

课程实验报告

课 程 名 称： 算法设计与分析

实验项目名称： 0-1背包问题

专 业 班 级： 信息安全1301班

姓 名： 吴炳潮

学 号： 201308060112

指 导 教 师： 张伟

完 成 时 间： 2016 年 06 月 11 日

**实验四 0-1背包问题**

1. 实验要求

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?

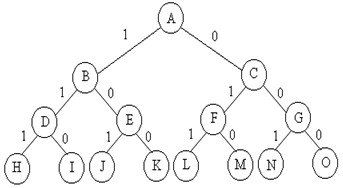
形式化描述：给定c >0, wi >0, vi >0 , 1≤i≤n.要求找一n元向量(x1,x2,…,xn,), xi∈{0,1}, ∋ ∑ wi xi≤c,且∑ vi xi达最大.即一个特殊的整数规划问题。

1. 实验分析

首先，要对输入数据进行预处理，将各物品依其单位重量价值从大到小进行排列。在优先队列分支限界法中，节点的优先级由已装袋的物品价值加上剩下的最大单位重量价值的物品装满剩余容量的价值和。

算法首先检查当前扩展结点的左儿子结点的可行性。如果该左儿子结点是可行结点，则将它加入到子集树和活结点优先队列中。当前扩展结点的右儿子结点一定是可行结点，仅当右儿子结点满足上界约束时才将它加入子集树和活结点优先队列。当扩展到叶节点时为问题的最优值。

例如当n=3时，w={16,15,15}, p={45,25,25}, c=30。优先队列式分支限界法：处理法则:价值大者优先。{}—>{A}—>{B,C}—>{C，D，E}—>{C，E}—>{C，J，K}—>{C}—>{F，G}—>{G，L，M}—>{G，M}—>{G}—>{N，O}—>{O}—>{}



1. 实验代码

#include<iostream>

#include<windows.h>

#include<queue>

#define N 200

using namespace std;

class HeapNode

{

public:

double uprofit;//节点的价值上界

double profit; //节点所对应的价值

double weight;//结点所对应的重量

int level;//活结点在子集树种所处的层序号

HeapNode\* parent;

bool valid;

};

queue<HeapNode\*> H; //堆

double w[N],p[N]; //

bool x[N]; //记录包裹的放入

bool y[N]; //记录包裹的未放入

double cw,cp,c;

int n; //总共的背包数目

HeapNode\* node;

double bestp=0;

int k=0;

/\*

计算上限的函数

int i：代表当前的层数

\*/

double Bound(int i)

{

double cleft=c-cw,b=cp;

while(i<=n&&w[i]<=cleft)

{

cleft-=w[i];

b+=p[i];

i++;

}

if(i<=n) b+=p[i]/w[i]\*cleft;

return b;

}

/\*

加入当前的堆中

参数：

1.double up：节点的价值上界

2.double cp; //节点所对应的价值

3.double cw;//结点所对应的重量

4.int level;//活结点在子集树种所处的层序号

\*/

void AddLiveNode(double up,double cp,double cw,bool ch,int level)

{

HeapNode\* nod=new HeapNode();

nod->uprofit=up;

nod->profit=cp;

nod->weight=cw;

nod->level=level;

nod->valid=ch;

nod->parent=node;

if(level<=n+1) H.push(nod);

}

/\*

\*/

double Knap()

{

int i=1;

cw=cp=0;

node=0;

double up=Bound(1);

while(i!=n+1)

{

double wt=cw+w[i];

if(wt<=c)

{

if(cp+p[i]>bestp) bestp=cp+p[i];

AddLiveNode(up,cp+p[i],cw+w[i],true,i+1);

k++;

}

up=Bound(i+1);

if(up>=bestp){

AddLiveNode(up,cp,cw,false,i+1);

k++;

}

//if(H.empty())

// return bestp;

node=H.front();

H.pop();

cw=node->weight;

cp=node->profit;

up=node->uprofit;

i=node->level;

}

//删选出最后一层的最优解

int a=H.size();

for(int i=0;i<H.size();i++){

if(node->uprofit<H.front()->uprofit){

node=H.front();

H.pop();

}

else{

H.pop();

}

}

for(int j=n;j>0;j--){

x[j]=node->valid;

node=node->parent;

}

}

int main()

{

cout<<"请输入n和c:"; cin>>n>>c;

cout<<"请输入w[i]"<<endl;

for(int i=1;i<=n;i++) cin>>w[i];

cout<<"请输入p[i]"<<endl;

for(int i=1;i<=n;i++) cin>>p[i];

Knap();

cout<<"最优值是："<<bestp<<endl;

//cout<<node->profit<<endl;

cout<<"各个背包存取的情况："<<endl;

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

cout<<x[i]<<" ";

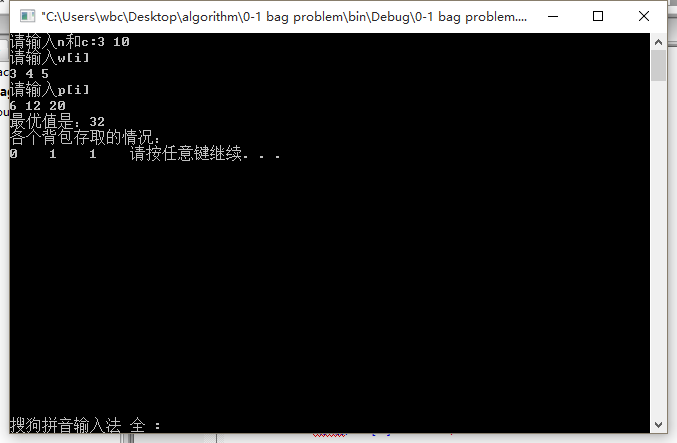
}

system("pause");

return 0;

}

1. 实验结果



1. 实验心得

之前参考了网上的代码和书上的代码，但是不能输出各个背包的放入是否，之后自己修改代码之后能得到正确的结果。