Reorder the given singly-linked list N1 -> N2 -> N3 -> N4 -> … -> Nn -> null to be N1- > Nn -> N2 -> Nn-1 -> N3 -> Nn-2 -> … -> null

Examples

* L = null, is reordered to null
* L = 1 -> null, is reordered to 1 -> null
* L = 1 -> 2 -> 3 -> 4 -> null, is reordered to 1 -> 4 -> 2 -> 3 -> null
* L = 1 -> 2 -> 3 -> null, is reordred to 1 -> 3 -> 2 -> null

Bo Chen

public ListNode reorder(ListNode head) {

// 1 -> 2 -> 3 4 -> 5

// mid

// rightPartHead

// corner case

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

return head;

}

ListNode mid = getMid(head);

ListNode rightPartHead = mid.next;

**mid.next = null;**

ListNode one = head;

ListNode two = reverse(rightPartHead);

return merge(one, two);

}

public ListNode reverse(ListNode head) {

// 1 <- 2 -> 3

// —------

// head

// base case

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

return head;

}

ListNode newHead = reverse(head.next);

head.next.next = head;

head.next = null;

return newhead;

}

public ListNode merge(ListNode one, ListNode two) {

// 1 -> 2 -> 3

// one

// 5 -> 4

// two

// dummy -> 1 -> 5 ->

// cur

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;

}

public ListNode getMid(ListNode head) {

ListNode slow = head;

ListNode fast = head;

while (fast.next != null && fast.next.next != null) {

slow = slow.next;

fast = fast.next.next;

}

return slow;

}

每个function要分开写成Helper ⇒ 可读性

1 -> 2 -> 3 -> 4 -> 5 -> 6

s

f

public ListNode reorder(ListNode head)

{

if(head == null || head.next == null) return head;

ListNode fast = head;

ListNode slow = head;

while(fast.next != null && fast.next.next != null)

{

Slow = slow.next;

Fast = fast.next.next;

}

ListNode remember2 = slow.next;

Slow.next =null;

**If(fast.next == null) {**

**ListNode head2 = fast;**

**}**Else {

ListNode head2 = fast.next;

} //head2

reverse(remember2);

ListNode cur = head, cur2 = head2, dummy = new ListNode(0);

ListNode remember = dummy;

while(head != null && head2 != null )

{

Dummy.next = cur;

Dummy.next.next = cur2;

Cur = cur.next;

Cur2 = cur2.next;

Dummy = dummy.next.next;

}

if(cur == null) dummy.next = cur2;

Else dummy.next = cur;

Return remember.next;

}

// dummy 不动得点

1 2 -> null

/ cur

dummy |

3 -> 4 -> null

cur2

public void reverse(ListNode head)

{

if(head.next == null) return;

reverse(head.next);

Head.next.next = head;

Head.next = null;

}