### 贪心搜索求解算法

1. 实验题目

利用贪心搜索算法（Greedy）求解背包问题

1. 算法分析

贪心算法（又称贪婪算法）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，不从整体最优上加以考虑，他所做出的是在某种意义上的局部最优解。贪心法不考虑他的子算法是否最优，只顾当前所以是一种“狭隘的算法“。

1. 实验目的

用贪心法解决背包问题，已知物品的重量和价值，分别按照价值最大优先，重量最小优先，和价容比最大优先放入背包，观察最后的结果和比较最优的方案

四、实验代码：

**package** text;

**public** **class greedy** {

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

**int**[] a ={5,2,3,8,7,6,4,1};//重量

**int**[] b ={3,8,5,4,7,5,1,2};//价值

**int** max = 15;//背包重量 15

System.***out***.println("按价值最大先放入的总价值是：" + *bigM*(a,b,max));

System.***out***.println("按重量最小先放入的总价值是：" + *smallW*(a,b,max));

System.***out***.println("按价容比最大先放入的总价值是：" + *bigB*(a,b,max));

}

**private** **static** **int** bigB(**int**[] a, **int**[] b, **int** max) {

**int** sum = 0;

**int**[] c = **new** **int**[a.length];

//求得价容比

**for** (**int** i = 0; i < c.length; i++) {

c[i] = b[i]/a[i];

}

//根据价容比从大到小进行排序

**for** (**int** i = 0; i < c.length-1; i++) {

**for** (**int** j = i+1; j < c.length; j++) {

**if** (c[i] < c[j]) {

**int** y = c[i];

c[i] = c[j];

c[j] = y;

**int** temp = a[i];

a[i] = a[j];

a[j] = temp;

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

b[i] = b[j];

b[j] = x;

}

}

}

//从价容比最大的开始放

**for** (**int** i = 0; i < c.length; i++) {

**if** (a[i]<=max) {

sum +=b[i];

max =max -a[i];

}**else** {

**break**;

}

}

**return** sum;

}

**private** **static** **int** smallW(**int**[] a, **int**[] b, **int** max) {

**int** sum = 0;

**int** temp;

//先按重量排序 从小到大

**for** (**int** i = 0; i < a.length-1; i++) {

**for** (**int** j = i+1; j < b.length; j++) {

**if** (a[i]>a[j]) {

temp =a[i];

a[i] = a[j];

a[j] = temp;

//价值数组的下标也随之改变

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

b[i]=b[j];

b[j]=x;

}

}

}

//按照从重量最轻开始放

**for** (**int** i = 0; i < a.length; i++) {

**if** (a[i]<=max) {

sum +=b[i];

max =max -a[i];

}**else** {

**break**;

}

}

**return** sum;

}

**private** **static** **int** bigM(**int**[] a, **int**[] b, **int** max) {

**int** sum = 0;

**int** temp;

//先按价值排序 从大到小

**for** (**int** i = 0; i < a.length-1; i++) {

**for** (**int** j = i+1; j < b.length; j++) {

**if** (b[i]<b[j]) {

temp =b[i];

b[i] = b[j];

b[j] = temp;

//重量数组的下标也随之改变

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

a[i]=a[j];

a[j]=x;

}

}

}

//按照从价值最高开始放

**for** (**int** i = 0; i < b.length; i++) {

**if** (a[i]<=max) {

sum +=b[i];

max =max -a[i];

}**else** {

**break**;

}

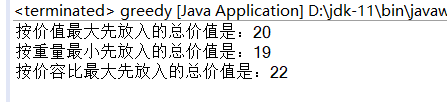
}

**return** sum;

}

}

1. 实验结果：



1. 实验总结：

由实验结果可知，按照价容比最大优先放入背包问题是最优的因为如果按照重量最小的放进背包，但是有可能一些最贵重的不能放入，如果按照价值最大的放入背包，也有可能它本身占据的重量很大，占据了很多空间，导致达不到最优解，相反，按照价容比最大放入，就可以把价值细分到每一个单位，单位重量的价值最高的先放入背包。每一个当前的选择都是最优的，所以得到的解也是最优的