数据结构作业 9.20

计科 陈昕琪 PB22111711

1. (3) B

在表头插入，移动 127 个.

在表尾插入，移动 0 个.

平均移动 $\frac{127+126+\cdots+0}{128}=63.5$

(8) A

当两个有序表都按顺序表存储时，

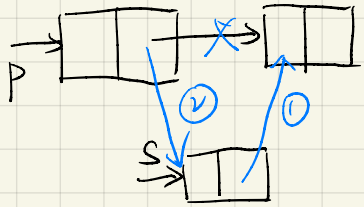
将两个表合并后仍是有序表时，

此时比较次数最少，即只需

表中的前1个元素与另一个表的元素比较，

则次数为 n

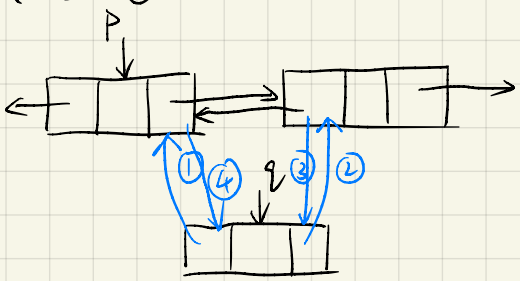
(13) D



① s->next = p->next

② p->next = s

(15) C



① q->prior = q

② q->next = p->next

③ p->next->prior = q

④ p->next = q

算法设计

（2）将两个非递减链表合并为一个非递增链表

```
#include<stdio.h>
#include<stdlib.h>
typedef struct link{        //定义链表
        int data;
        struct link *next;
}link;
link * initLink(int n){         //初始化链表，获取数据
        link *p,*q;
        printf("Input:");       //输入数据
        for(int i=0;i<n;i++){
                link *temp=(link*)malloc(sizeof(link));         //动态分配内存
                scanf("%d",&temp->data);        //填充数据
                temp->next=NULL;      //最后一个节点指针NULL
                if(i==0){
                        q=temp;         //给q,p赋值第一个结点
                        p=temp;
                }
                else{           //形成一条p链
                        p->next=temp;   //p陆续指向后边的结点
                        p=p->next;      //p移动到最后
                }
        }
        return q;   //返回头指针
}

link * mergeList(link *a,link *b){      //头插法合并两条传入的链表
        link *o,*p,*q;      //这里p中转，o指向最后一个元素，q指向最新元素
        int flag=1;                 //第一次合并使用
        while (a||b){
                if(!a){         //如果a空了，直接指向b的结点
                        //printf("get %d from b\n",b->data);
                        p=b;
                        b=b->next;
                }
                else if(!b||a->data<=b->data){//如果b空了或者a的数据<=b的数据时，p指
向a的结点，获取a指向的数据
                        //printf("get %d from a\n",a->data);
                        p=a;
                        a=a->next;
                }
                else {
                        //printf("get %d from b\n",b->data);//否则从b获取数据
                        p=b;
                        b=b->next;
                }
                if(flag){
```

```
                        o=q=p;          //初始化指针
                        flag=0;           //使用一次舍弃
                }
                else {
                        p->next=q;        //使取下的结点连上合并后的链
                        q=p;              //更新q的位置到最新结点
                }
        }
        o->next=NULL;   //使最后一个指向NULL输出结束条件
        free(b);        //a，b，q的空间可以释放
        free(a);
        return q;
}
int main(){
        int m,n;                //a，b的长度
        printf("The length of the first list:");
        scanf("%d",&m);
        link *a = initLink(m);  //初始化a
        printf("The length of the second list:");
        scanf("%d",&n);
        link *b = initLink(n);  //初始化b

        link *c = mergeList(a,b);//合并a，b
        printf("Result:");
        while(c){
                printf("%d ",c->data);
                c=c->next;
        }
        return 0;
}
```

(7)将所有结点链接方向原地逆转

```
#include<stdio.h>
#include<stdlib.h>
typedef struct link{        //定义链表
        int data;
        struct link *next;
}link;
link * initLink(int n){         //初始化链表，获取数据
        link *p,*q;
        printf("Input:");       //输入数据
        for(int i=0;i<n;i++){
                link *temp=(link*)malloc(sizeof(link));         //动态分配内存
                scanf("%d",&temp->data);        //填充数据
                temp->next=NULL;      //最后一个节点指针NULL
                if(i==0){
                        q=temp;         //给q,p赋值第一个结点
                        p=temp;
                }
```

```
                else{            //形成一条p链
                        p->next=temp;   //p陆续指向后边的结点
                        p=p->next;      //p移动到最后
                }
        }
        return q;   //返回头指针
}
link * turnlist(link *L){
    link *p,*q,*r;
    p=L;//p,q,r分别为三个不断移动的指针
    q=p->next;
    r=q->next;

    q->next=p;//进行第一个结点转换时要把第一个结点的尾指针变为空。
    L->next=NULL;

    while(r)
    {
       q->next=p;
       p=q;
       q=r;
       r=r->next;//改变结点后继续向下移动
    }//当r为空时，q指向最后一个元素
    q->next=p;
    return q;
}
int main(){
        int n;                //链表的长度
        printf("The length of the list:");
        scanf("%d",&n);
        link *L = initLink(n);  //初始化L
        L = turnlist(L);
        printf("Result:");
        while(L){//输出数据
                printf("%d ",L->data);
                L=L->next;
        }
        return 0;
}
```