MULTISOURCE BFS

It is a modification of BFS. We will put ~~all~~ the all source vertices to the queue at first rather than a single vertex which was in case of standard BFS. This way Multisource

BFS will first visit all the source vertices. After that it will visit the vertices which are at a distance of 1 from all source vertices, then at a distance of 2 from all source vertices and so on and so forth.

QUESTION LINK: CODECHEF.COM/PROBLEMS/SNSOCIAL

A brief summary of question:

Suppose we have this grid (Matrix)

$$\begin{matrix} 1 & 2 & 1 & 2 \\ 1 & 1 & 1 & 2 \\ 1 & 1 & 2 & 2 \end{matrix}$$

So, in each transition each neighbouring grid will be replaced by its maximum.

e.g. $\begin{matrix} 1 & 2 & 3 \\ 1 & 3 \end{matrix} \xrightarrow{\text{1 Transition}} \begin{matrix} 3 & 3 \\ 3 & 3 \end{matrix}$

∴ We have to ~~outp~~ output in how many transitions our grid will be completely at its max. eg:

$$\begin{matrix} 1 & 2 & 1 & 2 \\ 1 & 1 & 1 & 2 \\ 1 & 1 & 2 & 2 \end{matrix} \xrightarrow{\text{1st T}} \begin{matrix} 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 \\ 1 & 2 & 2 & 2 \end{matrix} \xrightarrow{\text{2ND T}}$$

2 2 2 2
2 2 2 2
2 2 2 2

∴ This grid will take 2 transitions.

APPROACH: We will our multisource BFS in it. First we will compute the positions of all the maximum node and put them in queue in the beginning and mark their level as 0 and then run BFS.

CODE

```
CONST INT N = 1e3+10;
CONST INT INF = 1e9+10;
-> To store value (nodes) of matrix (graph)
INT VAL[N][N];
-> To store visited nodes
INT VIS[N][N];
-> To store level of each node.
INT LEV[N][N];
INT n, m;

// All possible movements array
VECTOR<PAIR<INT, INT>> MOVEMENTS = {
        {0,1}, {0,-1}, {1,0}, {-1,0},
        {1,1}, {1,-1}, {-1,1}, {-1,-1}
};
```

Scanned with CamScanner

Page No.
Date

→ A function to check if a transition is valid or not

```
BOOL ISVALID (INT i, INT j)
{
    RETURN i ≥ 0 && j ≥ 0 && i < n && j < m;
}
```

→ Multisource BFS

```
INT BFS( )
{
    → Finding maximum value in the matrix
    INT MX = 0;
    FOR (INT i=0; i<n; i++)
    {
        FOR (INT j=0; j<m; j++)
        {
            MX = MAX(MX, VAL[i][j]);
        }
    }

    QUEUE <PAIR<INT, INT>> q;
    → To find and store indices of maximum
    value (they will be our sources)
    FOR (INT i=0; i<n; i++)
    {
        FOR (INT j=0; j<m; j++)
        {
            IF (MX = VAL[i][j])
            {
                q.PUSH({i,j});
                LEV[i][j] = 0;
```

Scanned with CamScanner

Page No.
Date

```
        }
        COUT << BFS( ) << "\n";
    }
    RETURN 0;
}
```

INPUT :

```
3 -> Testcases

2 2  -> Rows x columns
1 1  } MATRIX
1 1

2 2  -> R, C
1 1
1 2

3 4  -> R, C
1 2 1 2
1 1 1 2
1 1 2 2
```

OUTPUT :

```
0
1
2
```