# L22

Tuesday, January 25, 2022 3:12 PM

# Arrays - Problem solving – 2

# Saurav Hathi

https://www.youtube.com/channel/UCp6MFWao5vWRnyRCxBsKnfw

Q1. Write a class ElementCheckInEveryPair with a **public** method checkElement that takes two parameters one is arr of type int[] and second one is arg of type int and returns true if every pair of arr contains at least one arg.

Assumptions:

1. arr is never null

These are examples for your understanding:

```
Enter no of elements in the array:
6
Enter elements in the array seperated by space:
52 2 3 2 65 2
Enter the search element:
2
true
Enter no of elements in the array:
6
Enter elements in the array seperated by space:
4 5 4 1 1 4
Enter the search element:
4
false
```

**Test Case 1:**

**Expected Output:**

```
Enter no of elements in the array:
7
Enter elements in the array seperated by space:
1 2 3 2 2 4 2
Enter the search element:
2
true
```

**Test Case 2:**

**Expected Output:**

```
Enter no of elements in the array:
6
Enter elements in the array seperated by space:
1 2 2 1 1 2
Enter the search element:
2
false
```

**Test Case 3:**

**Expected Output:**

```
Enter no of elements in the array:
6
Enter elements in the array seperated by space:
52 2 3 2 65 2
Enter the search element:
2
true
```

**Test Case 4:**

**Expected Output:**

```
Enter no of elements in the array:
6
Enter elements in the array seperated by space:
4 5 4 1 1 4
Enter the search element:
4
false
```

q11082/ElementCheckInE | q11082/ElementCheckInE

```java

package q11082;

public class ElementCheckInEveryPair {
    /**
     * Find every pair of array contains atleast one arg
     *
     *
     *
     *
     *@return result
     */

    public boolean checkElement(int[] arr, int arg) {
        //Write your code here
        int size = arr.length;
        boolean result = false;

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

            if(arr[i]==arg || arr[i+1]==arg){

                result=true;

            } else if(arr[i]!=arg || arr[i+1]!=arg){

                result=false;
                break;
            }
        }
        return result;

    }
}
```

Q2. Write a class CompareArrays with a **public** method compare that takes two parameters arr1 and arr2 of type int[] find the difference between the corresponding arr1 and arr2 elements and returns the count of elements whose difference is less than 2 and greater than -2. The return type of compare should be int.

Assumptions:

1. arr1 and arr2 never will null
2. arr1 and arr2 have same length

Here is an example:

Enter no of elements in the arr1:
3
Enter elements in the arr1 seperated by space:
1 2 3
Enter no of elements in the arr2:
3
Enter elements in the arr2 seperated by space:
2 3 4
3

q11087/CompareArrays.ja | q11087/CompareArraysM

```java

package q11087;

public class CompareArrays {
    /**
     * Compute the difference between two arrays
     *
     * Find the count of elements which have difference less than 2 and greater than -2
     *
     *
     *
     * @return count
     */

    public int compare(int[] arr1, int[] arr2) {
        //Write your code here

        int count=0;
        for (int i = 0, j = 0; i <arr1.length; i++, j++){

            if(arr1[i]-arr2[j] > -2 && arr1[i]-arr2[j] < 2){

                count++;
            }
        }

        return count;


    }
}
```

Q3. Write a class CheckSurroundedElement with a **public** method checkElement that takes one parameter arr of type int[] and print all the elements in the arr that are surrounded by left and right elements and not equal to the left and right elements.

Here is an example:

Enter·no·of·elements·in·the·arr1:

```
5
Enter elements in the arr1 seperated by space:
1 2 1 1 3
2
```

q11088/CheckSurrounded | q11088/CheckSurrounded

```java
package q11088;

public class CheckSurroundedElement {
    /**
     * Find the elements in the array surrounded by left and right and not equal to the left and right elements
     *
     *
     *
     * @print the result
     */

    public void checkElement(int[] arr) {
        //Write your code here


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

            if(arr[i-1] != arr[i] && arr[i+1] != arr[i]){

                System.out.println(arr[i]);
            }
        }

    }
}
```

Q4. Write a class ReorderArray with a **public** method reorder that takes one parameter arr of type int[] and returns the arr such that all zeros should come in front of the arr.

Assumptions:

1. arr is never null

Here is an example:

```
Enter no of elements in the arr:
8
Enter elements in the arr seperated by space:
55 0 21 0 63 0 45 0
0
0
0
55
21
63
45
```

q11089/ReorderArray.java | q11089/ReorderArrayMain

```java
package q11089;

public class ReorderArray {
    /**
     *Arrange all the zeros should come infront of the array
     *
     *
     *
     *@return array
     *
     */

    public int[] reorder(int[] arr) {
        //Write your code here

        int size = arr.length;
        int i = size-1, j=size-1;

        while(i >= 0){
            if(arr[i] != 0){
                arr[j] = arr[i];
                j--;
            }
            i--;
        }

        while(j >= 0){
            arr[j] = 0;
            j--;
        }

        return arr;

    }
}
```

Q5. Write a class ReorderArray with a **public** method reorder that takes one parameter arr of type int[] and returns the arr such that all even numbers in the array come to the front of the arr.

Assumptions:

1. arr is never null

Here is an example:

Enter no of elements in the array:
6
Enter elements in the array seperated by space:
3 5 6 4 2 4
6
4
2
4
3
5

q11090/ReorderArray.java q11090/ReorderArrayMai

```java
package q11090;

public class ReorderArray {
    /**
     *Arrange all even numbers to infront of the array
     *
     *
     *
     *
     * @return result
     */

    public int[] reorder(int[] arr) {

        //Write your code here
            int size = arr.length;
            int temp=0,count=0;

            for(int i = 0; i < size; i++){
                if(arr[i]%2==0){

                    for(int j=i; j > count; j--){
                        temp = arr[j-1];
                        arr[j-1] = arr[j];
                        arr[j] = temp;
                    }
                    count++;
                }
            }

            return arr;
    }
}
```

Q6. Write a class MultiplesInArray with a **public** method findMultiples that takes three parameters arr of type int[] and other two are m1 and m2 are of type int. Print all the elements in the array, but if any element in the array is a multiple of m1, print **multiple of** (actual value of m1 should be printed instead of ). If it is a multiple of m2, print **multiple of** . If it is a multiple of both m1 and m2, **print multiple of m1 and m2**.

For example:

Enter no of elements in the array:
6
Enter elements in the array separated by space:
1 2 34 5 6 7
Enter the first multiple element:
2
Enter the second multiple element:
3
1
2 is multiple of 2
34 is multiple of 2
5
6 is multiple of 2 and 3
7

q11091/MultiplesInArray.j q11091/MultiplesInArrayM

```java
package q11091;

public class MultiplesInArray{
    public static void findMultiples(int[] arr, int m1, int m2){
        int size = arr.length;

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

            if(arr[i]%m1==0 && arr[i]%m2==0){
                System.out.println(arr[i]+" is multiple of "+m1+" and "+m2);
            } else if(arr[i]%m2==0){
                System.out.println(arr[i]+" is multiple of "+m2);
            } else if(arr[i]%m1==0){
                System.out.println(arr[i]+" is multiple of "+m1);
            } else{
                System.out.println(arr[i]);
            }
        }
    }
}
```

# Arrays - Problem solving – 2

Q7. Write a class FindCenteredAverage with a **public** method findCenteredAverage that takes one parameter arr of type int[] and returns the centered average of the elements in the arr

**Hint**: exclude the biggest and smallest numbers from the array and compute the average of the remaining numbers. If there is more than one smallest value excludes only one of those. Similarly for biggest also.

Here is an example:
Enter no of elements in the array:
9
Enter elements in the array separated by space:
1 5 1 1 9 9 1 9 2
4

```java
package q11092;
public class FindCenteredAverage{
    public int findCenteredAverage(int[] arr){


        int n = arr.length;
        int temp = 0;
        for(int i=0; i < n; i++){
            for(int j=1; j < (n-i); j++){
                if(arr[j-1] > arr[j]){
                    //swap elements
                    temp = arr[j-1];
                      arr[j-1] = arr[j];
                         arr[j] = temp;
                }
            }
        }

        int sum=0;
        int count=0;

        for(int i = 1; i < arr.length-1; i++){
            sum += arr[i];
            count++;
        }

        int av = sum/count;

        return av;

    }
}
```

Q8. Write a class FindSumIgnoringSection with a **public** method findSum that takes three parameters one is arr of type int[] and other two are ignore1 and ignore2 are of type int and returns the sum of all the elements in the array, if the numbers ignore1 and ignore2, both appear in the array, ignore all the elements between them, including these two numbers.

Assumptions:
1. arr is never null
2. arr will not contain duplicate elements

Here is an example:
Enter no of elements in the array:
5
Enter elements in the array separated by space:
1 3 6 9 5
Enter the first element:
3
Enter the second element:
9
Sum of remaining elements is:
6

```java
package q11093;
public class FindSumIgnoringSection {
    /**
     * Compute the sum of all the elements in the array ignoring the elements between two ignore1 and ignore2 elements
     *
     *
     * @return sum
     *
     */
    public int findSum(int[] arr, int ignore1, int ignore2) {

        int sum = 0;
        int size = arr.length;
        boolean result=true;
        for(int i = 0; i < size; i++){
            if(arr[i] != ignore1 && result == true ){
                sum += arr[i];
            } else if(arr[i] == ignore1){
                result = false;
            } else if(arr[i] == ignore2){
                result = true;
            }
        }
        return sum;
    }
}
```

Q9. Write a class EitherOfASequence with a **public** method checkSequences that takes one parameter arr of type int[] and returns true only if one of these two sequences is present in the array: 18, 28 and 33, 36, returns false if none of these sequences are present or both are present. The return type of checkSequences is boolean.

Assumptions:

1. arr is never null

These are examples for understanding:

Enter no of elements in the array:
5
Enter elements in the array seperated by space:
18 28 36 4 2
true
Enter no of elements in the array:
6
Enter elements in the array seperated by space:
18 28 5 6 33 36
false

```java
package q11094;
public class EitherOfASequence {
    /**
     * Find the given sequences are present in the arry or not
     * @return result
     */
    public boolean checkSequences(int[] arr) {
        //Write your code here
        int size = arr.length;
        boolean found = false, found1 = false, found2 = false, found3 = false;
        for(int i=0; i<size-1; i++) {
            if(arr[i]==18) {
                found = true;
                if(arr[i+1]==28) {
                    found1 = true;
                }
            } else if(arr[i]==33){
                found2 = true;
                if(arr[i+1]==36) {
                    found3 = true;
                }
            }
        }
        boolean flag1 = found && found1;
        boolean flag2 = found2 && found3;

        if(flag1 == flag2) {
            found = false;
        } else if(flag1 || flag2) {
            found = true;
          }
          return found;

    }
}
```

Q10. Write a class SequenceOfEvens with a **public** method checkEvenSequence that takes one parameter arr of type int[] and returns true if three consecutive even numbers are present in the arr. The return type of checkEvenSequence is boolean.

Assumptions:

1. arr is never null

Here is an example:

Enter no of elements in the array:
4
Enter elements in the array seperated by space:
2 4 6 5
true

```java
package q11095;

public class SequenceOfEvens {
    /**
     * Find three consecutive even numbers are present in the array or not
     *
     *
     *
     * @return result
     */

    public void checkEvenSequence(int[] arr) {
        //Write your code here

        boolean result=false;
        int size = arr.length;

        for (int i = 0; i < size-1; i++) {
            if(arr[i] % 2 == 0) {
                if(arr[i] % 2 == 0 && arr[i+1] % 2 == 0 && arr[i+2] % 2 == 0) {
                    result = true;
                    break;
                }
            }
        }
        System.out.println(result);

    }
}
```

Q11. Write a class SymmetricalArrayCheck with a **public** method checkSymmetry that takes two parameters one is arr of type int[] and second one is n of type int and returns true if the first n numbers are same as the last n numbers in the arr.

Assumptions:

1. arr is never null

Here is an example:

Enter no of elements in the array:
8
Enter elements in the array seperated by space:
1 2 3 5 6 1 2 3
Enter the search number you want to search:
3
true

```java
package q11096;

public class SymmetricalArrayCheck {
    /**
     * Find if the first n numbers or equal to the last n numbers or not
     *
     *
     * @return result
     */

    public boolean checkSymmetry(int[] arr, int n) {
        //Write your code here

        boolean checkSymmetry = true;

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

            if(arr[i] != arr[arr.length-n+i]){

                checkSymmetry=false;
                break;
            }
        }
        return checkSymmetry;

    }
}
```

Q12. Write a class SequenceCheck with a **public** method checkSequence that takes one parameter arr of type int[] and returns true if any three consecutive elements in arr are in incremental order.

Assumptions:

1. arr is never null

Here is an example:

Enter no of elements in the array:

6
Enter elements in the array seperated by space:
1 2 3 7 4 6
true

```java
package q11097;

public class SequenceCheck {
    /**
     * Find three consecutive elements in the array are in incremental order or not
     *
     *
     *
     * @return result
     */

    public boolean checkSequence(int[] arr) {
        //Write your code here

        boolean result = false;

       for(int i = 0; i < arr.length-1; i++){
        if(arr[i]==arr[i]){
            if(arr[i+1]==1+arr[i] && arr[i+2]==2+arr[i]){
            result = true;
            break;
            } else{
                result = false;

            }
        }
       }
       return result;

    }
}
```