【回溯法】－-多维背包问题

1、问题描述

　　给定n种物品和m个属性约束，并规定了属性约束的上限值，把在属性上限值内的并且满足全部属性的物品装到背包里面。属性约束的上限值p[100]，物品价值的数组v[100]，约束物品属性的数组a[100][100]，目前背包的价值pv，所求的最大价值maxv，如何选择物品装入背包中，并且使得装入背包中物品的当前总价值maxv最大。

例如：

规定物品数为6个，属性约束个数为10个，这六个物品的价值分别是100 600 1200 2400 500 2000，六个物品的十个属性分别为：

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

最后是十个属性的上限值80 96 20 36 44 48 10 18 22 24

最终得出这个背包的当前的最大价值为3800

２、算法分析

【整体思路】

　　多维背包属于找最优解问题，这里用回溯法来求解此问题，所以我们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，可以顺序依次考虑每个物品，这样就形成了一棵解空间树： 基本思想就是遍历这棵树，以枚举所有情况，最后进行判断，如果在规定的约束范围内物品的总和不超过背包容量，且价值最大的话，该方案就是最后的答案。

      在搜索状态空间树时，当该物品符合m个属性约束时就进入右子树（放入该物品），不满足时就进入左子树（不能够放入该物品）。当遍历到达根节点时，就可以得到一个解，接着释放该节点的属性值和价值，回溯到上一个节点进行搜索。当遍历完这课树时，通过解的比较，最终找出最大的价值。

【算法设计】

    利用回溯法设计一个算法求出多维背包问题的解，也就是求出对n个物品放或不放的一种的方案。  
　　　　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  
　　　　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。

【时间复杂度】

因为物品只有选与不选2个决策，而总共有n个物品，所以时间复杂度为https://img-blog.csdn.net/20180302134830410?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQvcWlhbjIyMTM3NjI0OTg=/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70。

因为递归栈最多达到n层，而且存储所有物品的信息需要多个一维和二维数组，所以最终的空间复杂度为O(n\*m)。

【源代码】

#include<stdio.h>

int n; //代表给定的物品个数

int m; //代表属性约束的个数

int p[100]; //代表属性约束的上限值

double v[100]; //代表物品价值的数组

int a[100][100]; //代表约束物品属性的数组

double pv; //代表目前背包的价值

double maxv;//代表所求的最大价值

void getBest(int i)//求最优解的函数

{

int j;

if(i>=n)//判断目前是否到达根节点

{

if(pv>maxv)//如果目前背包的价值大于目前的最大价值，则交换

maxv=pv;

return;

}

int count=0;//定义整型count用于判断约束条件成立的计数器

for(j=0;j<m;j++)//用于遍历m个属性

{

if(p[j]-a[j][i]>=0)

{

count++;//代表一个属性约束成立，则计数器就加1

}

else break;//当某个属性不满足约束条件时,即(p[j]-a[j][i]<0)，就跳出循环停止遍历

}

if(count==m)//这里代表m个属性的约束同时成立，进入右子树（代表该物品可以放入背包内）

{

for(j=0;j<m;j++)//再遍历m个属性

{

p[j]-=a[j][i];//放入物品后，相应的m个属性上限值要减少

}

pv+=v[i];

getBest(i+1);

for(j=0;j<m;j++)//遍历m个属性

{

p[j]+=a[j][i];//回溯到上一个节点，相当于把物品拿出来，所以相应的m个属性值上限增加

}

pv-=v[i];//释放拿出物品的价值

}

getBest(i+1);//进入左子树，就是不满足属性约束（不把该物品放入背包）

}

void main()

{

int i,j;

printf("请输入物品的个数及属性约束的个数：\n");

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

printf("输入这%d个物品的价值：\n",n);

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

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

printf("输入这%d个物品的属性约束值的数组：\n",n);

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

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

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

}

}

printf("输入这%d个属性约束的上限值：\n",m);

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

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

getBest(0);

printf("最后所得的最大价值为：%lf\n",maxv);

}