贪心法求解0-1背包问题

171491106 安南 硬件一班

一、实验题目

贪心法求解0-1背包问题

二、算法简介

贪心算法，又称贪婪算法(Greedy Algorithm)，是指在对问题求解时，总是做出在当前看来是最好的选择。也就是说，不从整体最优解出发来考虑，它所做出的仅是在某种意义上的局部最优解。贪婪算法是一种分阶段的工作，在每一个阶段，可以认为所做决定是最好的，而不考虑将来的后果。这种“眼下能够拿到的就拿”的策略是这类算法名称的来源。贪心算法没有固定的算法框架，算法设计的关键是贪心策略的选择。必须注意的是，贪心算法不是对所有问题都能得到整体最优解，选择的贪心策略必须具备无后效性，即某个状态以后的过程不会影响以前的状态，只与当前状态有关。所以对所采用的贪心策略一定要仔细分析其是否满足无后效性。

三、实验目的

给定n种物品和一个容量为C的背包，物品i的重量是wi，其价值为vi，背包问题是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?编写一个贪心算法来解决这个问题。

四、实验代码

**package** 贪心法;

**import** java.util.Arrays;

**import** java.util.Scanner;

**public** **class** 背包 {

**public** **static** **void** main(String[] args)

{

Scanner in=**new** Scanner(System.***in***);

System.***out***.println("请输入物品的数量:");

**int** n=in.nextInt();

**int**[] w=**new** **int**[n];

**int**[] v=**new** **int**[n];

System.***out***.println("现在请输入这些物品的重量:");

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

{

w[i]=in.nextInt();

}

System.***out***.println("现在请输入这些物品的价值):");

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

{

v[i]=in.nextInt();

}

System.***out***.println("现在请输入背包的容量:");

**int** c=in.nextInt();

/\*\*

\*按单位重量价值r[i]=v[i]/w[i]降序排列

\*/

**double** startTime=System.*currentTimeMillis*();

**double**[] r=**new** **double**[n];

**int**[] index=**new** **int**[n];

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

{

r[i]=(**double**)v[i]/(**double**)w[i];

index[i]=i;

}

**double** temp=0;

**for**(**int** i=0;i<n-1;i++)

{

**for**(**int** j=i+1;j<n;j++)

{

**if**(r[i]<r[j])

{

temp=r[i];

r[i]=r[j];

r[j]=temp;

**int** x=index[i];

index[i]=index[j];

index[j]=x;

}

}

}

/\*\*

\*排序后的重量和价值分别存到w1[]和v1[]中

\*/

**int**[] w1=**new** **int**[n];

**int**[] v1=**new** **int**[n];

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

{

w1[i]=w[index[i]];

v1[i]=v[index[i]];

}

System.***out***.println (Arrays.*toString*(w1));

System.***out***.println (Arrays.*toString*(v1));

/\*\*

\*初始化解向量x[n]

\*/

**int**[] x=**new** **int**[n];

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

{

x[i]=0;

}

/\*\*

\*求解并打印解向量

\*/

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

{

**if**(w1[i]<c)

{

x[i]=1;

c=c-w1[i];

}

}

System.***out***.println("解向量是:"+Arrays.*toString*(x));

/\*\*

\*根据解向量求出背包中存放物品的最大价值并打印

\*/

**int** maxValue=0;

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

{

**if**(x[i]==1)

maxValue+=v1[i];

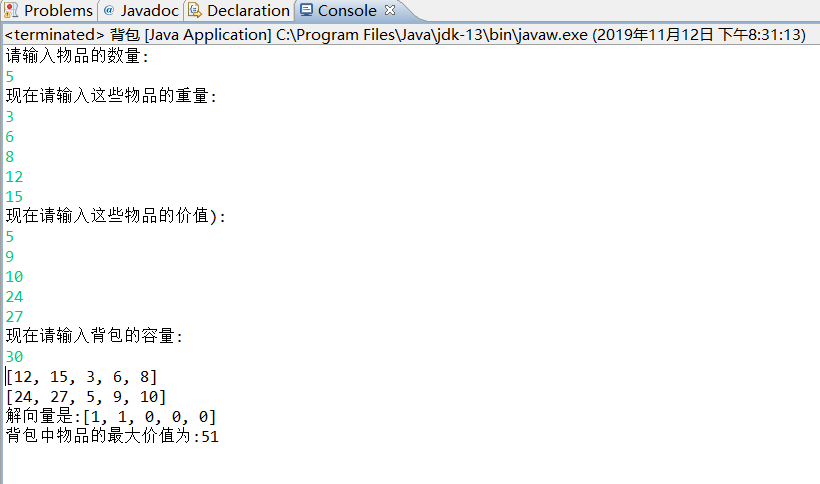
}

**double** endTime=System.*currentTimeMillis*();

System.***out***.println("背包中物品的最大价值为:"+maxValue);

}

五、实验结果



六、实验总结

用贪心法解决问题首先要建立数学模型来描述问题。主要思想就是把求解的问题分成若干个子问题，然后对每一子问题求解，得到子问题的局部最优解。贪心法一般适用于有着：局部最优策略能导致产生全局最优解。也就是当算法终止的时候，局部最优等于全局最优这些特点的问题。