

本节内容

单链表

查找

知识总览



单链表的查找

按位查找

按值查找

注：本节只探讨“带头结点”的情况

GetElem(L,i): 按位查找操作。获取表L中第i个位置的元素的值。

LocateElem(L,e): 按值查找操作。在表L中查找具有给定关键字值的元素。

按位查找

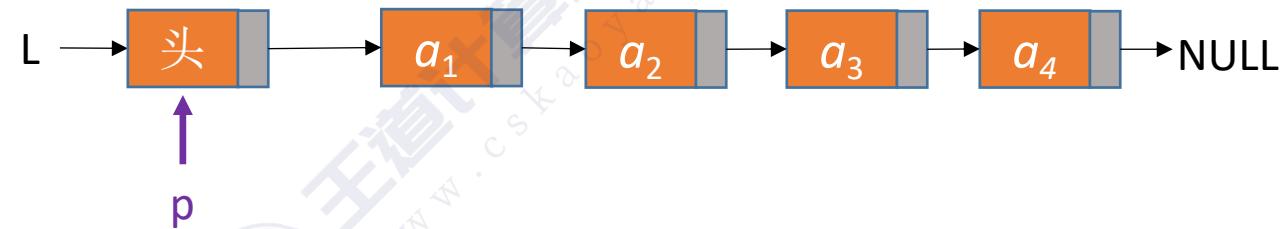
```
//在第 i 个位置插入元素 e (带头结点)
bool ListInsert(LinkList &L, int i, ElemType e){
    if(i<1)
        return false;
    LNode *p; //指针p指向当前扫描到的结点
    int j=0; //当前p指向的是第几个结点
    p = L; //L指向头结点, 头结点是第0个结点 (不存数据)
    while (p!=NULL && j<i-1) { //循环找到第 i-1 个结点
        p=p->next;
        j++;
    }
    if(p==NULL) //i值非法
        return false;
    LNode *s = (LNode *)malloc(sizeof(LNode));
    s->data = e;
    s->next=p->next;
    p->next=s; //将结点s连到p之后
    return true; //插入成功
}
```

```
bool ListDelete(LinkList &L, int i,ElemType &e){
    if(i<1)
        return false;
    LNode *p; //指针p指向当前扫描到的结点
    int j=0; //当前p指向的是第几个结点
    p = L; //L指向头结点, 头结点是第0个结点 (不存数据)
    while (p!=NULL && j<i-1) { //循环找到第 i-1 个结点
        p=p->next;
        j++;
    }
    if(p==NULL) //i值非法
        return false;
    if(p->next == NULL) //第i-1个结点之后已无其他结点
        return false;
    LNode *q=p->next; //令q指向被删除结点
    e = q->data; //用e返回元素的值
    p->next=q->next; //将*q结点从链中“断开”
    free(q); //释放结点的存储空间
    return true; //删除成功
}
```

按位查找

```
//按位查找，返回第 i 个元素（带头结点）  
LNode * GetElem(LinkList L, int i){  
    if(i<0)  
        return NULL;  
    LNode *p; //指针p指向当前扫描到的结点  
    int j=0; //当前p指向的是第几个结点  
    → p = L; //L指向头结点，头结点是第0个结点（不存数据）  
    while (p!=NULL && j<i) { //循环找到第 i 个结点  
        p=p->next;  
        j++;  
    }  
    return p;  
}
```

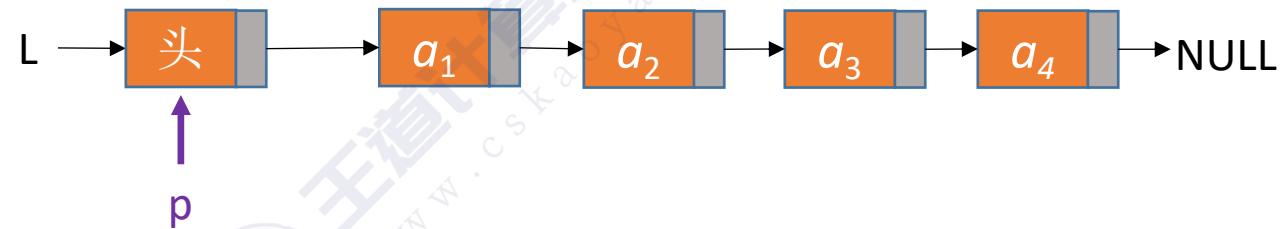
边界情况：
①如果 $i = 0$



按位查找

```
//按位查找, 返回第 i 个元素 (带头结点)
LNode * GetElem(LinkList L, int i){
    if(i<0)
        return NULL;
    LNode *p; //指针p指向当前扫描到的结点
    int j=0; //当前p指向的是第几个结点
    → p = L; //L指向头结点, 头结点是第0个结点 (不存数据)
    while (p!=NULL && j<i) { //循环找到第 i 个结点
    → p=p->next;
    j++;
}
return p;
}
```

边界情况:
②如果 $i = 8$



按位查找

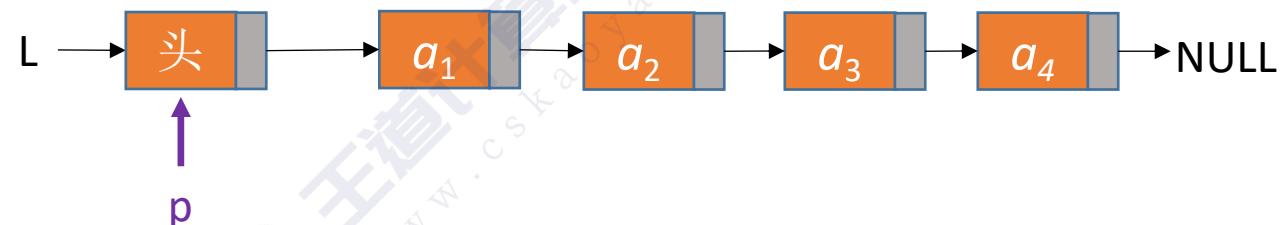
```
//按位查找, 返回第 i 个元素 (带头结点)
LNode * GetElem(LinkList L, int i){
    if(i<0)
        return NULL;
    LNode *p; //指针p指向当前扫描到的结点
    int j=0; //当前p指向的是第几个结点
    p = L; //L指向头结点, 头结点是第0个结点 (不存数据)
    while (p!=NULL && j<i) { //循环找到第 i 个结点
        p=p->next;
        j++;
    }
    return p;
}
```

平均时间复杂度: $O(n)$

普通情况:
③如果 $i = 3$



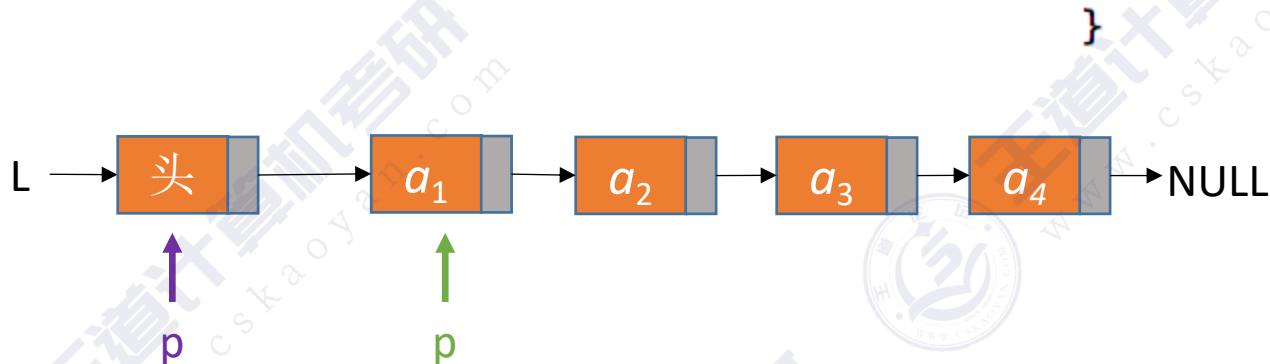
脑补一下咯



按位查找

王道书
版本

```
//按位查找，返回第 i 个元素（带头结点）  
LNode * GetElem(LinkList L, int i){  
    if(i<0)  
        return NULL;  
    LNode *p; //指针p指向当前扫描到的结点  
    int j=0; //当前p指向的是第几个结点  
    p = L; //L指向头结点，头结点是第0个结点（不存数据）  
    while (p!=NULL && j<i) { //循环找到第 i 个结点  
        p=p->next;  
        j++;  
    }  
    return p;  
}
```



```
LNode * GetElem(LinkList L, int i){  
    int j=1;  
    LNode *p=L->next;  
    if(i==0)  
        return L;  
    if(i<1)  
        return NULL;  
    while(p!=NULL && j<i){  
        p=p->next;  
        j++;  
    }  
    return p;
```

封装（基本操作）的好处

避免重复代码，
简洁、易维护

//在第 i 个位置插插入元素 e (带头结点)

```
bool ListInsert(LinkList &L, int i, ElemType e){  
    if(i<1)  
        return false;
```

LNode *p = GetElem(L, i-1); //找到第i-1个结点

int j=0; //当前p指向的是第j+1个结点

p = L; //L指向头结点, 头结点是第0个结点 (不存数据)

while (p!=NULL && j<i-1) { //循环找到第 i-1 个结点

p=p->next;

j++;

}

if(p==NULL) //i值不合法

return false;

return InsertNextNode(p, e);

s->data = e;

s->next=p->next;

p->next=s; //将结点s连到p之后

return true; //插入成功

p后插入新元素e

```
bool ListDelete(LinkList &L, int i, ElemType &e){  
    if(i<1)  
        return false;
```

LNode *p = GetElem(L, i-1); //找到第i-1个结点

p = L; //L指向头结点, 头结点是第0个结点 (不存数据)

while (p!=NULL && j<i-1) { //循环找到第 i-1 个结点

p=p->next;

j++;

//后插操作: 在p结点之后插入元素 e

```
bool InsertNextNode (LNode *p, ElemType e){
```

if (p==NULL)

return false;

LNode *s = (LNode *)malloc(sizeof(LNode));

if (s==NULL) //内存分配失败

return false;

s->data = e; //用结点s保存数据元素e

s->next=p->next;

p->next=s; //将结点s连到p之后

return true;

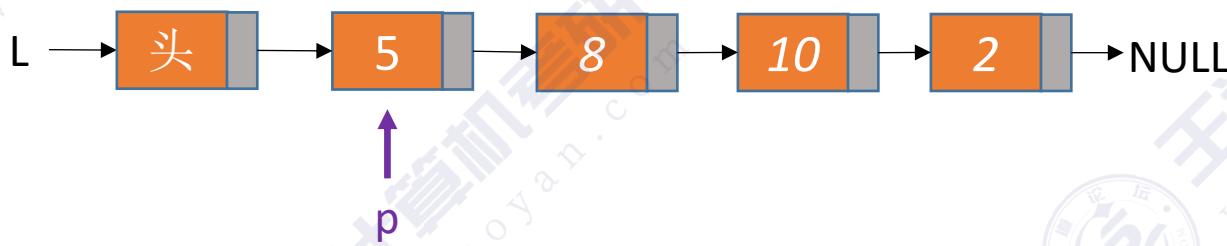
无其他结点

考虑到非法情
况: 健壮性

开”
|

按值查找

```
//按值查找，找到数据域==e 的结点  
LNode * LocateElem(LinkList L, ElemType e) {  
    LNode *p = L->next;  
    //从第1个结点开始查找数据域为e的结点  
    while (p != NULL && p->data != e)  
        p = p->next;  
    return p;    //找到后返回该结点指针，否则返回NULL  
}
```

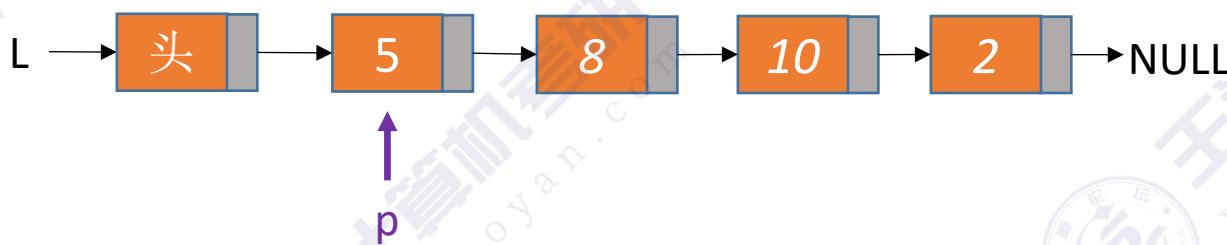


注：假设本例中
ElemType 是 int

能找到的情况：
①如果 $e = 8$

按值查找

```
//按值查找，找到数据域==e 的结点  
LNode * LocateElem(LinkList L, ElecType e) {  
    LNode *p = L->next;  
    //从第1个结点开始查找数据域为e的结点  
    while (p != NULL && p->data != e)  
        p = p->next;  
    return p;    //找到后返回该结点指针，否则返回NULL  
}
```



注：假设本例中
ElecType 是 int

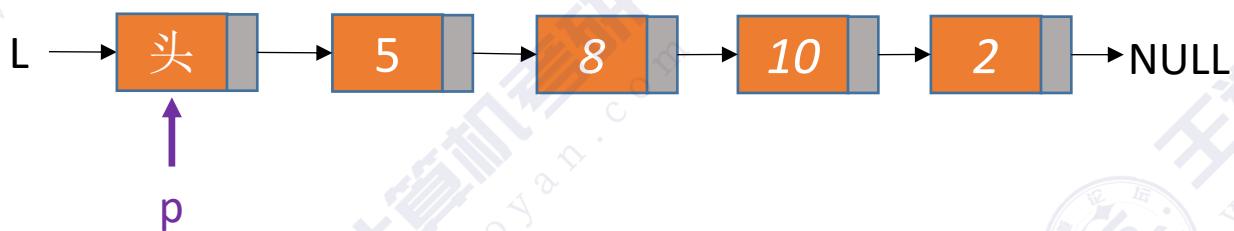
不能找到的情况：
②如果 $e = 6$

如果 ElecType 是更复
杂的结构类型呢？

平均时间复杂度： $O(n)$

求表的长度

```
//求表的长度  
int Length(LinkList L){  
    int len = 0;      //统计表长  
    LNode *p = L;  
    while (p->next != NULL){  
        p = p->next;  
        len++;  
    }  
    return len;  
}
```



脑补一下咯

如果不带头结点呢？

时间复杂度: $O(n)$

知识回顾与重要考点

单链表的查找

按位查找

注意与“顺序表”对比

按值查找

单链表不具备“随机访问”的特性，只能依次扫描

求单链表长度

三种基本操作的时间复杂度都是 $O(n)$

Key

如何写循环扫描各个结点的代码逻辑

注意边界条件的处理