**Karthick M \_ AI&DS \_ Day-2 Practice**

**1. 0 - 1 Knapsack Problem**

**Program:**

public class Knapsack {

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

int n = val.length;

int[][] dp = new int[n + 1][capacity + 1];

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

for (int w = 1; w <= capacity; w++) {

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

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

} else {

dp[i][w] = dp[i - 1][w];

}

}

}

return dp[n][capacity];

}

public static void main(String[] args) {

int[] val1 = {1, 2, 3};

int[] wt1 = {4, 5, 1};

int capacity1 = 4;

System.out.println("Maximum value for example 1: " + knapsack(capacity1, val1, wt1));

int[] val2 = {1, 2, 3};

int[] wt2 = {4, 5, 6};

int capacity2 = 3;

System.out.println("Maximum value for example 2: " + knapsack(capacity2, val2, wt2));

int[] val3 = {10, 40, 30, 50};

int[] wt3 = {5, 4, 6, 3};

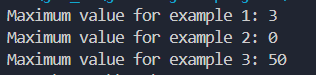
int capacity3 = 5;

System.out.println("Maximum value for example 3: " + knapsack(capacity3, val3, wt3));

}

}

**Output:**

****

**Time Complexity:** O(n×capacity)

**2.Floor in sorted array**

**Program:**

import java.util.\*;

import java.util.Scanner;

class problem1{

public static void main(String[] args) {

Scanner sc=new Scanner(System.in);

int max = Integer.MIN\_VALUE;

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

int n = sc.nextInt();

System.out.println("Enter the elements in array : ");

int[] arr = new int[n];

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

arr[i] = sc.nextInt();

}

System.out.println("Enter the floor elements : ");

int k=sc.nextInt();

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

{

if(arr[i]<=k)

{

if(arr[i]>=max)

{

max=arr[i];

}

}

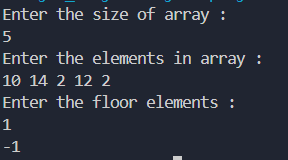
}

System.out.println(max);

}

}

**Output:**

****

**Time Complexity :** O(n)

**4. Palindrome Linked List:**

**Program:**

import java.util.ArrayList;

import java.util.Scanner;

class ListNode {

int val;

ListNode next;

ListNode() {}

ListNode(int val) { this.val = val; }

ListNode(int val, ListNode next) { this.val = val; this.next = next; }

}

class PalindromeLinkedList {

public boolean isPalindrome(ListNode head) {

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

ListNode temp = head;

while (temp != null) {

list.add(temp.val);

temp = temp.next;

}

int l = 0, r = list.size() - 1;

while (l <= r) {

if (!list.get(l).equals(list.get(r))) return false;

l++;

r--;

}

return true;

}

// Method to create a linked list from user input

public ListNode createLinkedList(int[] values) {

if (values.length == 0) return null;

ListNode head = new ListNode(values[0]);

ListNode current = head;

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

current.next = new ListNode(values[i]);

current = current.next;

}

return head;

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Get user input for the linked list

System.out.print("Enter the number of elements in the linked list: ");

int n = scanner.nextInt();

int[] values = new int[n];

System.out.println("Enter the elements of the linked list:");

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

values[i] = scanner.nextInt();

}

PalindromeLinkedList solution = new PalindromeLinkedList();

ListNode head = solution.createLinkedList(values);

// Check if the linked list is a palindrome

boolean isPalindrome = solution.isPalindrome(head);

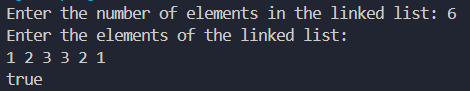
System.out.println(isPalindrome);

scanner.close();

}

}

**Output:**

****

**Time Complexity:** O(n)

**3. Check equal arrays:**

**Program:**

import java.util.HashMap;

import java.util.Scanner;

public class equalarray {

public static boolean areArraysEqual(int[] arr1, int[] arr2) {

if (arr1.length != arr2.length) {

return false;

}

HashMap<Integer, Integer> frequencyMap1 = new HashMap<>();

HashMap<Integer, Integer> frequencyMap2 = new HashMap<>();

for (int num : arr1) {

frequencyMap1.put(num, frequencyMap1.getOrDefault(num, 0) + 1);

}

for (int num : arr2) {

frequencyMap2.put(num, frequencyMap2.getOrDefault(num, 0) + 1);

}

return frequencyMap1.equals(frequencyMap2);

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the size of the arrays: ");

int size = scanner.nextInt();

int[] arr1 = new int[size];

int[] arr2 = new int[size];

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

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

arr1[i] = scanner.nextInt();

}

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

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

arr2[i] = scanner.nextInt();

}

if (areArraysEqual(arr1, arr2)) {

System.out.println("The arrays are equal.");

} else {

System.out.println("The arrays are not equal.");

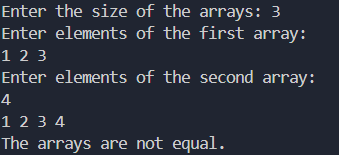
}

scanner.close();

}

}

**Output:**

****

**Time Complexity:** O(n)

**5. Balance Tree Check**

**Program:**

class Node {

int data;

Node left, right;

Node(int d) {

data = d;

left = right = null;

}

}

class Solution {

public int isBalanced(Node root) {

if (checkHeight(root) == -1) {

return 0;

}

return 1;

}

private int checkHeight(Node root) {

if (root == null) {

return 0;

}

int leftHeight = checkHeight(root.left);

int rightHeight = checkHeight(root.right);

if (leftHeight == -1 || rightHeight == -1) {

return -1;

}

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

return -1;

}

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

}

}

public class BalancedTreeCheck {

public static void main(String[] args) {

Node root = new Node(10);

root.left = new Node(20);

root.right = new Node(30);

root.left.left = new Node(40);

root.left.right = new Node(60);

Solution solution = new Solution();

int result = solution.isBalanced(root);

System.out.println("Is the tree balanced? " + (result == 1 ? "Yes" : "No"));

}

}

**Output:**

****

**Time Complexity:** O(n)

**6.Triplet sum in array**

**Program:**

import java.util.Arrays;

import java.util.Scanner;

public class tripletSum {

public static int find3Numbers(int[] arr, int n, int x) {

Arrays.sort(arr);

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

int left = i + 1;

int right = n - 1;

while (left < right) {

int currentSum = arr[i] + arr[left] + arr[right];

if (currentSum == x) {

return 1;

} else if (currentSum < x) {

left++;

} else {

right--;

}

}

}

return 0;

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

int n = scanner.nextInt();

int[] arr = new int[n];

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

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

arr[i] = scanner.nextInt();

}

System.out.print("Enter the target sum x: ");

int x = scanner.nextInt();

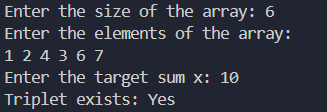
int result = find3Numbers(arr, n, x);

System.out.println("Triplet exists: " + (result == 1 ? "Yes" : "No"));

}

}

**Output:**

****

**Time Complexity:** O(n^2)