Linked List

Q) Partition List

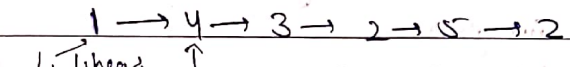
Given LL A & value B, partition it such that all nodes less than B come before nodes greater than or equal to B.

$1 \le |A| \le 10^6$, $1 \le A[i], B \le 10^9$

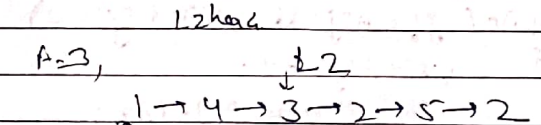
Ex: A = [1, 4, 3, 2, 5, 2] B = 3

Op = [1, 2, 2, 4, 3, 5]

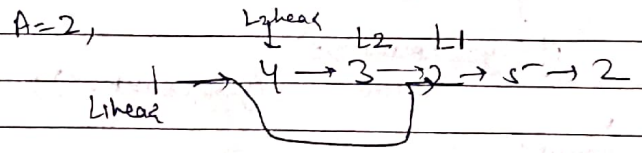
A=1, 4



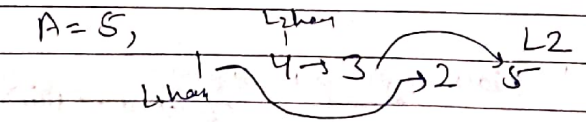
A=3,



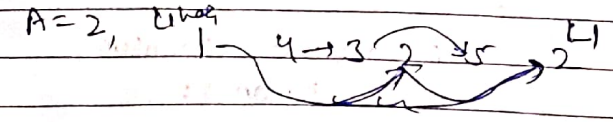
A=2,



A=5,



A=2,




```
public ListNode partition(ListNode A, int B)
{
    ListNode L1 = L2 = L1head = L2head = null;

    dl head = A;

    while (A != null)
    {
        if (A.val < B)
        {   if (dl1 == null)
            {   dl1 = A;
                dl1head = A;
            }
            else {  dl1.next = A;
                    dl1 = A;
                 }
        }
        else{
            if (dl2 == null)
            {   dl2 = A;
                dl2head = A;
            }
            else {  dl2.next = A;
                    dl2 = A;
                 }
        }
        A = A.next;
    }
    if (dl2 != null) dl2.next = null;
    if (dl1 != null) dl1.next = dl2head;
    return dlhead;
}
```

TC → O(N)

SC → O(1)

8) Palindrome List

Given a singly LL A, determine if its a palindrome. return 1 if yes, else 0.

$1 \le |A| \le 10^5$

ex: A = [1,2,2,1] Op = 1

A = [1,2,3] Op = 0

1 → 2 → 2 → 1

mid → reverse

1 → 2 | 1 → 2

compare if true = 1.

```
int lpalin (ListNode A)
{
    ListNode slow=A, fast=A;
    while ( slow!=null && fast!=null && fast.next!=null
                                  && fast.next.next!=null)
    {   slow = slow.next;
        fast = fast.next.next;
    }
    ListNode l2 = slow.next;
    slow.next = null;
    l2 = reverse(l2);
    l1 = A;
    int res = 1;
    while ( l1!=null && l2!=null)
    {
        if (l1.val != l2.val)
        {   res=0; break;
        }
        l1 = l1.next;
        l2 = l2.next;
    }
    return res;
}
ListNode reverse ( ListNode l2)
{   ListNode prev=null, curr=l2, next=l2;
    while ( next!=null)
    {   next = curr.next;
        curr.next = prev;
        prev = curr;
        curr = next;
    }
    return prev;
}
```

Scanned with Oken Scanner

Page No. _____
Date

Q) Remove loop from linked list.

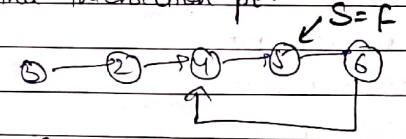
Given a LL with contain some loop.
Find the node which creates a loop & break it by making the node pt. to NULL.

$1 \le$ no. of nodes ≤ 1000

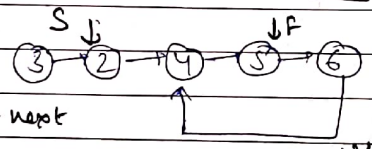
- return head of update LL.

ex: $3 \to 2 \to 4 \to 5 \to 6 \Rightarrow 3 \to 2 \to 4 \to 5 \to 6 \to$ ⊥

→ 1) find if there is a cycle, using slow fast ptr.
2) from the pt where S=F found. iterate to find intersection pt.



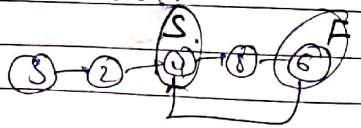
reset slow to ~~A~~.next



slow = A.next

while (fast.next != null)
{ slow = slow.next
fast = fast.next
}

fast.next = null.




```
public ListNode solve(ListNode A)
{
    ListNode slow=A, fast=A;
    while(slow && fast && fast.next!=null)
    {
        slow = slow.next;
        fast = fast.next.next;
        if(slow==fast)
            break;
    }
    if(slow==fast)
    {
        slow = A.next;
        while(fast.next != slow)
        {
            slow = slow.next;
            fast = fast.next;
        }
        fast.next = null;
    }
    return A;
}
```

TC → O(N)
SC → O(1)

Scanned with Oken Scanner

Page No. _____
Date

Q) Merge 2 Sorted Lists

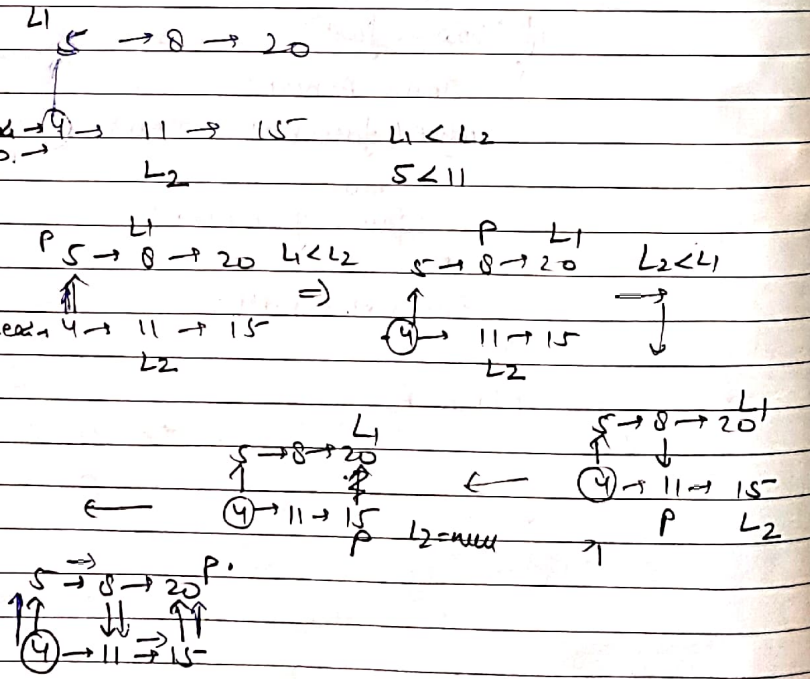
Merge 2 sorted linked lists A & B and return it as a new list.

$0 \le |A|, |B| \le 10^5$

ex: A = 5 → 8 → 20

B = 4 → 11 → 15

Op = 4 → 5 → 8 → 11 → 15 → 20



Page No. _____
Date

```
public ListNode mergeTwoLists(ListNode A, ListNode B)
{
    if (B == null) return A;
    if (A == null) return B;
    ListNode head, p, l1, l2;
    l1 = A;
    l2 = B;
    head = new ListNode(0);
    p = head;
    while(l1 != null && l2 != null)
    {
        if (l1.val < l2.val)
        {
            p.next = l1;
            p = l1;
            l1 = l1.next;
        }
        else
        {
            p.next = l2;
            p = l2;
            l2 = l2.next;
        }
    }
    if (l1 == null)
        p.next = l2;
    if (l2 == null)
        p.next = l1;
    return head.next;
}
```

TC → O(n)

SC → O(1)

Scanned with Oken Scanner

(Q) K reverse linked list

Given Singly LL A & an integer B, reverse the nodes of the list B at a time & return modified Linked list.

$1 \le |A| \le 10^3$

B always divide A

ex: A = [1,2,3,4,5,6] , B = 2

①

Op = [2,1,4,3,6,5]

② A = [1,2,3,4,5,6] B = 3

Op = [3,2,1,6,5,4]

```
public ListNode reverseList(ListNode A, int B)
{
    if (A == null) return null;
    if (B <= 1) return A;
    return Reverse(A, B);
}

ListNode Reverse(ListNode head, int B)
{
    ListNode prev = null, curr = head, next = head;
    int count = 0;
    while (next != null && count < B)
    {
        next = curr.next;
        curr.next = prev;
        prev = curr;
        curr = next;
        count++;
    }
    if (next != null)
        head.next = reverse(next, B);
    return prev;
}
```

(Q) Remove nth node from list end

Given Singly LL A, remove Bth node from the end of list & return its head.

ex: 1 2 3 4 5

1→2→3→(4)→5 , B = 2

Op = 1→2→3→5. $1 \le |A| \le 10^6$

→ if B > size of list, remove first node of list.

```
public ListNode removeNthFromEnd(ListNode A, int B)
{
    ListNode P1 = A;
    ListNode P2 = A;
    int i = 0;
    while (i < B && P2.next != null)
    {
        i++;
        P2 = P2.next;
    }
    if (i < B) return A.next;

    while (P2.next != null)
    {
        P1 = P1.next;
        P2 = P2.next;
    }
    P1.next = P1.next.next;
    return A;
}
```

0 1 2 3 4

1→2→3→4→5

P1 P1 P2

P1 P2

1→2→3→4→5

1→2→3→5

(Q) Reorder list

Given a singly LL A

A: $A_0 \to A_1 \to \cdots \to A_{n-1} \to A_n$

reorder it to

$A_0 \to A_n \to A_1 \to A_{n-1} \to A_2 \to A_{n-2} \to \cdots$

$1 \le |A| \le 10^6$

ex: [1, 2, 3, 4, 5]

O/P ⇒ [1, 5, 2, 4, 3]

1 → 2 → 3 → 4 → 5 → 6 → 7 → 8.

Find middle & reverse second half.

P1 ⇒ 1 → 2 → 3 → 4

P2 ⇒ 8 → 7 → 6 → 5

now merge both the list.

```
ListNode reverse(ListNode A)
{   ListNode head=A, prev=null, curr=A, next=A;

    while(next!=null)
    {   next = curr.next;
        curr.next = prev;
        prev = curr;
        curr = next;
    }
    return prev;
}
```

```
public ListNode reorderList(ListNode A)
{   if(A==null) return null;
    if(A.next==null) return A;

    ListNode slow = A, fast = A;

    while(fast!=null && fast.next!=null)
    {   slow = slow.next;
        fast = fast.next.next;
    }

    ListNode l1 = A, l1head, l2head;
    ListNode l2 = slow.next;
    slow.next = null;
    ListNode l2 = reverse(l2);

    while(l1.next!=null && l2!=null)
    {   l1head = l1.next;
        l2head = l2.next;
        l1.next = l2;
        l2.next = l1head;
        l1 = l1head;
        l2 = l2head;
    }

    return A;
}
```

TC → O(n)

SC → O(1)

Page No.
Date

(Q) Reverse Linked List II

Reverse a linked list from position B to C.

A = 1 → 2 → 3 → 4 → 5

B = 2, C = 4

Op = 1 → 4 → 3 → 2 → 5

```
ListNode reverseBetween (ListNode A, int B, int C)
{
    if (A==null) return null;
    if (B==C) return A;
    ListNode head = A, prev = null;
    int count = 1;

    while (head != null && count < B)
    {
        prev = head;
        head = head.next;
        count++;
    }
    if (prev != null)
        prev.next = reverse (head, C-B+1);
    else
        return reverse (head, C-B+1);
    return A;
}
```

Page No.
Date

```
ListNode reverse (ListNode head, int B)
{
    ListNode prev = null, curr = head, next = head;
    int count = 0;
    while (next != null && count < B)
    {   count++;
        next = curr.next;
        curr.next = prev;
        prev = curr;
        curr = next;
    }
    head.next = curr;
    return prev;
}
```

1 → 2 → 3 → 4 → 5

p ↑ ↑ H

Page No.
Date

8) Longest palindromic list

Given LL. find & return the length of the longest palindrome list that exist in LL.

Expected space complexity: O(1)

ex: 1) 2→3→3→3

Op = 3 (3→3→3)

2) 2→1→2→1→2→2→1→3→2→2

Op = 5 (2→1→2→1→2)

```
public int solve( ListNode A)
{
    ListNode prev=null, curr=A, next=A;
    int result=0;
    while( next!=null)
    {   next= curr.next;
        curr.next= prev;
        //odd len
        result= math.max( result, count(prev,next)+1);
        // even len
        result=math.max(result, count(curr,next));
        prev = curr;
        curr = next;
    }
    return result;
}
```

Page No.
Date

```
int count( ListNode a, ListNode b)
{
    int cnt=0;
    while( a!=null && b!=null)
    {   if (a.val!=b.val)
            break;
        cnt+=2;
        a= a.next;
        b= b.next;
    }
    return cnt;
}
```

P C N
2 → 3 → 3 → 3

P C N
2 → 3 → 3 → 3 → res=0

Null

P C N
2 → 3 → 3 → 3 count(P,N) ≠ count(C,N) = 2

P C N
3 → 2 → 3 → 3 (P,N) 3=3, 2+1=3 (C,N) ≠ 2≠3

P C N
3 → 2 → 3 → 3

C P N
3 → 3 → 2 → 3 (P,N) 2+1=3 (C,N) 2

Scanned with Oken Scanner

Page No. Date

Sort a Linked List-

in TC → O(n log n)
SC → O(1)

Ex: LL : 3 → 4 → 2 → 8
O/P : 2 → 3 → 4 → 8

```
LN sortList (LN A)
{   if (A==null || A.next==null) return A;
    return mergesort(A);
}

LN mergesort (LN head)
{
    if (head==null || head.next==null) return head;
    LN mid = getmiddle(head);
    LN A = head;
    LN B = mid.next;
    mid.next = null;

    return LN l1 = mergesort(A);
           LN l2 = mergesort(B);

    return merge(l1, l2);
}

LN getmiddle (LN head)
{   LN slow = head, fast = head;
    if (fast.next!=null && fast.next.next!=null)
    {   slow = slow.next;
        fast = fast.next.next;
    }
    return slow;
}
```

Page No. Date

```
LN merge (ListNode l1, ListNode l2)
{
    if (l1==null) return l2;
    if (l2==null) return l1;

    LN head = new ListNode(-1);
    LN p = head;

    while (l1!=null && l2!=null)
    {
        if (l1.val < l2.val)
        {   p.next = l1;
            p = l1;
            l1 = l1.next;
        }
        else
        {   p.next = l2;
            p = l2;
            l2 = l2.next;
        }
    }
    if (l1==null) p.next = l2;
    if (l2==null) p.next = l1;

    return head.next;
}
```

Scanned with Oken Scanner

Add 2 nos as LLs

Give A & B LL representing 2 non negative nos. Digits are stored in reverse order & each node contain single digit.
Add 2 nos & return as linked list

ex: A = [2, 4, 3]
B = [5, 6, 4]
Op = [7, 0, 8]

A = [342]
B = 465
807

A = [9, 9]
B = [1]
Op = [0, 0, 1]

A = 99
B = 1
100

```
LN addtwonos(LN A, LN B)
{   ListNode head = new LN(-1);
    LN C = head;
    int carry = 0, sum = 0;
    while(A != null || B != null || carry != 0)
    {
        sum = carry;
        if(A != null)
            sum += A.val;
            A = A.next;
        }
        if(B != null)
            sum += B.val;
            B = B.next;
        }
        C.next = new LN(sum % 10);
        carry = sum / 10;
        C = C.next;
    }
    return head.next;
}
```

Clone a linked list

Given doubly LL of integers with one pointer of each node pointing to the next node (just like a singly linked list) while second ptr, can pt. to any node in the list & not just the previous node.
Create a copy of the list & return head pointer of the duplicated list.

```
ListNode CloneList(ListNode A)
{   ListNode head = A, temp = null, temp1 = null, prev = null;

    while(head != null)
    {
        temp = new ListNode(head.val);
        temp.next = head.next;
        head.next = temp;
        head = head.next.next;
    }
    ListNode x = A, y = A.next;
    while(x != null && x.next != null)
    {
        x.next.random = x.random.next;
        x = x.next.next;
    }

    ListNode res = A.next;
        y = A.next;
    while(y != null && y.next != null)
    {
        y.next = y.next.next;
        y = y.next;
    }
    return res;
}
```

Scanned with Oken Scanner