链表（下）

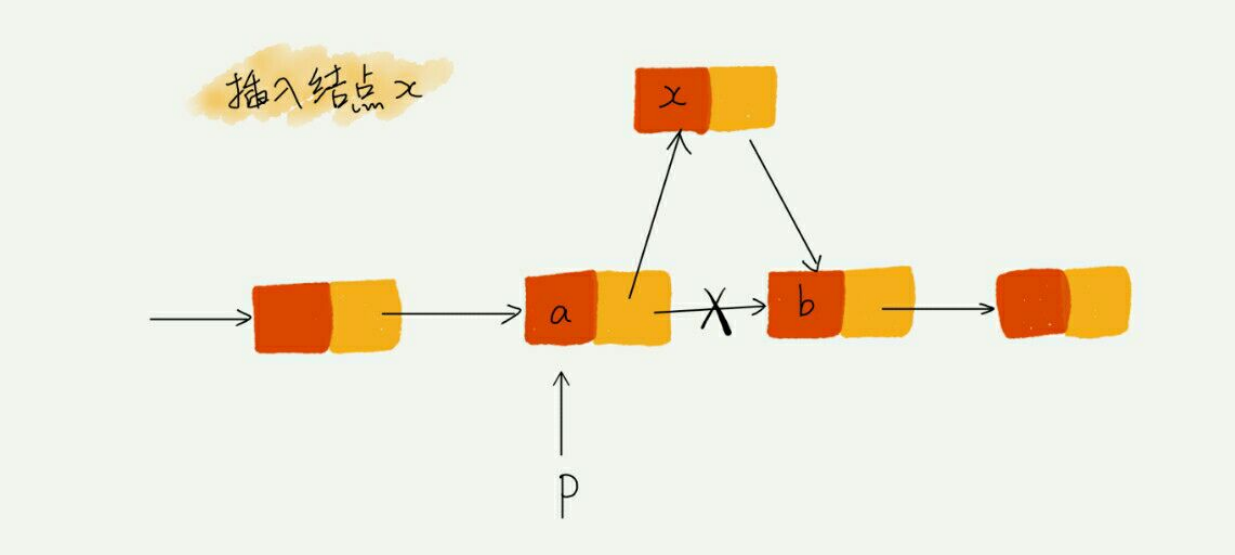
数据结构与算法

链表代码书写技巧

理解指针或引用的含义

将某个变量赋值给指针，实际上就是将这个变量的地址赋值给指针。代码：`p->next = q` 指 p 结点的 next 指针存储了 q 结点的内存地址。代码：`p->next = p->next->next` 指 p 结点的 next 指针存储了 p 结点的下下一个结点的内存地址。

警惕指针丢失和内存泄露



实现这一操作的代码

```
x->next = p->next; // 将 x 结点的 next 指针指向 b 结点
p->next = x;       // 将 p 的 next 指针指向 x
```

如果这两行代码顺序错误，就会产生错误，C 语言的话会产生内存泄露。

利用哨兵简化实现难度

如果我们在 p 结点后面插入一个新的结点，则只需下面两行代码即可完成

```
new_node->next = p->next;
p->next = new_node;
```

但是，如果这是一个空链表，这个逻辑就是错误的，需要进行特殊处理。

```
```

```
if(head == null){
    head = new_node;
}
```

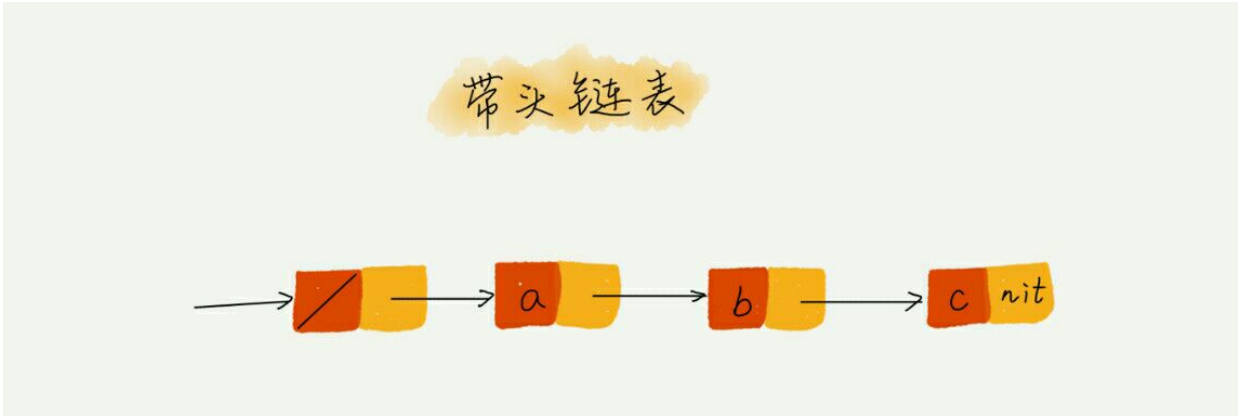
同理，对于删除操作，如果不是尾结点，代码如下

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

如果是尾结点，代码如下

```
if (head->next == null){
    head == null;
}
```

这种逻辑操作很繁琐，所以通常我们会引入哨兵结点，在任何时候，不管链表是不是空，head 指针都会一直指向这个哨兵结点。这种链表称为带头链表。

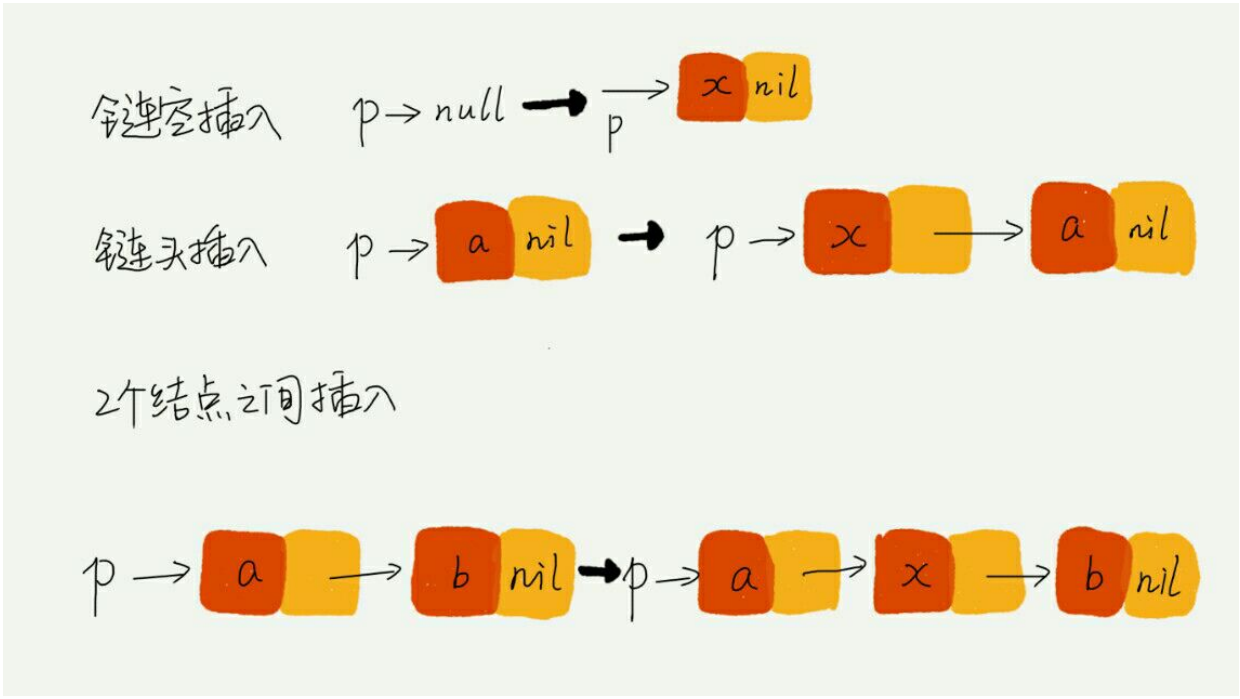


留意边界条件的处理

检验链表代码是否正确的几个边界条件

- 链表为空时，能否正常工作。
- 链表只包含一个结点时，能否正常工作。
- 链表只包含两个结点时，能否正常工作。
- 代码逻辑在处理头结点和尾结点时，能否正常工作。

举例画图，辅助思考



参考自：极客时间《数据结构与算法之美 》专栏