

Ranjith R 22IT085 Day 2 DSA Practice

1)0 - 1 Knapsack Problem

Difficulty: Medium Accuracy: 31.76% Submissions: 447K+ Points: 4

You are given the weights and values of items, and you need to put these items in a knapsack of capacity capacity to achieve the maximum total value in the knapsack. Each item is available in only one quantity.

In other words, you are given two integer arrays val[] and wt[], which represent the values and weights associated with items, respectively. You are also given an integer capacity, which represents the knapsack capacity. Your task is to find the maximum sum of values of a subset of val[] such that the sum of the weights of the corresponding subset is less than or equal to capacity. You cannot break an item; you must either pick the entire item or leave it (0-1 property).

Examples :

Input: capacity = 4, val[] = [1, 2, 3], wt[] = [4, 5, 1]

Output: 3

Explanation: Choose the last item, which weighs 1 unit and has a value of 3.

Input: capacity = 3, val[] = [1, 2, 3], wt[] = [4, 5, 6]

Output: 0

Explanation: Every item has a weight exceeding the knapsack's capacity (3).

PROGRAM:

```
package JavaPractice;

public class Array{

    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));

}

}

```

Time Complexity: $O(n \times \text{capacity})$

OUTPUT:

```

Desktop - JavaPractice/src/JavaPractice/Array.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer x
  > ArrayList1
  > BookingSystem2
  > JavaPractice
  > Main
  > TicketBooking
demo.java Practice.java Booking.java BookingSystem.java Array.java x
2 public class Array{
3
4
5 public static int knapsack(int capacity, int[] val, int[] wt) {
6     int n = val.length;
7     int[][] dp = new int[n + 1][capacity + 1];
8
9     for (int i = 1; i <= n; i++) {
10         for (int w = 1; w <= capacity; w++) {
11             if (wt[i - 1] <= w) {
12                 dp[i][w] = Math.max(dp[i - 1][w], val[i - 1] + dp[i - 1][w - wt[i - 1]]);
13             } else {
14                 dp[i][w] = dp[i - 1][w];
15             }
16         }
17     }
18     return dp[n][capacity];
19 }
20
21 public static void main(String[] args) {
22     int[] val1 = {1, 2, 3};
23     int[] wt1 = {4, 5, 1};
24     int capacity1 = 4;
25     System.out.println("Maximum value for example 1: " + knapsack(capacity1, val1, wt1));
26
27     int[] val2 = {1, 2, 3};
28     int[] wt2 = {4, 5, 6};
29     int capacity2 = 3;
30     System.out.println("Maximum value for example 2: " + knapsack(capacity2, val2, wt2));
31
32     int[] val3 = {5, 4, 6, 3};
33     int capacity3 = 5;
34     System.out.println("Maximum value for example 3: " + knapsack(capacity3, val3, wt3));
35 }
36 }
Problems Javadoc Declaration Console x
<terminated> Array [Java Application] C:\Program Files\Java\jdk-23\bin\javaw.exe (14 Nov 2024, 10:26:37 pm - 10:26:37 pm) [pid: 18208]
Maximum value for example 1: 3
Maximum value for example 2: 0
Maximum value for example 3: 50

```

2)Floor in a Sorted Array

Given a sorted array `arr[]` (with unique elements) and an integer `k`, find the index (0-based) of the largest element in `arr[]` that is less than or equal to `k`. This element is called the "floor" of `k`. If such an element does not exist, return -1.

Examples

Input: arr[] = [1, 2, 8, 10, 11, 12, 19], k = 0

Output: -1

Explanation: No element less than 0 is found. So output is -1.

Input: arr[] = [1, 2, 8, 10, 11, 12, 19], k = 5

Output: 1

Explanation: Largest Number less than 5 is 2 , whose index is 1.

Input: arr[] = [1, 2, 8], k = 1

Output: 0

Explanation: Largest Number less than or equal to 1 is 1 , whose index is 0.

PROGRAM:

```
package JavaPractice;

public class Array {

    static int findFloor(int[] arr, int k) {

        if (arr[0] > k) {

            return -1;

        } else {

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

                if (arr[i - 1] <= k && arr[i] > k) {

                    return i - 1;

                } else if (arr[i] == k) {

                    return i;

                }

            }

        }

    }

}
```

```

    }

    return arr.length - 1;
}

public static void main(String[] args) {

    int[] arr1 = {1, 2, 8, 10, 11, 12, 19};

    int k1 = 0;

    System.out.println("Output for arr1 with k=0: " +
        findFloor(arr1, k1));

    int k2 = 5;

    System.out.println("Output for arr1 with k=5: " +
        findFloor(arr1, k2));

    int k3 = 10;

    System.out.println("Output for arr1 with k=10: " +
        findFloor(arr1, k3));

}
}

```

Time Complexity:O(n)

OUTPUT:

```
File Edit Source Refactor Navigate Search Project Run Window Help
demo.java Practice.java Booking.java BookingSystem.java Array.java x
Package Explorer x
ArrayList1
BookingSystem2
JavaPractice
Main
TicketBooking

3 public class Array {
4
5     static int findFloor(int[] arr, int k) {
6         if (arr[0] > k) {
7             return -1;
8         } else {
9             for (int i = 1; i < arr.length; i++) {
10                if (arr[i - 1] <= k && arr[i] > k) {
11                    return i - 1;
12                } else if (arr[i] == k) {
13                    return i;
14                }
15            }
16        }
17        return arr.length - 1;
18    }
19
20    public static void main(String[] args) {
21        int[] arr1 = {1, 2, 8, 10, 11, 12, 19};
22        int k1 = 0;
23        System.out.println("Output for arr1 with k=0: " + findFloor(arr1, k1));
24
25        int k2 = 5;
26        System.out.println("Output for arr1 with k=5: " + findFloor(arr1, k2));
27
28        int k3 = 10;
29        System.out.println("Output for arr1 with k=10: " + findFloor(arr1, k3));
30    }
31 }
32

Problems Javadoc Declaration Console x
<terminated> Array [Java Application] C:\Program Files\Java\jdk-23\bin\javaw.exe (14 Nov 2024, 10:42:40 pm - 10:42:40 pm) [pid: 4316]
Output for arr1 with k=0: -1
Output for arr1 with k=5: 1
Output for arr1 with k=10: 3
```

Check Equal Arrays

Given two arrays arr1 and arr2 of equal size, the task is to find whether the given arrays are equal. Two arrays are said to be equal if both contain the same set of elements, arrangements (or permutations) of elements may be different though.

Note: If there are repetitions, then counts of repeated elements must also be the same for two arrays to be equal.

Examples:

Input: arr1[] = [1, 2, 5, 4, 0], arr2[] = [2, 4, 5, 0, 1]

Output: true

Explanation: Both the array can be rearranged to [0,1,2,4,5]

Input: arr1[] = [1, 2, 5], arr2[] = [2, 4, 15]

Output: false

Explanation: arr1[] and arr2[] have only one common value.

PROGRAM:

```

package JavaPractice;

import java.util.Arrays;

public class Array{

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

        Arrays.sort(arr1);
        Arrays.sort(arr2);

        for (int i = 0; i < arr1.length; i++) {
            if (arr1[i] != arr2[i]) {
                return false;
            }
        }

        return true;
    }

    public static void main(String[] args) {

        int[] arr1 = {1, 2, 5, 4, 0};
        int[] arr2 = {2, 4, 5, 0, 1};

        System.out.println("Output for arr1 and arr2: " + check(arr1,
arr2));

        int[] arr3 = {1, 2, 5};
        int[] arr4 = {2, 4, 15};

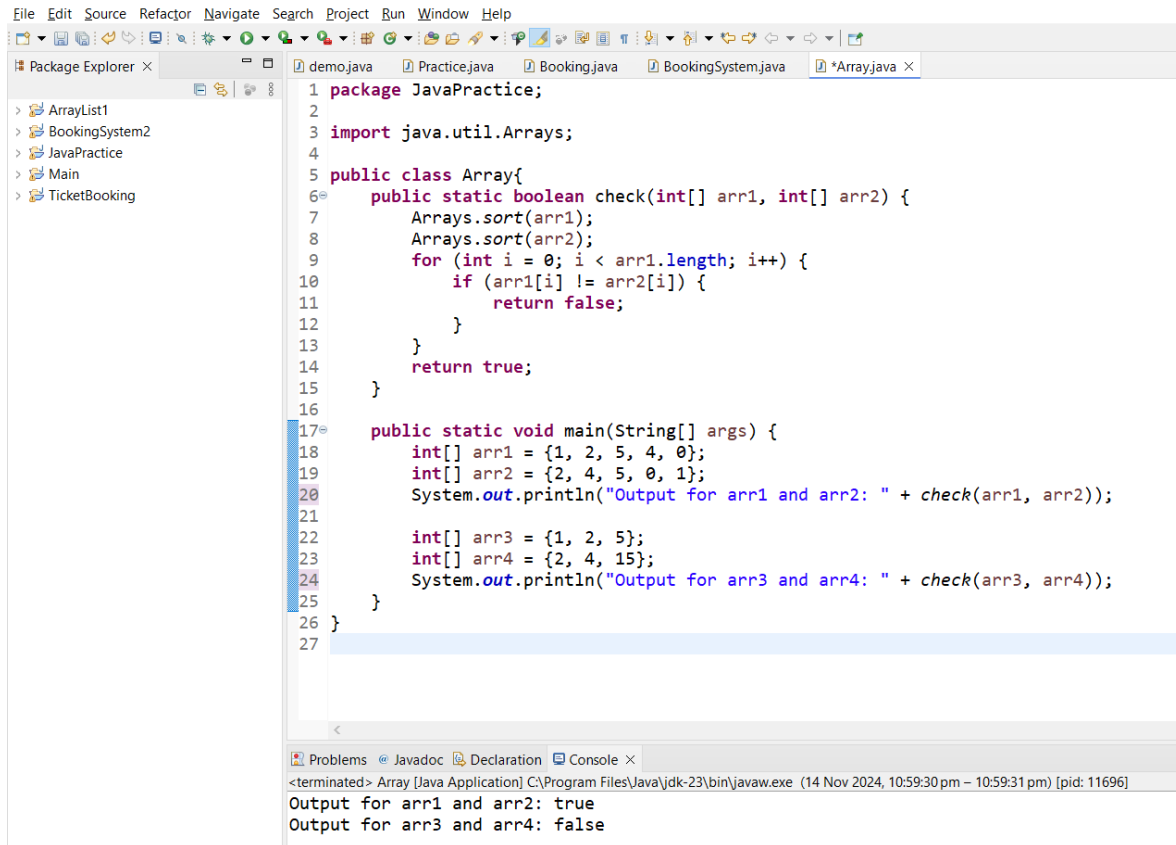
        System.out.println("Output for arr3 and arr4: " + check(arr3,
arr4));

    }
}

```

Time Complexity: $O(n \log n)$

OUTPUT:



The screenshot shows an IDE with a package explorer on the left containing 'ArrayList1', 'BookingSystem2', 'JavaPractice', 'Main', and 'TicketBooking'. The main editor displays the code for 'Array.java' within the 'JavaPractice' package. The code defines a 'check' method that sorts two integer arrays and compares them, and a 'main' method that tests this with two pairs of arrays. The console at the bottom shows the output of the program.

```
1 package JavaPractice;
2
3 import java.util.Arrays;
4
5 public class Array{
6     public static boolean check(int[] arr1, int[] arr2) {
7         Arrays.sort(arr1);
8         Arrays.sort(arr2);
9         for (int i = 0; i < arr1.length; i++) {
10             if (arr1[i] != arr2[i]) {
11                 return false;
12             }
13         }
14         return true;
15     }
16
17     public static void main(String[] args) {
18         int[] arr1 = {1, 2, 5, 4, 0};
19         int[] arr2 = {2, 4, 5, 0, 1};
20         System.out.println("Output for arr1 and arr2: " + check(arr1, arr2));
21
22         int[] arr3 = {1, 2, 5};
23         int[] arr4 = {2, 4, 15};
24         System.out.println("Output for arr3 and arr4: " + check(arr3, arr4));
25     }
26 }
27
```

Problems Javadoc Declaration Console X
<terminated> Array [Java Application] C:\Program Files\Java\jdk-23\bin\javaw.exe (14 Nov 2024, 10:59:30 pm – 10:59:31 pm) [pid: 11696]
Output for arr1 and arr2: true
Output for arr3 and arr4: false

3)Palindrome Linked List

Given a singly linked list of integers. The task is to check if the given linked list is palindrome or not.

Examples:

Input: LinkedList: 1->2->1->1->2->1

Output: true

Explanation: The given linked list is 1->2->1->1->2->1 , which is a palindrome and Hence, the output is true.

Input: LinkedList: 1->2->3->4

Output: false

Explanation: The given linked list is 1->2->3->4, which is not a palindrome and Hence, the output is false.

PROGRAM:


```
package JavaPractice;
```

```
class Node {
```

```
    int data;
```

```
    Node next;
```

```
    Node(int data) {
```

```
        this.data = data;
```

```
        this.next = null;
```

```
    }
```

```
}
```

```
public class Array{
```

```
    boolean isPalindrome(Node head) {
```

```
        Node fast = head;
```

```
        Node slow = head;
```

```
        Node prev, temp;
```

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

```
            slow = slow.next;
```

```
            fast = fast.next.next;
```

```
        }
```

```
        prev = slow;
```

```
        slow = slow.next;
```

```
        prev.next = null;
```

```

    while (slow != null) {
        temp = slow.next;
        slow.next = prev;
        prev = slow;
        slow = temp;
    }

    fast = head;
    slow = prev;

    while (slow != null) {
        if (slow.data != fast.data) {
            return false;
        }
        slow = slow.next;
        fast = fast.next;
    }

    return true;
}

static Node createLinkedList(int[] values) {
    Node head = new Node(values[0]);
    Node current = head;
    for (int i = 1; i < values.length; i++) {
        current.next = new Node(values[i]);
        current = current.next;
    }
}

```

```

        return head;
    }

    public static void main(String[] args) {
        Array solution = new Array();

        int[] values1 = {1, 2, 1, 1, 2, 1};
        Node head1 = createLinkedList(values1);
        System.out.println("Output for linked list 1: " +
            solution.isPalindrome(head1));

        int[] values2 = {1, 2, 3, 4};
        Node head2 = createLinkedList(values2);
        System.out.println("Output for linked list 2: " +
            solution.isPalindrome(head2));
    }
}

```

Time complexity: $O(n)$

OUTPUT:

The screenshot shows an IDE with a package explorer on the left containing 'ArrayList1', 'BookingSystem2', 'JavaPractice', 'Main', and 'TicketBooking'. The main editor displays the code in 'Array.java' with line numbers 16 to 45. The code implements a linked list reversal algorithm using slow, fast, and prev pointers. The console at the bottom shows the output for two linked lists: 'true' for the first and 'false' for the second.

```
16 Node slow = head;
17 Node prev, temp;
18
19 while (fast != null && fast.next != null) {
20     slow = slow.next;
21     fast = fast.next.next;
22 }
23
24 prev = slow;
25 slow = slow.next;
26 prev.next = null;
27
28 while (slow != null) {
29     temp = slow.next;
30     slow.next = prev;
31     prev = slow;
32     slow = temp;
33 }
34
35 fast = head;
36 slow = prev;
37
38 while (slow != null) {
39     if (slow.data != fast.data) {
40         return false;
41     }
42     slow = slow.next;
43     fast = fast.next;
44 }
45
```

Problems @ Javadoc Declaration Console ×
<terminated> Array [Java Application] C:\Program Files\Java\jdk-23\bin\javaw.exe (14 Nov 2024, 11:19:19 pm – 11:19:20 pm) [pid: 3840]
Output for linked list 1: true
Output for linked list 2: false

5)Balanced tree check

Given a binary tree, find if it is height balanced or not. A tree is height balanced if difference between heights of left and right subtrees is not more than one for all nodes of tree.

Examples:

Input:

1

/

2

\

3

Output: 0

Explanation: The max difference in height of left subtree and right subtree is 2, which is greater than 1. Hence unbalanced

Input:

```
    10
   /  \
  20   30
 /  \
40  60
```

Output: 1

Explanation: The max difference in height of left subtree and right subtree is 1. Hence balanced.

PROGRAM:

```
package JavaPractice;

class Node {
    int data;
    Node left, right;
    Node(int d) {
        data = d;
        left = right = null;
    }
}

class Array {
    private 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 boolean isBalanced(Node root) {
    return checkHeight(root) != -1;
}

```

```

public static void main(String[] args) {
    Array tree = new Array();

    Node root1 = new Node(1);
    root1.left = new Node(2);
    root1.left.right = new Node(3);

    System.out.println("Is tree 1 balanced? " +
tree.isBalanced(root1));

    Node root2 = new Node(10);
    root2.left = new Node(20);

```

```

        root2.right = new Node(30);

        root2.left.left = new Node(40);

        root2.left.right = new Node(60);

        System.out.println("Is tree 2 balanced? " +
tree.isBalanced(root2));

    }

}

```

Time Complexity: $O(n)$

```

File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer x demo.java Practice.java Booking.java BookingSystem.java Array.java x
> ArrayList1
> BookingSystem2
> JavaPractice
> Main
> TicketBooking
19
20     int rightHeight = checkHeight(node.right);
21     if (rightHeight == -1) return -1;
22
23     if (Math.abs(leftHeight - rightHeight) > 1) return -1;
24
25     return Math.max(leftHeight, rightHeight) + 1;
26 }
27
28 public boolean isBalanced(Node root) {
29     return checkHeight(root) != -1;
30 }
31
32 public static void main(String[] args) {
33     Array tree = new Array();
34
35     Node root1 = new Node(1);
36     root1.left = new Node(2);
37     root1.left.right = new Node(3);
38
39     System.out.println("Is tree 1 balanced? " + tree.isBalanced(root1));
40     Node root2 = new Node(10);
41     root2.left = new Node(20);
42     root2.right = new Node(30);
43     root2.left.left = new Node(40);
44     root2.left.right = new Node(60);
45
46     System.out.println("Is tree 2 balanced? " + tree.isBalanced(root2));
47 }
48 }
Problems Javadoc Declaration Console x
<terminated> Array [Java Application] C:\Program Files\Java\jdk-23\bin\javaw.exe (14 Nov 2024, 11:37:29 pm - 11:37:30 pm) [pid: 20948]
Is tree 1 balanced? false
Is tree 2 balanced? true

```

6) Triplet sum in array

Given an array arr of size n and an integer x. Find if there's a triplet in the array which sums up to the given integer x.

Examples

Input: n = 6, x = 13, arr[] = [1,4,45,6,10,8]

Output: 1

Explanation: The triplet {1, 4, 8} in the array sums up to 13.

Input: n = 6, x = 10, arr[] = [1,2,4,3,6,7]

Output: 1

Explanation: Triplets {1,3,6} & {1,2,7} in the array sum to 10.

Input: n = 6, x = 24, arr[] = [40,20,10,3,6,7]

Output: 0

Explanation: There is no triplet with sum 24.

PROGRAM:

```
package JavaPractice;

import java.util.Arrays;

class Array {

    public static boolean 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 sum = arr[i] + arr[left] + arr[right];

                if (sum == x) {

                    return true;

                }

            }

        }

    }

}
```



```

        } else if (sum < x) {
            left++;
        } else {
            right--;
        }
    }
}

return false;
}

```

```

public static void main(String[] args) {
    Array sol = new Array();

    int arr1[] = {1, 4, 45, 6, 10, 8};
    int x1 = 13;
    System.out.println(Array.find3Numbers(arr1, arr1.length, x1));

    int arr2[] = {1, 2, 4, 3, 6, 7};
    int x2 = 10;
    System.out.println(Array.find3Numbers(arr2, arr2.length, x2));

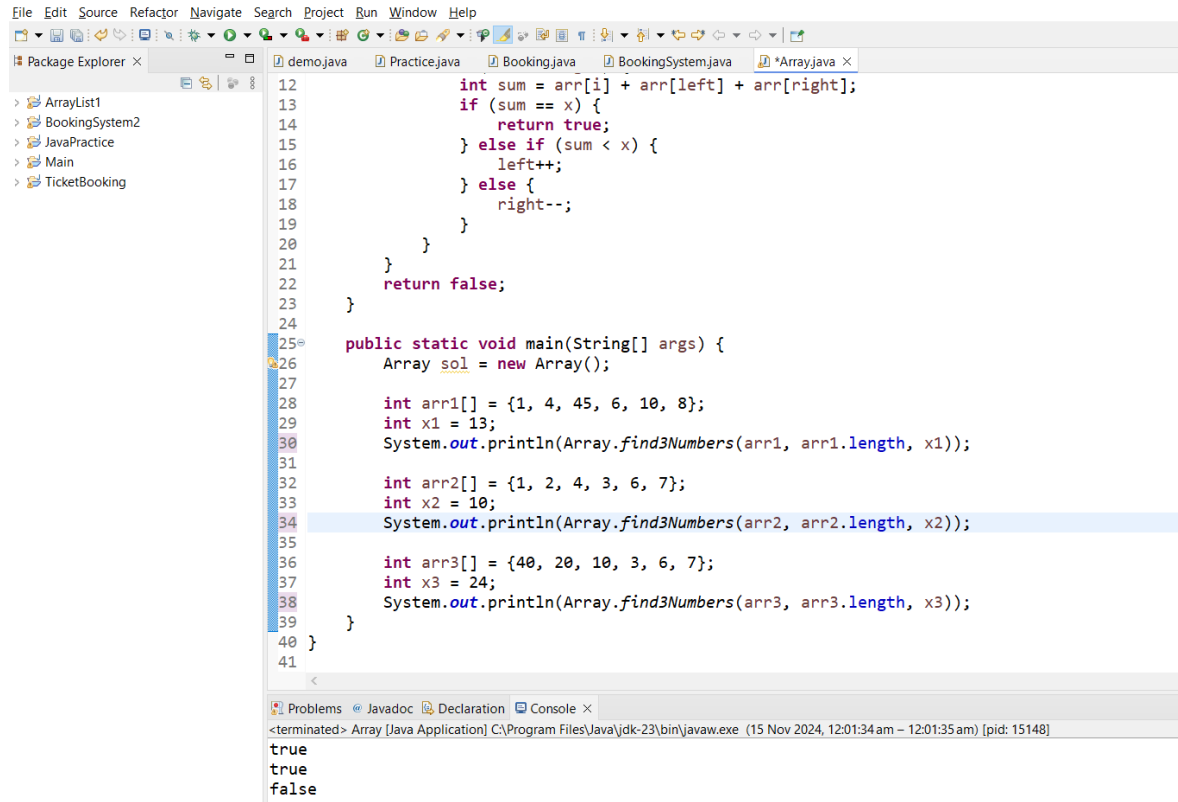
    int arr3[] = {40, 20, 10, 3, 6, 7};
    int x3 = 24;
    System.out.println(Array.find3Numbers(arr3, arr3.length, x3));
}

```

}

Time Complexity: $O(n^2)$

OUTPUT:



The screenshot shows an IDE with a Java project. The Package Explorer on the left lists: ArrayList1, BookingSystem2, JavaPractice, Main, and TicketBooking. The main editor displays the code for `*Array.java`. The code defines a `find3Numbers` method that checks if three elements in an array sum up to a target `x`. It uses a two-pointer approach with `left` and `right` indices. The `main` method tests this with three arrays: `[1, 4, 45, 6, 10, 8]` (target 13), `[1, 2, 4, 3, 6, 7]` (target 10), and `[40, 20, 10, 3, 6, 7]` (target 24). The console at the bottom shows the output: `true`, `true`, and `false`.

```
12         int sum = arr[i] + arr[left] + arr[right];
13         if (sum == x) {
14             return true;
15         } else if (sum < x) {
16             left++;
17         } else {
18             right--;
19         }
20     }
21 }
22 return false;
23 }
24
25 public static void main(String[] args) {
26     Array sol = new Array();
27
28     int arr1[] = {1, 4, 45, 6, 10, 8};
29     int x1 = 13;
30     System.out.println(Array.find3Numbers(arr1, arr1.length, x1));
31
32     int arr2[] = {1, 2, 4, 3, 6, 7};
33     int x2 = 10;
34     System.out.println(Array.find3Numbers(arr2, arr2.length, x2));
35
36     int arr3[] = {40, 20, 10, 3, 6, 7};
37     int x3 = 24;
38     System.out.println(Array.find3Numbers(arr3, arr3.length, x3));
39 }
40 }
41
```

Problems Javadoc Declaration Console ×
<terminated> Array [Java Application] C:\Program Files\Java\jdk-23\bin\javaw.exe (15 Nov 2024, 12:01:34 am – 12:01:35 am) [pid: 15148]
true
true
false