Lecture 26 LinkedList

1. <https://leetcode.com/problems/sort-list/>

class Solution {

ListNode\* midPoint(ListNode\*head)

{

ListNode\* slow = head;

ListNode\* fast = head->next;

while(fast!=NULL && fast->next!=NULL)

{

fast = fast->next->next;

slow = slow->next;

}

return slow;

}

ListNode\* merge(ListNode\*a, ListNode\*b)

{

if(a==NULL)

{

return b;

}

if(b==NULL)

{

return a;

}

ListNode \*c;

if(a->val <= b->val)

{

c=a;

c->next=merge(a->next, b);

}

else

{

c=b;

c->next=merge(a, b->next);

}

return c;

}

public:

ListNode\* sortList(ListNode\* head) {

if(head==NULL || head->next==NULL)

{

return head;

}

// 1. mid point

ListNode\* mid = midPoint(head);

ListNode \*a = head;

ListNode \*b = mid->next;

mid->next = NULL;

// 2. recursively sort

a = sortList(a);

b = sortList(b);

// 3. merge

ListNode \*c = merge(a, b);

return c;

}

};

1. <https://leetcode.com/problems/remove-duplicates-from-sorted-list/>

class Solution {

public:

ListNode\* deleteDuplicates(ListNode\* head) {

ListNode \*temp = head;

while(temp!=NULL && temp->next!=NULL)

{

if(temp->val==temp->next->val)

{

temp->next = temp->next->next;

}

else

{

temp = temp->next;

}

}

return head;

}

};

1. <https://leetcode.com/problems/rotate-list/>

class Solution {

public:

ListNode\* rotateRight(ListNode\* head, int k) {

if(head==NULL || head->next==NULL || k==0)

{

return head;

}

ListNode \*curr = head;

int length = 1;

while(curr->next!=NULL)

{

curr = curr->next;

length++;

}

curr->next = head;

k = k%length;

k = length-k;

while(k--)

{

curr = curr->next;

}

head = curr->next;

curr->next = NULL;

return head;

}

};

1. <https://leetcode.com/problems/add-two-numbers-ii/>

class Solution {

ListNode \*reverse(ListNode\*head)

{

ListNode\* c = head;

ListNode\* p = NULL;

ListNode\* n;

while(c!=NULL)

{

n = c->next;

c->next = p;

p = c;

c = n;

}

return p;

}

public:

ListNode\* addTwoNumbers(ListNode\* l1, ListNode\* l2) {

l1 = reverse(l1);

l2 = reverse(l2);

ListNode \*dummy = new ListNode();

ListNode \*temp = dummy;

int carry = 0;

while(l1!=NULL || l2!=NULL || carry!=0)

{

int sum = 0;

if(l1!=NULL)

{

sum += l1->val;

l1 = l1->next;

}

if(l2!=NULL)

{

sum += l2->val;

l2 = l2->next;

}

sum += carry;

carry = sum/10;

ListNode \* newNode = new ListNode(sum%10);

temp->next = newNode;

temp = temp->next;

}

ListNode \*l3 = reverse(dummy->next);

return l3;