

D.K.T.E. Society's Textile and Engineering Institute, Ichalkaranji.

(An Autonomous Institute, Affiliated to Shivaji University, Kolhapur)

Accredited with 'A+' Grade by NAAC

Department of Computer Science & Engineering

2020-2021



THE PROJECT REPORT ON

Flood-It

[ACM-ICPC]

Under The Guidance Of

Mr. S. J. Murchite

DEVELOPED BY:

- | | |
|--------------------------------------|-----------------|
| 1. Pranav Prasad Kulkarni | 19UCS068 |
| 2. Riteshkumar Sharad Kandane | 19UCS056 |
| 3. Ritesh Subodh Koli | 18UCS048 |

**D.K.T.E. Society's Textile and Engineering Institute,
Ichalkaranji.**

An Autonomous Institute with NAAC A+ grade

Department of Computer Science & Engineering

CERTIFICATE

This is to certify that

- | | |
|-------------------------------|----------|
| 1. Pranav Prasad Kulkarni | 19UCS068 |
| 2. Riteshkumar Sharad Kandane | 19UCS056 |
| 3. Ritesh Subodh Koli | 18UCS048 |

Have successfully completed the project work, entitled,

Flood-It

In partial fulfillment for the award of degree of Bachelor of Technology in Computer Science and Engineering. This is the record of their work carried out during academic year 2020-2021.

Date:

Place: Ichalkaranji

Mr. S.J.Murchite
[Project Guide]

[External Examiner]

Prof.Dr.D.V.Kodavade
[Head of Department]

Prof.Dr.P.V.Kadole
[Director]

DECLARATION

We the undersigned students of S.Y.C.S.E. declare that the Project work report entitled “Flood-It” written and submitted under the guidance of Mr. S. J. Murchite is our original work. The empirical findings in this report are based on the data collected by us. The matter assimilated in this report is not reproduction from any readymade report.

Date:

Place:Ichalkaranji

Name

Signature

1. Pranav Prasad Kulkarni
2. Riteshkumar Sharad Kandane
3. Ritesh Subodh Koli

INDEX

| CONTENTS | PAGE NO. |
|------------------------------|-----------------|
| 1. Abstract | 5 |
| 2. Problem Statement | 5 |
| 3. Problem Description | 6 |
| 4. Requirement Specification | 7 |
| 5. Requirement Analysis | 8 |
| 6. Problem Solution | 11 |
| 7. Flowcharts | 12 |
| 8. Snapshot | 13 |
| 9. Conclusion | 14 |
| 10. References | 15 |

ABSTRACT

Flood-it is a fun game in which we have to choose a colour from available colours which is present more frequently around left corner colour. When we choose that colour, then the left corner colour changes in the colour chosen by us. When we choose another colour which is present around the previously chosen colour in more count, the previous colour get changed in our chosen colour. So we have to fill all the blocks of table in same colour in minimum no of steps.

When all the table is complete in given steps then game is completed.

PROBLEMSTATEMENT

Find the minimum number of steps required to fill all the blocks in same colour(whole matrix in a single digit). Also the frequency of each colour or digit selected saperately.

PROBLEMDESCRIPTION

A player makes a move by choosing one of the colours. After the choice is made, all tiles that are connected to the origin are changed to the chosen colour. The game proceeds until all tiles have the same colour. The goal of the game is to change all the tiles to the same colour preferably with the Fewest number moves possible.

INPUT

The Input contents of multiple test cases. The first line of input is a single integer, not more than 20, indicating the number of test cases to follow. Each case starts with a line containing the integer n ($1 \leq n \leq 20$). The next n lines each contain n characters, giving the initial colours of the $n \times n$ board of tiles. Each colour is specified by a digit from 1 to 6.

OUTPUT

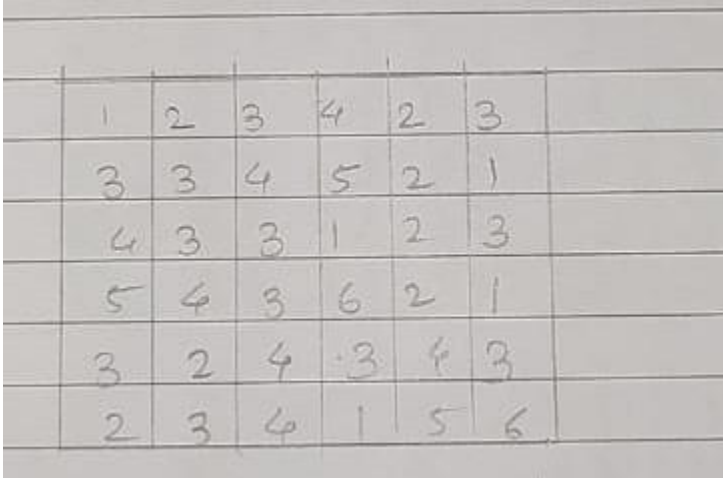
For each case, display two lines of output. The first line specifies the number of moves needed to change all the tiles to the same colour. The second line specifies 6 integers separated by a single space. The i th integer gives the number of times colour i is chosen as a move in the game.

REQUIREMENTS SPECIFICATION

1. The input must be an Integer value.
2. You start with a square of a given colour in the upper left corner.
3. You choose any colour and then the spaces near the starting square is filled with the desired colour.
4. Your goal is to flood all the squares in a given number of moves.

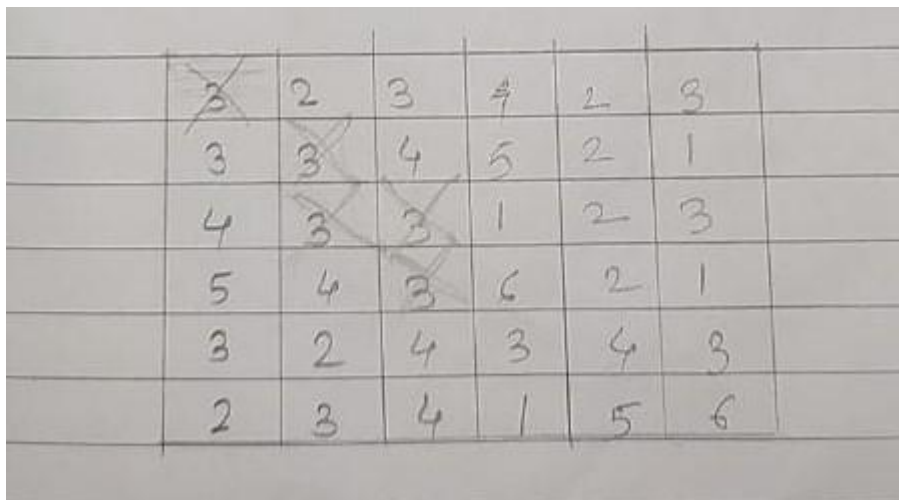
REQUIREMENT ANALYSIS

1. for each move, choose the colour that will result in the largest number of tiles connected to the origin;
2. if there is a tie, break ties by choosing the lowest numbered colour. To illustrate this, we look at the sample test case in the sample input, the original board is:



| | | | | | |
|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 2 | 3 |
| 3 | 3 | 4 | 5 | 2 | 1 |
| 4 | 3 | 3 | 1 | 2 | 3 |
| 5 | 4 | 3 | 6 | 2 | 1 |
| 3 | 2 | 4 | 3 | 4 | 3 |
| 2 | 3 | 4 | 1 | 5 | 6 |

3. If we choose colour 3 for the first move, the result will be:



| | | | | | |
|--------------|--------------|--------------|---|---|---|
| 3 | 2 | 3 | 4 | 2 | 3 |
| 3 | 3 | 4 | 5 | 2 | 1 |
| 4 | 3 | 3 | 1 | 2 | 3 |
| 5 | 4 | 3 | 6 | 2 | 1 |
| 3 | 2 | 4 | 3 | 4 | 3 |
| 2 | 3 | 4 | 1 | 5 | 6 |

4.where the tiles connected to the origin are shaded. In the next move, we choose colour 4 because we can increase the number of tiles connected to the origin by 5 tiles:

| | | | | | |
|--------------|---|---|---|---|----|
| 4 | 2 | 3 | 4 | 2 | 3 |
| 4 | 4 | 4 | 5 | 2 | 1 |
| 4 | 4 | 4 | 1 | 2 | 3 |
| 5 | 4 | 4 | 6 | 2 | 1 |
| 3 | 2 | 4 | 3 | 4 | 3 |
| 2 | 3 | 4 | 1 | 5 | 6. |

5. As we move forward the changes takes place as follows:

| | | | | | |
|--------------|--------------|--------------|--------------|---|----|
| 3 | 2 | 3 | 4 | 2 | 3 |
| 3 | 3 | 3 | 5 | 2 | 1 |
| 3 | 3 | 3 | 1 | 2 | 3 |
| 5 | 3 | 3 | 6 | 2 | 1 |
| 3 | 2 | 3 | 3 | 4 | 3 |
| 2 | 3 | 3 | 1 | 5 | 6. |

| | | | | | |
|--------------|--------------|--------------|--------------|--------------|---|
| 2 | 2 | 2 | 4 | 2 | 3 |
| 2 | 2 | 2 | 5 | 2 | 1 |
| 2 | 2 | 2 | 1 | 2 | 3 |
| 5 | 2 | 2 | 6 | 2 | 1 |
| 3 | 2 | 2 | 2 | 4 | 3 |
| 2 | 2 | 2 | 1 | 5 | 6 |

| | | | | | |
|--------------|--------------|--------------|--------------|---|---|
| 4 | 4 | 4 | 4 | 2 | 3 |
| 4 | 4 | 4 | 5 | 2 | 1 |
| 4 | 4 | 4 | 1 | 2 | 3 |
| 5 | 4 | 4 | 6 | 2 | 1 |
| 3 | 4 | 4 | 4 | 4 | 3 |
| 4 | 4 | 4 | 1 | 5 | 6 |

| | | | | | |
|--------------|--------------|--------------|--------------|--------------|---|
| 5 | 5 | 5 | 4 | 2 | 3 |
| 5 | 5 | 5 | 5 | 2 | 1 |
| 5 | 5 | 5 | 5 | 2 | 3 |
| 5 | 5 | 5 | 6 | 2 | 1 |
| 3 | 5 | 5 | 5 | 4 | 3 |
| 5 | 5 | 5 | 5 | 5 | 6 |

| | | | | | |
|---|---|---|---|---|---|
| 3 | 3 | 3 | 4 | 2 | 3 |
| 3 | 3 | 3 | 3 | 2 | 1 |
| 3 | 3 | 3 | 3 | 2 | 3 |
| 3 | 3 | 3 | 3 | 2 | 3 |
| 3 | 3 | 3 | 3 | 2 | 3 |
| 3 | 3 | 3 | 3 | 2 | 3 |

| | | | | | |
|--------------|--------------|--------------|--------------|--------------|---|
| 2 | 2 | 2 | 4 | 2 | 3 |
| 2 | 2 | 2 | 2 | 2 | 1 |
| 2 | 2 | 2 | 2 | 2 | 3 |
| 2 | 2 | 2 | 6 | 2 | 1 |
| 3 | 2 | 2 | 2 | 4 | 3 |
| 2 | 2 | 2 | 2 | 3 | 6 |

| | | | | | |
|--------------|--------------|--------------|--------------|--------------|--------------|
| 3 | 3 | 3 | 4 | 3 | 3 |
| 3 | 3 | 3 | 3 | 3 | 1 |
| 3 | 3 | 3 | 3 | 3 | 3 |
| 3 | 3 | 3 | 6 | 3 | 1 |
| 3 | 3 | 3 | 3 | 4 | 3 |
| 3 | 3 | 3 | 3 | 3 | 6 |

X X X 4 Y Y
X X X Y Y Y
X X X Y Y X
X X Y 6 X X
X X Y Y 4 3
X X X X X 6

6 6 6 6 6 6
6 6 6 6 6 6
6 6 6 6 6 6
6 6 6 6 6 6
6 6 6 6 4 3
6 6 6 6 6 6

4 4 4 4 4 4
4 4 4 4 4 4
4 4 4 4 4 4
4 4 4 4 4 4
4 4 4 4 4 4
4 4 4 4 4 4

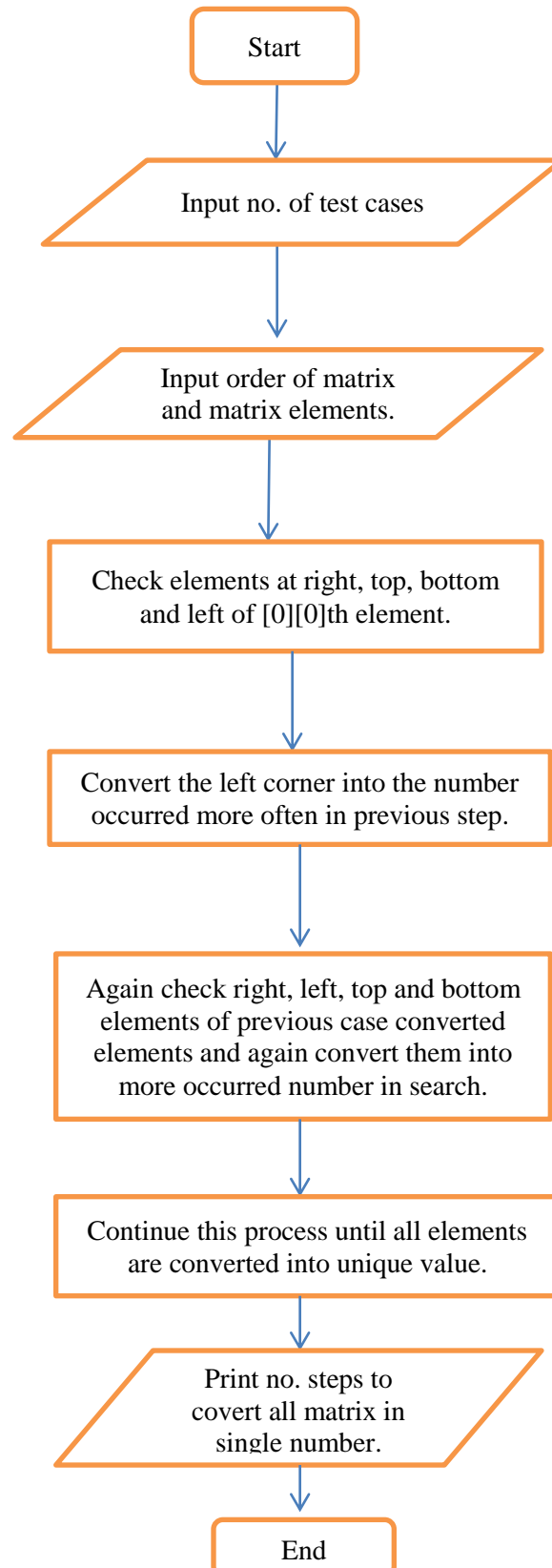
3 3 3 3 3 3
3 3 3 3 3 3
3 3 3 3 3 3
3 3 3 3 3 3
3 3 3 3 3 3
3 3 3 3 3 3

PROBLEMSOLUTION

Algorithm:

1. Start
2. Enter no of test cases and matrices.
3. Begin with first test case.
4. Check the digit at left corner. Count the digit at right and bottom side of element. Count the digit which has more frequency and also count a step.
5. In next step, check the elements at top, bottom, left and right of the linked elements in the last step.
6. Count the digit which has more frequency and also count this step in the different counter.
7. Store frequency of each element in different counters and also the total steps also in the different counter.
8. Repeat steps 4, 5, 6 & 7 until the whole matrix is covered and all steps are counted.
9. Perform the same operation for remaining all test cases.
10. Print the total steps required to flood the matrix.
11. In next line, print indisual digits taken to flood the matrix in the order.
12. End.

FLOWCHART



SNAPSHOT

INPUTS AND OUTPUTS:

```
4
6
1 2 3 4 2 3
3 3 4 5 2 1
4 3 3 1 2 3
5 4 3 6 2 1
3 2 4 3 4 3
```

```
2 3 4 1 5 6
12
2 2 4 2 1 1
```

```
5
1 2 1 2 1
2 1 2 1 2
1 2 1 2 1
2 1 2 1 2
```

```
1 2 1 2 1
8
4 4 0 0 0 0
```

```
5
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
```

```
1 2 3 4 5
4
0 1 1 1 1 0
```

```
5
1 1 1 3 1
1 2 2 1 1
3 1 3 1 1
2 1 1 1 1
```

SNAPSHOT

```
1 1 1 1 1
4
1 2 1 0 0 0
```

CONCLUSION

As an overall conclusion we have achieved the purpose of the project to find the minimum steps needed to flood all the data in a same number and total steps taken by each 1-6 numbers in order.

REFERENCES

Books:

- Let Us C- Yashavant Kanetkar

Web-links:

- <https://www.geeksforgeeks.org/c-program-find-largest-element-array/>