**算法设计与分析**

学号：20170640250 班级：计算机172班 姓名：王永丽

**一、题目分析**

题目是多维背包问题，.多维背包问题是0-1背包的一个延伸，0-1背包问题的约束条件只有重量，但多维背包问题是多个约束条件，求最优解问题，本题目是给多个约束条件和多个物品属性值和物品的价值，求放入背包的最大价值。其实我们也可以叫它为多约束背包问题，它带有一组约束，比如说重量、尺寸、可靠性、体积等等。该问题也可以简单描述为n个物品要放入m个称重不同的背包，它与0-1背包问题有些类似，0-1背包问题是给出n种物品和一个背包，使得装入背包中的物品价值最大，背包问题制定了背包的最大容量，求不超过背包最大容量的，且价值最大的最优解。而多维背包问题与之不同的是，物品放入不同背包的装量是不同的，首先我们要确定是否放入背包，然后还要确定我们要放入哪个背包，最终目的就是求解我们物品最高的价值比重，也就是最优解。

二、**分析数据**

变量数（物品个数）n：6 约束数（每个物品的属性）m：10

最终价值：3800

放入背包的价值:100 600 1200 2400 500 2000

十行约束：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

**三、设计算法**

我们综合运用所学过的算法设计与分析方法，选择运用回溯法来解决我们的多维背包问题，回溯法是一种以深度优先方式系统搜索问题解的算法，它的基本思想是确定解空间的组织结构后，回溯法从根结点出发，以深度优先的方式搜索整个解空间树，这个开始结点成为活结点，同时也成为当前的扩展结点。在当前扩展结点处，搜索向纵深方向移至一个新结点。这个新结点成为新的活结点，并成为当前扩展结点。如果在当前扩展结点处不能再向纵深方向移动，则当前扩展结点就成为死结点。此时，应往回移动（回溯）至最近的活结点处，并使这个活结点成为当前扩展结点。回溯法以这种工作方式递归地在状态空间中搜索，直到找到所要求的解或解空间中已无活结点时为止，设计算法并用c语言代码经过调试后实现。

**设计代码思路分析：**我运用的是c语言，开头部门是定义三个库函数，然后在定义物品重量、物品价值、结果最好的价值等变量，定义一个记录结果的数组，定义一个递归函数，函数里面包含now 当前价值 ,nw当前每组所求价值，w每组的约束价值等参数，该用于判断是否到达叶子结点和判断当前价值是否大于最后价值更新结果，如果大于就将交换结果，如果到了叶子结点就会回溯上一层，判断是否有结点，有就计算，反之继续回溯到上一层，以此类推，最后在调用函数时令参数j=0，开始计算他的价值，通过getans函数递归得出最优解最后输出这个最优解。

**算法复杂度：** O(n4)，根据语句中for循环的次数。

由于代码中只出现过几次if语句的判断，还有四次for循环，也只是简单的输出将要输入的属性，里面还调用了两次函数，运用回溯法，然后经过比较，选出的最大物品价值。

**c源码：**

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

#define MaxN 51

int weight[MaxN][MaxN];//物品重量

double value[MaxN];//物品价值

double ans=-1;//结果最好的价值

int x[MaxN];//记录结果的数组

//now 为当前价值 ,nw为当前每组所求价值，w为每组的约束价值

void getans(int n,int m,int i,int a[],double now,int nw[],int w[]) //递归函数

{

if(i>n-1){

if(now>ans){//如果当前价值大于最好价值更新结果

ans=now;

int k;

for(k=0;k<n;++k)//更新结果

x[k]=a[k];

}

return ;

}

int j;

int flag=0;

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

if(w[j]<nw[j]+weight[j][i])

flag=1;

}//查找加入这个值后每组是否符合条件，符合的话flag=1

if(!flag){

a[i]=1;//标记选这个值 等于1表示选，0表示不选

for(j=0;j<m;++j)//更新当前每组的价值

nw[j]+=weight[j][i];

getans(n,m,i+1,a,now+value[i],nw,w);//下一组

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

nw[j]-=weight[j][i];//取消更新当前每组的价值

}

a[i]=0;//不选

getans(n,m,i+1,a,now,nw,w);//下一组

}

int main()

{

int n,m,i,j;

int w[MaxN];

int a[MaxN];

int nw[MaxN];

memset(nw,0,sizeof(nw));

memset(a,0,sizeof(a));

printf("请输入物品个数和属性个数:\n");

scanf("%d%d",&n,&m);

printf("请依次输入物品的价值:\n");

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

scanf("%lf",&value[i]);

printf("请依次输入物品的属性的数值:\n");

for(i=0;i<m;++i)

for(j=0;j<n;++j)

scanf("%d",&weight[i][j]);

for(i=0;i<m;++i)

scanf("%d",&w[i]);

getans(n,m,0,a,0,nw,w);

printf("回溯法:\nx=[");

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

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

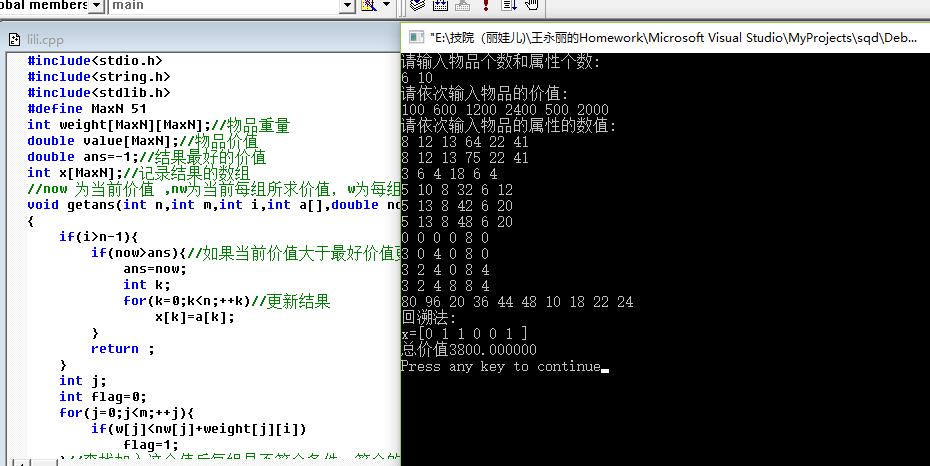
printf("]\n总价值%lf\n",ans);

return 0;

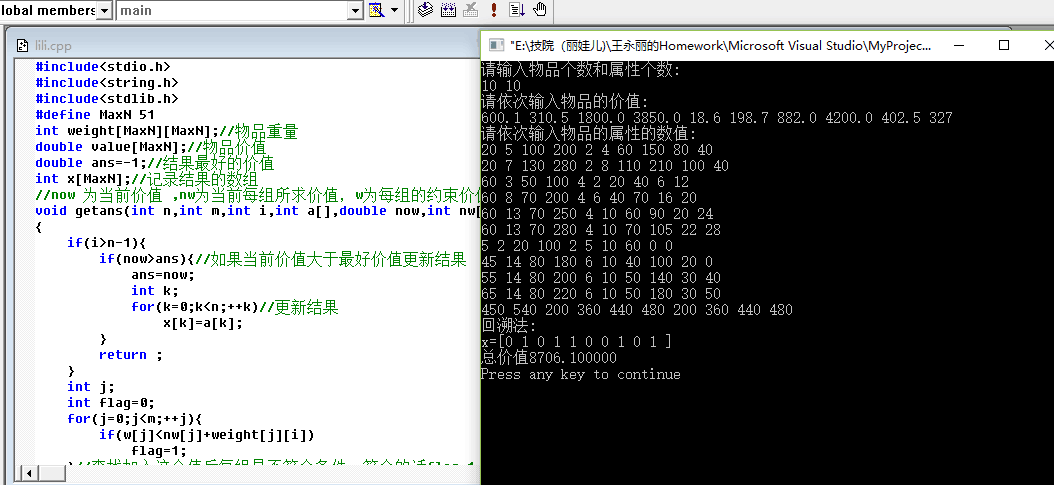
}

**四、运行结果**

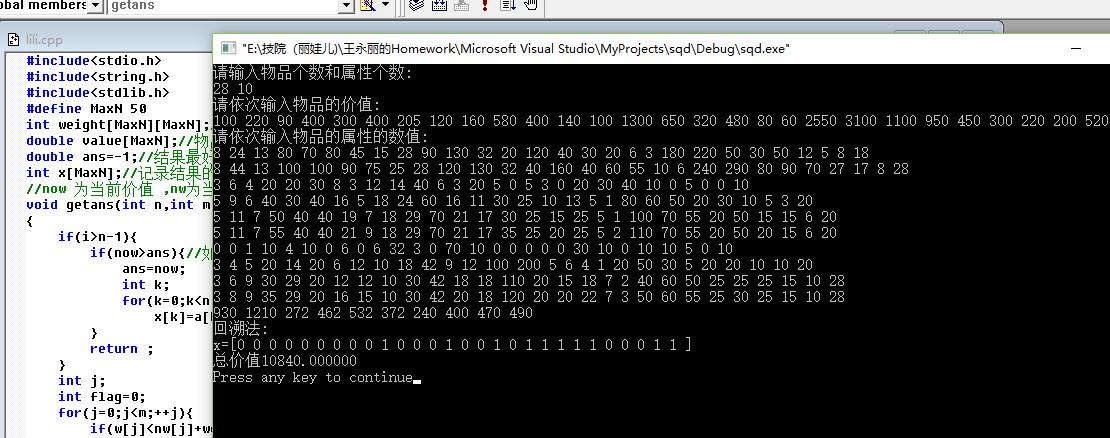
6个物品，10个约束条件时：



10个物品，10个约束条件时：



28个物品，10个约束条件时：



50个物品，5个约束条件时：

不能运行出结果