**姓名：李永宾 学号：201706401203 班别：计算机172**

# 算法设计与分析

## 问题分析

该问题是一个多维背包问题，也就是多个物品多个约束，求解最大价值的一种背包问题，也是01背包问题的另外一种形式，可以采用穷举法，动态规划，回溯法，分支限界等多种方法进行求解，我使用的方法是回溯法加上界函数求解问题。

首先先分析所给的其中一组测试数据，并分析：

个数 属性 最优解

6 10 3800

价格：100 600 1200 2400 500 2000

一 二 三 四 五 六

8 12 13 64 22 41

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

约束条件：80 96 20 36 44 48 10 18 22 24每个约束条件是指选取物品的每个属性不能超过约束条件，否则不选这个物品，输出最优解3800。

从这组数据可以看出，总共有6个物品，每个物品有10个属性和与之对应的6个价值，所给的背包容量就是约束条件，所求的最优解就是在满足约束条件的前提下达到所存放物品的价值最大。

## **算法设计思路**

开始

流程图：

当前添加第n个是否超过物品总数N

是

打印答案

输出结果

否

当前添加第n个物品的约束是否大于背包总量b

否

判断当前最优解是否大于当前价值（剪枝函数调用，判断是否剪枝）

是

否

放入第n个物品

对下一个物品（n+1物品）进行判断，也就是进行递归，返回一个最优解

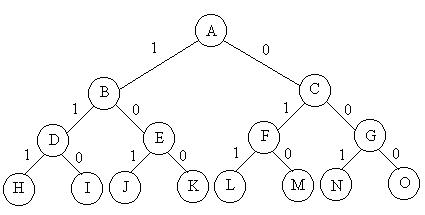
拿走第n表物品

对下一个物品（n+1物品）进行判断，也就是递归

我采用的是C语言进行算法的实现，在上诉分析中我有如下算法设计思路：

大致思路：使用一个二维数组a[][]存放物品约束条件（物品的属性），一个一维数组v[]存放物品价格，一个一维数组b[]存放背包容量，一个一维数组d[]存放所取到物品的属性，如果取到多个则进行累加，然后通过这个数组d[]和b[]进行比较，如果b[]里面所有的数值都大于d[]，则表示可以取到这个物品，然后通过比较价值得到答案，最后以此类推下去一直取还是不取得到一个最优解。

1. 从键盘输入对应的数值：先确定有多少个物品和每个物品有多少个属性，所以先通过键盘键入物品个数n和物品属性个数m，确定好后，使用一个一维数组存入每个物品所对应的价值，因为物品多个，每个物品又有多个约束条件，所以用一个二维数组a[][]存放每个物品的属性，在使用一个一维数组b[]存放背包容量（约束条件）。
2. 设计回溯法计算最优解：我是先了解什么是回溯法。



如图所示：回溯法就是首先会从根节点A开始取或者不取物品，如果取物品放入背包中，就会给一个标记1，不取就给一个标记0（1表示取了物品放入背包，0表示没取物品放入背包），1和0的作用是通过与物品的属性（也就是a[][]\*1或\*0）相乘，乘1就是取了物品，乘0就是没取到物品（任何数乘1都是等于原来的数，乘0都是为0 ，所以可以表示取还是不取），举个简单的例子，在实现算法时，回溯法会先一直都不取，也就是沿着如图A->C->G->O一直不取下去，当取到最后一个时，得到一个当前价值，通过比较当前价值是否大于最优解，因为都不取，所以此时的最优解为0，它会通过递归返回到树的上一层，也就是到G的位置，上一次是不取，递归会这一层后就是取该物品，然后继续向下执行取当前节点的物品，也就是到达到节点N，在此存入一个当前价值，将当前价值与上一次得到的最优解比较，如果大于最优解，将当前价值赋予最优解，然后继续递归调用回上一层一直到C，然后继续向下执行，一直到树的左后一层，又会得到一个当前价值，当前价值又会与上一次取到的最优解比较，以此循环遍历完整颗树，得到一个最大的最优解。

1. 优化算法，让程序执行更快，所以设置一个上界函数：使用背包容量减去所选取物品的属性，将剩下的背包容量和所选物品属性和之前已经放入背包里的物品的属性的和进行比较，如果大于就表示可以放入背包，小于就进行舍去，也就是进行剪枝。（如果没有上界函数，计算结果是第二组数据都会计算很久，有了上界函数计算时间将大幅度减少，但最后几组数据测试时间也很久）

## C语言伪代码和源码

伪代码：

int(k,i,t,d[])//定义连续判断函数

void back(int t)

{

If(t>n)

output（x）//输出函数

Else

{

For(i=0;i<=1;i++)’//判断是否放入物品；

Bound(t);//调用上界函数；

If(lianxupanduan(0,i,t,d))//调用连续判断函数，判断当前节点的物品属性和是否大于背包容量

{

d[t];//累加当前所选物品属性

curvalue;//保存当前价值

back(t+1);//递归

D[t];//减去上次所选胡物品属性

Curvalue;//减去上次的所选物品的价值

}

If(bound(t));//判断是否剪枝

back(t+1);//递归

}

}

Int bound(int i)

{

int left;//存储当前剩余背包容量

int b;//存储当前背包价值

Panduan();//判断当前背包容量是否大于剩余的背包容量

return b;//返回结果b

}

void main()

{

scantf();//从键盘键入测试数据

back(0);

printf();//输出结果

}

源码：

#include<stdio.h>

#define M 1000//给M足够大的内存空间，确保数值全部键入

int n,m;//物品个数，属性个数

int a[M][M];//物品的属性

int b[M];//存放约束条件，就是背包容量

double v[M];//存放价值

double bestvalue=0;//存放最优解

double curvalue=0;//存放当前价值

int best[M];//存放所取物品,已经取了的

double x[M]={0,0,0}; //x[i]=1代表物品i放入背包，0代表不放

double d[M];//存放当前已取的物品，并把其各属性存放在其中，如果取到多个则累加约束条件

int left[M];//在使用上界函数时存储剩余的背包容量

int lianxupanduan(int k,int i,int t,double d[])//定义一个连续判断的函数，i是指0或1，t是指当前节点（获取的是back函数中的t），d[]是指当前所选物品属性全部加起来的值

{

if(k<m)

if(d[k]+i\*a[k][t]<=b[k])//如果当前所选物品属性全部加起来的值小于等于背包容量

{

lianxupanduan(k+1,i,t,d);//递归lianxupanduan函数，如果全部满足约束，则return 1，也就是判断为真，向下执行back函数

}

else

{

return 0;//否则返回0，判断为假，就不取物品，然后跳出判断语句执行后面两个语句（if(bound(i+1)>bestvalue)//判断是否符合条件，如果不符合则进行剪枝back(t+1);//一直遍历到最后，也就是一直得到最新的最优解）

}

}

void back(int t)

{

if(t>n)//遍历当前根节点

{

if(curvalue>bestvalue) //判断当前价值是否大于最优解

{

bestvalue=curvalue;//将当前价值赋予最优解

for(int i=0;i<n;i++) best[i]=x[i];//存储取到的物品

}

}

else

{

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

{//放还是不放

double bound(int i);//定义上界函数，获取当前节点，调用上界函数

x[t]=i;

//添加判断所选物件是否大于约束条件

if(lianxupanduan(0,i,t,d))//连续判断物品当前背包余量是否大于背包容量

{

for(int c=0;c<m;c++)

{

d[c]+=a[c][t]\*x[t];//累加当前所选物品的属性

}

curvalue+=v[t]\*x[t];//累加所选物品的价值

back(t+1);//执行回溯

for(int f=0;f<m;f++)

{

d[f]-=a[f][t]\*x[t];//如果超出了容量，减去上一次所选物品的属性

}

curvalue-=v[t]\*x[t]; //减去上一次物品的价值

}

if(bound(i+1)>bestvalue)//判断是否符合条件，如果不符合则进行剪枝

back(t+1);//一直遍历到最后，也就是一直得到最新的最优解

}

}

}

double bound(int j)//设置上界函数，j是当前树的节点，对应回溯法back（）中的i

{

for(int f=0;f<=m;f++)

{

left[f]=b[f]-d[f];//存储剩余的背包容量，用背包容量（能装物品属性的最大值）减去已装入背包的物品属性

}

int b=curvalue;//记录背包当前总价值

for(int o=0;o<=m;o++)

{

while(j<=n &&a[o][j]<=left[o])//判断当前节点索取物品的属性总和是否大于剩余的背包容量

{

left[j]-=a[o][j];//符合条件存入物品到背包中，就是背包容量减去所选的物品属性

b+=v[j];//存储当前总价值

j++;//向下继续取

}

}

if(j<=n)

{

for(int q=0;q<=m;q++)

{

b+=v[j]/a[q][j]\*left[j];//存储当前价值

}

return b;

}

}

void main()

{

printf("请输入物品个数:\n");

scanf("%d",&n);//获取键盘输入的值，物品的数量

printf("请输入物品属性个数:\n");

scanf("%d",&m);//获取键盘输入的值，物品属性的个数

printf("请依次输入物品的价值:\n");

for(int i=0;i<n;i++)

scanf("%lf",&v[i]);//获取键盘输入的值，物品的价值

printf("请依次输入每个物品的属性的值:\n");

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

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

scanf("%d",&a[j][i]);}//获取键盘输入的值，每个物品的属性

}

printf("请依次输入物品的属性限制:\n");

for(int w=0;w<m;w++)

scanf("%d",&b[w]);//获取键盘输入的值，背包容量

for(int e=1;e<m;e++)

d[e]=0;

back(0);//调用回溯法

printf("最大价值为:\n");

printf("%lf\n",bestvalue);//输出最优解

printf("被选中的物品有(1是选中，0是没有选中)\n");

for(int q=1;q<=n;q++)

printf("%d ",best[q]);//输出选中物品的数组

printf("\n");

}

## 结果展示

第一组数据：

6 10 3800

100 600 1200 2400 500 2000

8 12 13 64 22 41

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

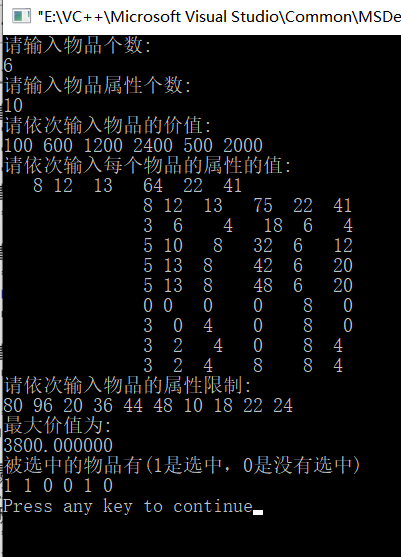
0 0 0 0 8 0

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



第二组数据：

10 10 8706.1

600.1 310.5 1800 3850 18.6 198.7 882 4200 402.5 327

20 5 100 200 2 4 60 150 80 40

20 7 130 280 2 8 110 210 100 40

60 3 50 100 4 2 20 40 6 12

60 8 70 200 4 6 40 70 16 20

60 13 70 250 4 10 60 90 20 24

60 13 70 280 4 10 70 105 22 28

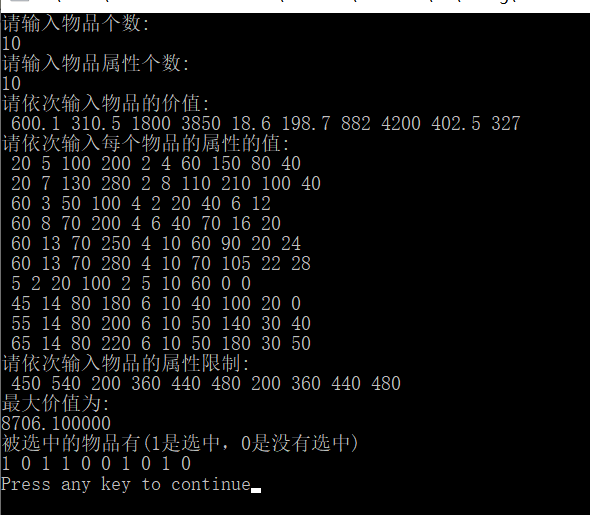
5 2 20 100 2 5 10 60 0 0

45 14 80 180 6 10 40 100 20 0

55 14 80 200 6 10 50 140 30 40

65 14 80 220 6 10 50 180 30 50

450 540 200 360 440 480 200 360 440 480



第三组数据：

15 10 4015

100 220 90 400 300 400 205 120 160 580 400 140 100 1300 650

8 24 13 80 70 80 45 15 28 90 130 32 20 120 40

8 44 13 100 100 90 75 25 28 120 130 32 40 160 40

3 6 4 20 20 30 8 3 12 14 40 6 3 20 5

5 9 6 40 30 40 16 5 18 24 60 16 11 30 25

5 11 7 50 40 40 19 7 18 29 70 21 17 30 25

5 11 7 55 40 40 21 9 18 29 70 21 17 35 25

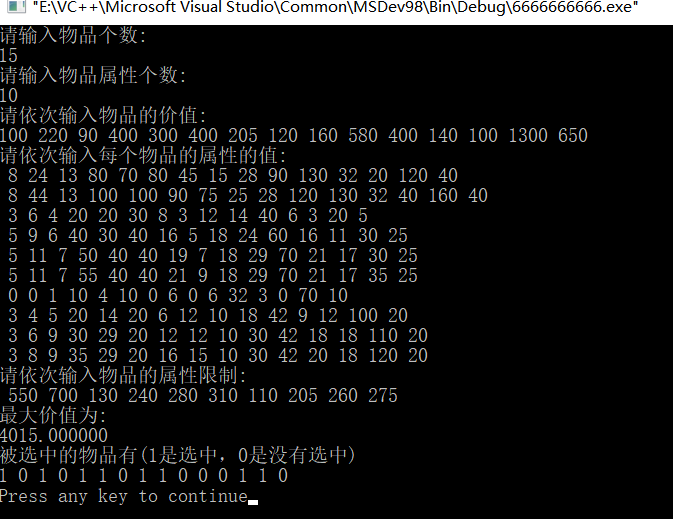
0 0 1 10 4 10 0 6 0 6 32 3 0 70 10

3 4 5 20 14 20 6 12 10 18 42 9 12 100 20

3 6 9 30 29 20 12 12 10 30 42 18 18 110 20

3 8 9 35 29 20 16 15 10 30 42 20 18 120 20

550 700 130 240 280 310 110 205 260 275



第四组数据：

20 10 6120

100 220 90 400 300 400 205 120 160 580 400 140 100 1300 650

320 480 80 60 2550

8 24 13 80 70 80 45 15 28 90 130 32 20 120 40

30 20 6 3 180

8 44 13 100 100 90 75 25 28 120 130 32 40 160 40

60 55 10 6 240

3 6 4 20 20 30 8 3 12 14 40 6 3 20 5

0 5 3 0 20

5 9 6 40 30 40 16 5 18 24 60 16 11 30 25

10 13 5 1 80

5 11 7 50 40 40 19 7 18 29 70 21 17 30 25

15 25 5 1 100

5 11 7 55 40 40 21 9 18 29 70 21 17 35 25

20 25 5 2 110

0 0 1 10 4 10 0 6 0 6 32 3 0 70 10

0 0 0 0 0

3 4 5 20 14 20 6 12 10 18 42 9 12 100 20

5 6 4 1 20

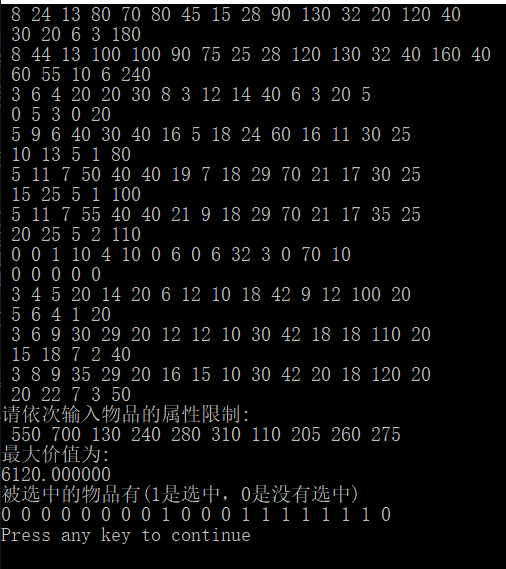
3 6 9 30 29 20 12 12 10 30 42 18 18 110 20

15 18 7 2 40

3 8 9 35 29 20 16 15 10 30 42 20 18 120 20

20 22 7 3 50

550 700 130 240 280 310 110 205 260 275

：

第五组数据：

28 10 12400

100 220 90 400 300 400 205 120 160 580 400 140 100 1300 650

320 480 80 60 2550 3100 1100 950 450 300 220 200 520

8 24 13 80 70 80 45 15 28 90 130 32 20 120 40

30 20 6 3 180 220 50 30 50 12 5 8 18

8 44 13 100 100 90 75 25 28 120 130 32 40 160 40

60 55 10 6 240 290 80 90 70 27 17 8 28

3 6 4 20 20 30 8 3 12 14 40 6 3 20 5

0 5 3 0 20 30 40 10 0 5 0 0 10

5 9 6 40 30 40 16 5 18 24 60 16 11 30 25

10 13 5 1 80 60 50 20 30 10 5 3 20

5 11 7 50 40 40 19 7 18 29 70 21 17 30 25

15 25 5 1 100 70 55 20 50 15 15 6 20

5 11 7 55 40 40 21 9 18 29 70 21 17 35 25

20 25 5 2 110 70 55 20 50 20 15 6 20

0 0 1 10 4 10 0 6 0 6 32 3 0 70 10

0 0 0 0 0 30 10 0 10 10 5 0 10

3 4 5 20 14 20 6 12 10 18 42 9 12 100 20

5 6 4 1 20 50 30 5 20 20 10 10 20

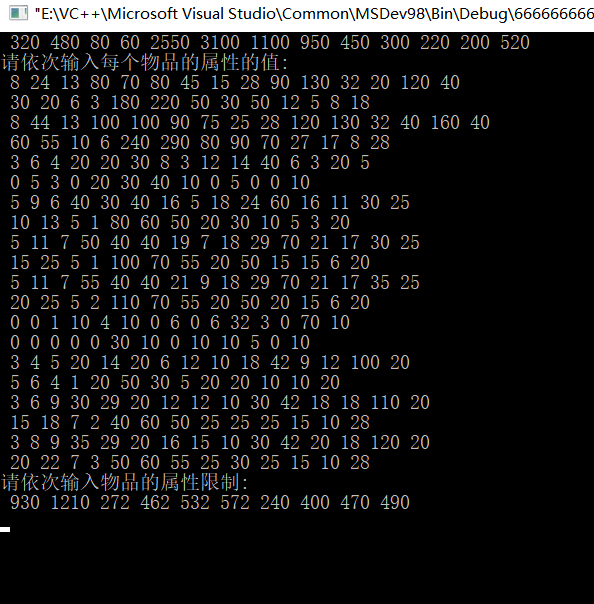
3 6 9 30 29 20 12 12 10 30 42 18 18 110 20

15 18 7 2 40 60 50 25 25 25 15 10 28

3 8 9 35 29 20 16 15 10 30 42 20 18 120 20

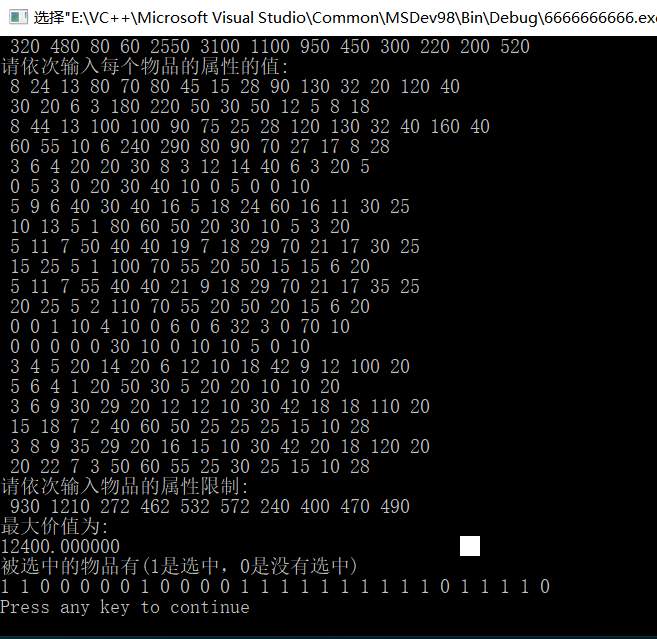
20 22 7 3 50 60 55 25 30 25 15 10 28

930 1210 272 462 532 572 240 400 470 490



第五组数据没有出现结果，因为计算时间太久了。

大概5分钟时算出答案：



## 时间复杂度

回溯法时间复杂度：O(n2n)

上界函数时间复杂度：O(n)，最坏情况下O(2n)

## 算法的缺点

此算法计算物品个数太多，所涉及计算量以指数形式增加，在计算答案时计算量太大，会照成结果计算不出来，所以我写的这个算法应该适合数值较少的计算，太多就会出现在计算第五组数据时出现的问题，结果运算时间太久。当然，还有其他确定，比如代码过于长，也就是并不是优化到最好的，有很多可以舍去的地方，占用内存空间太多，照成不必要的资源浪费。