Maze Generation Algorithm

GRAPH THEORY BASED

Using Depth-first search algorithm:

In this algorithm we traverse the whole tree starting from root and explores as far as possible along each branch and then backtrack to the root. We use stack in this method. We consider the space for maze being a large grid of cells where each cell has four cells which represents four directions which it can move.

A random cell is chosen to be starting point and then computer selects a random neighboring cell that is not yet visited. Then the computer removes the wall between the cells and the neighboring cell is now current cell. The current cell is marked as visited and adds to the stack. This process continues until there is until there is a cell with visited neighbors and then backtracks to a cell where it has unvisited neighbors and this process continues until every cell gets visited and reaches to the beginning cell.

How Recursive Backtracking works:

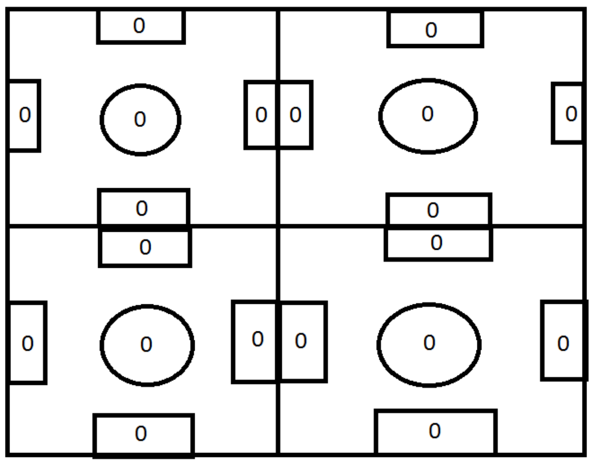
First we create a space of lists with number of columns and number of rows with each list (or cell) containing five variables. Those five variables contain the information of directions and whether the cell is visited (Left, Up, Right, Down, Visited). Then in while loop we start at an initial cell and mark it as visited and add it to a stack.

Then a neighbor is chosen randomly and its information of previous cell and new cell gets updated (e.g., if the right cell is chosen the initial cell information: (0,0,1,0,1) and the current cell information: (1,0,0,0,1)). The current cell marked as visited and is pushed to the stack. If a cell has no directions to move then it pops off the element in the stack and moves the position to that popped off cell. This backtracks every cell until the position gets back to (0, 0).

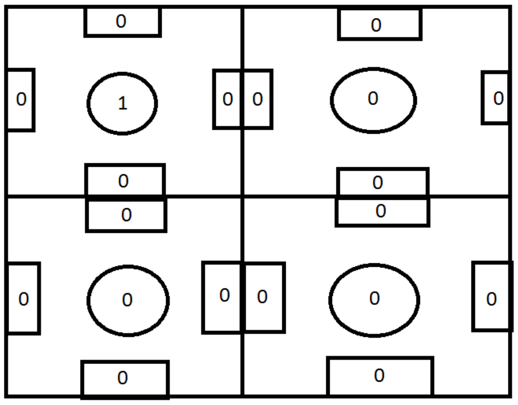
Graphical Representation:

For example we take a 2*2 lists with all its information.

Here all the sides and the center (visited) parts have the value zero in the initial case.

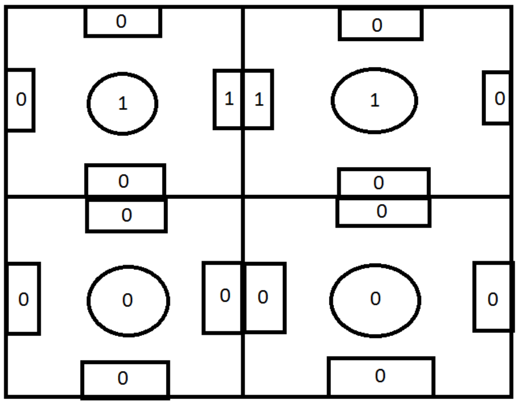


Let us consider we start at (0, 0) position. The cell gets visited therefore its value gets updated.

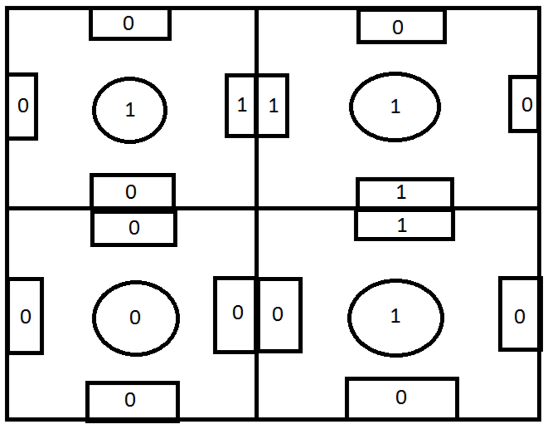


Now the computer randomly choses possible direction to traverse either right or down.

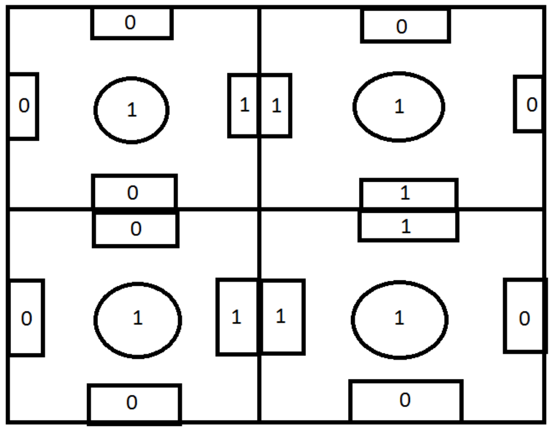
Let us consider right side. Now the right value of previous cell gets updated and left value gets updated and the present cell gets visited and the position gets added to the stack.



Now it can either go left or down. As left cell is visited it traverse to the down cell.



Now we can only traverse to left side

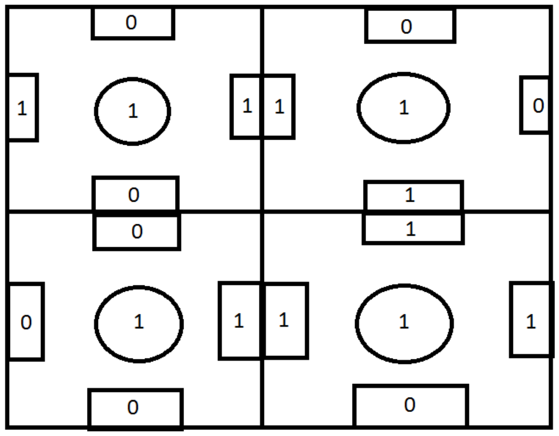


At (1, 0) it has two neighbors which are visited now we pop off the element from the stack i.e. (1, 1) and goes to that position.

In (1, 1) similar case arises hence pops off previous cell tuple.

Finally we backtrack to (0, 0) while loop ends.

Let the entry be at (0, 0) and exit at (1, 0)



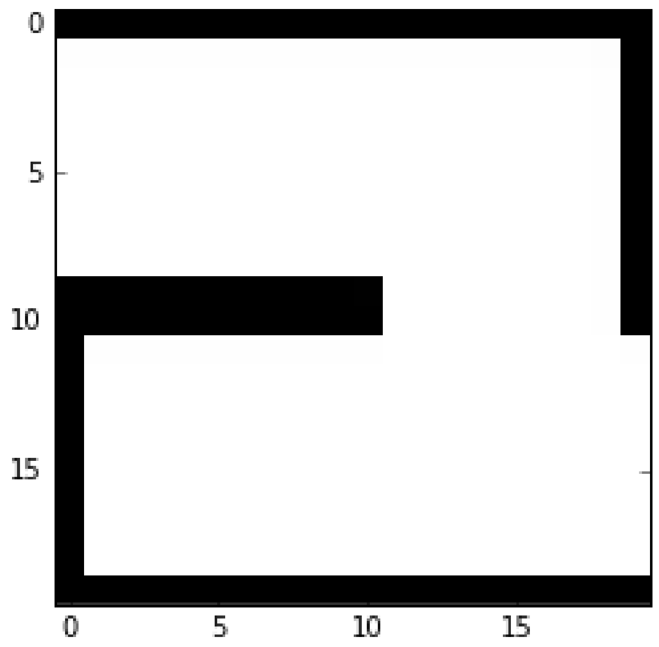
Now all 1 walls are removed to get a unique solution of the maze.

Code:

- We create list of arrays of five zeros using numpy
- The initial position is given and a stack is crated with the position tuple.
- Every visited cell gets added to the stack
- In while loop all the checks are done and if there is no cell unvisited the previous cell gets pop from the stack
- When it traverse backs to the initial point we plot the image of the maze.

Maze.pdf

The maze for above example:


0
5
10
15
0
5
10
15