第 2 章 线性

朱世轩 计 2 1752528

注：所有未特别说明的链表，均为带头结点

1、 设线性表有 n 个元素，以下操作中，_A____在顺序表上实现比在链表上实现效率更高

A 输出第 i 个元素值（i 在 1-n 之间）

B 交换第 1 个元素与第 2 个元素的值

C 顺序输出这 n 个元素的值

D 输出与给定值 x 相等的元素在线性表中的序号

2、 设线性表中有 2n 个元素，以下操作中，__B___在单链表上实现要比在顺序表上实现效率更高

A 删除指定的元素

B 在最后一个元素的后面插入一个新元素

C 顺序输出前 k 个元素

D 交换第 i 个元素和第 2n-i-1 个元素的值(i 在 0 - n-1 间)

3、 如果最常用的操作是取第 i 个结点及其前驱，则采用_D__存储方式最节省时间

A 单链表

B 双链表

C 单循环链表

D 顺序表

4、 将两个各有 n 个元素的有序顺序表(某个表中的元素，两个表之间的元素，值均有可能相同)归并成一个有序顺序表，其最少比较次数是__A___

A n

B 2n-1

C 2n

D n-1

5、 一个长度为 n(n>1)的带头结点单链表 h 上,另设有尾指针 r(指向尾结点),执行__B__的操作与链表的长度有关

A 删除单链表中的第一个元素

B 删除单链表的最后一个元素

C 在单链表的第一个元素前插入一个新元素

D 在单链表的最后一个元素后插入一个新元素

6、 双向循环链表中,在 p 结点之前插入 q 结点的操作是_D___

A
```
p->prior=q;
q->next=p;
p->prior->next=q;
q->prior=p->prior;
```

B
```
p->prior=q;
p->prior->next=q;
q->next=p;
q->prior=p->prior;
```

C
```
q->next=p;
q->prior=p->prior;
p->prior=q;
p->prior->next=q;
```

D
```
q->next=p;
```

```
q->prior=p->prior;
p->prior->next=q;
p->prior=q;
```

7、 在一个单链表中删除 p 结点(假设 p 不是尾结点)时，应执行如下操作:

(1) q=p->next;

(2) p->data=p->next->data;

(3) p->next=____q->next_____;

(4) free(q);

8、 在一个单链表中的 p 结点之前插入一个 s 结点，可执行如下操作：

(1) s->next=___p->next_____

(2) p->next=s;

(3) t=p->data;

(4) p->data=____s->data_____

(5) s->data=____t_____

9、 在一个双向循环链表中删除 p 结点时，应执行如下操作:

(1) _p->next->prior_____ = p->prior;

(2) p->prior->next = __p->next_____;

(3) free(p);

10、 在单链表、双向链表和单循环链表中，若仅知道指针 p 指向某结点，不知道头指针，能否将 p 从相应的链表中删除(不允许进行结点之间数据域的复制)？若可以，时间复杂度各为多少？

单链表不可以

双向链表和单循环链表可以

双向链表只需要直接取 p 的前驱和后继，时间复杂度为 O(1);

单循环链表需要循环查找链表一次来寻找 p 的前驱，时间复杂度为 O(n)

11、 设计一个高效算法，将顺序表的所有元素逆置，要求算法的空间复杂度为 O(1)

```
void UpsidedownList(List &L)
{
    Length=ListLength(L);  //顺序表长度
    for(i=0;i<Length/2;i++)
    {
        t=L[i];               //交换第 i 个和第 length-i-1 个元素的值
        L[i]=L[Length-i-1];
        L[Length-i-1]=t;
    }
}
```

12、 设计一个高效算法，从顺序表中删除所有元素值为 x 的元素，要求空间复杂度为 O(1)

```
void UpsidedownList(List &L,ElemType x,int & Length)
{
    Length=ListLength(L);  //顺序表长度
    for(i=0;i<Length;i++)
```

```
        {
            if(L[i]==x)
            {
                for(j=i;j < Length-1;i++)
                {
                    L[j] = L[j+1];
                }//end of for
                Length--
            }//if
        }//end of for
    }
```

13、 用顺序表表示集合，设计一个求集合交集的算法

```
 void Inter(List &La, List Lb)
{
    La_len = Listlength(La);
    Lb_len = Listlength(Lb);
    for (i = 1; i <= La; i++)
    {
        GetElem(La, i, e);                  //取La中第i个元素
        if (!LocateElem(Lb, e, equal))      //Lb中不存在与e相同的元素，则删除La中对应元素
        {
            ListDelete(La, i);
            La_len--;                       //la长度减1
        }
    }
}//Inter
```

14、 从带头结点的循环单链表中删除值为 x 的第一个结点

```
void ListDelete(List &L,ElemType e)
{
    List p = L;
    while (p->next != L);
    {
        if (compare(p->next->data, e)==TRUE)
        {
            q = p->next->next;
            p = q;
            free(p->next);
            break;
        }
        else
        {
            p = p->next;
        }
    }
    if (p->next == L)
        return FAIL;
    else
        return OK;
```

```
}//Inter
```

15、 假定有一个带头结点的链接表，头指针为 HL，每个结点含三个域:data，next 和 range，其中 data 为值域，next 和 range 均为指针域，现在所有结点已经由 next 域链接起来，试编一算法，利用 range 域(此域的初始值均为 NULL)把所有结点按照其值从小到大的顺序链接起来（next 域不变）

```
void Link(LinkList  *L)
{
    for (p = *L; p; p = p->next)
    {
        for (r = *L; r->range; r = r->rang)
        {
            if (p->data < r->range->data)
            {
                p->range = r->range->range;
                r->range = p;
            }
        }
        if (!r->range)
            r->range;
    }
}
```

16、 已知带头结点的单链表 L 是一个递增有序表，设计一个高效算法，删除表中 data 值在[min .. max]之间的所有结点，并分析算法的时间复杂度

```
void DelMintoMax(LinkList *&L, ElemType min, ElemType max)
{
    LinkList *p = L->next, *q, *r = L;
    while (p != NULL && p->data < min)
    { //p指向第一个大于等于min的结点
        r = p;                      //r为*p的前驱结点
        p = p->next;
    }
    q = p;
    while (q != NULL && q->data <= max) //q指向第一个大于max的结点
        q = q->next;
    printf("p:%d q:%d\n", p->data, q->data); //删除*p到*q之间的所有结点
    r->next = q;
    r = p->next;
    while (r != q) { //释放被删除结点的空间
        free(p);
        p = r; r = r->next;
    }
}
```

算法中三个循环，前两个循环至多将链表扫描一遍，第三个循环至多将链表扫描一遍，删除结点只用了一次操作，总执行次数与链表长度成正比，所以时间复杂度是 O(n)

17、 有一个值按非递减有序排列的单链表，设计一个算法删除值域重复的结点，并分析算法的时间复杂度

```
void DelSame(LinkList *&L)
{
    LinkList *p = L->next, *q;
    while (p->next != NULL)
    {
```

```
        if (p->data == p->next->data) //找到重复值的结点
        {
            q = p->next;        //q指向这个重复值的结点
            p->next = q->next; //删除*q结点
            free(q);
        }
        else p = p->next;
    }
}
```

算法中循环将链表扫描一遍，执行次数与链表长度成正比，所以时间复杂度为O(n)

18、 用单链表表示集合，设计一个算法表示集合的交

```
void Inter(LinkList &La, LinkList Lb)
{
    pa=La->next;
    while(pa)
    {
        if (!LocateElem(Lb, pa ->data, equal))    //Lb中不存在与e相同的元素，则删除La中对应结点
        {
            qa = pa->next;        //q指向第i个结点
            pa->next = qa->next; //第i-1个结点的next域指向第i+1个
            free(qa);
        }
        pa=pa->next;
    }
}//Inter
```

19、 写出将带头结点的双向循环链表倒置的算法

```
Status reverse(ListLink_Dul  &L)
{
    p = L;
    q = p->next;
    LinkList_DUL s;
    while (q! =L)
    {
        r = q->next;
        q->next = p;
        p->prior = q;
        p = q;
        q = r;
    }
    head->next = p;
    p->prior = head;
    return OK;
}
```

20、 设有一个双向链表 h,设计一个算法查找第一个元素值为 x 的结点，将其与后继结点进行交换

```
Status linklistsearch (LinkList &L)
```

```
{
    p = L->next;
    while(p)
    {
        if(compare(p->data ,x)==TRUE)
        {
            q = p->next;                //抓住下一个结点

            q->prior = p->prior;            //实现与后继结点对调
            q->next->prior = p;
            p->next = q->next;
            p->prior->next = q;

            q->next = p;
            p->prior = q;
            break;                      //对调完第一个就跳出循环
        }
        p = p->next;
    }
    return OK;
}
```

【作业要求：】

1、5 月 15 日前网上提交本次作业（直接在本文件中作答，转换为 PDF 后提交即可）

2、每题所占平时成绩的具体分值见网页

3、超过截止时间提交作业会自动扣除相应的分数，具体见网页上的说明

4、**答案用蓝色标注(选择题将正确选项直接设置为蓝色文字即可)**