DSA PROBLEM DAY 7

PROBLEM 1:

NEXT PERMUTATION :

CODE:

import java.util.\*;

public class Main {

public static void main(String... argv) {

Scanner scan = new Scanner(System.in);

System.out.println("Enter the size of the array :");

int n = scan.nextInt();

int[] arr = new int[n];

for(int i=0;i<n;i++){

arr[i] = scan.nextInt();

}

int i = n-2;

while(i>=0 && arr[i]>=arr[i+1]){

i--;

}

if(i>=0){

int j = n-1;

while(j>=0 && arr[j]<=arr[i]){

j--;

}

swap(i,j,arr);

}

reverse(i+1,n,arr);

System.out.println("Next permutation is");

for(int i1=0;i1<n;i1++){

System.out.print(arr[i1]+" ");

}

}

public static void swap(int i,int j,int[] arr){

int temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

public static void reverse(int start,int n,int[] arr){

int end = n-1;

while(start<end){

swap(start,end,arr);

start++;

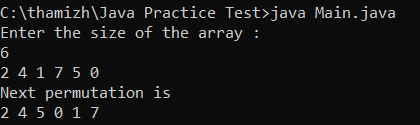
end--;

}

}

}

OUTPUT:



PROBLEM 2:

Longest Substring Without Repeating Characters

CODE:

import java.util.\*;

public class Main {

public static void main(String... argv) {

Scanner scan = new Scanner(System.in);

System.out.println("Enter the String :");

String s = scan.next();

Map<Character,Integer> map = new HashMap<>();

int max=0;

int left=0;

int right=0;

while(right<s.length()){

if(map.containsKey(s.charAt(right))){

max = Math.max(max,right-left);

left = Math.max(left,map.get(s.charAt(right))+1);

}

map.put(s.charAt(right),right);

right++;

}

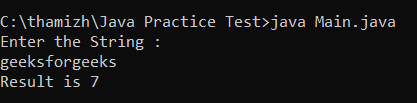
max = Math.max(max,right-left);

System.out.println("Max is " +max);

}

}

OUTPUT:



PROBLEM 3:

REMOVE LINKED LIST ELEMENTS

CODE:

class Solution {

public ListNode removeElements(ListNode head, int val) {

while(head!=null && head.val==val){

head = head.next;

}

ListNode prev = head;

ListNode temp = head;

while(temp!=null){

if(temp.val==val){

prev.next = temp.next;

temp = temp.next;

}else{

prev = temp;

temp = temp.next;

}

}

return head;

}

}

PROBLEM 4:

class Solution {

// Function to check whether the list is palindrome.

boolean isPalindrome(Node head) {

Node mid = findMid(head);

Node SecondHead = reverse(mid);

while(head!=null && SecondHead!=null){

if(head.data != SecondHead.data) return false;

head = head.next;

SecondHead = SecondHead.next;

}

return true;

}

public Node reverse(Node head){

Node prev = null;

Node temp = head;

while(temp!=null){

Node front = temp.next;

temp.next = prev;

prev = temp;

temp = front;

}

return prev;

}

public Node findMid(Node head){

Node slow = head;

Node fast = head;

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

slow = slow.next;

fast = fast.next.next;

}

return slow;

}

}

PROBLEM 5:

MINIMUM PATH SUM:

CODE:

class Solution {

public int minPathSum(int[][] grid) {

if(grid.length==0){

return 0;

}

int m = grid.length;

int n = grid[0].length;

int[][] dp = new int[m][n];

for(int i=0;i<dp.length;i++){

for(int j=0;j<dp[0].length;j++){

dp[i][j] += grid[i][j];

if(i>0 && j>0){

dp[i][j] += Math.min(dp[i][j-1],dp[i-1][j]);

}else if(i>0){

dp[i][j] += dp[i-1][j];

}else if(j>0){

dp[i][j] += dp[i][j-1];

}

}

}

return dp[dp.length-1][dp[0].length-1];

}

}

PROBLEM 6:

VALID BST :

CODE:

class Solution {

// Function to check whether a Binary Tree is BST or not.

boolean isBST(Node root) {

return helper(root,Integer.MIN\_VALUE,Integer.MAX\_VALUE);

}

public boolean helper(Node root,int min,int max){

if(root==null) return true;

if(root.data<=min || root.data>=max) return false;

return helper(root.left,min,root.data) && helper(root.right,root.data,max);

}

}

PROBLEM 7: