**算法设计与分析**

**实验报告**

|  |  |
| --- | --- |
| **专业班级** | **计算机科学与技术21-1班** |
| **学生姓名** | **何超** |
| **学 号** | **202001021107** |
| **指导老师** | **张鹏** |

**山东科技大学**

|  |  |  |  |
| --- | --- | --- | --- |
| **实验名称** | 01背包问题的回溯算法 | | |
| **实验日期** | **2022/06/17** | **实验地点** | J13-332 |
| **指导老师** | **张鹏** | **实验成绩** |  |

1. **实验目的和要求：**

要求用回溯法求解01背包问题

交互输入背包容量，物品重量数组，物品价值数组

使用回溯法计算背包所含最大价值

1. **实验内容：**

**解决问题：**

有N件物品和一个最多能背重量为W的背包，第i件物品的重量是weight[i]，其价值是value[i]，每件物品只能背一次，求解将哪些物品放到背包里面物品价值的总和最大。

**具体内容：**

1.确定动态规划在该问题中dp数组其下标的含义

2.确定递推公式

3.确定dp数组中数值的计算方式和计算顺序

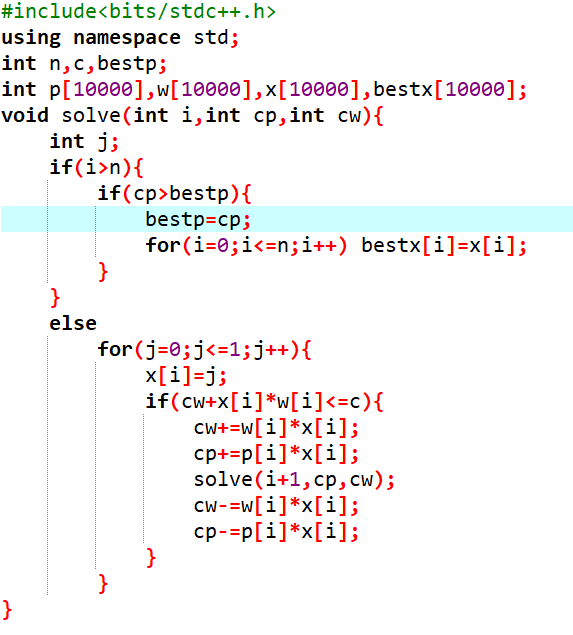
4.编写代码实现过程

**三、算法描述及实验步骤：**

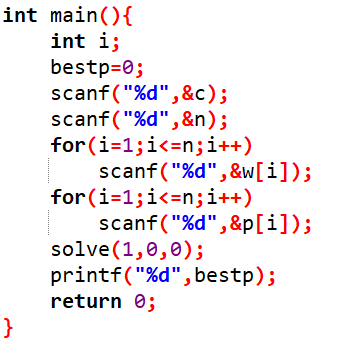
1.分别用p[i]和w[i]存储第i种物品的价值和重量，用x[i]标记第i种物品是否装入背包，用bestx[i]存储第i种物品的最优装载方案；  
2.用递归函数Backtrack (i,cp,cw)来实现回溯法搜索子集树（i表示递归深度，n用来控制递归深度，形式参数cp和cw表示当前总价值和总重量，bestp表示当前最优总价值），若i >n，则算法搜索到一个叶结点，判断当前总价值是否最优，若cp>bestp，更新当前最优总价值为当前总价值（即bestp=cp），更新装载方案（即bestx[i]=x[i]( 1≤i≤n)）；  
3.采用for循环对物品i装与不装两种情况进行讨论（0≤j≤1），若总重量不大于背包即（cw+x[i]\*w[i]<=c），则更新当前总价 br=""> 值和总重量（cw+=w[i]\*x[i],cp+=p[i]\*x[i]），对物品i+1调用递归函数Backtrack(i+1,cp,cw) 继续进行装载，函数Backtrack(i+1,cp,cw)调用结束后则返回当前总价值和总重量（即 cw-=w[i]\*x[i],cp-=p[i]\*x[i]），当j>1时，for循环结束。  
4.当i=1时，若已测试完所有装载方案，外层调用就全部结束；  
5.编写主函数，在其中调用一次backtrack(1,0,0)即可完成整个回溯搜索过程，最终得到的bestp和bestx[i]即为所求最大总价值和最优装载方案。

**四、实验调试过程及输入输出结果：**

1.根据算法设计的思路与动态规划思想，编写函数solve实现算法。首先建立数组物品重量w，物品价值p，物品暂存x，物品选中bestx，最大容量c，物品数量n，当前包内物品重量cw，当前包内物品价值cp，输入数据，代码如下：

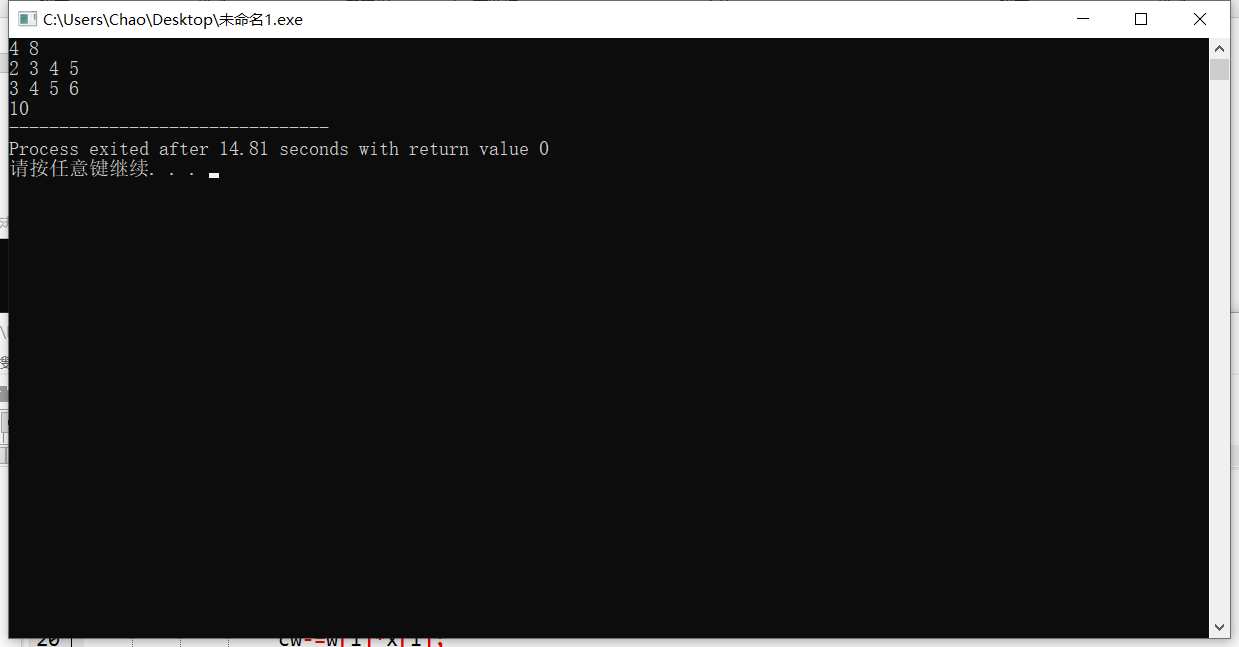
****

2.编写main函数接收用户数据，调用solve函数，输出结果：



3.输入测试数据，验证程序正确性：

使用与之前动态规划解决的01背包问题相同的测试数据。得到相同测试结果，程序正确。

****

**五、实验总结：**

在本次实验中，我们使用回溯算法解决01背包问题的实现，使用回溯法解决此类问题大概分为三步，首先，需要确定解空间：决定装入哪几种物品，其次，确定易于搜索的解空间结构，我们选择使用用数组p,w分别表示各个物品价值和重量。并记录是否选中物品，最后，搜索解空间，并在搜索过程中进行适当剪枝。回溯法是一个既带有系统性又带有跳跃性的的搜索算法。它在包含问题的所有解的解空间树中，按照深度优先的策略，从根结点出发搜索解空间树。算法搜索至解空间树的任一结点时，总是先判断该结点是否肯定不包含问题的解。如果肯定不包含，则跳过对以该结点为根的子树的系统搜索，逐层向其祖先结点回溯。否则，进入该子树，继续按深度优先的策略进行搜索。回溯法在用来求问题的所有解时，要回溯到根，且根结点的所有子树都已被搜索遍才结束。而回溯法在用来求问题的任一解时，只要搜索到问题的一个解就可以结束。这种以深度优先的方式系统地搜索问题的解的算法称为回溯法，适用于解一些组合数较大的问题。