**2020-2021第二学期**

**网络工程专业2018级**

**《算法数据与分析》课程设计**

**班级：网络工程**

**学号：181124039**

**姓名：张怀周**

**0-1背包问题：**

1. **问题描述**

给定n种物品和一背包，物品i的重量是wi，其价值为vi，背包的容量为c。问应如何选择装入背包的物品，使得装入背包中物品的总价值最大？在选择装入背包的物品时，对每种物品i只有两种选择，即装入背包或不装入背包，不能将物品i装入背包多次，也不能只装入部分的物品i。

1. **算法分析**

01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，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。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。对于右子树时，先计算上界函数，以判断是否将其减去（剪枝）。

如果当前背包剩余容量不足，物品i放不进去，则要返回,然后当前的背包容量要加物品i的容量，然后跳物品i，继续下一个物品的遍历。当物品数量大于n时，如果当前背包价值大于最大的背包价值(初始背包价值为零），则把现在的选择方案赋值给最佳选择方案，然后把当前的背包价值赋值给最大价值，然后返回最大的价值。

1. **算法描述**

利用回溯法求出0-1背包问题的解，也就是求出一个解向量ｘ[i ]（即对n个物品放或不放的一种的方案）  
其中，ｘ[i] = 0表示物体ｉ不放入背包，ｘ[i] ＝1表示把物体ｉ放入背包  
在递归函数backtrack中，

当t>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t<n时，当前扩展结点位于排列树的第（t-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，对不满足上界约束条件的结点，则剪去相应的子树。

void Backtrack(int i,int cp,int cw)

{ //cw当前包内物品重量，cp当前包内物品价值

int j;

if(i>n)//回溯结束

{

if(cp>bestp)

{

bestp=cp;

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

bestx[i]=x[i];

}

}

else

for(j=0;j<=1; j++)

{

x[i]=j;

if(cw+x[i]\*w[i]<=c)

{

cw+=w[i]\*x[i];

cp+=p[i]\*x[i];

Backtrack(i+1,cp,cw);

cw-=w[i]\*x[i];

cp-=p[i]\*x[i];

}

}

}

1. **算法实现**

#include<stdio.h>

int n,c,bestp;//物品的个数，背包的容量，最大价值

int p[10000],w[10000],x[10000],bestx[10000];//物品的价值，物品的重量，x[i]暂存物品的选中情况,物品的选中情况

void Backtrack(int i,int cp,int cw)

{ //cw当前包内物品重量，cp当前包内物品价值

int j;

if(i>n)//回溯结束

{

if(cp>bestp)

{

bestp=cp;

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

bestx[i]=x[i];

}

}

else

for(j=0;j<=1; j++)

{

x[i]=j;

if(cw+x[i]\*w[i]<=c)

{

cw+=w[i]\*x[i];

cp+=p[i]\*x[i];

Backtrack(i+1,cp,cw);

cw-=w[i]\*x[i];

cp-=p[i]\*x[i];

}

}

}

int main()

{

int i;

bestp=0;

printf("请输入物品个数和背包最大容量:\n");

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

printf("请依次输入物品的价值:\n");

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

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

printf("请依次输入物品的重量:\n");

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

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

Backtrack(1,0,0);

printf("最大价值为:\n");

printf("%d\n",bestp);

printf("被选中的物品依次是(0表示未选中，1表示选中)\n");

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

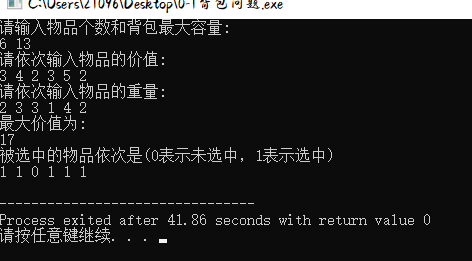
printf("%d ",bestx[i]);

printf("\n");

return 0;

}

1. *}***测试样例**



1. **时空分析**

深度优先搜索的策略，

算法从Backtrack(1,0,0) 开始，调用void Backtrack(int i,int cp,int cw)函数，

当n = 1时，直接就根据物品重量与背包重量比较，只执行一步O(n)=O(1);

当n > 1时，进入for循环

for(j=0;j<=1; ++j) // 枚举i所有可能的路径，在满足限界函数和约束条件if(cw+x[i]\*w[i]<=c)下，连续排列，直到碰壁，然后进行下一条路（另一个条件），进行下一次的遍历，直到结束，最坏情况下有O（2n）个右儿子结点用限界函数，故计算0-1背包问题的回溯算法的时间复杂度为O(n2的n次方)，在搜索过程中的任何时刻，仅保留从开始结点到当前可扩展结点的路径，其空间需求为O（从开始结点起最长路径的长度）。所以，空间复杂度为O（n）

7.**个人心得体会**

算法中使用回溯算法问题进行求解，在回溯中求解中对不符合要求的解进行剪枝处理，并将最优解保存下来，知道算法结束为止，即求出问题的最优解。