DSA PRACTICE PROBLEM – SET – 2

Problem 1 :

* 1. knapsack problem.

Code :

import java.util.\*;

public class Main {

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

Scanner scan = new Scanner(System.in);

System.out.println("Enter the Total Capacity :");

int W = scan.nextInt();

System.out.println("Total Capacity is : " + W );

System.out.println("Enter the Number of items :");

int n = scan.nextInt();

int[] weight = new int[n];

System.out.println("Enter the Weight of each Item :");

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

weight[i] = scan.nextInt();

}

int[] value = new int[n];

System.out.println("Enter the Value of Each Item :");

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

value[i] = scan.nextInt();

}

int prev[] = new int[W+1];

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

for(int wei = W ; wei >= 0;wei--){

int notTaken = prev[wei];

int taken = 0;

if(weight[item] <= wei){

taken = value[item] + prev[wei - weight[item]];

}

prev[wei] = Math.max(taken,notTaken);

}

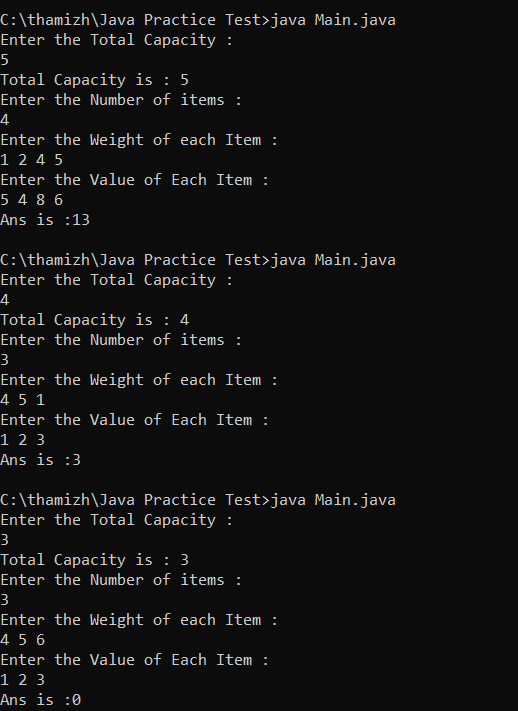
}

System.out.println("Ans is :" + prev[W]);

}

}

OUTPUT :



TIME COMPLEXITY :

PROBLEM 2 :

Floor in sorted array

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];

System.out.println("Enter the Elements in Sorted Order :");

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

arr[i] = scan.nextInt();

}

System.out.println("Enter the Target Element :");

int target = scan.nextInt();

int low = 0;

int high = n-1;

int result = -1;

boolean found = true;

if(arr[0] > target){

System.out.println("No such Element exist :" + result);

found = false;

}

while(low<high){

int mid = low + (high - low)/2;

if(arr[mid] < target){

result = arr[mid];

low = mid+1;

}else{

high = mid - 1;

}

}

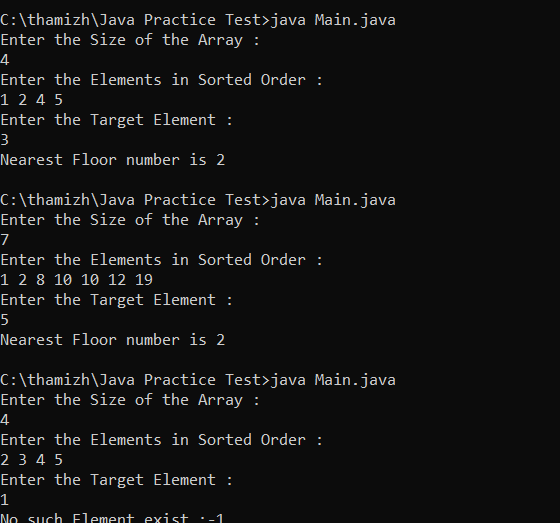
if(found)

System.out.println("Nearest Floor number is "+result);

}

}

OUTPUT :



PROBLEM 3:

Check equal arrays

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[] arr1 = new int[n];

int[] arr2 = new int[n];

System.out.println("Enter the Elements in the Array1 :");

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

arr1[i] = scan.nextInt();

}

System.out.println("Enter the Elements in the Array2 :");

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

arr2[i] = scan.nextInt();

}

Arrays.sort(arr1);

Arrays.sort(arr2);

if(Arrays.equals(arr1,arr2)){

System.out.println("Equal");

}else{

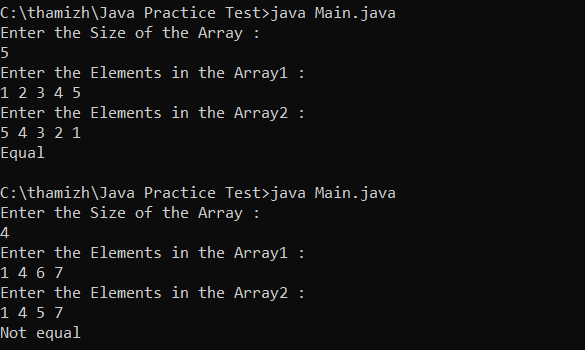
System.out.println("Not equal");

}

}

}

OUTPUT :



TIME COMPLEXITY :

PROBLEM 4:

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 LinkedList :");

int n = scan.nextInt();

System.out.println("Enter the head of the LinkedList :");

int h = scan.nextInt();

Node head = new Node(h);

Node temp = head;

System.out.println("Enter the rem node val :");

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

int num = scan.nextInt();

Node node = new Node(num);

temp.next = node;

temp = temp.next;

}

Node mid = middle(head);

Node secondHead = reverse(mid);

boolean polin = true;

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

if(head.val != secondHead.val){

polin = false;

break;

}

head = head.next;

secondHead = secondHead.next;

}

if(polin){

System.out.println("Polindrome");

}else{

System.out.println("Not a Polindrome");

}

}

public static Node middle(Node head){

Node fast = head;

Node slow = head;

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

fast = fast.next.next;

slow = slow.next;

}

return slow;

}

public static 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 class Node{

int val;

Node next;

public Node(int val){

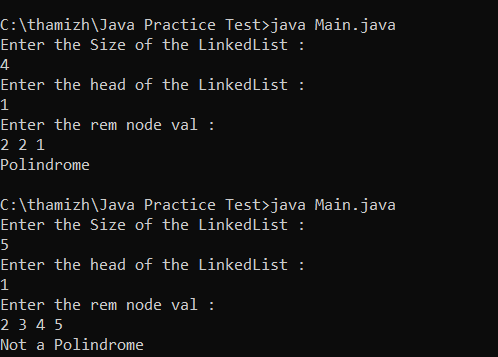
this.val = val;

next = null;

}

}

OUTPUT :



PROBLEM 5:

BALANCED TREE CHECK

CODE :

import java.util.\*;

public class Main {

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

TreeNode root = new TreeNode(1);

TreeNode node2 = new TreeNode(2);

TreeNode node3 = new TreeNode(3);

TreeNode node4 = new TreeNode(4);

TreeNode node5 = new TreeNode(5);

TreeNode node6 = new TreeNode(6);

TreeNode node7 = new TreeNode(7);

root.left = node2;

root.right = node3;

node2.left = node4;

node3.right = node5;

node5.left = node6;

node5.right = node7;

if(helper(root)!=-1){

System.out.println("BALANCED");

}else{

System.out.println("NOT BALANCED");

}

}

public static int helper(TreeNode root){

if(root==null) return 0;

int left = helper(root.left);

int right = helper(root.right);

if(left==-1 || right==-1) return -1;

if(Math.abs(left-right)==-1) return -1;

return Math.max(left,right)+1;

}

}

class TreeNode {

int val;

TreeNode left;

TreeNode right;

TreeNode(int val) {

this.val = val;

left = null;

right = null;

}

}

OUTPUT:



PROBLEM 6 :

TRIPLET SUM :

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];

System.out.println("Enter the Elements in Array :");

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

arr[i] = scan.nextInt();

}

System.out.println("Enter the Number to find the Triplet ");

int x = scan.nextInt();

Arrays.sort(arr);

boolean found = false;

for (int i = 0; i < n - 2; i++) {

int l = i + 1;

int r = n - 1;

while (l < r) {

int sum = arr[i] + arr[l] + arr[r];

if (sum == x) {

found = true;

break;

} else if (sum < x) {

l++;

} else {

r--;

}

}

}

if(found){

System.out.println("EXIST");

}else{

System.out.println("NOT EXIST");

}

}

}

OUTPUT:

