Rat maze (amt7)

locked

Submitted a few seconds ago • Score: 1.00 Status: Accepted

✔ Test Case #0	✔ Test Case #1	✔ Test Case #2
✔ Test Case #3		

Submitted Code

Language: C++ Open in editor

```
1 #include <stdio.h>
2 #include <iostream>
3 // Maze size
```

```c
#define N 4

bool solveMazeUtil(int maze[N][N], int x, int y, int sol[N][N]);

/* A utility function to print solution matrix sol[N][N] */
void printSolution(int sol[N][N])
{
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++)
            printf("%d ", sol[i][j]);
        printf("\n");
    }
}

/* A utility function to check if x, y is valid index for N*N maze */
bool isSafe(int maze[N][N], int x, int y)
{
    // if (x, y outside maze) return false
    if (x >= 0 && x < N && y >= 0 && y < N && maze[x][y] == 1)
        return true;

    return false;
}

/* This function solves the Maze problem using Backtracking.  It mainly
   uses solveMazeUtil() to solve the problem. It returns false if no
   path is possible, otherwise return true and prints the path in the
   form of 1s. Please note that there may be more than one solutions,
   this function prints one of the feasible solutions.*/
bool solveMaze(int maze[N][N])
{
    int sol[N][N] = { { 0, 0, 0, 0 },
                      { 0, 0, 0, 0 },
                      { 0, 0, 0, 0 },
                      { 0, 0, 0, 0 } };
```

```
39
40      if (solveMazeUtil(maze, 0, 0, sol) == false) {
41          printf("Solution doesn't exist");
42          return false;
43      }
44
45      printSolution(sol);
46      return true;
47  }
48
49  /* A recursive utility function to solve Maze problem */
50  bool solveMazeUtil(int maze[N][N], int x, int y, int sol[N][N])
51  {
52      // if (x, y is goal) return true
53      if (x == N - 1 && y == N - 1 && maze[x][y] == 1) {
54          sol[x][y] = 1;
55          return true;
56      }
57
58      // Check if maze[x][y] is valid
59      if (isSafe(maze, x, y) == true) {
60          // mark x, y as part of solution path
61          sol[x][y] = 1;
62
63          /* Move forward in x direction */
64          if (solveMazeUtil(maze, x + 1, y, sol) == true)
65              return true;
66
67          /* If moving in x direction doesn't give solution then
68             Move down in y direction  */
69          if (solveMazeUtil(maze, x, y + 1, sol) == true)
70              return true;
71
72          /* If none of the above movements work then BACKTRACK:
73              unmark x, y as part of solution path */
```

```
74          sol[x][y] = 0;
75          return false;
76      }
77
78      return false;
79  }
80
81  // driver program to test above function
82  int main()
83  {
84      int maze[N][N] ;
85      for(int i=0;i<4;i++)
86          for(int j=0;j<4;j++)
87              std::cin>>maze[i][j];
88
89      solveMaze(maze);
90      return 0;
91  }
```