桂林电子科技大学

实验 1 线性表的基本操作、算法和应用 实验报告

实验名称	线性表的基本操作、算法和应用						
院　　系	计算机与信息安全学院		专业	信息安全			
学　　号	2000301708		姓名	蔡响			
实验日期	2021	年	10	月	22	日	

辅导员意见：

成绩　　辅导员签　名

实验目的

1. 链表的基本操作
2. 链表的算法设计
3. 链表的应用

实验内容

1. 链表插入算法
2. 链表的删除算法
3. 移动链表中的最大值到尾部
4. 合并两个递增有序的单循环链表
5. 链表中奇偶结点的移动
6. 多项式的加法

实验环境

在PTA平台进行实验

实验要求

根据每个实训的要求完成代码提交和测评

实验步骤

链表的含义

将位于不同地址的元素通过指针链接起来，在读取链表中的数据是连续的，所以叫做链表。

链表的插入算法

读题：

要求在链表llist中的元素x之后插入一个元素y的操作。就是把链表断了，填上，再连接。

可能需要考虑以下两种特殊情况：

1. 链表llist中是否存在对应的元素x
2. 元素x是链表llist中的最后一个元素

思路：

遍历至数据域为元素x的节点。

对于普通情况的处理，就是创立一个新节点，把新节点连上后续节点，最后让x->next指向新节点。这个操作相比于先用a变量记住x->next，再让x->next变为新变量p，最后把p->next指向a的操作要优秀，就省去了新建变量。

如果插入成功，直接return 结束函数。

如果链表llist中不存在对应的元素x，则打印报错信息。

实现：

在链表的插入算法中主要需要考虑三种情况：1. 头部插入 2. 中间插入 3. 尾部插入

但是头部与中间插入在本题中相同，直接采让新建的结点需要先连接后面再连接前面的，进行实现。对于尾部插入，则只需直接连接前面的节点，等同于链表的正常定义生成。

链表的删除算法

读题：

删除链表llist中的指定元素deldata。思路简单，即断掉链表：把deldata的前一个元素与与deldata的下一个元素相接。

可能需要考虑这些特殊情况：

1. deldata不存在时
2. 需要删除首元素
3. 删除中间元素
4. 删除尾部元素

思路：

遍历至数据域为元素deldata的节点。并用before记为deldata的前节点，after为deldata的后节点，为如下关系：

$$before \longrightarrow deldata \longrightarrow after$$

既然需要链表连接的操作，那么采用双指针的方式进行最为方便。找到对应的deldata之后，让before->next=after即可。

实现：

对于deldata位于首，中，尾做不同的删除处理。只要进行了删除操作就return结束，如果没有进行删除的操作，就打印错误信息。

移动链表中的最大值到尾部

读题：

只把最大值Max移到尾部，其他结点的相对次序保持不变。涉及到删除节点以及链表的尾部插入。

思路：

删除与尾部插入，前面已经实现。这题的主要难点是将两个部分的功能进行融合，以及找到链表中的最大值。只有将链表遍历完成，才能找到最大值Max，而节点的删除需要最大值Max的前节点及后节点。

- 法一：

两次遍历。一次遍历找出最大值Max，第二次的遍历过程中将原来的最大值结点删除然后移动到尾部

- 法二：

只采用一次遍历。找出最大值Max，并通过变量before以及after记住Max的前节点以及后节点，在变量完成后进行删除与移动。

$before \longrightarrow Max \longrightarrow after$

实现：

明显法二的时间复杂度较低为$O(n)$，采用法二。将删除节点以及链表的尾部插入两部分的代码进行融合，并采用“打擂台”的思想进行记录Max。

合并两个递增有序的单循环链表

读题：

要求将有序单循环链表tail1和tail2的合并，要求合并时实现去重操作，并且合并后的链表为递增有序链表。感觉难度突然直线上升。合并涉及到链表插入，去重涉及到链表删除。但是，总体牵扯到的元素较多。

思路：

- 法一：

由于两个链表都是递增有序，所以只需要新建一个空链表tail3，然后然后变量对比两个链表中的元素，让小的节点先插入新链表tail3。如果已存在相同元素则跳过。

- 法二：

雷同于法一的做法，将链表tail1作为法一中tail3，将tail2中节点在tail1中合适的地方插入。

- 法三：

让tail1的尾节点连接tail2的头结点，先拼合两个链表，然后进行排序，排序后去重。

实现：

三个方法，法一与法二本质差不多，法一比较暴力，法二的空间复杂度比较低，法三的时间复杂度高，且实现复杂。故采用法二。

第一次遇到循环链表，循环链表还是比较神奇的：让尾指针指向head->next。

循环链表变单链表：

由于本题存在递增有序，所以只要下一个元素变小，就跳出循环。为了减低时间复杂度，我们将其直接添加在合并的代码，作为跳出循环的判断。

个人感觉这个方法十分巧妙

```
LinkList head = tail1->next->next;
tail1->next->next = tail2->next->next;
tail2->next->next = NULL;
```

合并：

通过for循环嵌套的方式进行遍历两个链表的元素，在遍历过程中将通过比较将链表tail2中的数值依次插入链表tail1中。为实现去重效果，在插入节点的时候进行判断，如果插入的节点与上一个节点相同，则continue。

这里简化了程序，将排序与剔除合在一起，降低时间复杂度：

```
for (pre = head; pre->next != NULL; pre = pre->next)
{
    for (q = pre->next; q != NULL; q = q->next)
    {
        if (pre->data > q->data)
        {
            flage = pre->data;
            pre->data = q->data;
            q->data = flage;
        }
        else if (pre->data == q->data)
        {
            if (q->next == NULL)
            {
                pre->next = NULL;
                free(q);
            }
            else
            {
                pre->next = q->next;
                free(q);
                q = pre;
            }
        }
    }
}
```

将排序后的链表形成循环链表并return头结点：

```
tail2->next->next = head;
//tail2->next指向的地址不变，所以依然是最大
return tail2;
```

链表中奇偶结点的移动

读题：

移动单循环链表中奇数和偶数结点，奇前偶后，且结点之间的相对顺序不变。与排序有关，需判断奇偶。

思路：

- 法一：

涉及到奇偶两个类型的数据，所以考虑两条链，一条为奇数链，另一条为偶数链。顺序遍历链表，将其中节点插入到对应的奇偶链表中，最后将两个链表拼接，生成最终的链表。

- 法二：

创建一个空链表，遍历原链表两次，分别在新链表中添加奇节点与偶节点。

实现：

就时间复杂度，选择法一。

选择直接在原链表上操作，定义两个头指针odd与even，遍历链表，让它们分别指向链表中的第一个奇数与第一个偶数，继续遍历链表，添加节点。最终让odd的链尾指向even即可。

多项式的加法（编程题）

读题：

用链表表示多项式，并实现多项式的加法运算。逗号前是系数，逗号后是指数。相加后系数为0则无需输出，存在指数无序与指数有序的情况。

思路：

创建两个链表分别存储两个多项式，直接在原有的某个链表上进行数据域的加减，以及未有数据节点的添加，添加完成后可以采用冒泡的方式进行排序，排序后打印节点，并free()。

这题的重难点是如何在降低算法的时间复杂度的同时简化算法。

我认为，应该先将两个多项式进行排序，这样有利于后续的指数相加计算，实现起来也比较简单。时间复杂度为$O(n^2+m^2+n+m)$

其中$O(n^2+m^2)$冒泡排序，n与m为对应的链表长度，$O(n+m)$为加法运算时消耗的时间。

还存在另一种方法，先计算再排序。时间复杂度为$O(n*m+(n+m)^2)$。

其中$O(n*m)$为加法运算，$O((n+m)^2)$为排序。

通过简单的数学运算，我们可以发现$O(n^2+m^2+n+m)<O(n*m+(n+m)^2)$故选择前面的方法。

实现：

踩雷：不知道如何将输入的指数与下标通过逗号隔开，打算输入字符串后，分割字符串进行处理。后面才知道可以通过在scanf中加入逗号隔开：

```
scanf("%d,%d",&coefficient,&index);
```

冒泡排序：

采用最简单的冒泡排序。此处不同于数组，结束的标准被设置为p->next!=NULL，或者可以储存一下链表的长度，通过长度作为结束位的判断依据。

合并：

链表已经过排序，合并的过程即可采用双指针的方式进行，无需采用双层for循环遍历。

1. 指数相同，则相加
2. 指数不同，则直接插入节点

代码的实现就类似于上面的”合并两个递增有序的单循环链表“，这里就不进行额外的赘述。

问题记录和实验总结

本次实验主要让我了解了线性表的基本操作、算法，掌握了线性表应用。然我重新熟悉了许久没碰的C语言，更熟练的掌握结构体等链表知识。在完成本次实验的过程中，也悟出了一些相关的解题思路：

1. 在节点中添加其他的信息。定义的结构体不仅仅只存储data与next，还可以存放一些其他的信息，比如这个链表中的最大值，链表长度等等，可以极大的方便后续解题操作。
2. 双指针的妙用。可以说链表的各种增删改查等许多基本操作都可以采用双指针进行，双指针的运用极大降低相关代码的时间复杂度，简化运行，提高效率。
3. 基础不牢地动山摇。在做多项式的加法时，不知道在scanf 的时候可以通过逗号隔开输入，导致在做这题时费力不讨好，浪费了大把的时间。