Reorder the given singly-linked list N1 - > N2 -> N3 -> N4 -> … -> Nn -> null to be N1 -> Nn -> N2 -> Nn-1 -> N3 … -> null

public class ReorderList {

public ListNode reorder(ListNode head) {

if (head == null || head.next == null) {

return head;

}

// 1. find the middle node

ListNode mid = middleNode(head);

ListNode one = head;

ListNode two = mid.next;

// de-link the second half from the list;

mid.next = null;

// 2. reverse the second half

ListNode secondReversed = reverse(two);

// 3. merge the two halves

return merge(one, secondReversed);

}

private ListNode middleNode(ListNode head) {

ListNode slow = head;

ListNode fast = head;

|  |  |
| --- | --- |
| 错误  while (fast.next != null) {  if (fast.next.next == null) {  return slow;  } else {  slow = slow.next;  fast = fast.next.next  }  } | 正确  while (fast.next != null || fast.next.next != null)  slow = slow.next;  fast = fast.next.next  } |

}

private ListNode reverse(ListNode head) {

|  |  |
| --- | --- |
| 错误  if ( head.next == null) {  return head;  }  ListNode dummyhead = null;  ListNode pre = dummyhead;  ListNode cur = head;  ListNode next = head.next;  head = next.next;  cur.next = pre; | 正确  if ( head == null || head.next == null) {  return head;  }  ListNode prev = null;  while (head != null) {  ListNode next = head.next;  head.next = prev;  prev = head;  head = next;  }  return prev;  } |

private ListNode merge(ListNode one, ListNode two) {

ListNode dummy = new ListNode(0);

ListNode cur = dummy;

while ( one != null && two != null) {

cur.next = one;

one = one.next;

cur.next.next = two;

two = two.next;

cur = cur.next.next;

}

if (one != null) {

cur.next = one;

} else {

cur.next = two;

}

return dummy.next;

}

}

Recursion

public LIstNode Reverse(ListNode head) {

if ( head == null || head.next == null) {

return head;

}

ListNode newhead = reverse(head.next);

head.next.next = head;

head.next = null;

return newhead;

}