2D Array

Assignment Solutions

PW SKILLS

Q1: Take m and n input from the user and m * n integer inputs from user and print the following: [Easy]

number of positive numbers
number of negative numbers
number of odd numbers
number of even numbers
number of 0.

Input 1:

```
1 2 -3 4
0 0 -4 2
1 -1 2 3
-4 -5 -7 0
```

Output :
number of positive numbers = 7
number of negative numbers = 6
number of odd numbers = 7
number of even numbers = 9
number of 0 = 3

Solution:

ASS_CODE1.java

Output:

```
enter the number of rows=4
enter the number of column=3
enter the  matrix element=

-2 0 9
0 0 8
-1 -2 -9
1 1 8
Number of positives = 5
Number of negatives = 4
Number of odds = 5
Number of evens = 7
Number of zeroes = 3
```

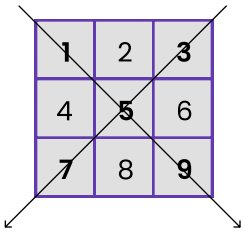
Approach:

- In this question we have simply taken the 2D matrix as input from the user and traversed the matrix and checked for every element whether it is positive, negative, zero, odd or even.
- We have used if conditions and not if-else because one element of the matrix can play multiple roles. It can be even as well as positive and there are similar other possibilities as well.

Time complexity: O(n*m) where n = number of rows , m = number of columns
Space complexity: O(1)

Q2: write a program to print the elements above the secondary diagonal in a user inputted square matrix. [Medium]

Input 1:



secondary diagonal **primary diagonal**

Output : 1 2 4

Solution:

ASS_CODE2.java

Output:

```
enter the number of rows : 3
enter the number of column : 3
enter the  matrix element :
1 2 3
4 5 6
7 8 9
Elements above secondary diagonal are as follows :
1  2  4
```

Approach:

-

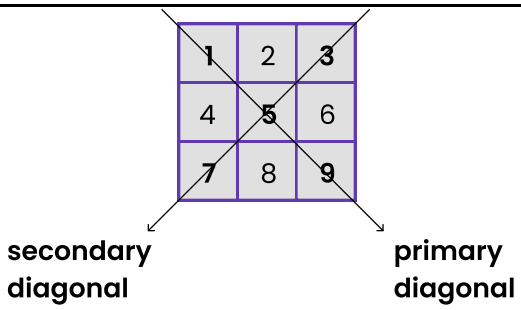
0 0	0 1	0 2	0 3
1 0	1 1	1 2	1 3
2 0	2 1	2 2	2 3
3 0	3 1	3 2	3 3

- The secondary diagonal is highlighted.
- The elements that lie above the secondary diagonal are also highlighted.
- In all these blue cells one thing is common.
- The sum of (i + j)th coordinate is always lesser than the column number - 1.
- 0 + 0 < 3 , 0 + 1 < 3 , 0 + 2 < 3 , 1 + 0 < 3 and so on.
- We have used this condition to print all those elements.

Time complexity: O(n*n) where n = number of rows
Space complexity: O(1)

Q3: write a program to print the elements of both the diagonals in a user inputted square matrix in any order. [Medium]

Input 1:



Output 1: 1 3 5 7 9

ASS_CODE3.java

Output:

```
enter the number of rows : 5
enter the number of column : 5
enter the  matrix element :
1 2 3 4 5
3 4 5 6 7
5 6 7 8 9
0 1 4 2 3
9 7 6 5 8
Elements of both the diagonals are as follows :
1  5  4  6  7  1  2  9  8
```

Approach:

- For primary diagonal as highlighted by blue color we can observe the similarities in the coordinate that the ith coordinate is always equal to the jth coordinate.
- For the secondary diagonal as highlighted by red color we can observe the similarities in the coordinate that the sum of (i + j)th coordinate is always equal to 1 less in the number of columns i.e m - 1.[it can be also treated as n 1 because the matrix is square matrix.]
-

0 0	0 1	0 2
1 0	1 1	1 2
2 0	2 1	2 2

- For an odd matrix we can see one thing that the middle element (highlighted by blue and red colors) is also an element of primary as well as secondary diagonal but it needs to be printed just once. For that we have used the if-else condition so that either the primary diagonal condition meets or the secondary diagonal condition and not both simultaneously.

Time complexity: O(n*n) where n = number of rows
Space complexity: O(1)

Optimised Approach.
As we have a square matrix, and we know this special property of Diagonal element, then for one diagonal both indices are the same and for the other the sum of both indices = m-1.

So we will optimize our solution by just traversing one time and getting both the values together.

To skip the middle element twice we will add an check extra i.e. i != m-1-i.

This condition will ensure that for the middle element as we are already printing by i,then we shouldn't print by i != m-1-i

Time complexity: O(n) where n = number of rows
Space complexity: O(1)

ASS_CODE_3_OPTIMISED.java

Q4: Write a java program to find the largest element of a given 2D array of integers. [Easy]

Input 1:

```
1 2 4 0
2 5 7 -1
4 2 6 9
```

Output : 9

Solution:

ASS_CODE4.java

Output:

```
enter the number of rows : 3
enter the number of column : 4
enter the  matrix element :
1 3 6 9
0 9 8 2
7 0 6 -1
The maximum element in this matrix is : 9
```

Approach:

- We have simply traversed the matrix and updated the "maximum" if any other element is greater than the currently set "maximum".

Time complexity: O(m*n) where m = number of rows and n = number of columns
Space complexity: O(1)

Q5: Write a function which accepts a 2D array of integers and its size as arguments and displays the elements of middle row and the elements of middle column. Printing can be done in any order. [Assuming the 2D Array to be a square matrix with odd dimensions i.e. 3x3, 5x5, 7x7 etc...] [Medium]

Input 1:

```
1 2 3 4 5
3 4 5 6 7
7 6 5 4 3
8 7 6 5 4
1 2 37 8 0
```

Output : 3 5 5 6 37 76 4 3

Solution:

ASS_CODE5.java

Output:

```
enter the number of rows :
3
enter the  matrix element :
1 2 3
4 5 6
7 8 9
The elements  of the middle row and middle column are as follows :
2  5  8  4  6
```

Approach:

- We have simply printed the elements when the value of row is fixed as “m/2” and when value of column is fixed as “m/2”.
- Since the middle element i.e “arr[m/2][m/2]” will be repeated twice but we need to print it once so we simply applied an if condition in the second for loop that if the value of column is also “m/2” then we can simply continue over this element because it has been already counted in the row’s for loop.

Time complexity: O(m) where m = number of rows and for complete code if we consider than it is O(m*m) as we are taking input in 2 for loops.
Space complexity: O(1)