**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背包问题的动态规划算法实现 | | |
| **实验日期** | **2022/06/10** | **实验地点** | J13-332 |
| **指导老师** | **张鹏** | **实验成绩** |  |

1. **实验目的和要求：**

分析01背包问题的动态规划可行性

确定01背包问题动态规划解决的递推方程

使用C++语言编写01背包问题的动态规划程序设计

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.我们定义dp数组下标含义：dpvalue[i][j]表示在在容量限制为j时，决定第i件物品是否装入背包时背包中的总价值为dpvalue[i][j]

2.由数组定义与动态规划思想可得：程序应讨论两种情况：

一，物品i不放入背包中

二，物品i放入背包中

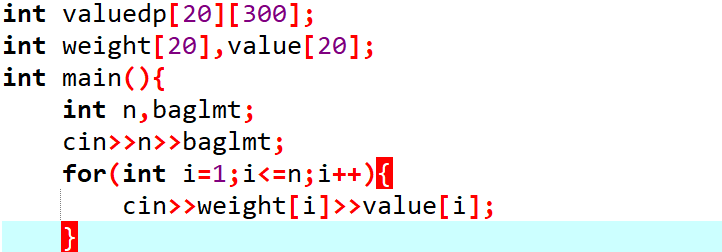
对于情况一，不放入的原因应当是容量不足以放入物品，否则就应当放入，成为情况二。由此，我们确定了递推公式：dp[i][j]=max(dp[i-1][j] , dp[i-1][j-weight[i]]+value[i])。

3.我们使用先遍历物品，在遍历物品时不断试探背包容量的逻辑关系。

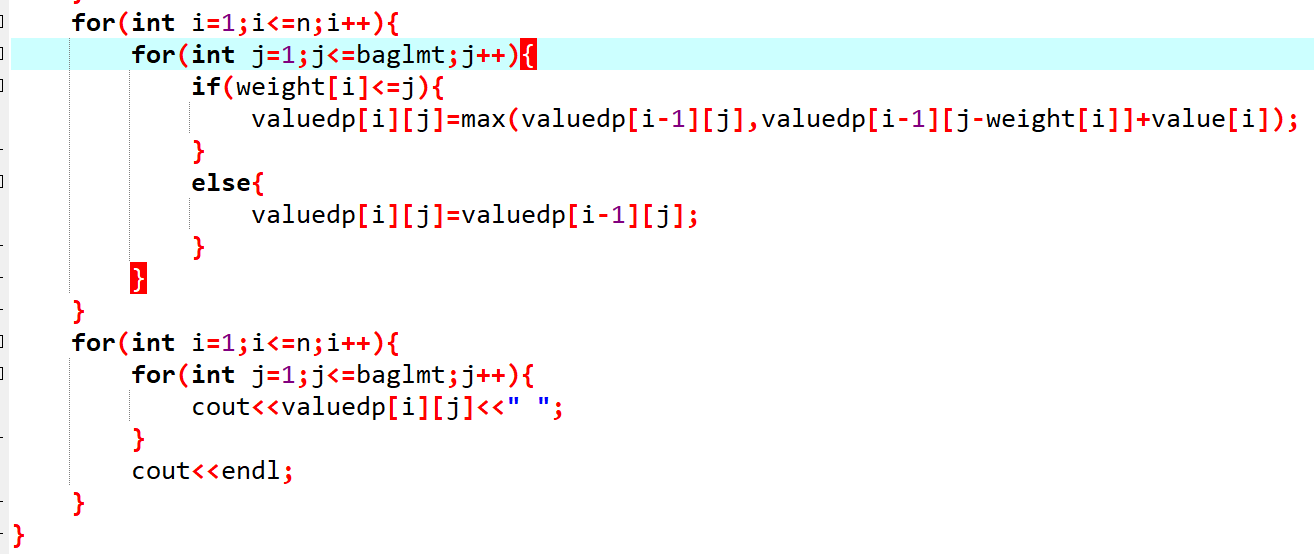
4.最后，在确定所有的算法细节后，编写main函数，实现对算法设计的程序语言实现，编写部分数据，测试代码编写是否正确。

**四、实验调试过程及输入输出结果：**

根据算法设计的思路与动态规划思想，在确定了递推公式dp[i][j]=max(dp[i-1][j] , dp[i-1][j-weight[i]]+value[i])后，直接在main函数中实现算法。首先建立两个数组weight和value，物品数量n，输入数据，代码如下：



根据算法逻辑，如果当前物品超过了当前容量限制，则保持不变，dpvalue表中的值保持和没放置物品i的价格一致，而若可以放置，则需要比较未放置时背包容量扣去物品重量时的价值和物品i价值的加和与不放置时的最大值。遍历dpvalue表的C++代码实现如下：



**程序运行测试：**

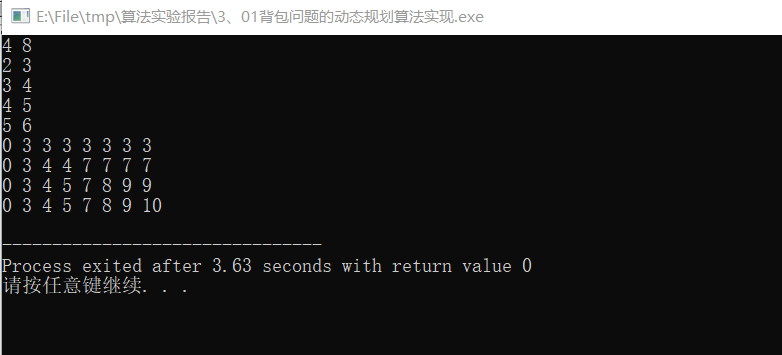
编写数据：物品数量4，背包载重5，

物品1：重量2价值3

物品2：重量3价值4

物品3：重量4价值5

物品4：重量5价值6



由运行结果可知，dpvalue表中右下角值10即为该背包在不超过载重的情况下面对当前物品的最大价值

**五、实验总结：**

在本次实验中，我们使用动态规划思想解决了01背包问题的实现，在实验过程中我们发现此类问题一般分为5个步骤，也就是确定dp数组下标含义，确定递推公式，初始化dp数组，确定遍历的顺序，确定推导dp数组的具体方法及其中数值计算的代码实现。在使用动态规划时要注意其问题应该满足最优子结构性质，最优子结构的性质指不论初始状态和初始决策如何，对于前面决策所造成的某一状态而言，其后各阶段的决策序列必须构成最优策略，通俗来讲也就是可以把大问题拆分成小问题，再找大小问题间的递推关系。由最基本的小问题开始解决，逐渐解决大问题。在探究01背包问题的解法时，我们可以发现动态规划与分治法的不同点是分治法不会记忆中间的过程，所以需要重复计算子问题。而动态规划往往以填表的形式将已经解决了的小问题的答案都记录下来，以供后续计算使用，避免了重复计算而节省时间。在问题满足最优性原理后，动态规划解决问题的核心就在于填表，表填写完毕，就可以找到最优解。