Ezhilarasan R-AI&DS-DSA-Practice-2

1)0-1 knapsack problem:

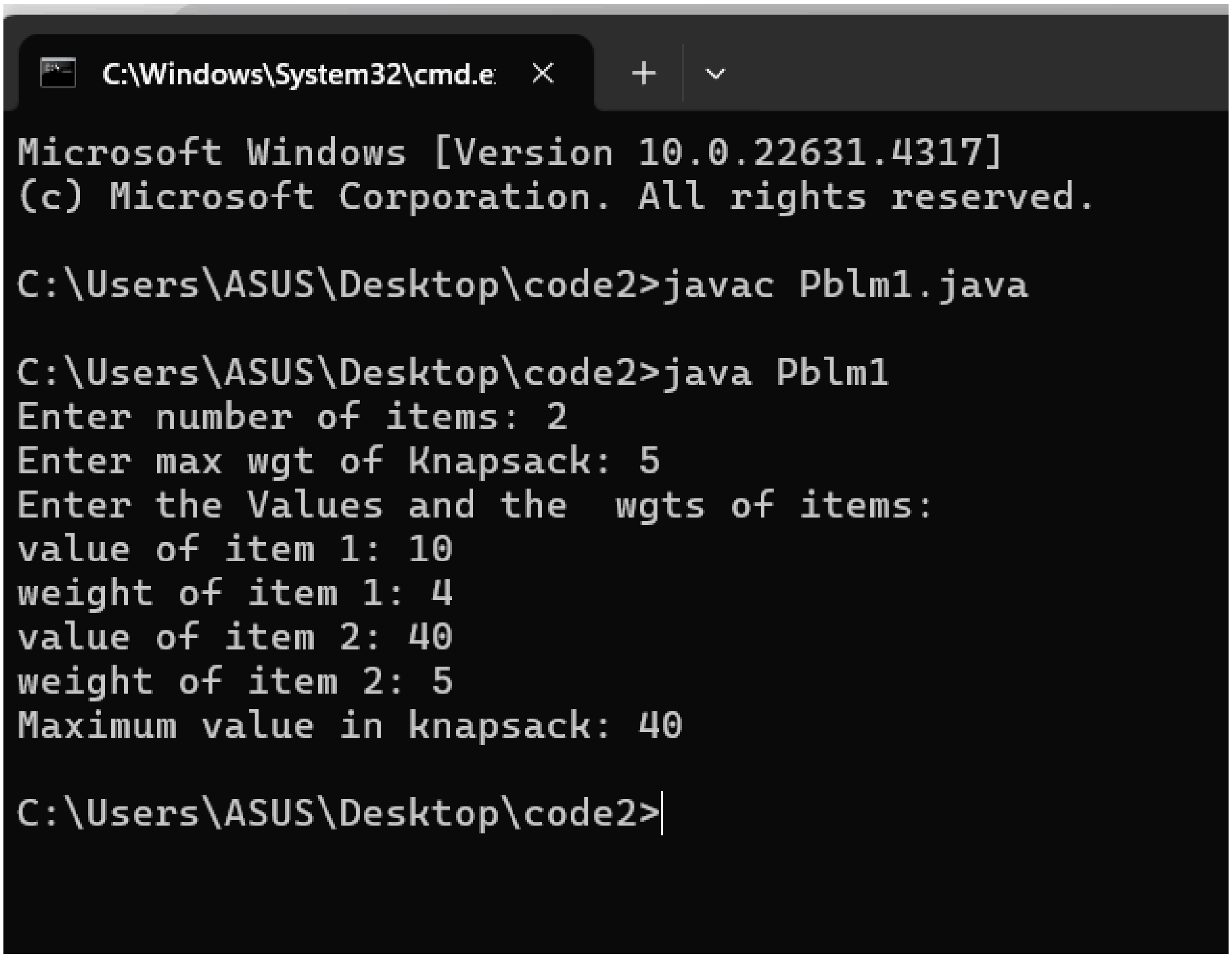
Code:

```
import java.util.Scanner;
public class Pblm1 {
public static int knapsack(int W, int[] wt, int[] val, int n) {
        int[][] dp = new int[n + 1][W + 1];
for (int i = 0; i <= n; i++) {
            for (int w = 0; w <= W; w++) {
                if (i == 0 || w == 0)
                    dp[i][w] = 0;
                else if (wt[i - 1] <= w)
                    dp[i][w] = Math.max(val[i - 1] + dp[i - 1][w - wt[i - 1]], dp[i -
1][w]);
         else
                    dp[i][w] = dp[i - 1][w];
            }
        }
return dp[n][W];
    }

    public static void main(String[] args) {
               Scanner sc = new Scanner(System.in);
        System.out.print("Enter number of items: ");
        int n = sc.nextInt();
        int[] val = new int[n];
        int[] wt = new int[n];
System.out.print("Enter max wgt of Knapsack: ");
        int W = sc.nextInt();
System.out.println("Enter the Values and the  wgts of items:");
        for (int i = 0; i < n; i++) {
            System.out.print("value of item " + (i + 1) + ": ");
            val[i] = sc.nextInt();
            System.out.print("weight of item " + (i + 1) + ": ");
```

```
          wt[i] = sc.nextInt();
 }
int maxProfit = knapsack(W, wt, val, n);
    System.out.println("Maximum value in knapsack: " + maxProfit);
             sc.close(); }}
```

Output:



```
Microsoft Windows [Version 10.0.22631.4317]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ASUS\Desktop\code2>javac Pblm1.java

C:\Users\ASUS\Desktop\code2>java Pblm1
Enter number of items: 2
Enter max wgt of Knapsack: 5
Enter the Values and the  wgts of items:
value of item 1: 10
weight of item 1: 4
value of item 2: 40
weight of item 2: 5
Maximum value in knapsack: 40

C:\Users\ASUS\Desktop\code2>
```

Time complexity: O(n * W)

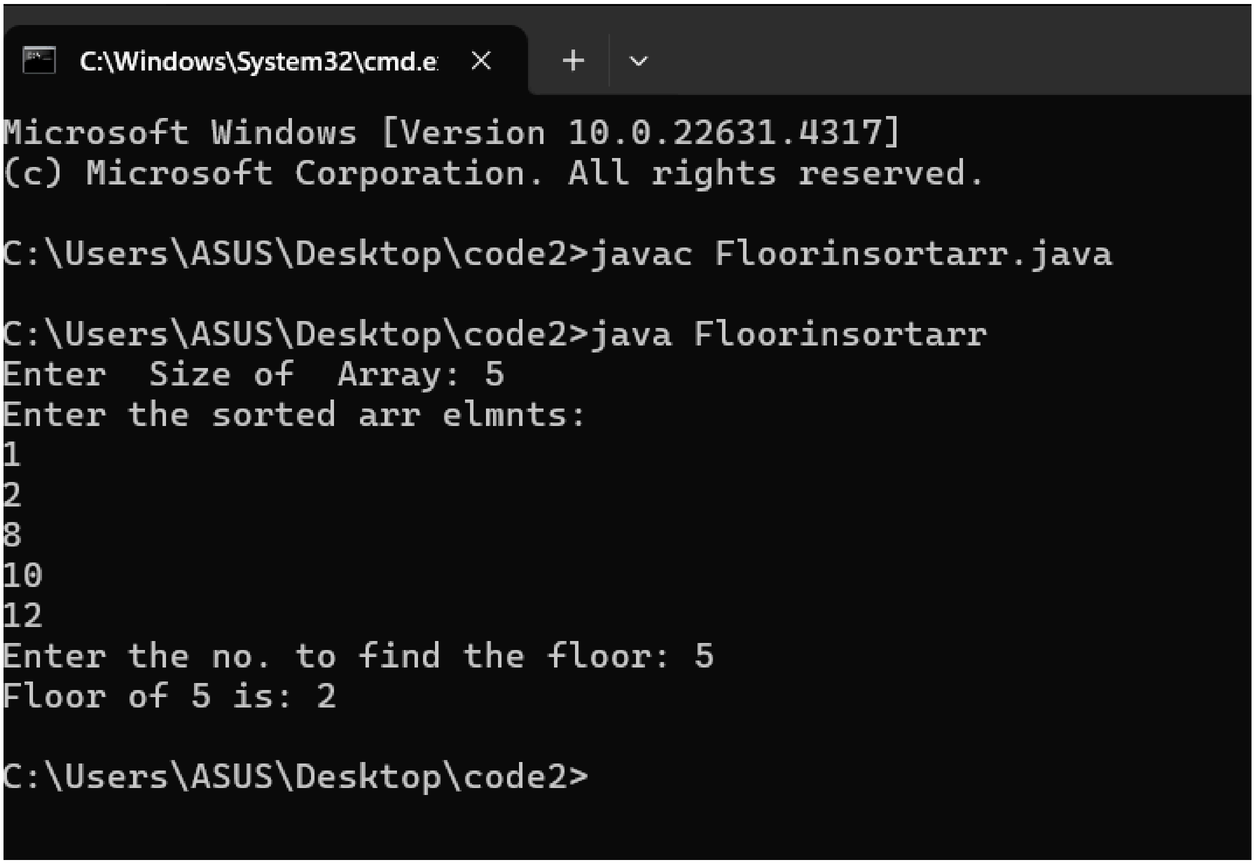
2)Floor in sorted array

Code:

```
import java.util.Scanner;
public class Floorinsortarr {
    public static int findFloor(int[] arr, int x) {
        int low = 0, high = arr.length - 1, floor = -1;
        while (low <= high) {
            int mid = low + (high - low) / 2;
            if (arr[mid] == x) return arr[mid];
            else if (arr[mid] < x) {
                floor = arr[mid];
                low = mid + 1;
            } else high = mid - 1;
        }
        return floor;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter  Size of  Array: ");
        int n = sc.nextInt();
        int[] arr = new int[n];
        System.out.println("Enter the sorted arr elmnts:");
        for (int i = 0; i < n; i++) arr[i] = sc.nextInt();
        System.out.print("Enter the no. to find the floor: ");
        int x = sc.nextInt();
        int floor = findFloor(arr, x);
        System.out.println("Floor of " + x + " is: " + floor);
        sc.close();
    }
}
```

Output:



```
Microsoft Windows [Version 10.0.22631.4317]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ASUS\Desktop\code2>javac Floorinsortarr.java

C:\Users\ASUS\Desktop\code2>java Floorinsortarr
Enter  Size of  Array: 5
Enter the sorted arr elmnts:
1
2
8
10
12
Enter the no. to find the floor: 5
Floor of 5 is: 2

C:\Users\ASUS\Desktop\code2>
```

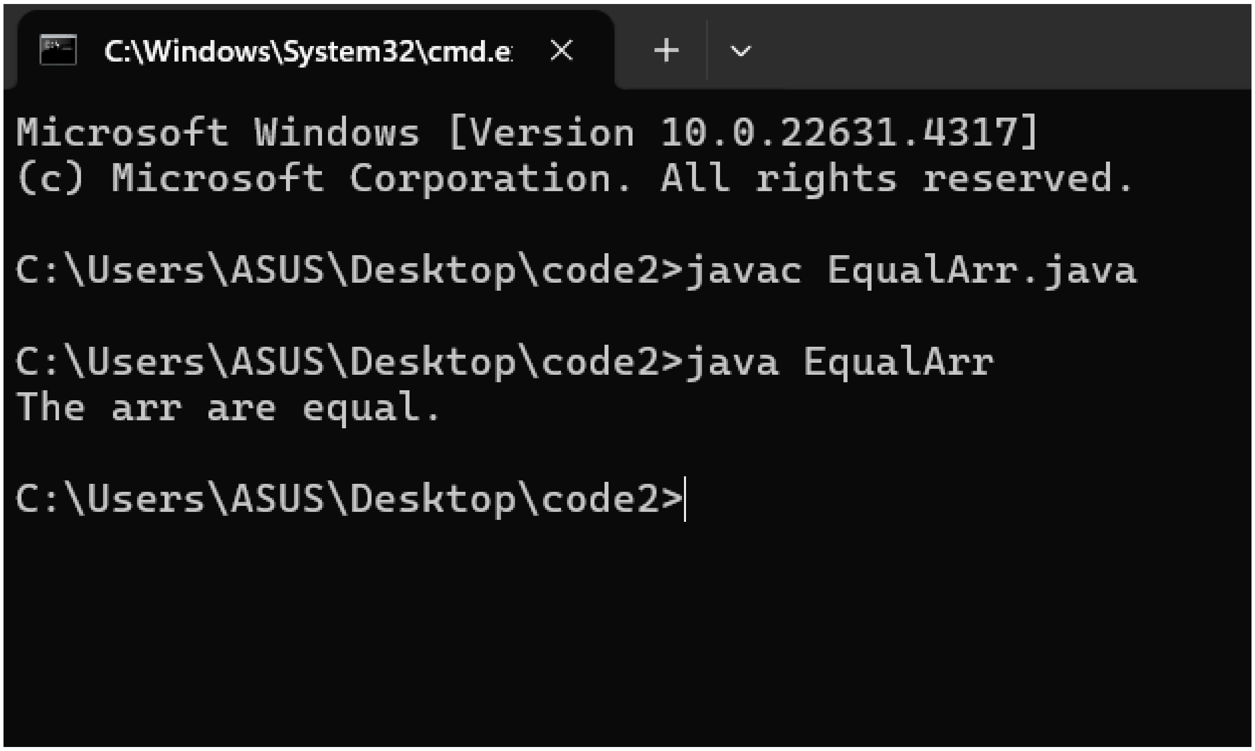
Time complexity:O(logn)

3)Check equal arrays

Code:

```
import java.util.Arrays;
public class EqualArr {
public static boolean areArraysEqual(int[] arr1, int[] arr2) {
        if (arr1.length != arr2.length) return false;
        Arrays.sort(arr1);
        Arrays.sort(arr2);
        return Arrays.equals(arr1, arr2);
    }
 public static void main(String[] args) {
        int[] arr1 = {1, 2, 3, 4};
        int[] arr2 = {4, 3, 2, 1};
        if (areArraysEqual(arr1, arr2)) {
            System.out.println("The arr are equal.");
        } else {
            System.out.println("The arr are no equal.");
        }
    }
}
```

Output:



```
Microsoft Windows [Version 10.0.22631.4317]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ASUS\Desktop\code2>javac EqualArr.java

C:\Users\ASUS\Desktop\code2>java EqualArr
The arr are equal.

C:\Users\ASUS\Desktop\code2>
```

Time complexity:O(n logn)

4)Palindrome linked list

Code:

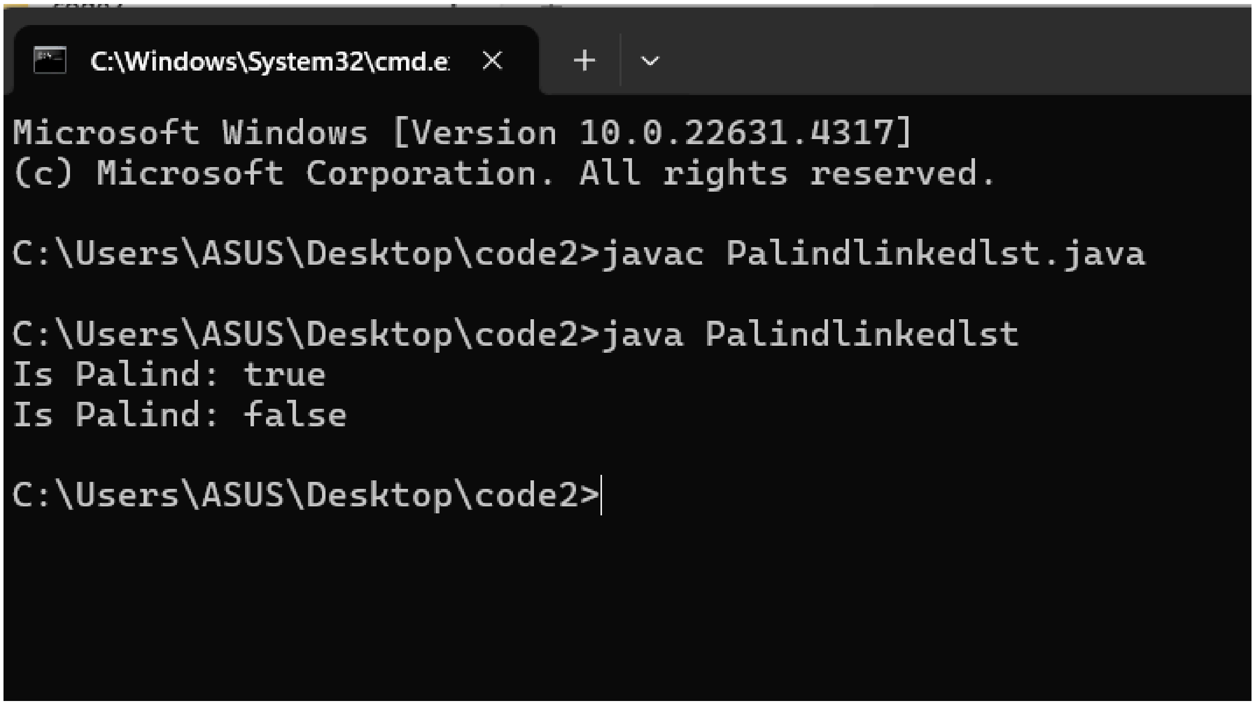
```
class ListNode{
    int val;
    ListNode next;
    ListNode(int val){this.val=val;}
```

```
}

public class Palindlinkedlst{
    public static boolean isPalindrome(ListNode head){
        if(head==null||head.next==null)return true;
        ListNode slow=head,fast=head;
        while(fast!=null&&fast.next!=null){
            slow=slow.next;
            fast=fast.next.next;
        }
        ListNode secondHalf=reverseList(slow);
        ListNode firstHalf=head;
        ListNode tempSecondHalf=secondHalf;
        while(secondHalf!=null){
            if(firstHalf.val!=secondHalf.val){
                reverseList(tempSecondHalf);
                return false;
            }
            firstHalf=firstHalf.next;
            secondHalf=secondHalf.next;
        }
        reverseList(tempSecondHalf);
        return true;
    }

    private static ListNode reverseList(ListNode head){
        ListNode prev=null;
        while(head!=null){
            ListNode nextNode=head.next;
            head.next=prev;
            prev=head;
            head=nextNode;
        }
        return prev;
    }
    public static void main(String[] args){
        ListNode head=new ListNode(1);
        head.next=new ListNode(2);
        head.next.next=new ListNode(2);
        head.next.next.next=new ListNode(1);
        System.out.println("Is Palind: "+isPalindrome(head));
        head=new ListNode(1);
        head.next=new ListNode(2);
        System.out.println("Is Palind: "+isPalindrome(head));
    }
}
```

Output:



```
Microsoft Windows [Version 10.0.22631.4317]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ASUS\Desktop\code2>javac Palindlinkedlst.java

C:\Users\ASUS\Desktop\code2>java Palindlinkedlst
Is Palind: true
Is Palind: false

C:\Users\ASUS\Desktop\code2>
```

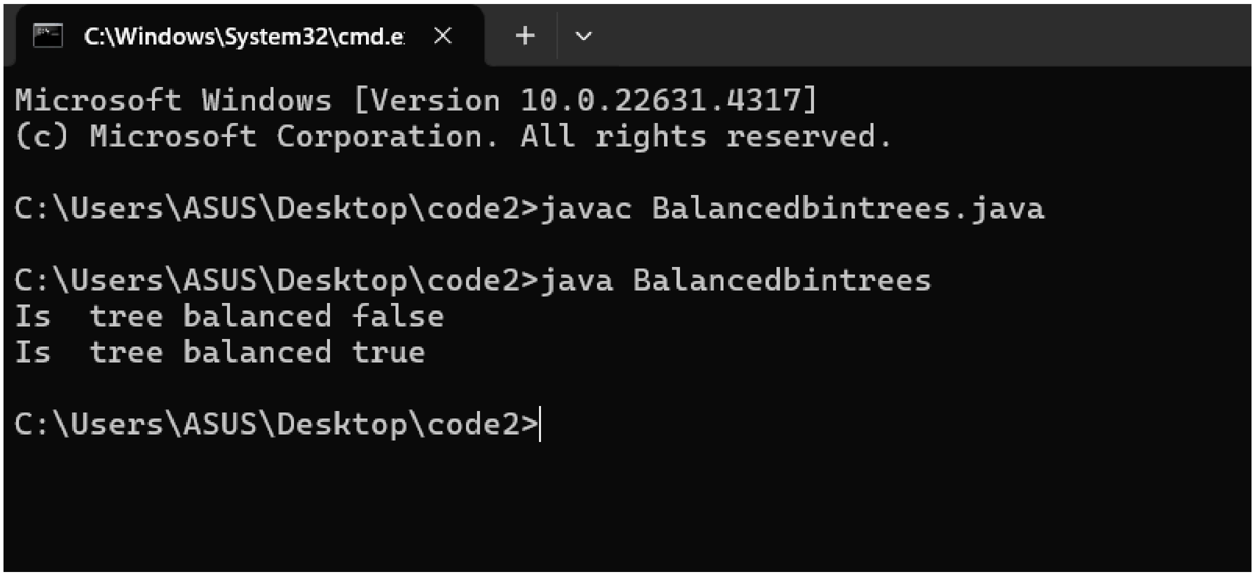
Time complexity:O(n)

5)Balanced tree check

Code:

```
class TreeNode{
    int val;
    TreeNode left,right;
    TreeNode(int val){this.val=val;}
}
public class Balancedbintrees{
    public static boolean isBalanced(TreeNode root){
        return checkBalance(root)!=-1;
    }
    private static int checkBalance(TreeNode root){
        if(root==null)return 0;
        int leftHeight=checkBalance(root.left);
        if(leftHeight==-1)return -1;
        int rightHeight=checkBalance(root.right);
        if(rightHeight==-1)return -1;
        if(Math.abs(leftHeight-rightHeight)>1)return -1;
        return Math.max(leftHeight,rightHeight)+1;
    }
    public static void main(String[] args){
        TreeNode root=new TreeNode(1);
        root.left=new TreeNode(2);
        root.right=new TreeNode(2);
        root.left.left=new TreeNode(3);
        root.left.right=new TreeNode(3);
        root.left.left.left=new TreeNode(4);
        root.left.left.right=new TreeNode(4);
        System.out.println("Is  tree balanced "+isBalanced(root));
        TreeNode balancedRoot=new TreeNode(1);
        balancedRoot.left=new TreeNode(2);
        balancedRoot.right=new TreeNode(2);
        System.out.println("Is  tree balanced "+isBalanced(balancedRoot));
    }
}
```

Output:



```
Microsoft Windows [Version 10.0.22631.4317]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ASUS\Desktop\code2>javac Balancedbintrees.java

C:\Users\ASUS\Desktop\code2>java Balancedbintrees
Is  tree balanced false
Is  tree balanced true

C:\Users\ASUS\Desktop\code2>
```

Time complexity:O(n)

6)Triplet sum in an array:

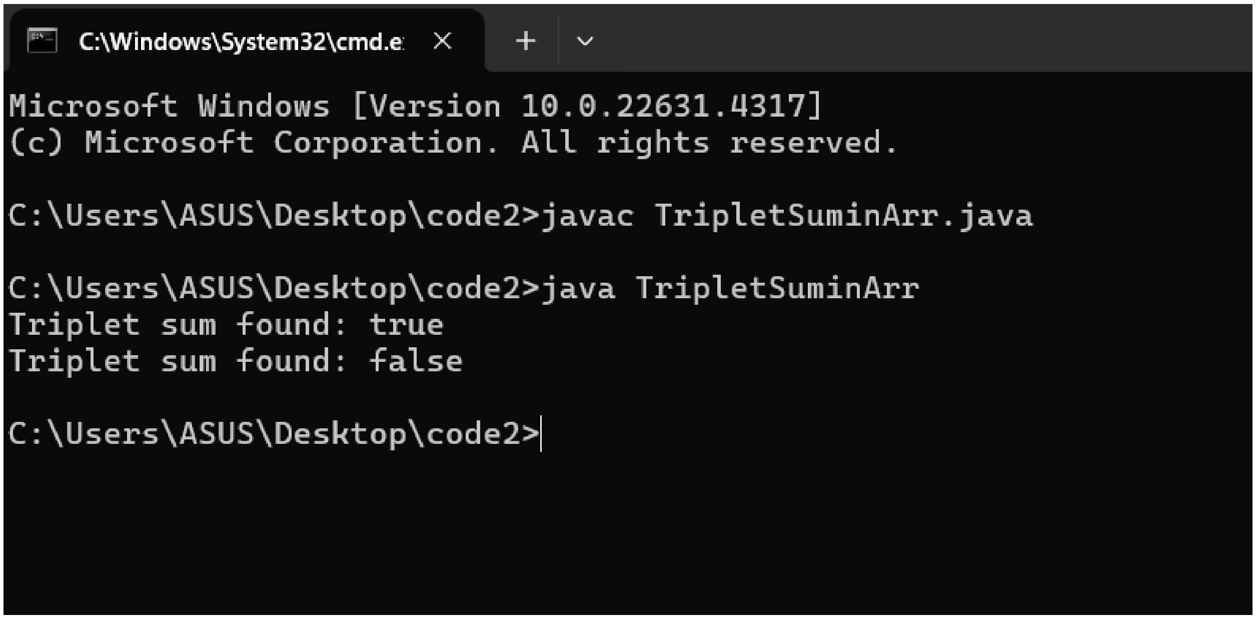
Code:

```
import java.util.Arrays;
public class TripletSuminArr {
    public static boolean findTriplet(int[] arr, int target) {
        Arrays.sort(arr);
        for (int i = 0; i < arr.length - 2; i++) {
            int left = i + 1;
            int right = arr.length - 1;
            while (left < right) {
                int sum = arr[i] + arr[left] + arr[right];
                if (sum == target) {
                    return true;
                } else if (sum < target) {
                    left++;
                } else {
                    right--;
                }
            }
        }
        return false;
    }

    public static void main(String[] args) {
        int[] arr = {12, 3, 4, 1, 6, 9};
        int target = 24;
        System.out.println("Triplet sum found: " + findTriplet(arr, target));

        int[] arr2 = {1, 2, 3, 4, 5};
        int target2 = 15;
        System.out.println("Triplet sum found: " + findTriplet(arr2, target2));
    }
}
```

Output:



```
Microsoft Windows [Version 10.0.22631.4317]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ASUS\Desktop\code2>javac TripletSuminArr.java

C:\Users\ASUS\Desktop\code2>java TripletSuminArr
Triplet sum found: true
Triplet sum found: false

C:\Users\ASUS\Desktop\code2>
```

Time complexity:O(n^2)