**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 | 01背包问题 | | | | | |
| 实验目的 | 根据动态规划解题步骤（问题抽象化、建立模型、寻找约束条件、判断是否满足最优性原理、找大问题与小问题的递推关系式、填表、寻找解组成）找出01背包问题的最优解以及解组成，然后编写代码实现。 | | | | | |
| 实验原理 | 动态规划法与分治法类似都是对于最优子结构问题将大问题划分为小问题但不同的是，动态规划法的子问题相互之间是有联系的，所以在进行一般方法的计算的时候往往会重复计算，为了减少重复计算就有了动态规划法，记录某一子问题当前最优解然后成为下一个大问题的最优解，对于这个01背包问题，当前背包装不装物体是一个大问题如果装那么就需要有足够的容量，并且选取装物体还是不装物体哪一种方式价值最大，如果不装就判断下一个物体装不装，从而将当前物体装不装的大问题划分成下一个物体装不装的小问题 | | | | | |
| 实验步骤 | 1，输入数据然后进行当前物体的判断，如果当前背包容量大于物体重量可以装进背包，在不装该物品和装该物品两种方式中选取一个价值最大的方式，如果当前背包容量小于物体重量那么就不装判断下一个物体装不装。  2，思考的方式是自顶向下，但是实现需要自底向上，建立一个二维数组行为物品标号从小到大，列为背包容量，一行一行的开始补。  3，最后输出二维数组中最后一个元素的值即可。 | | | | | |
| 关键代码 | for(int i=0;i<=n;i++)  {  for(int j=0;j<bsg;j++)  {      y[i][j]=0;    }  }  for(int i=1;i<=n;i++)  {  for(int j=1;j<=bsg;j++)  {  if(w[i]>j)  {  y[i][j]=y[i-1][j];  }  else  {  y[i][j]=max(y[i - 1][j], y[i - 1][j - w[i]] + v[i]);  }  }  }  1，第一个for循环是初始化  2，第二个for循环注意一个细节，i，j从1开始这是因为背包容量为0的时候价值肯定是0，物品的标号是从1开始不是从0开始所以第一行为0。  3，如果背包容量大于当前物品重量那么就得选择是装还是不装物品y[i - 1][j]代表的是不装物品，那么就和前面i-1个物品价值相同并且背包容量还是j，如果装就是 y[i - 1][j - w[i]] + v[i]，在装物品之前的最大价值就是y[i - 1][j - w[i]]然后再加上装进这个物品之后的价值就是总价值。  4，如果背包容量不够装就和之前i-1个物品最大价值相同 | | | | | |
| 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 | 01背包问题采用动态规划法可以发现就是在填一个二维数组的表格，会发现，整个算法时间开销最多的地方就是二维数组的填写，行与物品个数n有关，列与背包容量m有关，所以这个问题的时间复杂度就是O（mn）由于背包的容量可以是很大的一个数，当这个数为n次幂的时候01背包问题就变成了np问题，此外采用这种动态规划法解题无法解决当背包容量，物体重量不是整数或者背包容量很大时候的问题，因此看似一个简单的问题其实稍微改一改，难度就不是一个水平了。 | | | | | |
| 实验得分 |  | 助教签名 | |  | | |

**附录：完整代码**