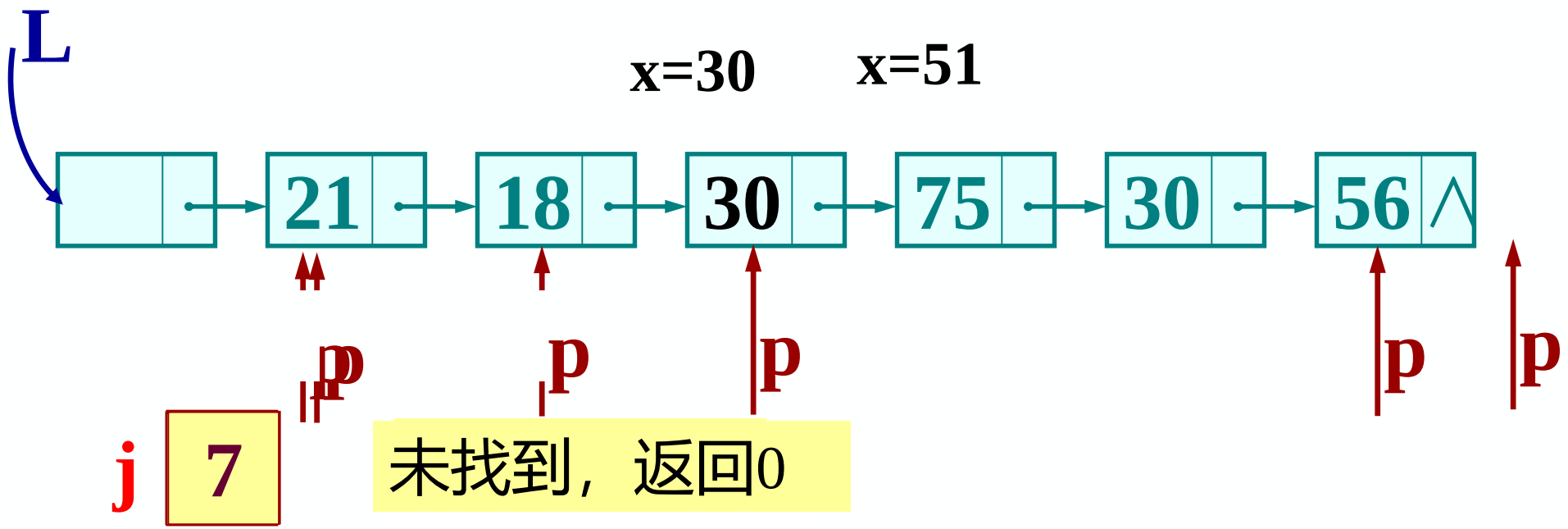
3.查找（根据指定数据获取数据所在的位置）



✓从第一个结点起，依次和e相比较。

✓如果找到一个其值与e相等的数据元素，则返回其在链表中的“位置”或地址；

✓如果查遍整个链表都没有找到其值和e相等的元素，则返回0或“NULL”。

【算法描述】

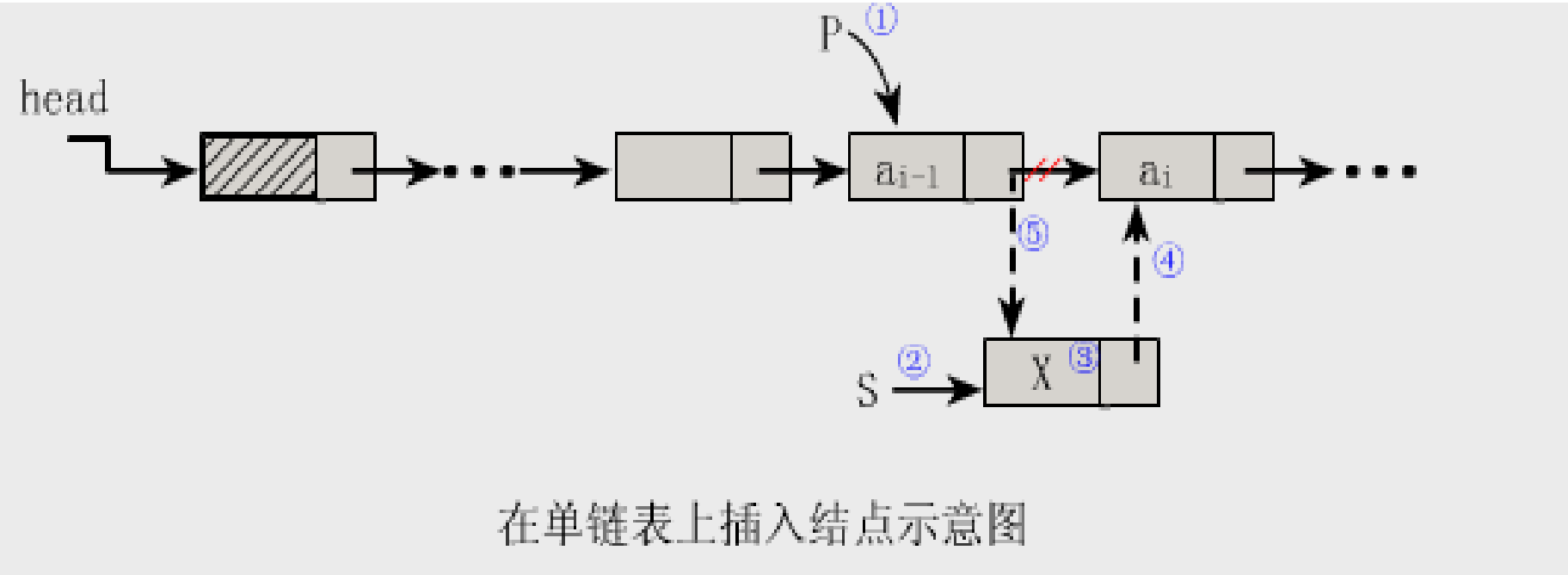
```
//在线性表L中查找值为e的数据元素
LNode *LocateELem_L (LinkList L，Elemtype e) {
 //返回L中值为e的数据元素的地址，查找失败返回NULL
  p=L->next;
  while(p &&p->data!=e)
        p=p->next;
  return p;
}
```

【算法描述】

```
//在线性表L中查找值为e的数据元素
int LocateELem_L (LinkList L，Elemtype e) {
 //返回L中值为e的数据元素的位置序号，查找失败返回0
  p=L->next; j=1;
  while(p &&p->data!=e)
        {p=p->next;  j++;}
  if(p) return j;
  else return 0;
}
```

3. 插入（插在第 i 个结点之前）

- 将值为x的新结点插入到表的第i个结点的位置上，即插入到a_{i-1}与a_i之间



在单链表上插入结点示意图

```
s->next=p->next;      p->next=s
```

思考：步骤1和2能互换么?

16:15

【算法步骤】

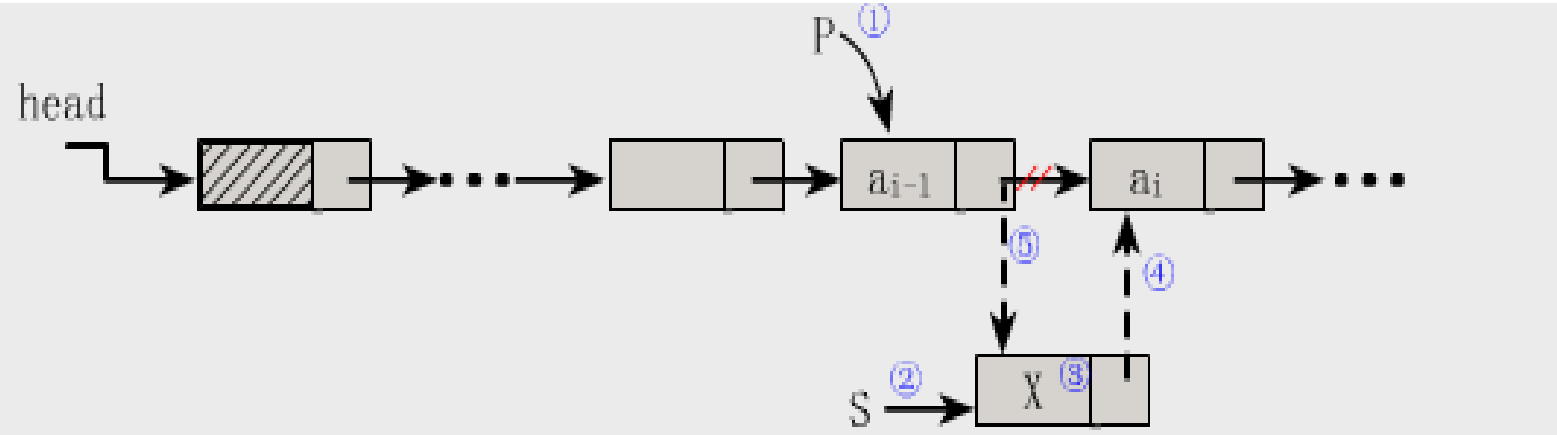
（1）找到a_{i-1}存储位置p

（2）生成一个新结点*s

（3）将新结点*s的数据域置为x

（4）新结点*s的指针域指向结点a_i

（5）令结点*p的指针域指向新结点*s



在单链表上插入结点示意图

【算法描述】

```
//在L中第i个元素之前插入数据元素e
Status ListInsert_L(LinkList &L,int i,ElemType e){
    p=L;j=0;
     while(p&&j<i−1){p=p->next;++j;}      //寻找第i−1个结点
     if(!p||j>i−1)return ERROR;      //i大于表长 + 1或者小于1
     s=new LNode;                    //生成新结点s
     s->data=e;                      //将结点s的数据域置为e
     s->next=p->next;                //将结点s插入L中
     p->next=s;
     return OK;
}//ListInsert_L
```

5. 删除（删除第 i 个结点）

- 将表的第i个结点删去
- 步骤：

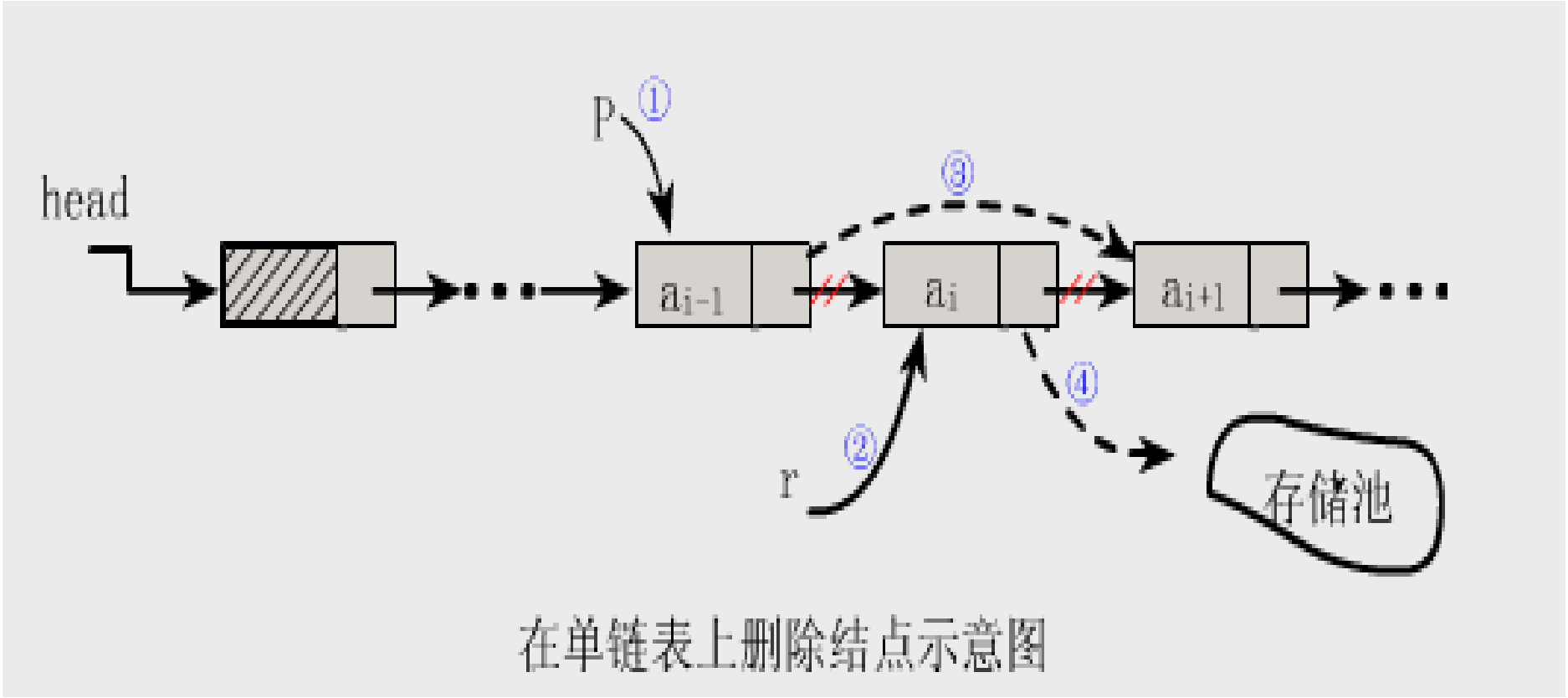
（1）找到a_{i-1}存储位置p

（2）保存要删除的结点的值

（3）令p->next指向a_i的直接后继结点

（4）释放结点a_i的空间

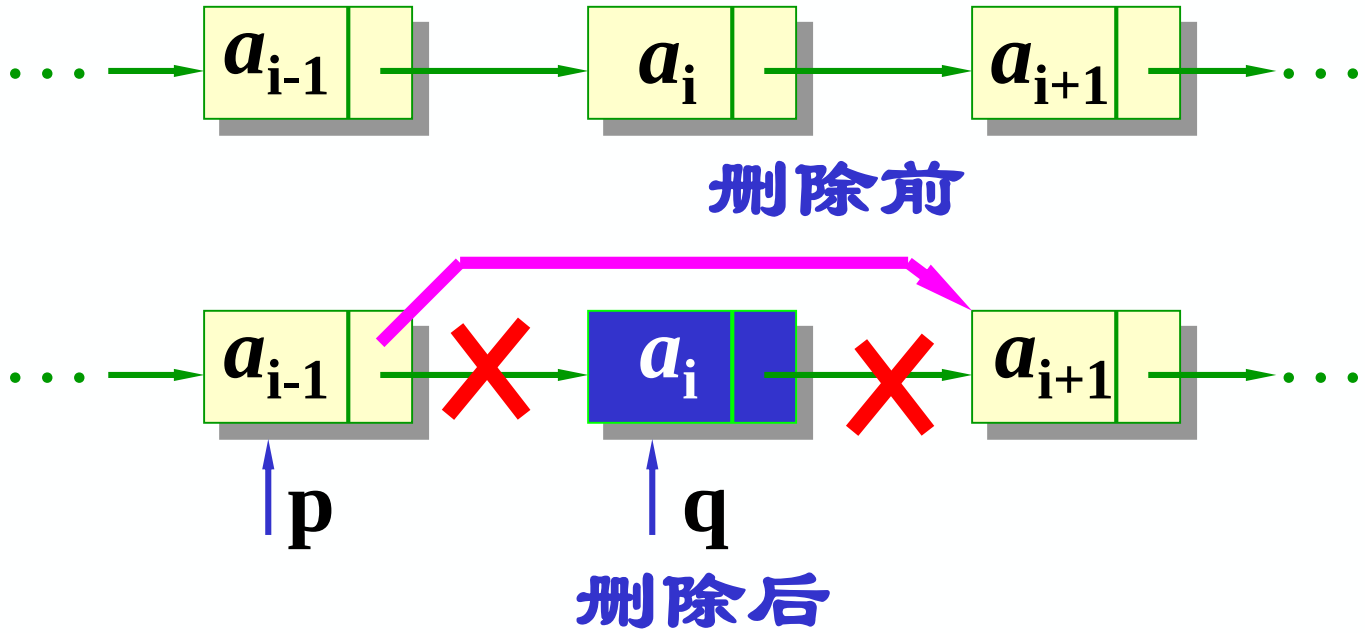
5. 删除（删除第 i 个结点）



在单链表上删除结点示意图

16:15

5. 删除（删除第 i 个结点）



p->next = p->next->next ???

【算法步骤】

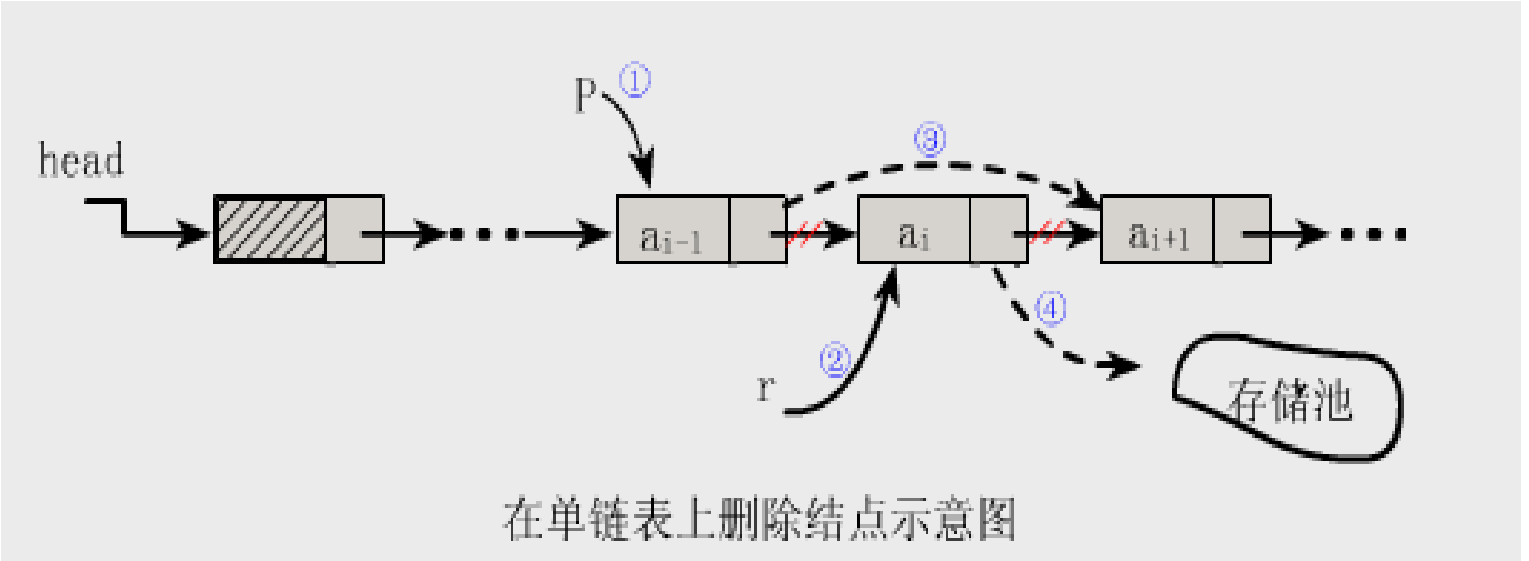
（1）找到a_{i-1}存储位置p

（2）临时保存结点a_i的地址在q中，以备释放

（3）令p->next指向a_i的直接后继结点

（4）将a_i的值保留在e中

（5）释放a_i的空间



在单链表上删除结点示意图

【算法描述】

```
//将线性表L中第i个数据元素删除
Status ListDelete_L(LinkList &L,int i,ElemType &e){
  p=L;j=0;
  while(p->next &&j<i-1){//寻找第i个结点，并令p指向其前驱
    p=p->next; ++j;
  }
  if(!(p->next)||j>i-1) return ERROR; //删除位置不合理
  q=p->next; //临时保存被删结点的地址以备释放
  p->next=q->next;  //改变删除结点前驱结点的指针域
  e=q->data; //保存删除结点的数据域
  delete q;     //释放删除结点的空间
return OK;

}//ListDelete_L
```

链表的运算时间效率分析

1. **查找：**因线性链表只能顺序存取，即在查找时要从头指针找起，查找的时间复杂度为 O(n)。

2. **插入和删除：**因线性链表不需要移动元素，只要修改指针，一般情况下时间复杂度为 O(1)。

但是，如果要在单链表中进行前插或删除操作，由于要从头查找前驱结点，所耗时间复杂度为 O(n) 。