湖南大學

数据结构

课程实验报告

题　　目：折点计数 （CCF201909-1）

学生姓名　魏子铖

学生学号　201726010308

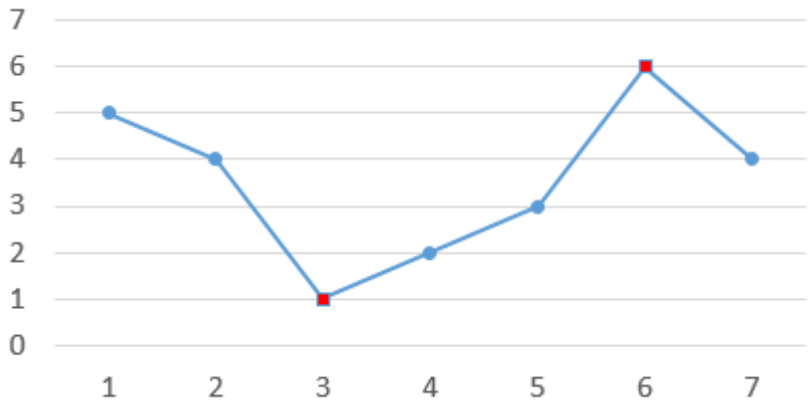
专业班级　软件 1703

完成日期　2018.10.3

一、 需求分析

0.问题描述

给定 n 个整数表示一个商店连续 n 天的销售量。如果某天之前销售量在增长，而后一天销售量减少，则称这一天为折点，反过来如果之前销售量减少而后一天销售量增长，也称这一天为折点。其他的天都不是折点。如下图中，第 3 天和第 6 天是折点。



给定 n 个整数 $a_1, a_2, \cdots, a_n$ 表示销售量，请计算出这些天总共有多少个折点。

为了减少歧义，我们给定的数据保证：在这 n 天中相邻两天的销售量总是不同的，即 $a_{i-1} \neq a_i$。注意，如果两天不相邻，销售量可能相同。

1.问题分析

①存储一组销售量数据

②统计根据这组数据绘制的折线图上拐点的数量

2.输入数据

输入的第一行包含一个整数 n。

第二行包含 n 个整数，用空格分隔，分别表示 $a_1, a_2, \cdots, a_n$。

所有评测用例满足：$1 \leqslant n \leqslant 1000$，每天的销售量是不超过 10000 的非负整数。

3.输出数据

输出一个整数，表示折点出现的数量。

4.测试样例设计

样例一：

9

2 6 9 8 5 1 6 3 2

输出一：

3

样例二：

12

35 69 81 401 215 59 7 199 800 1593 12 1

输出二：

3

说明：样例一与样例二为一般情况，用于检验算法的正确性

样例三：

8

40 40 40 40 40 40 40 40

输出三：

0

样例四：

8

2 3 4 5 6 7 8 9

输出四：

0

样例五：

8

9 8 7 6 5 4 3 2

输出五：

0

说明：样例三、样例四与样例五中不存在折点，用于检验边界条件情况下的正确性

二、概要设计

1.抽象数据类型

由于输入的数据元素均为非负整数，所以可以将输入存储在线性表中。

抽象数据类型设计：

数据对象：n 个互相独立的整数

数据关系：从键盘读取的 n 个整数，按照输入的先后顺序存储，其形式满足线性特征，即$< a_i, a_{i+1} > (0 \leq a < n)$

基本操作：开辟能够存储 n 个整型数据的内存空间

将输入的数据存储在线性表中

取出线性表中特定位置的数据

ADT：

```
template <typename E>
class Array_List : public List<E> {
private:
    int maxSize;//顺序表的最大长度
    int listSize;//顺序表的长度
    int curr;//当前位置
    E* listArray;//顺序表的物理结构

public:
    explicit Array_List(int size = defaultSize);//含默认参数的构造函数
    ~Array_List();//析构函数

    void append(const E&);//在顺序表后新增一个元素
    void moveToStart();//移动到顺序表的开头
    const E& getPrev() const;//获取当前位置的前驱的值
    const E& getNext() const;//获取当前位置的后继的值
    int length() const;//获取顺序表的长度
    const E& getValue() const;//获取当前位置的值
};
```

2.算法的基本思想

折点计数问题中的数据是一个个整数，第一天与第 n 天的销售量即为“第一元素、最后

元素”，并且第二天至第 n-1 天的数据是存在“唯一的前驱和后继的”，符合线性表的特点，由于输入是按照天数顺序输入的，所以可以采用顺序表来实现线性表，完成折点数量的统计。

核心算法主要分为两步：

1、 初始化一个计数器 count = 0；

2、 从第二天开始，一直到第 n-1 天，依次比较前后连续三天的销售量数据。

根据题目中折点的定义，第一天与第 n 天的销售量不存在出现折点的情况，所以我们从第二天开始查询。如果在查询的连续三天中，中间那天的销售量低于或高于前后两天的销售量，那么计数器的数值自增 1，直到查询到第 n-1 天，循环结束后输出计数器的值即为所求结果。

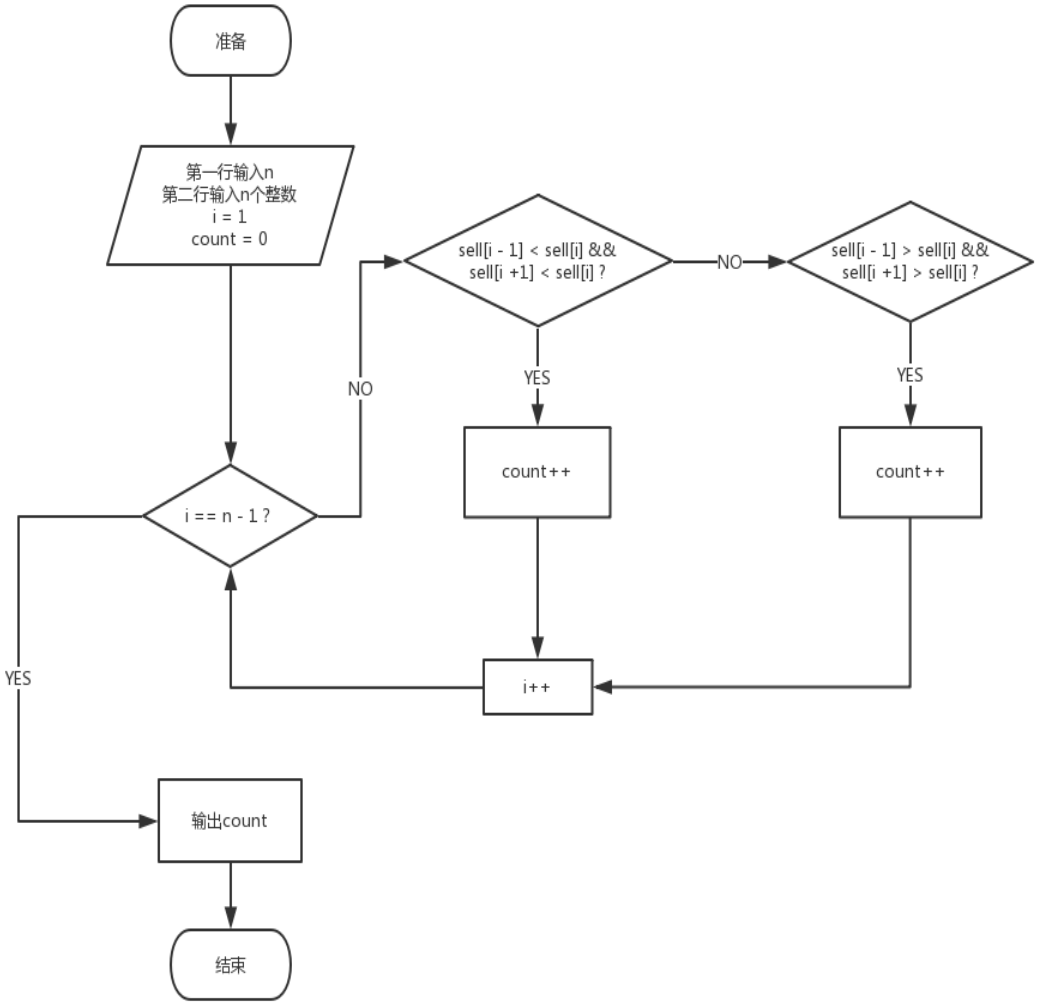
3.程序的流程

① 输入模块：无提示语句，第一行输入总数 n，第二行依次输入整数，中间用空格隔开。

② 处理模块：将输入元素存储在顺序表中，从顺序表第二个位置开始依次查询相邻连续三个位置的数据，并给句查询结果更新计数器的值。

③ 输出模块：在 DOS 下，根据输入按照要求输出折点的数量。

流程图如下：



三、详细设计

1.物理数据类型

由于输入数据是整型数据，且所有评测用例满足：1 ≤ n ≤ 1000，每天的销售量是不超过 10000 的非负整数，为了节省内存空间，数据类型用 short；由于输入按照日期先后顺序，满足线性特征，所以物理数据结构为线性结构。

2.输入和输出的格式

从键盘输入数据规模 n 和每天的销售量 sell[i],程序将处理这些数据并将结果通过 DOS 输出。输出结果为折点的数量。

3.算法的具体步骤

1、设置数组，通过 for 循环将数据存入

2、已知输入总数 n，依次根据题目中折点的定义，第一天与第 n 天的销售量不存在出现折点的情况，所以我们从顺序表的第二位开始查询。如果在查询的前后连续三个数据中，中间位置的数据同时小于或大于前后位置的数据，那么计数器的数值自增 1，直到查询到第 n-1 个数据，循环结束后输出计数器的值即为所求结果。

ADT 具体实现：

```
Array_List<int> sell(n);
for (int i = 0; i < n; i++) {
    int a = 0;
    cin >> a;
    sell.append(a);
}

int count = 0;
for (sell.moveToStart(), sell.next(); sell.currPosition() <
sell.length() - 1; sell.next()) {
    int day1 = sell.getPrev();
    int day2 = sell.getValue();
    int day3 = sell.getNext();
    if (day1 > day2 && day3 > day2)
        count++;
    if (day1 < day2 && day3 < day2)
        count++;
}

cout << count << endl;
```

4.算法的时空分析

1、初始化部分循环赋值，时间复杂度为Θ(n)，空间复杂度为Θ(n)；

2、处理部分循环 n-2 次，每次进行两次判断，时间复杂度为Θ(n)；

综上，该算法的时间复杂度与空间复杂度均为Θ(n)。

四、调试分析

1.调试方案设计

调试目的：

发现思维逻辑与代码实现上的区别，改进代码结构，排除语法、逻辑上的错误。

样例：

9

2 6 9 8 5 1 6 3 2

调试计划：

单步调试，在调试过程中注意 i 与 count 的值的改变，发现问题。

设置：

在循环开始处设置断点，单步调试。

2.调试过程和结果，及分析

一开始由于对折点定义的不充分理解，导致从顺序表的第一位开始查询，但第一位的数据没有前驱，于是发现了一个越界的问题。修改逻辑上的问题后，样例完美通过调试，提交到评测机上，该份代码得分 100 分。

五、测试结果

样例一

```
C:\Users\istil\Desktop\istillmessup\DataStructures\List\cmake-build-debug\List.exe
9
2 6 9 8 5 1 6 3 2
9
2 6 9 8 5 1 6 3 2
3

Process finished with exit code 0
```

折点数量为 3，结果正确

样例二

```
C:\Users\istil\Desktop\istillmessup\DataStructures\List\cmake-build-debug\List.exe
12
35 69 81 401 215 59 7 199 800 1593 12 1
12
35 69 81 401 215 59 7 199 800 1593 12 1
3

Process finished with exit code 0
```

折点数量为 3，结果正确

样例三

```
C:\Users\istil\Desktop\istillmessup\DataStructures\List\cmake-build-debug\List.exe
8
40 40 40 40 40 40 40 40
8
40 40 40 40 40 40 40 40
0

Process finished with exit code 0
```

折点数量为 0，结果正确

样例四

```
C:\Users\istil\Desktop\istillmessup\DataStructures\List\cmake-build-debug\List.exe
8
2 3 4 5 6 7 8 9
8
2 3 4 5 6 7 8 9
0

Process finished with exit code 0
```

折点数量为 0，结果正确

样例五

```
C:\Users\istil\Desktop\istillmessup\DataStructures\List\cmake-build-debug\List.exe
8
9 8 7 6 5 4 3 2
8
9 8 7 6 5 4 3 2
0

Process finished with exit code 0
```

折点数量为 0，结果正确

六、实验日志

这是第一次接触数据结构这个概念，对抽象数据类型的理解还不够深刻，尽管是入门级别的顺序表，在构建的过程中，还是会遗漏掉一些细节上的事情，比如检测当前位置是否是表的开头或结尾，或检测是否越界的问题。在顺序表的应用上，初步体会到了封装的意义：将数据项与相关的操作结合为一个整体，将数据结构的外部特性与内部实现相分离来提供一致且标准的对外接口，隐藏内部实现细节。数据结构的设计更像是面向对象思想的应用，与传统的面向过程来解题的思路有很大的差别，不过最后完成了，还是感到十分满足哇哈哈。