数据结构和算法

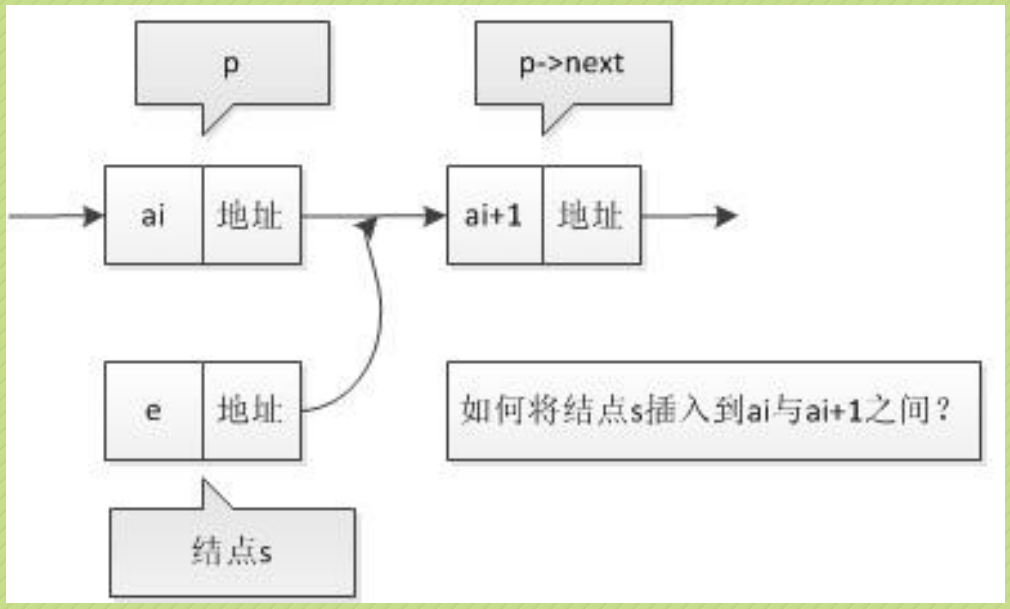
作者：小甲鱼

让编程改变世界

Change the world by program

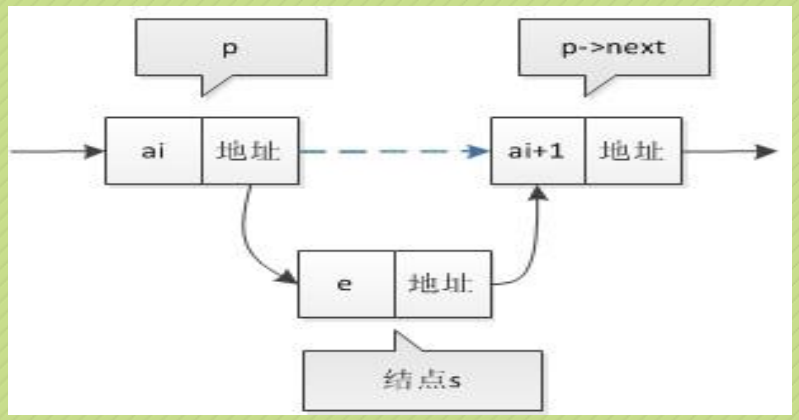
单链表的插入

- 我们先来看下单链表的插入。假设存储元素e的结点为s，要实现结点p、p->next和s之间逻辑关系的变化，大家参考下图思考一下：



单链表的插入

- 我们思考后发觉根本用不着惊动其他结点，只需要让s->next和p->next的指针做一点改变。
  - s->next = p->next;
  - p->next = s;
- 我们通过图片来解读一下这两句代码。



单链表的插入

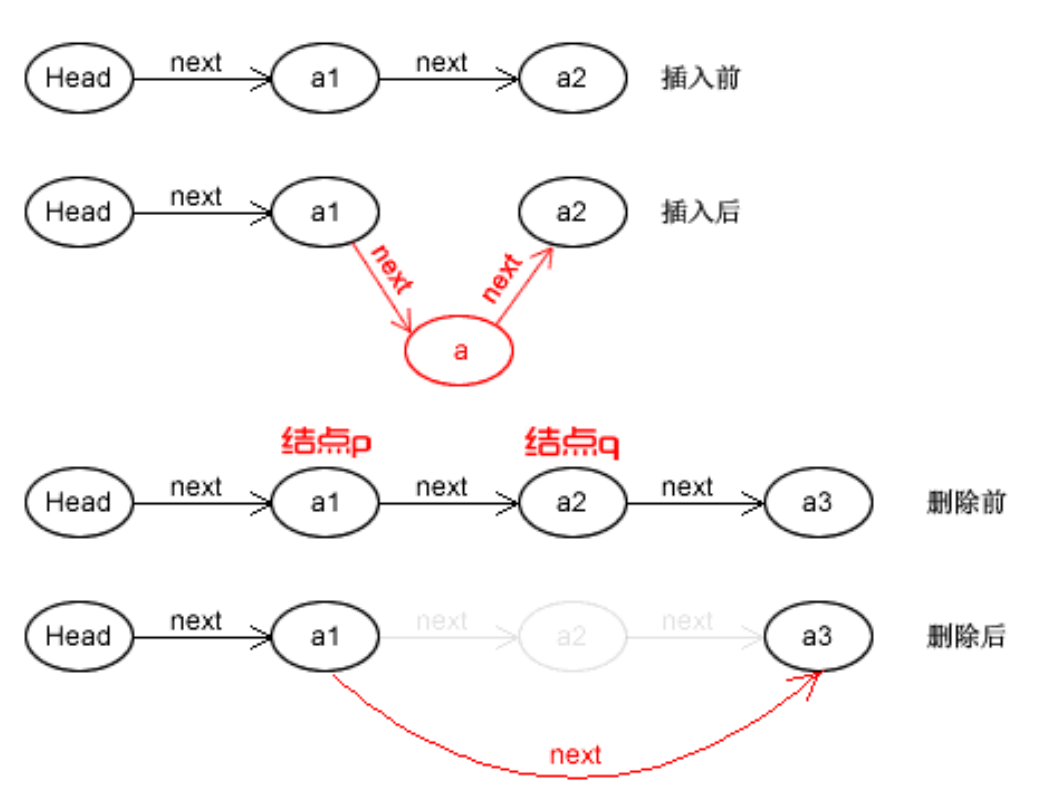
- 那么我们考虑一下大部分初学者最容易搞坏脑子的问题：这两句代码的顺序可不可以交换过来？
  - 先p->next = s;
  - 再s->next = p->next;
- 大家发现没有？如果先执行p->next的话会先被覆盖为s的地址，那么s->next = p->next其实就等于s->next = s了。
- 所以这两句是无论如何不能弄反的，这点初学者一定要注意咯~

单链表的插入

- 单链表第i个数据插入结点的算法思路：
  - 声明一结点p指向链表头结点，初始化j从1开始；
  - 当j<1时，就遍历链表，让p的指针向后移动，不断指向下一结点，j累加1；
  - 若到链表末尾p为空，则说明第i个元素不存在；
  - 否则查找成功，在系统中生成一个空结点s；
  - 将数据元素e赋值给s->data；
  - 单链表的插入刚才两个标准语句；
  - 返回成功。

单链表的删除

- 现在我们再来看单链表的删除操作。



单链表的删除

- 假设元素a2的结点为q，要实现结点q删除单链表的操作，其实就是将它的前继结点的指针绕过指向后继结点即可。
- 那我们所要做的，实际上就是一步：
  - 可以这样：p->next =p->next->next;
  - 也可以是：q=p->next; p->next=q->next;
- 那么我给大家提供算法的思路，由大家来写一下代码吧~

单链表的删除

- 单链表第i个数据删除结点的算法思路：
  - 声明结点p指向链表第一个结点，初始化j=1；
  - 当j<1时，就遍历链表，让P的指针向后移动，不断指向下一个结点，j累加1；
  - 若到链表末尾p为空，则说明第i个元素不存在；
  - 否则查找成功，将欲删除结点p->next赋值给q；
  - 单链表的删除标准语句p->next =q->next；
  - 将q结点中的数据赋值给e，作为返回；
  - 释放q结点。

效率PK

- 我们最后的环节是效率PK，我们发现无论是单链表插入还是删除算法，它们其实都是由两个部分组成：第一部分就是遍历查找第i个元素，第二部分就是实现插入和删除元素。
- 从整个算法来说，我们很容易可以推出它们的时间复杂度都是O(n)。
- 再详细点分析：如果在我们不知道第i个元素的指针位置，单链表数据结构在插入和删除操作上，与线性表的顺序存储结构是没有太大优势的。

效率PK

- 但如果，我们希望从第i个位置开始，插入连续10个元素，对于顺序存储结构意味着，每一次插入都需要移动n-i个位置，所以每次都是O(n)。
- 而单链表，我们只需要在第一次时，找到第i个位置的指针，此时为O(n)，接下来只是简单地通过赋值移动指针而已，时间复杂度都是O(1)。
- 显然，对于插入或删除数据越频繁的操作，单链表的效率优势就越是明显啦~