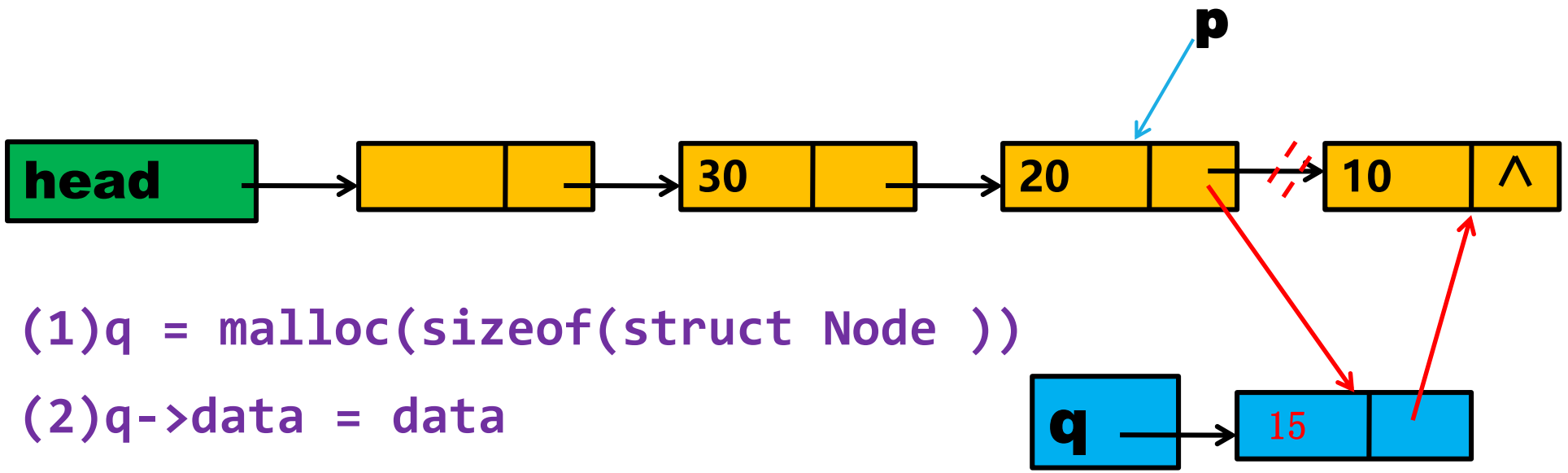
2.7 单链表的插入：后插法




```
(1)q = malloc(sizeof(struct Node ))
(2)q->data = data
(3)q->next = p->next
(4)p->next = q
```

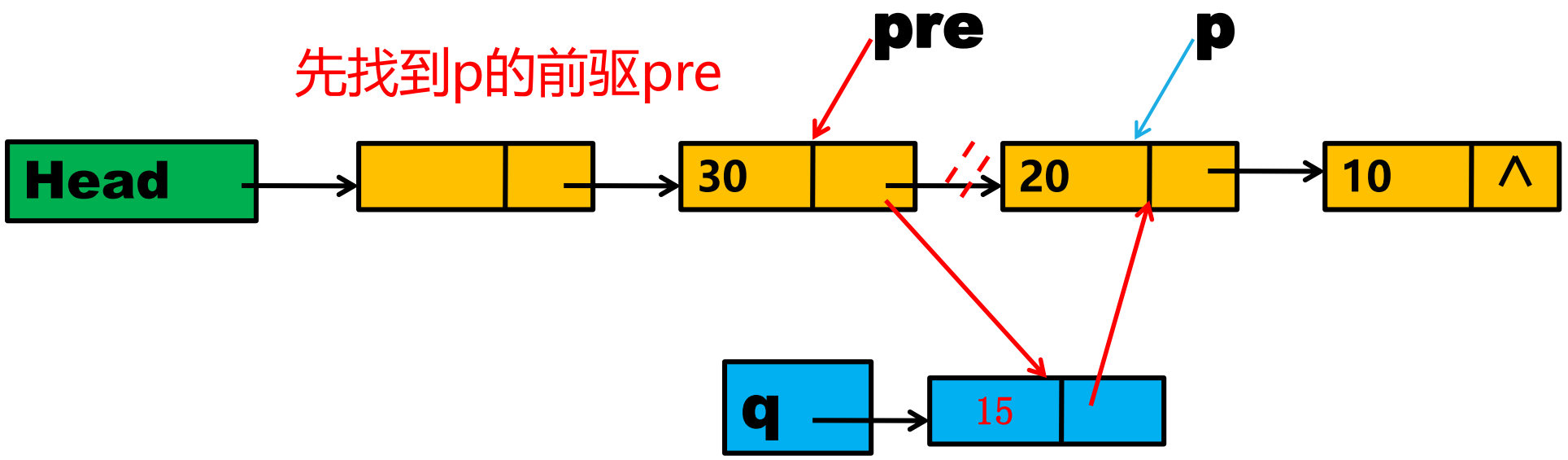
2.7 单链表的插入：后插法

算法2-17

```
1   //在llist链表中的p位置之后插入值为x的结点
2   int InsertPost_link(LinkList llist,PNode p,DataType x)
3   {
4           PNode q;
5           if(p==NULL) { printf("para failure!\n");return 0;}
6           q = (PNode)malloc(sizeof(struct Node));
7           if(q == NULL)
8          {
9                   printf("out of space!\n");  return 0;
10          }
11          else
12         {
13                  q->data=x;  q->next=p->next;
14                  p->next=q;  return 1;
15          }
16  }
```

算法时间复杂度
O(1)

2.7 单链表的插入：前插法




```
(1)q = malloc(sizeof(struct Node ))
(2)q->data = data
(3)q->next = p
(4)pre->next = q
```

2.7 单链表的插入：前插法

算法2-18

```
1   int InsertPre_link(LinkList llist,PNode p,DataType x)
2   { //在llist链表中的p位置之前插入值为x的结点
3           PNode pre = llist; PNode q = NULL;
4          if(p==NULL) { printf("para failure!\n");return 0;}
5           while (pre->next != p) /*定位p的前驱结点*/
6           {
7                   pre = pre->next;
8           }
9           q = (PNode)malloc(sizeof(struct Node));
10          q->data = x;
11          q->next = p;
12          pre->next = q;
13         return 1;
14  }
```

算法时间复杂度
O(n)