# 第二次算法设计作业

姓名： 学号：

【说明】共两道题，至少完成一题，全完成可加分。解答应包括解题思路（方法、时间复杂度分析）、代码以及按结果验证中的输入得到的输出截图。作业提交时请只保留选做的题，删除空白题。

【题一】背包问题是一个经典的动态规划问题，在给定背包容量和一组物品的重量和价值情况下，要求选择装入背包的物品，使得装入的物品总价值最大。0-1背包问题是背包问题的一个特例，即每种物品最多只能选择一次装入背包。编写一个程序，实现0-1背包问题的求解，并给出示例的输入输出。

【示例】输入：背包容量：10

物品重量：[2, 3, 4, 5]

物品价值：[3, 4, 5, 6]

输出：选择的物品编号：[2, 3]（这里编号从1开始计数）

总价值：9

【解题思路】

【代码】

【结果验证】1、输入： 背包容量：10

物品重量：[2, 3, 4, 5]

物品价值：[3, 4, 5, 6]

1. 输入： 背包容量：8

物品重量：[1, 3, 4, 5]

物品价值：[2, 4, 5, 7]

【题二】最长回文子序列

给定一个字符串，找出其中的最长回文子序列的长度。编写一个程序，实现该问题，并给出示例的输入输出。

【示例】输入：字符串： "bbbab"

输出：最长回文子序列的长度：4 （最长回文子序列可以是 "bbbb" 或者 "baab"。）

【解题思路】

对于一个子序列而言，如果它是回文子序列，并且长度大于 2，那么将它首尾的两个字符去除之后，它仍然是个回文子序列。因此可以用动态规划的方法计算给定字符串的最长回文子序列。

用 dp[i][j] 表示字符串 s 的下标范围 [i,j][i, j][i,j] 内的最长回文子序列的长度。假设字符串 s 的长度为 n，则只有当 0≤i≤j<n 时，才会有 dp[i][j]>0，否则 dp[i][j]=0。

由于任何长度为 1 的子序列都是回文子序列，因此动态规划的边界情况是，对任意 0≤i<n，都有 dp[i][i]=1。

当 i<j时，计算 dp[i][j] 需要分别考虑 s[i] 和 s[j] 相等和不相等的情况：

如果 s[i]=s[j]，则首先得到 s 的下标范围 [i+1,j−1] 内的最长回文子序列，然后在该子序列的首尾分别添加 s[i] 和 s[j]，即可得到 s 的下标范围 [i,j] 内的最长回文子序列，因此 dp[i][j]=dp[i+1][j−1]+2；

如果 s[i]≠s[j]，则 s[i] 和 s[j] 不可能同时作为同一个回文子序列的首尾，因此 dp[i][j]=max(dp[i + 1][j], dp[i][j - 1])。

由于状态转移方程都是从长度较短的子序列向长度较长的子序列转移，因此需要注意动态规划的循环顺序。

最终得到 dp[0][n−1] 即为字符串 s 的最长回文子序列的长度。

【代码】

public class T516 {  
 public static void main(String[] args) {  
 Solution solution = new Solution();  
 String s1 = "bbbab", s2 = "cbbd";  
 int ans = solution.longestPalindromeSubseq(s1);  
 System.out.println(ans);  
 ans = solution.longestPalindromeSubseq(s2);  
 System.out.println(ans);  
 }  
  
 static class Solution {  
 public int longestPalindromeSubseq(String s) {  
 int n = s.length();  
 int[][] dp = new int[n][n];  
 for (int i = 0; i < n; i++) {  
 dp[i][i] = 1;  
 }  
 for (int i = n - 1; i >= 0; i--) {  
 char c1 = s.charAt(i);  
 for (int j = i + 1; j < n; j++) {  
 char c2 = s.charAt(j);  
 if (c1 == c2) {  
 dp[i][j] = dp[i + 1][j - 1] + 2;  
 } else {  
 dp[i][j] = Math.max(dp[i + 1][j], dp[i][j - 1]);  
 }  
 }  
 }  
 return dp[0][n - 1];  
 }  
 }  
}

【结果验证】1、输入：字符串： "bbbab"

1. 输入：字符串： "cbbd"

