**算法分析与设计实验报告**

**第 二 次实验**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 姓名 | 李平凡 | 学号 | 201907040102 | | 班级 | 计科1905 |
| 时间 | 4.20 | 地点 | 软件大楼 | | | |
| 实验名称 | 贪心法求解背包问题 | | | | | |
| 实验目的 | 理解贪心算法的概念  掌握贪心算法的基本要素1，最优子结构性质 2，贪心选择性质  理解贪心算法的一般理论 | | | | | |
| 实验原理 | 背包问题就是在有限容量内使得背包里的物品的价值最大，由于每个物品的价值和重量不定，所以需要一个标准来判断物品放不放，并且背包问题允许物品只放一部分，所以选定标准后，每次放置当前重量较小并且当前价值最大的物品即可求解问题。 | | | | | |
| 实验步骤 | 1. 输入数据然后选定一个当前标准来判断物品放不放，由于是在有限容量内保证物品价值最大，那么就选择单位重量的价值作为判断依据，将所有物品按此标准从大到小排个序。 2. 然后判断当前背包容量是不是大于这个物品重量如果大于就可以将这个物品放进去，当到某一个物品背包容量不允许该物品放进去就跳出循环 3. 将该物品的一部分放进去，也就是（weight/w[i]）\*v[i]   4，最后得到总的结果 | | | | | |
| 关键代码 | int i=0;  double value;  while(weight>A[i].w)  {  X[i]=1;  weight-=A[i].w;  value=value+A[i].v;  i++;  }  if(weight>0)  {  value=value+(weight/A[i].w)\*A[i].v;  X[i]=weight/A[i].w;  }  return value;   1. 第一个循环就是找到当前所有可以完整放进去的物品，同时构造一个解向量 2. 第二个判断就是将背包最后的一点容量用来装当前单位重量价值最大物品的一部分 | | | | | |
| 测试结果 | 由于十万数据规模可能存在数组溢出所以没有显示出最终结果 | | | | | |
| 实验心得 | 与01背包问题相比，背包问题可以保证物品的一部分装进背包中，于是就可以采用贪心算法，每次装的物品都是当前单位重量价值最大的物品，如果不能装进物品的一部分，那么就会发现这种贪心算法求解背包问题是不能得到最优解的因为不能保证最后一点背包容量填充，这个时候如果还选用贪心算法就需要改进，考虑到不能填充背包的问题，由于贪心算法中采用了算法头文件中的排序函数所以最大的时间开销就是O(nlogn) | | | | | |
| 实验得分 |  | 助教签名 | |  | | |