**QuickSort on Doubly Linked List**

**Medium**

Sort the given doubly linked list of size **N** using quicksort. Just complete the partition function using the quicksort technique.

**Example 1:**

**Input:**

LinkedList: 4->2->9

**Output:**

2 4 9

**Explaination**: After sorting output will look like this.

**Example 2:**

**Input:**

LinkedList: 1->4->9->2

**Output:**

1 2 4 9

**Explaination**: After sorting output will look like this.

**Your Task:**  
Your task is to complete the given function **partition**(), which accepts the first and last node of the given linked list as input parameters and returns the pivot's address.

**Expected Time Complexity**: O(NlogN)  
**Expected Auxilliary Space**: O(1)

**Constraints:**  
1 <= N <= 2000  
1 <= Each Element of Double Linked List <= 105

**Company Tags**

[**HSBC**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=HSBC)

//{ Driver Code Starts

import java.util.\*;

import java.lang.\*;

class Node

{

int data;

Node next;

Node prev;

Node(int val)

{

data = val;

next = prev = null;

}

}

class QsortDLL

{

static Node head;

public static void addToTheLast(Node node)

{

if(head == null)

head = node;

else

{

Node temp = head;

while(temp.next != null)

temp = temp.next;

temp.next = node;

node.prev = temp;

}

}

public static void main (String[] args) {

Scanner sc = new Scanner(System.in);

int t = sc.nextInt();

while(t-- > 0)

{

int n = sc.nextInt();

int a1 = sc.nextInt();

Node head = new Node(a1);

addToTheLast(head);

for(int i = 1; i < n; i++)

{

int a = sc.nextInt();

addToTheLast(new Node(a));

}

quickSort(head);

printList(head);

System.out.println();

}

}

public static void quickSort(Node head)

{

Node h = lastNode(head);

\_quickSort(head, h);

}

public static Node lastNode(Node node)

{

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

node = node.next;

return node;

}

public static void \_quickSort(Node l, Node h)

{

CodingMaxima obj = new CodingMaxima ();

if (h != null && l != h && l != h.next)

{

Node p = obj.partition(l, h);

\_quickSort(l, p.prev);

\_quickSort(p.next, h);

}

}

public static void printList(Node node)

{

while(node != null)

{

System.out.print(node.data + " ");

node = node.next;

}

}

}

// } Driver Code Ends

class CodingMaxima

{

public static Node partition(Node l, Node h)

{

Node j=l;

Node pivot=h;

for(Node i=l ;i!=h;i=i.next){

if(pivot.data>i.data){

swap(i, j);

j=j.next;

}

}

swap(j, h);

return j;

}

public static void swap(Node a, Node b){

int temp=a.data;

a.data=b.data;

b.data=temp;

}

}