理的是第几个物品，j表示当前还剩下多少的质量，k表示现在还剩下多少的空间。M(I,j,k)表示的意义与3-3的内容相类似，表示在这种j和k的条件下，背包能装下的最优的价值。**

**2.动规方程:**

**值得注意的是，虽然看上去有两个条件4种排列组合，但是实际上却只有2种分类，因为仅有两个条件均符合才能放入背包，剩余调节均可以归为一类，因此动规方程并没有想象中的复杂。**

****

**3.根据上述规则，倒着依次填完二维表（i= n到1），一个一个平面填写，填写的时候先覆盖再判断**

**Dp[i][j][k] = dp[i+1][j][k],判断后再修改。**

**（三）算法实施步骤和流程**（伪代码/流程图等方式描述）

int TDkna(int\* v, int\* w, int\* b, int n, int weightlimi, int airlimi,int\* tb)

{

int dp[n + 1][weightlimi + 1][airlimi + 1];

for (int i = 0; i <= weightlimi; i++) {

for (int j = 0; j <= airlimi; j++) {

dp[n][i][j] = 0;

}

}

for (int i = w[n]; i <= weightlimi; i++) {

for (int j = b[n]; j <= airlimi; j++) {

dp[n][i][j] = v[n];

}

}

for (int i = n - 1; i > 0; i--){

for (int j = 0; j <= weightlimi; j++) {

for (int k = 0; k <= airlimi; k++) {

dp[i][j][k] = dp[i+1][j][k];

if (w[i] <= j && b[i] <= k) {

if((dp[i + 1][j - w[i]][k - b[i]] + v[i])>dp[i][j][k])

dp[i][j][k] = dp[i + 1][j - w[i]][k - b[i]] + v[i];

}

}

}

}

int i = 1,j = weightlimi,k = airlimi;

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

if(dp[i][j][k]>dp[i+1][j][k]){

tb[i] = 1;j-=w[i];k-=b[i];

}

}

if(dp[n][j][k]>0) tb[n] = 1;

return dp[1][weightlimi][airlimi];

}

**（四）源代码**（通过了编译运行的正确程序）

# include<stdio.h>

int TDkna(int\* v, int\* w, int\* b, int n, int weightlimi, int airlimi,int\* tb)

{

int dp[n + 1][weightlimi + 1][airlimi + 1];

for (int i = 0; i <= weightlimi; i++) {

for (int j = 0; j <= airlimi; j++) {

dp[n][i][j] = 0;

}

}

for (int i = w[n]; i <= weightlimi; i++) {

for (int j = b[n]; j <= airlimi; j++) {

dp[n][i][j] = v[n];

}

}

for (int i = n - 1; i > 0; i--){

for (int j = 0; j <= weightlimi; j++) {

for (int k = 0; k <= airlimi; k++) {

dp[i][j][k] = dp[i+1][j][k];

if (w[i] <= j && b[i] <= k) {

if((dp[i + 1][j - w[i]][k - b[i]] + v[i])>dp[i][j][k])

dp[i][j][k] = dp[i + 1][j - w[i]][k - b[i]] + v[i];

}

}

}

}

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

printf("i = %d\n",i);

for (int j = weightlimi; j >= 0; j--) {

for (int k = 0; k <= airlimi; k++) {

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

}

printf("\n");

}

}

int i = 1,j = weightlimi,k = airlimi;

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

if(dp[i][j][k]>dp[i+1][j][k]){

tb[i] = 1;j-=w[i];k-=b[i];

}

}

if(dp[n][j][k]>0) tb[n] = 1;

return dp[1][weightlimi][airlimi];

}

int main(void){

int v[7] = { 0,2,5,5,1,10,6 };

int w[7] = { 0,2,3,4,1,5,6 };

int b[7] = { 0,2,3,4,1,5,6 };

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

int n = 6;//物品数量

int weightlimi = 13;//容量

int airlimi = 13;

int result = TDkna(v, w, b, n, weightlimi, airlimi,tb);

printf("输出结果为：%d",result);printf("\n选取结果为：");

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

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

}

return 0;

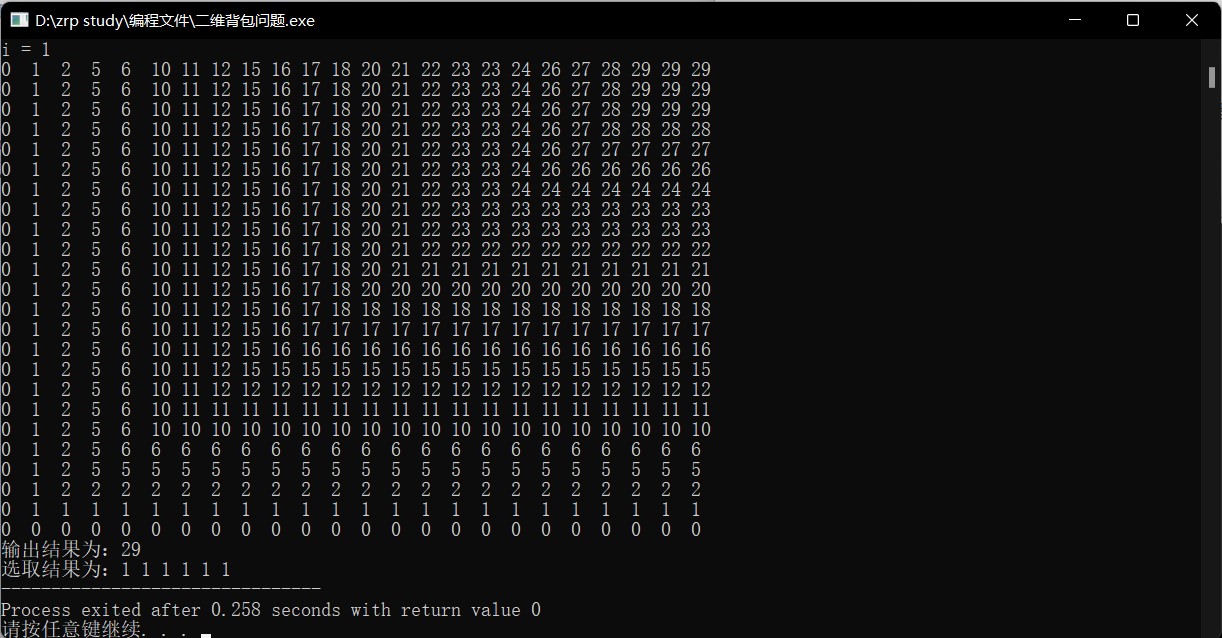
}

**（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）

int v[7] = { 0,2,5,5,1,10,6 };int w[7] = { 0,2,3,4,1,5,6 };

int b[7] = { 0,2,3,4,1,5,6 }; int weightlimi = 23;//容量

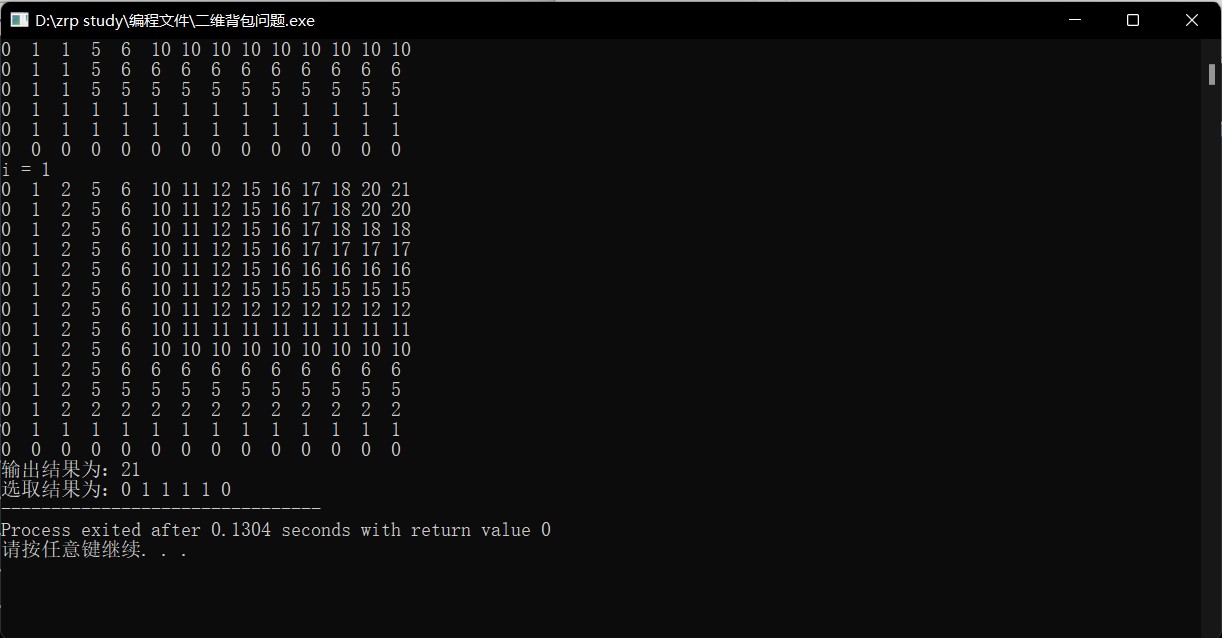
int airlimi = 23;//空出了第一个位置，这个测试用例是老师要求

****

int v[7] = { 0,2,5,5,1,10,6 };int w[7] = { 0,2,3,4,1,5,6 };

int b[7] = { 0,2,3,4,1,5,6 }; int weightlimi = 13;//容量

int airlimi = 13;

****

**我把三维数组全部打印出来（分n次打印，每次打印一个二维数组），方便检查错误，截屏只截了最后i= 1的矩阵**

**选取结果和输出的结果均打印输出。**

**（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）

1.一开始看到有两个限制条件，不知所措，冷静分析发现，其实换汤不换药，**虽然看上去有两个条件4种排列组合，但是实际上却只有2种分类，因为仅有两个条件均符合才能放入背包，剩余调节均可以归为一类，因此动规方程并没有想象中的复杂。即使换成4维数组，做法其实也差不多。**

**2.算法的时间复杂度应该是O(nwA),n表示的是物品的个数，w表示的是背包质量限制，a表示的是空间限制，相当于遍历了n\*w\*a大小的三维数组，剩下的操作均为线性。**

**3.算法设计要举一反三，像诸如此类的题目其实和老师上课讲的内容相差无几，应该能够独立完成。**

**暨南大学本科实验报告专用纸(附页)**