MICROSOFT PREPARATION

DAY : 06

0/1 Matrix :

Problem Link : https://leetcode.com/problems/01-matrix/description/

Test Cases Passed : 50 / 50

Time Used : 28.20

Difficulty Level : MEDIUM

Approach Used :

- Calculating the dimensions of grid
- Creating a visited vector
- Creating a result vector
- Creating an empty queue having pair of indexes and distances
- Inserting elements already 0 in queue to have distance 0 and mark them as visited
- Traverse until the queue becomes empty :
    - Extract first elements of the queue
    - Update the row and col of the result vector with the distance or number of steps
    - traverse for all of the adjacent elements :
        - Check for validity
            - Check if not visited and element is 1 :
                - Add element to queue with one more step and row column indexes
                - Mark it as visited
- Return the result vector

Solution :

```
vector<vector<int>> updateMatrix(vector<vector<int>>& mat) {
        // calculating the dimensions of the grid
        int n = mat.size();
        int m = mat[0].size();
        // declaring a visited vector to check if the element has been used
        vector<vector<int>> visited(n,vector<int>(m,0));
```

```
        // creating a result vector to store the resulting vector
        vector<vector<int>> result(n,vector<int>(m,0));
        // creating a queue to contain the pair of index and distance
        queue<pair<pair<int, int>,int>> q;
        // traversing for all elements and inserting the distances of 0 as 0 and
marking them visited
        for(int i=0;i<n;i++)
        {
          for(int j=0;j<m;j++)
          {
            if(mat[i][j]==0)
            {
              q.push({{i,j},0});
              visited[i][j]=1;
            }
          }
        }
        // traversing until the queue becomes empty
        while(!q.empty())
        {
          // extracting the first attributes of the row elements
          int row = q.front().first.first;
          int col = q.front().first.second;
          int steps = q.front().second;
          // pop the element of the queue
          q.pop();
          // assigning the steps to reach in result
          result[row][col] = steps;
          // traversing for all 4 adjacent directions
          int delRow[] = {1,0,-1,0};
          int delCol[] = {0,1,0,-1};

          for(int i=0;i<4;i++)
          { // calculating adjacent direction indexes
            int nrow = row+delRow[i];
            int ncol = col+delCol[i];
            // checking if indexes are valid and not visited and element is 1
            if(nrow<n && ncol<m && nrow>=0 && ncol>=0 && !visited[nrow][ncol] &&
mat[nrow][ncol]==1)
            { // insert element into queue with an incremental step and marking it
visited
              q.push({{nrow,ncol},steps+1});
              visited[nrow][ncol]=1;
            }
          }
        }
        // return the result of vector
```

```
        return result;
    }
```