Day 16

Pairwise Swap Elements

Start from heap node and traverge a list. while traversing swap data of each node with its next node's data.

Programme :

```
Class Node :

    def __init__(self, data):
        self.data = data
        self.next = None

Class linkedlist :

    def __init__(self):
        self.head = None

    def push(self, new_data):
        new_node = Node(new_data)
        new_node.next = self.head
        self.head = new_node
```

```
def pairwiseswap(self):
    temp = self.head
    if temp is None:
        return
    while (temp is not None and
                temp.next is not None):
        If (temp.data == temp.data.next):
            temp = temp.next.next
        else:
            temp.data, temp.next.data
                = temp.next.data, temp
            temp = temp.next.next

def printlist(self):
    temp = self.head
    while (temp):
        print (temp.data),
        temp = temp.next
```

```
L = linked list()
L.push(5)
L.push(4)
L.push(3)
L.push(2)
L.push(1)
L.print list()
L.pairwise swap()
L.print list
```

Output:

1 2 3 4 5

2 1 4 3 5

Time Complexity:

O(n)

Merge two sorted list

The Strategy here uses a temporary dummy Node as the start of the result list. The pointer tail always points to the last node in the result list.

The dummy node gives tail something to point to initially when the result list is empty. This dummy node is efficient. Since it is only temporary, and it is allocated in the stack. The loop proceeds, removing one Node from either 'a' or 'b' and adding it to tail.

```
Class Node:
    def __init__(self, data):
        self.data = data
        self.next = None.
```

```
class linkedlistt:
    def __init__(self):
        self.head = None

    def added list(self, new_data):
        newNode = Node(newdata)
        if self.head is None:
            self.head = newNode
        return
        last = self.head
        while last.next:
            last = last.next
        last.next = newNode

def mergelist(headA, headB):
    dummyNode = Node(0)
    tail = dummyNode
    while True:
        If headA is None:
            tail.next = headB
            break
```

```
        if headB is None:
            tail.next = headA
            break
        if headA.data <= headB.data:
            tail.next = headA
            headA = headA.next
        else:
            tail.next = headB
            headB = headB.next

        tail = tail.next.
    return dummyNode.next
```

```
ListA = LinkedList()
ListB = LinkedList()
ListA.addtolist(5)
ListA.addtolist(10)
ListA.addtolist(15)
ListB.addtolist(2)
ListB.addtolist(3)
ListB.addtolist(20)
listA.head = mergelists(listA.head, listB.head)
listA.printlist()
```

Time Complexity :

O(n)

Output

2 3 5 10 15 20