# 实验 2 线性表的应用

————利用单链表对两个多项式的求和

一、问题分析

本实验要对已经封装的单链表进行操作，以求两个多项式的和。并以题目要求的形式进行输出。对于输入样例：

3
3 5
-2 1
4 0
4
2 3
-1 2
1 1
3 0

会输出：

3 5
2 3
-1 2
-1 1
7 0

这是由于多项式 3x^5-2x+4 与多项式 2x^3-x^2+x+3 相加，计算结果：3x^5+2x^3-x^2-x+7。对于本题而言，我们定义了 6 个单链表，a1、b1 单链表分别存放第一个多项式每一项的系数和阶数，a2、b2 单链表分别存放第二个多项式每一项的系数和阶数，a3、b3 单链表分别存放输出的多项式的每一项的系数和阶数。通过判断找出最大的阶数，由此构造出两个具有最大阶数个节点的两个链表 a3、b3，使得 b3 的第 i 个结点存放 maxs-i+1，a3 初始化为 0。然后更新链表 a3，使得 a3 的每一个节点的值与具有相同阶数的链表 a1 相同。然后再次更新 a3，使得 a3 的每一个节点的值等于具有相同阶数的链表 a1 和 a2 值的和。最后遍历输出，当系数不为 0 时即输出系数和对应的阶数。

对于输入样例：

2
3 5
3 1
0

会输出：

3 5
3 1

这是由于多项式 3x^5+3x 与多项式 0 相加，计算结果：3x^5+3x。对于本题而言，我们同样定义了 6 个单链表，a1、b1 单链表分别存放第一个多项式每一项的系数和阶数，a2、b2 单链表分别存放第二个多项式每一项的系数和阶数，a3、b3 单链表分别存放输出的多项式的每一项的系数和阶数。通过判断找出最大的阶数，由此构造出两个具有最大阶数个节点的两个链表 a3、b3，使得 b3 的第 i 个结点存放 maxs-i+1，a3 初始化为 0。然后更新

链表 a3，使得 a3 的每一个节点的值与具有相同阶数的链表 a1 相同。然后再次更新 a3，使得 a3 的每一个节点的值等于具有相同阶数的链表 a1 和 a2 值的和。最后遍历输出，当系数不为 0 时即输出系数和对应的阶数。

二、数据结构和算法设计

1. 该数据结构（链表）的 ADT 有如下几种：

```
virtual void clear() = 0;

// Insert an element at the current location.
// item: The element to be inserted
virtual void insert(const E& item) = 0;

// Append an element at the end of the list.
// item: The element to be appended.
virtual void append(const E& item) = 0;

// Remove and return the current element.
// Return: the element that was removed.
virtual E remove() = 0;

// Set the current position to the start of the list
virtual void moveToStart() = 0;

// Set the current position to the end of the list
virtual void moveToEnd() = 0;

// Move the current position one step left. No change
// if already at beginning.
virtual void prev() = 0;

// Move the current position one step right. No change
// if already at end.
virtual void next() = 0;

// Return: The number of elements in the list.
virtual int length() const = 0;

// Return: The position of the current element.
virtual int currPos() const = 0;

// Set current position.
// pos: The position to make current.
virtual void moveToPos(int pos) = 0;
```

```
// Return: The current element.
virtual const E& getValue() const = 0;
```

2. 物理对象设计：

本实验定义了 6 个 int 类型的单链表，a1、b1 单链表分别存放第一个多项式每一项的系数和阶数，a2、b2 单链表分别存放第二个多项式每一项的系数和阶数，a3、b3 单链表分别存放输出的多项式的每一项的系数和阶数。通过对这六个单链表的操作实现对两个多项式求和。

3. 算法思想

通过判断找出最大的阶数，由此构造出两个具有最大阶数个节点的两个链表 a3、b3，使得 b3 的第 i 个结点存放 maxs-i+1，a3 初始化为 0。然后更新链表 a3，使得 a3 的每一个节点的值与具有相同阶数的链表 a1 相同。然后再次更新 a3，使得 a3 的每一个节点的值等于具有相同阶数的链表 a1 和 a2 值的和。最后遍历输出，当系数不为 0 时即输出系数和对应的阶数。

4. 关键功能的算法步骤

关键步骤主要在于对 a3，b3 链表的处理，即加法过程。

```
for(int i=0;i<n;i++)
{
    创建新节点使其值等于第一个多项式对应阶数的节点值；
    删除旧节点；
    插入新节点至原位置；
    a1.next();
    b1.next();
}

for(int i=0;i<m;i++)
{
    创建新节点使其值等于第一个多项式对应阶数的节点值和第二个多项式对应阶数的节点值之和；
    删除旧节点；
    插入新节点至原位置；
    a2.next();
    b2.next();
}
```

三、算法性能分析

该算法进行了 4 次单重 for 循环，2 次 for 循环中嵌套了一个平均需要执行 n/2 次的 for 循环，故可以认为该算法是$\theta(n^2)$