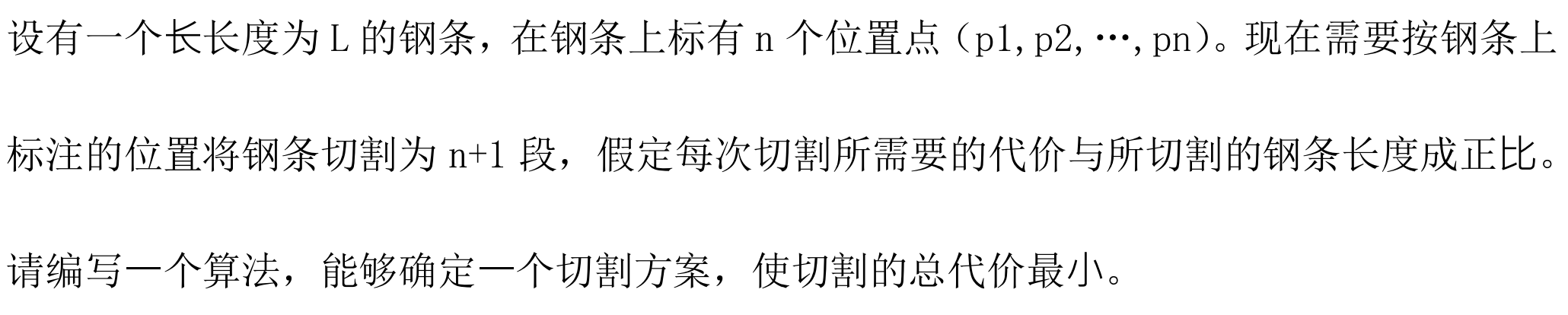
西安交通大学实验报告

课程名称：算法设计与分析 实验名称： 动态规划

学 院：\_\_\_\_\_\_\_电信学部\_\_\_\_\_\_\_\_\_\_\_\_ 实 验 日 期 2022年4 月 24 日

|  |  |  |
| --- | --- | --- |
| **刘沁宇** | **计算机002** | **2203613019** |

#### 问题描述



#### 问题分析

对于求解切割钢条的最小代价，可以看成合并已经切割好的钢条的最小代价，但合并钢条时需要确保每次只能合并相邻的两个钢条，且切割钢条具有最优子结构，可以利用动态规划解决。

#### 算法设计

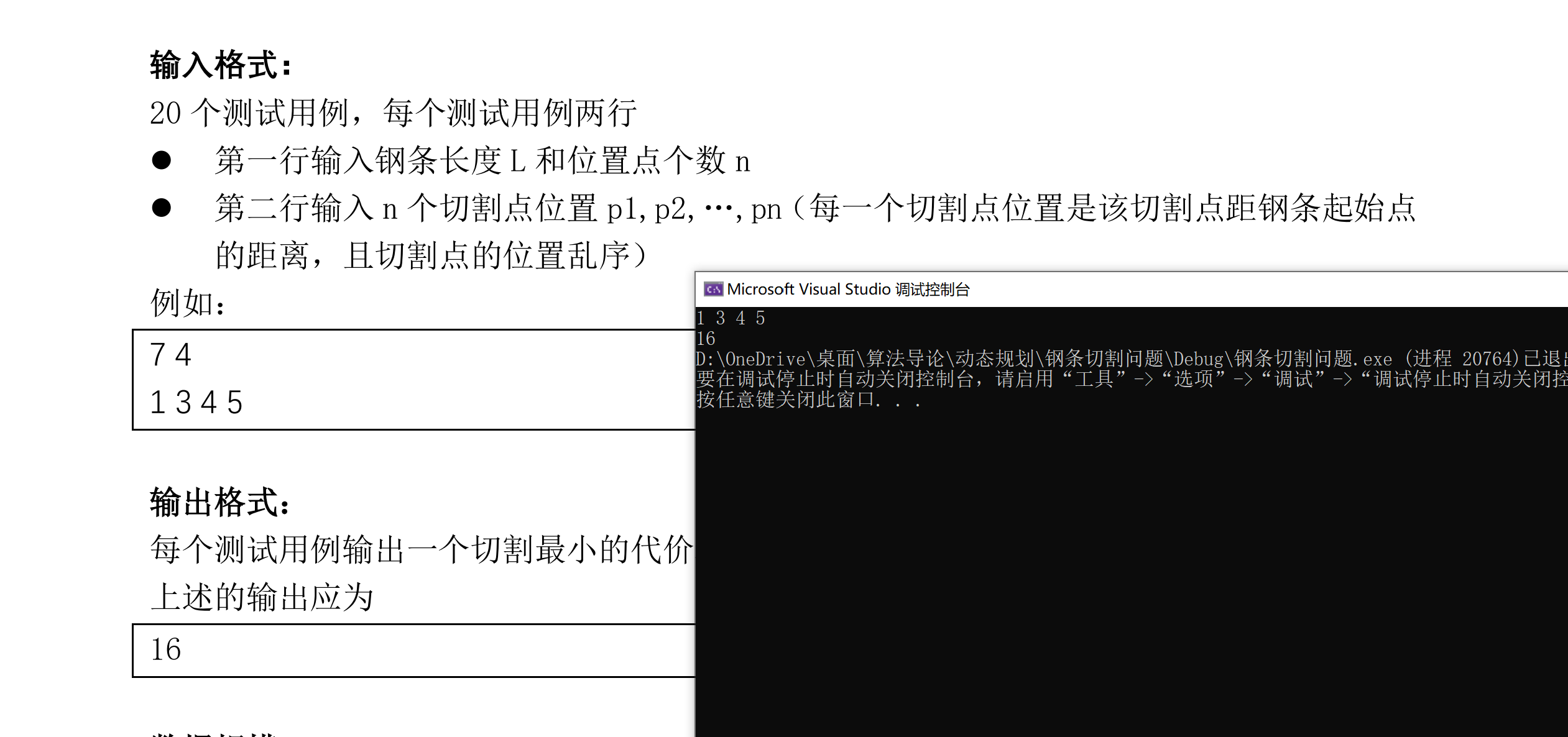
令表示合并第段到第段的钢条所需要的最小代价，则状态转移方程为：

且需要用三层循环实现，时间复杂度约为O()。

#### 算法实现

|  |
| --- |
| 1. #include <iostream> 2. #include <vector> 3. **using** **namespace** std; 5. **class** solution 6. { 7. **public**: 8. **void** minCost(**int** L, **int** n) 9. { 10. vector<**int**> length(n + 2);  //记录第0段钢条到到第i条钢条的长度和 11. vector<vector<**int**>> dp(n + 1); 12. **for** (**int** i = 0; i <= n; i++) 13. { 14. dp[i].resize(n + 1, INT\_MAX); 15. dp[i][i] = 0;  //合并任意单独的一段代价为0 16. } 17. length[0] = 0; 18. **for** (**int** i = 1; i <= n; i++) 19. { 20. cin >> length[i]; 21. } 22. length[n + 1] = L; 23. **for**(**int** i=n;i>=0;i--)    //区间dp,三层循环 24. { 25. **for** (**int** j = i+1; j <= n; j++) 26. { 27. **for** (**int** k = i; k < j; k++) 28. { 29. dp[i][j] = min(dp[i][j], dp[i][k] + dp[k + 1][j] + length[j + 1] - length[i]); //注意段落的序号和切割点序号的对应关系 30. } 31. } 32. } 33. cout << dp[0][n]; 34. } 35. }; 37. **int** main() 38. { 39. solution A; 40. A.minCost(7,4); 41. } |

#### 实验结果



**图表 1 输出结果与实验所给用例一致**