**西 南 石 油 大 学 实 验 报 告**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **课程** | **算法分析与设计** | **实验项目** | | **动态规划算法设计与实现** | | | |
| **专业班级** | **计算机科学与技术2022级** | | **姓名** | | **王磊** | **学号** | **202231060435** |

# 1、实验内容

## （1）实验步骤

· 理解0-1背包问题的定义和动态规划解决方法。

· 设计并实现动态规划算法。

· 测试算法，记录运行结果。

· 分析算法的时间复杂度和空间复杂度。

## 算法原理

0-1背包问题可以用动态规划来解决。其**基本思想**是：

创建一个二维数组`dp`，其中`dp[i][w]`表示前`i`个物品中总重量不超过`w`时 的最大价值。

状态转移方程为：

dp[i][w] = max(dp[i-1][w], dp[i-1][w-w\_i] + v\_i)

其中，(w\_i)是第`i`个物品的重量，(v\_i)是第`i`个物品的价值。

### 算法设计：

#### 1）确定dp数组以及下标的含义：

对于背包问题，使用二维数组，即dp[i][j] 表示从下标为[0-i]的物品里任意取，放进容量为j的背包，价值总和最大是多少。

#### 2）确定递推公式：

两个方向推出来dp[i][j]，

不放物品i：由dp[i - 1][j]推出，即背包容量为j，里面不放物品i的最大价值，此时dp[i][j]就是dp[i - 1][j]。(其实就是当物品i的重量大于背包j的重量时，物品i无法放进背包中，所以背包内的价值依然和前面相同。)

放物品i：由dp[i - 1][j - weight[i]]推出，dp[i - 1][j - weight[i]] 为背包容量为j - weight[i]的时候不放物品i的最大价值，那么dp[i - 1][j - weight[i]] + value[i] （物品i的价值），就是背包放物品i得到的最大价值

所以**递归公式**： dp[i][j] = max(dp[i - 1][j], dp[i - 1][j - weight[i]] + value[i]);

#### 3）dp数组如何初始化：

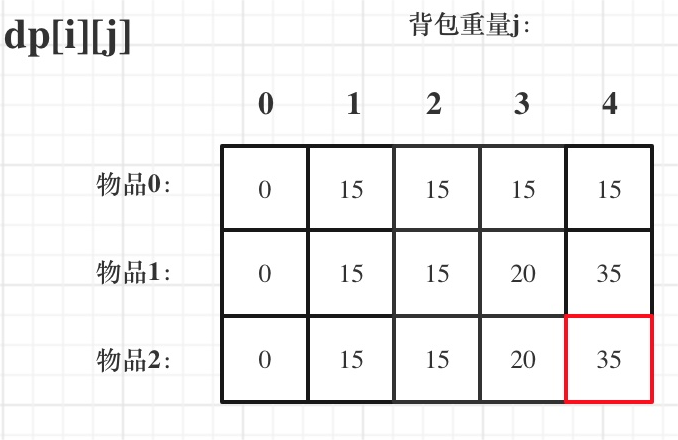
首先从dp[i][j]的定义出发，如果背包容量j为0的话，即dp[i][0]，无论是选取哪些物品，背包价值总和一定为0；在看其他情况。状态转移方程 dp[i][j] = max(dp[i - 1][j], dp[i - 1][j - weight[i]] + value[i]); 可以看出i 是由 i-1 推导出来，那么i为0的时候就一定要初始化。 dp[0][j]，即：i为0，存放编号0的物品的时候，各个容量的背包所能存放的最大价值。那么很明显当 j < weight[0]的时候，dp[0][j] 应该是 0，因为背包容量比编号0的物品重量还小。当j >= weight[0]时，dp[0][j] 应该是value[0]，因为背包容量放足够放编号0物品。其它下标对问题影响不大，统一把dp数组统一初始为0。

#### 确定遍历顺序

先遍历物品，然后遍历背包重量

（两种遍历顺序即可，我这里选择该遍历顺序）

#### 举例推导dp数组



验证自己的推算过程没有问题

## 测试数据

#### 输入描述：

第一行包含两个正整数，第一个整数 M 代表物品的种类，第二个正整数 N，代表背包空间。 第二行包含 M 个正整数，代表每种物品的所占空间。 第三行包含 M 个正整数，代表每种物品的价值。

测试数据一：

输入：

6 1

2 2 3 1 5 2

2 3 1 5 4 3

输出： 5

测试数据二：

输入：

1 0

5

10

输出：0

测试数据三：

输入：

3 10

1 2 3

100 100 100

输出：300

测试数据四：

输入：

5 5

5 5 5 5 5

10 20 30 40 50

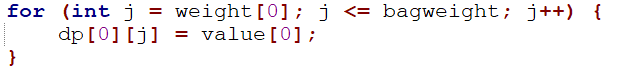
输出： 50

## （4）程序主要流程

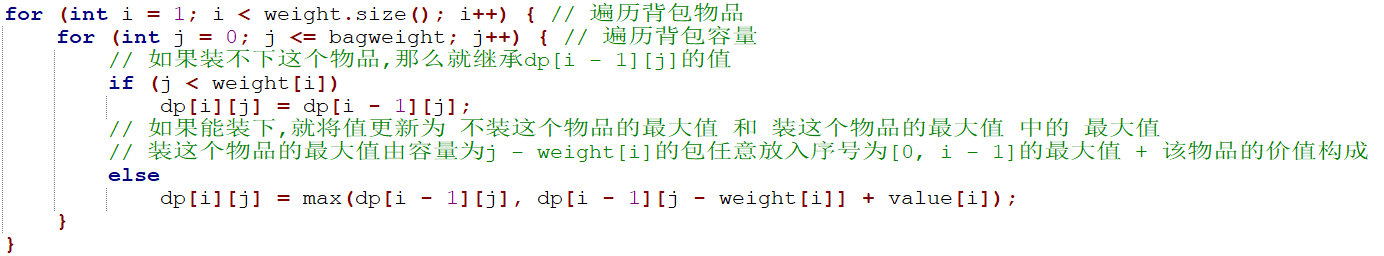
1. 输入数据
2. 初始化二维数组
3. 迭代每个物品，并更新`dp`数组。
4. 返回`dp[i][j]`，其中`i`是物品数量，`j`是背包容量。

### 主要模块功能

* 初始化dp数组



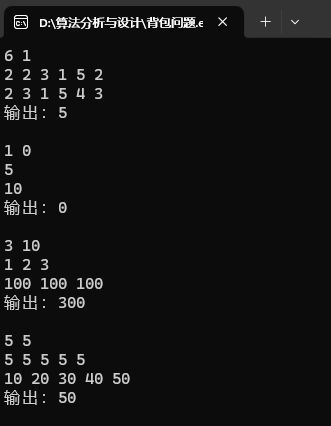
* 迭代更新‘dp’数组



## 数据结构设计

使用二维数组dp来存储中间结果。

## 运行结果（截图）



## （8）算法时间复杂度

时间复杂度：O(i\*j)

### 1、小结

（自己对于实验的结论、认识、心得/体会等）

在本次实验中，我学习、练习并深入理解了动态规划算法，从它的算法思维中收获颇丰。

在实验之初，由于难以理解算法，思索良久无从下笔，后经查阅大量资料和观看教学视频，开始为我所要解决的问题设计算法并编写代码，由于对算法理解不够透彻，经常犯一些错误，例如最初写的时候忘了初始化代码、遍历的时候不确定先遍历物品还是背包空间以及没理解清楚不知如何实现遍历过程，还有在递归时经过了多次学习和思考才理解了递归公式的含义及过程，好在最后成功解决问题。

经过本次实验，我理解了动态规划算法的核心思想是通过将原问题分解为相对简单的子问题，并在解决子问题的过程中保存中间结果，避免了重复计算，显著减少问题的计算量，来提高算法效率。也更深入理解和练习了递归的使用。

### 附录：源代码

（完整代码）

