Date: 11/11/2024

Problems:

0-1 knapsack problem

Floor in sorted array

Check equal arrays

Palindrome linked list

Balanced tree check

Triplet sum in array

1. 0-1 Knapsack: (TC: O(N^2)

import java.util.\*;

class Main {

public static int knapsack(int[] wt,int[] val,int capacity){

int arr[][]=new int[wt.length+1][capacity+1];

for(int i=1;i<=wt.length;i++)

{

for(int j=0;j<=capacity;j++)

{

if(wt[i-1]<=j){

arr[i][j]=Math.max(arr[i-1][j],val[i-1]+arr[i-1][j-wt[i-1]]);

}

else{

arr[i][j]=arr[i-1][j];

}

}

}

return arr[wt.length][capacity];

}

public static void main(String[] args) {

int wt[]={4, 5, 1};

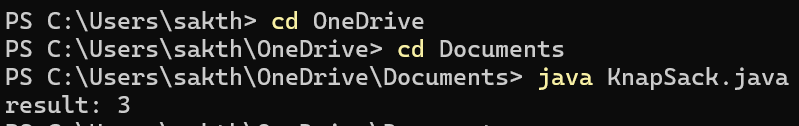
int val[]={1,2,3};

int capacity=4;

int res=knapsack(wt,val,capacity);

System.out.println("result: "+res);}}

Output:



1. Floor in sorted array:(TC: O(N))

import java.util.\*;

public class FloorFind{

public static void main(String[] args) {

int arr[]={1, 2, 8, 10, 11, 12, 19};

int k=5;

ArrayList<Integer> a=new ArrayList<>();

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

if(arr[i]<=k)

{

a.add(arr[i]);

}

}

// write code here

int max=-1;

if(a.size()==0)

{

System.out.println(-1);

}

for(int i=0;i<a.size();i++){

max=Math.max(max,a.get(i));

}

int res=0;

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

{

if(arr[i]==max)

{

res+=i;

}

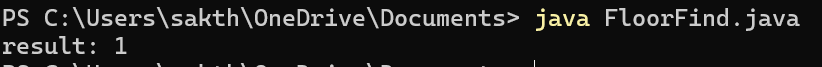
}

System.out.println("result: "+ res);

}

}

Output:



3.Check Equal Arrays(TC: O(n log n))

import java.util.\*;

public class CheckEquals {

public static void main(String[] args) {

int arr1[]={1, 2, 5, 4, 0};

int arr2[] = {2, 4, 5, 0, 1};

Arrays.sort(arr1);

Arrays.sort(arr2);

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

System.out.println("Result: "+"true");

}

else{

System.out.println("Result: " + "false");

}

}

}

Output:



4.Palindrome LinkedList(TC: O(n))

import java.util.\*;

class Node {

int data;

Node next;

Node(int data) {

this.data = data;

this.next = null;

}

}

class Main {

static boolean isPalindrome(Node head) {

// Your code here

ArrayList<Integer> a=new ArrayList<>();

if(head==null || head.next==null){

return true;

}

else{

Node cur=head;

while(cur!=null){

a.add(cur.data);

cur=cur.next;

}

}

int i=0;

int j=a.size()-1;

while(i<=j)

{

if(!a.get(i).equals(a.get(j))){

return false;

}

i++;

j--;

}

return true;

}

public static void main(String[] args) {

Node head = new Node(1);

head.next = new Node(2);

head.next.next = new Node(2);

head.next.next.next = new Node(1);

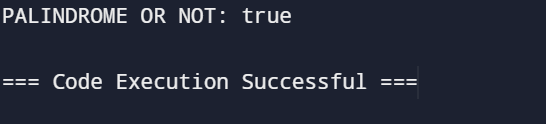
boolean result = isPalindrome(head);

System.out.println("PALINDROME OR NOT: " + result);

}

}

Output:



5.Balanced Tree Check(TC: O(n))

class Node {

int data;

Node left, right;

Node(int d) {

data = d;

left = right = null;

}

}

class BinaryTree {

Node root;

boolean isBalanced(Node node) {

return checkHeight(node) != -1;

}

int checkHeight(Node node) {

if (node == null) {

return 0;

}

int leftHeight = checkHeight(node.left);

if (leftHeight == -1) {

return -1;

}

int rightHeight = checkHeight(node.right);

if (rightHeight == -1) {

return -1;

}

if (Math.abs(leftHeight - rightHeight) > 1) {

return -1;

}

return Math.max(leftHeight, rightHeight) + 1;

}

}

public class Main{

public static void main(String args[]) {

BinaryTree bt = new BinaryTree();

bt.root = new Node(1);

bt.root.left = new Node(2);

bt.root.right = new Node(3);

bt.root.left.left = new Node(4);

bt.root.left.right = new Node(5);

bt.root.left.left.left = new Node(8);

if (bt.isBalanced(bt.root))

System.out.println("Tree is balanced");

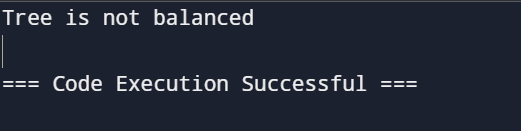
else

System.out.println("Tree is not balanced");

}

}

Output:



6. Triplet sum array: (TC: O(n log n))

import java.util.\*;

class tripletsum{

static boolean triplesum(int[] arr,int x)

{

Arrays.sort(arr);

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

int l=i+1;

int r=arr.length-1;

while(l<r){

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

if(x==sum)

{

return true;

}

else if(x>sum){

l++;

}

else{

r--;

}

}

}

return false;

}

public static void main(String[] args) {

int arr[]={1,4,45,6,10,8};

int k=13;

boolean res=triplesum(arr,k);

System.out.println("Sum Present or Not: "+ res);

}

}

Output:

