删除链表的中间节点和 a/b 处的节点

【题目】

给定链表的头节点 head,实现删除链表的中间节点的函数。例如:

不删除任何节点:

1->2, 删除节点1;

1->2->3, 删除节点 2;

1->2->3->4, 删除节点 2;

1->2->3->4->5, 删除节点3:

进阶:

给定链表的头节点 head、整数 a 和 b,实现删除位于 a/b 处节点的函数。例如:

链表: 1->2->3->4->5, 假设 a/b 的值为 r。

如果 r 等于 0, 不删除任何节点;

如果 r 在区间(0, 1/5]上, 删除节点 1;

如果 r 在区间(1/5, 2/5]上, 删除节点 2;

如果 r 在区间(2/5, 3/5]上, 删除节点 3;

如果 r 在区间(3/5, 4/5]上, 删除节点 4;

如果 r 在区间(4/5, 1]上, 删除节点 5;

如果r大于1,不删除任何节点。

【难度】

士 ★☆☆☆

【解答】

先来分析原问题,如果链表为空或者长度为 1,不需要调整,则直接返回;如果链表的长度为 2,将头节点删除即可;当链表长度到达 3,应该删除第 2 个节点;当链表长度为 4,应该删除第 2 个节点;当链表长度为 5,应该删除第 3 个节点······也就是链表长度每增加 2(3,5,7...),要删除的节点就后移一个节点。删除节点的问题在之前的题目中我们已经讨论过,如果要删除一个节点,则需要找到待删除节点的前一个节点。

具体过程请参看如下代码中的 removeMidNode 方法。

```
public class Node (
        public int value;
        public Node next;
        public Node (int data) {
                this.value = data:
- 1
public Node removeMidNode (Node head) {
        if (head == null || head.next == null) {
                return head;
        if (head.next.next == null) {
                return head.next;
        Node pre = head;
        Node cur = head.next.next:
        while (cur.next != null && cur.next.next != null) {
                pre = pre.next;
                cur = cur.next.next;
        pre.next = pre.next.next;
        return head;
 }
```

接下来讨论进阶问题,首先需要解决的问题是,如何根据链表的长度 n,以及 a 与 b 的值决定该删除的节点是哪一个节点呢?根据如下方法:

先计算 $\underline{\text{double r} = ((\text{double}) (a * n)) / ((\text{double}) b)}$ 的值, 然后 $\underline{\text{r}}$ 向上取整之后的整数值代表该删除的节点是第几个节点。

下面举几个例子来验证一下:

如果链表长度为7, a=5, b=7。

r=(7*5)/7=5.0,向上取整后为5,所以应该删除第5个节点。

如果链表长度为7, a=5, b=6。

r = (7*5)/6 = 5.8333 …,向上取整后为 6,所以应该删除第 6 个节点。如果链表长度为 7, a=1, b=6。

r=(7*1)/6=1.1666···,向上取整后为2,所以应该删除第2个节点。

知道该删除第几个节点之后,接下来找到需要删除节点的前一个节点即可。具体过程请参看如下代码中的 removeByRatio 方法。

```
public Node removeByRatio(Node head, int a, int b) {
       if (a < 1 | | a > b) {
              return head;
       int n = 0;
       Node cur = head;
       while (cur != null) {
               n++;
               cur = cur.next;
       n = (int) Math.ceil(((double) (a * n)) / (double) b);
       if (n == 1) {
              head = head.next;
       if (n > 1) {
               cur = head;
               while (--n != 1) {
                      cur = cur.next;
               cur.next = cur.next.next;
       return head;
}
```

反转单向和双向链表

【题目】

分别实现反转单向链表和反转双向链表的函数。

【要求】

如果链表长度为N,时间复杂度要求为O(N),额外空间复杂度要求为O(1)。

【难度】

士 ★☆☆☆