第一题最小差值

一、问题分析

【问题描述】

必须利用实验一实现的线性表 ADT，完成下面的题目。

对于给定 n 个数，请找出其中相差（差的绝对值）最小的两个数，输出它们的差值的绝对值。

【输入形式】

输入第一行包含一个整数 n。

第二行包含 n 个正整数，相邻整数之间使用一个空格分隔。

【输出形式】

输出一个整数，表示答案。

本题要处理的对象是 n 个正整数，要实现的功能是从给定的 n 个正整数中找出其中相差（差的绝对值）最小的两个数，并把它们的差值的绝对值输出。

1. 求解方法

从第一个正整数开始，依次计算它与后面的正整数的差值，然后计算第二个正整数与后面的正整数的差值，以此类推。如果计算所得差值小于当前最小差值，就对当前最小差值更新，直到计算到最后一个正整数与它前一个数的差值，结束操作，当前最小差值即为最终最小差值。

用（a，b）表示 a 和 b 差值的绝对值。

2. 样例推导

样例 1：

(1,5)=4(更新当前最小差值),(1,4)=3(更新当前最小差值),(1,8)=7,(1,20)=19

(5,4)=1(更新当前最小差值),(5,8)=3,(5,20)=15

(4,8)=4,(4,20)=16

(8,20)=12

操作结束,最终最小差值为 1。

样例 2：

(9,3)=6(更新当前最小差值),(9,6)=3(更新当前最小差值),(9,1)=8,(9,3)=6

(3,6)=3,(3,1)=2(更新当前最小差值),(3,3)=0(更新当前最小差值)

(6,1)=5,(6,3)=3

(1,3)=2

操作结束，最终最小差值为 0。

二、数据结构和算法设计

1. 抽象数据类型设计

要处理的数据对象为一组整型数，线性结构，故应设计线性表 ADT。

- 数据对象：线性表是由 n（n≥0）个数据元素（结点）a[0]，a[1]，a[2]…，a[n-1]组成的有限序列。
- 数据关系：这种结构具有下列特点：存在一个唯一的没有前驱的（头）数据元素；存在一个唯一的没有后继的（尾）数据元素；此外，每一个数据元素均有一个直接前驱和一个直接后继数据元素。
- 基本操作：

List() {}　　// Default constructor 构造函数 无输入和输出

virtual ~List() {} // Basedestructor 析构函数 无输入和输出

// Clear contents from thelist, to make it empty. 清空列表中的内容 无输入和输出

virtual void clear() = 0;

// Insert an element at the current location.

// item: The element to be inserted 在当前位置插入元素 item 输入 item，无输出

virtual void insert(const E& item) = 0;

// Append an element at the end of the list. 输入 item，无输出

// item: The element to be appended 在表尾添加元素 item

virtual void append(const E& item) = 0;

// Remove and return the current element.

// Return: the element that was removed. 删除当前元素，并将其作为返回值 无输入，输出 item

virtual E remove() = 0;

// Set the current position to the start of the list. 将当前位置设置为顺序表起始处 无输入和输出

virtual void moveToStart() = 0;

// Set the current position to the end of the list. 将当前位置设置为顺序表末尾 无输入和输出

virtual void moveToEnd() = 0;

// Move the current position one step left. No change

// if already at beginning. 将当前位置左移一步，如果当前位置在首位就不变 无输入和输出

virtual void prev() = 0;

```
// Move the current position one step right. No change

// if already at end. 将当前位置右移一步，如果当前位置在末尾就不变 无输入和输出

  virtual void next() = 0;

// Return: The number of elements in the list. 返回列表当前的元素个数 无输入，输出列表当前的元素个数

  virtual int length() const = 0;

// Return: The position of the current element. 返回当前元素的位置 无输入，输出当前元素的位置

  virtual int currPos() const = 0;

// Set current position.

// pos: The position to make current. 将当前位置设置为 pos 输入 pos 作为当前位置，无输出

  virtual void moveToPos(int pos) = 0;

// Return: The current element. 返回当前元素 无输入，输出当前元素

  virtual const E& getValue()  = 0;
```

2. 物理数据对象设计

因为本题中数据输入是追加在表尾，顺序表和链表效率差不多，但链表空间开销较大，故采用顺序表的物理实现方式。

3. 算法思想的设计

从第一个正整数开始，依次计算它与后面的正整数的差值，然后计算第二个正整数与后面的正整数的差值，以此类推。如果计算所得差值小于当前最小差值，就对当前最小差值更新，直到计算到最后一个正整数与它前一个数的差值，结束操作，当前最小差值即为最终最小差值。

4. 关键功能的算法步骤

1) 数据输入部分

```
for(int i=0;i<n;i++) //将数据填入线性表中
{
    cin>>m; //输入数据
    A.append(m); //将数据存到线性表末尾
}
```

2) 数据处理部分

```
for(int i=0;i<A.length();i++) //从第一个正整数开始
{
    for(int j=i+1;j<A.length();j++) //遍历第 i 个数后的每一个数
    {
        int temp=abs(A[i]-A[j]); //计算两个数差的绝对值
        if(temp<min) min=temp; //如果小于当前最小差值，就更新
    }
}
```

三、算法性能分析

1. 数据输入部分

for 循环共执行 n 次，所以时间复杂度为 O(n),空间复杂度为 O(n)。

2. 数据处理部分

两层 for 循环嵌套，时间复杂度为 $O(n^2)$，空间复杂度为 O(1)。