# Maze problem

A robot is asked to navigate a maze. It is placed at a certain position (the starting position) in the maze and is asked to try to reach another position (the goal position). Positions in the maze will either be open or blocked with an obstacle. Positions are identified by (x,y) coordinates.

[Example of a simple maze] At any given moment, the robot can only move 1 step in one of 4 directions. Valid moves are:

Go North: (x,y) -> (x,y-1)

Go East: (x,y) -> (x+1,y)

Go South: (x,y) -> (x,y+1)

Go West: (x,y) -> (x-1,y)

Note that positions are specified in zero-based coordinates (i.e., 0...size-1, where size is the size of the maze in the corresponding dimension).

The robot can only move to positions without obstacles and must stay within the maze.

The robot should search for a path from the starting position to the goal position (a solution path) until it finds one or until it exhausts all possibilities. In addition, it should mark the path it finds (if any) in the maze.

## Representation

To make this problem more concrete, let's consider a maze represented by a matrix of characters. An example 6x6 input maze is:

S#####

.....#

#.####

#.####

...#.G

##...#

'.' - where the robot can move (open positions)

'#' - obstacles (blocked positions)

'S' - start position (here, x=0, y=0)

'G' - goal (here, x=5, y=4)

Aside: Remember that we are using x and y coordinates (that start at 0) for maze positions. A y coordinate therefore corresponds to a row in the matrix and an x coordinate corresponds to a column.

A path in the maze can be marked by the '+' symbol...

A path refers to either a partial path, marked while the robot is still searching:

+#####

++++.#

#.####

#.####

...#.G

##...#

(i.e., one that may or may not lead to a solution). Or, a solution path:

S#####

++...#

#+####

#+####

.++#+G

##+++#

which leads from start to goal.

## Some examples

**Maze1 :**

S...##

#.#...

#.##.#

..#.##

#...#G

#.#...

Solution :

S+xx##

#+#xxx

#+##x#

.+#x##

#+++#G

#.#+++

**Maze2:**

S...##

###...

#.##.#

..#..#

#...##

G.#...

Solution:

S+++##

###++x

#x##+#

xx#++#

#+++##

G+#xxx

> Note : 'x' in the solutions you see shows the invalid paths the algorithm taken

> while going over solution . You do not have to have the same x's in your solution.