1.Algorithm

Each time dfs(i,j) represents the point we tried to get to (i,j) from the previous point. We first check the legality of i,j, then we see if it can go, if it can't go straight back to 0, if it can go, then check whether it was calculated and saved in mark, neither, we will go to the next step dfs

Whole program

```
matrix[][] ; given
mark[][] ; same size as matrix
dfs(i,j,pre) {
    if (i < 0 || j < 0 || i >= m || j >= n) {
        return 0;
    }
    if (matrix[i][j] <= pre && pre != inf ) {
        return 0;
    }
    if (mark[i][j] != 0) {
        return mark[i][j];
    }
    int tem = dfs(matrix, i + 1, j, matrix[i][j], mark);
    tem = max(tem,dfs(matrix, i, j + 1, matrix[i][j], mark));
    tem = max(tem,dfs(matrix, i - 1, j, matrix[i][j], mark));
    tem = max(tem,dfs(matrix, i, j - 1, matrix[i][j], mark));
    tem ++;
    mark[i][j] = tem;
    return mark[i][j];
}
Main (){
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            res = max(res, dfs(matrix, i, j, -inf, mark));
        }
    }
    return res;
}
```

2.Essential parts of code with sufficient comments

```
.ORIG X3000
... some reg init
WHILE1              ;for int i =0 ; i < n ; i++
    LD R3,NEGN
    ADD R3,R1,R3    ; I -N
    BRZ WHILE1OUT   ; for int j =0 ; j< m ; j++
        WHILE2
        LD R3,NEGM
```

```
        ADD R3,R2,R3    ; J - M
        BRZ WHILE2OUT   ; ANS = MAX (ANS , DFS(I,J,-1))
        ST  R1,K1
        ST  R5,K2
        JSR DFS         ; CALL (DFS(I,J,-1))
        LD R5,K2
        ADD R1,R5,#0    ; R1 = ANS
        JSR GETMAX      ; R0 = MAX (R1,R0)
        ADD R5,R0,#0    ; R5 = MAX
        LD R1,K1
        ADD R2,R2,#1
        BR WHILE2
    WHILE2OUT
    AND R2,R2,#0 ; R2 =j
    ADD R1,R1,#1
    BR WHILE1
WHILE1OUT
ADD R2,R5,#0
HALT
some constant used

DFS ; R0 AS OUTPUT , R1,R2 AS IJ R4 AS PRE
...Some stack push
...Some basic check if (i < 0 || j < 0 || i >= m || j >= n) return 0
ADD R3,R4,#0    ;check matrix[i][j] <= pre && pre != inf
NOT R3,R3
ADD R3,R3,#1 ; -PRE
LD  R5,INF
ADD R3,R3,R5    ; INF - PRE
BRZ SKIP        ; IF PRE == INF ,THEN SKIP COMPARE
LDR R5,R0,#0    ; R5 = MATRIX[I][J]
ADD R3,R3,R5    ;
BRNZ RETURN0    ; matrix[i][j] <= pre return 0
SKIP
LDR R5,R0,#0    ; R5 = MATRIX[I][J]
ADD R3,R0,#0    ;
LD R0,OFFSET    ;
ADD R3,R0,R3    ; R3 = &MARK[i][j]
LDR R0,R3,#0    ; R0 = MARK[I][J]
BRNP RETURN
ADD R4,R5,#0    ; let R5 be tem
ADD R1,R1,#-1
JSR DFS         ; dfs(matrix, i + 1, j, matrix[i][j], mark);
ADD R5,R0,#0
ADD R1,R1,#2    ; tem = max(tem,dfs(matrix, i, j + 1, matrix[i][j], mark));
JSR DFS
ST  R1,K3
ADD R1,R5,#0
JSR GETMAX
LD  R1,K3
ADD R5,R0,#0
... other two dfs Omit
ADD R2,R2,#-1
ADD R5,R5,#1    ; TEM++
STR R5,R3,#0    ; MARK[i][j] = TEM
```

```
BR RETURN       ; returm mark[i][j]
RETURN0
AND R0,R0,#0
BR FINISH
RETURN
JSR GETMAPIJ
LD R3,OFFSET
ADD R0,R3,R0 ;R0 = &MARK[][]
LDR R0,R0,#0 ;RETURN MARKIJ
FINISH ... some stack pop
RET
;; Function GETMAX(R0,R1)
GETMAX ;; R0 = MAX(R0,R1) detail Omit

FUNCTION GETMAPij(R1,R2,R4)
GETMAPij ;R0=&Matrix[i][j] r1 = i ,r2 = j ,A + I*M + J  ; compute MAP + R1*M + R2
, OMIT
.END
.ORIG X5000 ; here is array mark
.BLKW 2600
.END
```

3.Questions

1.Explain your algorithm briefly

see part 1.

2.What is the maximum number of layers of the stack in the recursive process

The worst case is M*N, because the operation calling dfs to press the stack is only reachable at points (i,j), and there is no corresponding mark record. Since mark is at most M*N in size, it is guaranteed that there are at most M*N points without mark records, and once all points are pushed into the stack, there must be a point that ends dfs, records mark, and exits the stack. (We can prove this by proof by contradiction. If you can call dfs after M*N points have been pushed into the stack, then this point must not be a point in the stack, and the stack already contains all mark(i,j)==0 points.)
Then, once once out of the stack, then number of points has mark(i,j)==0 is at most M*N -1, and again DFS will not exceed this value, proving the same above, so that recursion we know that the worst case is M*N