Contd:

```
void dfs(int[][] m, int row, int col) {
    m[row][col] = 0;   // we are allowed to change m so clear the 1s as 0. so that they are not revisited. else use boolean matrix to track visited cells. */

    /* now explore 8 neighbors of this cell */
    for (int i=0; i< rowIdx.length; i++){
        if(isSafe(m, row + rowIdx[i], col + colIdx[i]))
            dfs(m, row + rowIdx[i], col + colIdx[i])
    }
}

int[] rowIdx = new int[]{-1,-1,-1,0,0,1,1,1}
int[] colIdx = new int[]{-1,0,1,-1,1,-1,0,1}

boolean isSafe(int[][] m, int row, int col){
// check cell is within boundary.
    return (row >=0 && row < m.length && col >=0
            && col < m[0].length && m[row][col]==1);
}
```