# 一、算法分析：

1. 算法问题：设计MPK问题的一种算法，分析复杂性，实现算法（语言不限），调试测试，讲解代码。
2. 问题描述：给定n个物品和m个约束条件，输入代表每个物品的价值，输入m行的n个物品属性，输入对应于前面m行每个属性的约束值，在上述属性约束值的条件范围下，求出当前最优价值bestv。
3. 算法使用方法：多维背包问题不能使用贪心算法，不能保证能得到最优价值。多维背包属于找最优解问题，用回溯法需要构造解的子集树（多叉树）。在搜索状态空间树时，只要左子节点是可一个可行结点，搜索就进入其左子树。相对于其右子树时，先计算上界函数，以判断是否满足其约束条件和背包数量将其舍去，以减少算法复杂性！

（四）整体思路：多维背包属于找最优解问题，用回溯法需要构造解的子集树。对于每一个物品i，对于该物品只有选与不选2个决策，总共有n个物品，不考虑其大小排序，可以顺序依次考虑每个物品，这样就形成了一棵解空间树：基本思想就是根据条件输入按序按需利用变量和数组存储数据，首先设计递归结束的判定条件，即当i>n背包数量时，递归结束；当约束条件不满足时，即当前属性+已选物品属性>其对应约束条件时，结束运算；遍历这棵树，以枚举所有情况，最后进行判断，如果重量不超过背包容量，且价值最大的话，该方案就是最后的答案，并将其记录；如此循环再遍历下一个节点，将节点k<n为判断条件，采取递归的形式进行搜索解空间树；采取回溯法将价值、当前属性值进行回溯，继续递归选取下一个节点。

# 算法设计：

（一）利用回溯法设计一个算法求出多维背包问题的解：

在递归函数getBest()中

（1）、当i>n时，算法搜索至叶子结点，得到一个新的物品装包方案。此时算法适时更新当前的最优价值。  
 （2）、当i<n时，当前扩展结点位于排列树的第（i-1）层，此时算法选择下一个要安排的物品，以深度优先方式递归的对相应的子树进行搜索，用约束条件来进行判断取舍。

（二）时间复杂度：

因为物品只有选与不选两张情况，总共有n个物品，递归来看时间复杂度为O（2n）

而该多维背包回溯算法时间复杂度为O（n！），但是该问题总的来说应该还是O（n2）。

（三）源代码：

#include <stdio.h>

#include <stdlib.h>

int n; //物品的数量

int m; //物品的约束条件数量

int w[100][100]; //物品的属性

int c[100]; //约束条件

int cw[100]={0}; //当前属性值

double v[100]; //物品价值

double cv=0; //当前背包价值

double bestv=0;//当前最大的价值

void getBest(int i)

{

int k;

if(i>=n) //递归结束的判定条件，参数i超过物品数量，结束

return;

for(k=0;k<m;k++) //对物品约束条件的数量进行遍历

if(cw[k]+w[i][k]>c[k]) //当前属性+已选物品属性超过了属性的约束条件，则不满足约束条件，结束

return ;

for(k=0;k<m;k++) //否则自身加上当前选择的物品的属性值 ，记录到当前属性值cw[]数组中

cw[k]+=w[i][k];

cv+=v[i]; //并且，加上当前选择物品的价值，用当前背包价值cv记录

if(cv>bestv) //如果当前背包价值>当前最大价值

bestv=cv; //更新当前最大收益

k=i+1; //计算下一个节点要不要加入背包

while(k<n) //k<物品数量

getBest(k++); //对右边节点调用函数自身，递归下去

cv-=v[i]; //此时，回溯还原价值到上一个状态

for(k=0;k<m;k++) //同时

cw[k]-=w[i][k]; //回溯还原属性约束到上一个状态

getBest(i+1); //继续选取下一个节点求最佳

}

int main(){

int i,j;

printf("请输入物品的数量n和约束条件m的个数：");

scanf("%d %d",&n,&m); //记录到n和m中

printf("请输入这%d个物品的价值:\n",n);

for(i=0;i<n;i++){ //物品数和价值数相等

scanf("%lf",&v[i]); //价值记录到数组v[]中，这里共n个物品

}

for(j=0;j<m;j++){ //以i列j行输入为例子，实现提示输入物品的属性，存到数组w[][]中

printf("请依次输入%d个物品的第%d种属性:\n",n,j+1);

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

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

}

}

printf("请依次输入该%d个物品属性的约束值:\n",m);//输入n个物品m钟约束条件数量的约束值

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

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

}

getBest(0); //参数0，调用函数

printf("最优价值为：%lf\n",bestv);//输出最大价值

system("PAUSE");

return 0;

}

# 算法讲解



# 算法调试测试

测试MTP算例：mknapinfo.html；

