# Unit 3 - Lesson 4. Maze Algorithms – Recursive Backtracker and Depth First

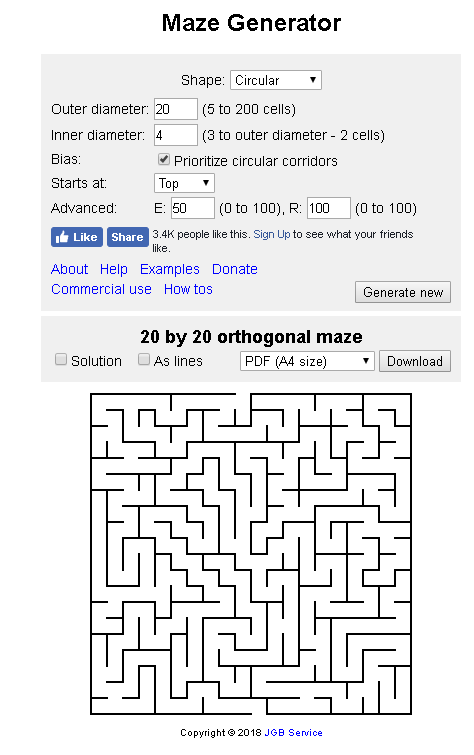
**Aim:** What is the recursive backtracker algorithm and what is the depth first the algorithm? How do we use the algorithms to generate a maze?

**Objectives:** After the lesson, students should be able to:

* Understand the recursive backtracker algorithm
* Understand the depth first algorithm
* Design a simple maze generator using the recursive backtracker
* Design a simple maze generator using the depth first

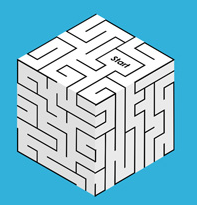
**CLASS PROCEDURE:**

***Do Now:*** Go to <http://www.mazegenerator.net/>, use the generator to generate a circular maze and solve the maze.



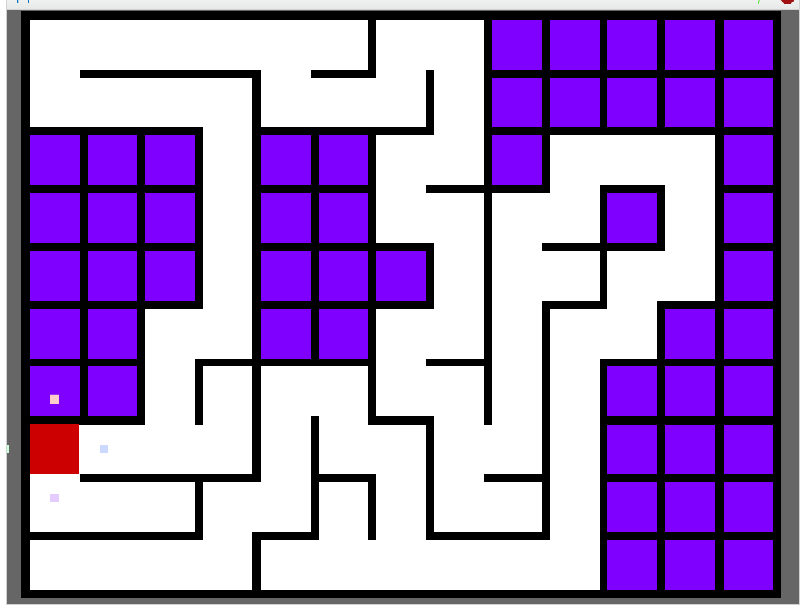
***Class Discussion / Presentation:***

1. How do we design a 3D maze? Answer: WE HAVE TO USE ALGORITHMS!



1. ***What is the recursive backtracker algorithm?***
2. Here’s the high level view of recursive backtracking:
3. Choose a starting point in the field.
4. Randomly choose a wall at that point and carve a passage through to the adjacent cell, but only if the adjacent cell has not been visited yet. This becomes the new current cell.
5. If all adjacent cells have been visited, back up to the last cell that has uncarved walls and repeat.
6. The algorithm ends when the process has backed all the way up to the starting point.
7. Here’s a demo on backtracking:

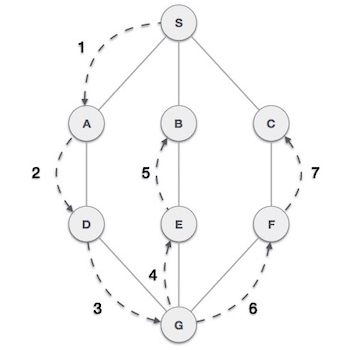
<https://scratch.mit.edu/projects/25744490/>



1. Use backtracking to create a maze using the grid below:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
| 6 | 7 | 8 | 9 | 10 |
| 11 | 12 | 13 | 14 | 15 |
| 16 | 17 | 18 | 19 | 20 |
| 21 | 22 | 23 | 24 | 25 |
| 26 | 27 | 28 | 29 | 30 |

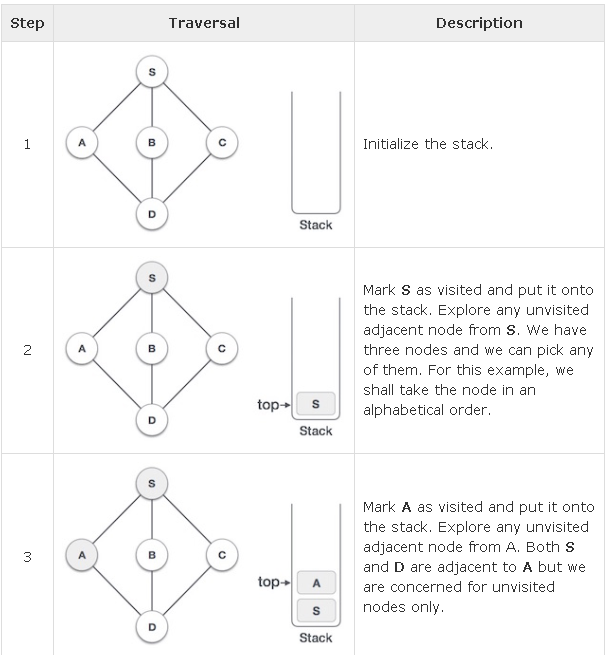
1. ***What is depth – first algorithm (DFS)? How do we use depth – first to create a maze?***
   1. Depth First Search (DFS) algorithm traverses a graph in a depthward motion and uses a stack to remember to get the next vertex to start a search, when a dead end occurs in any iteration.

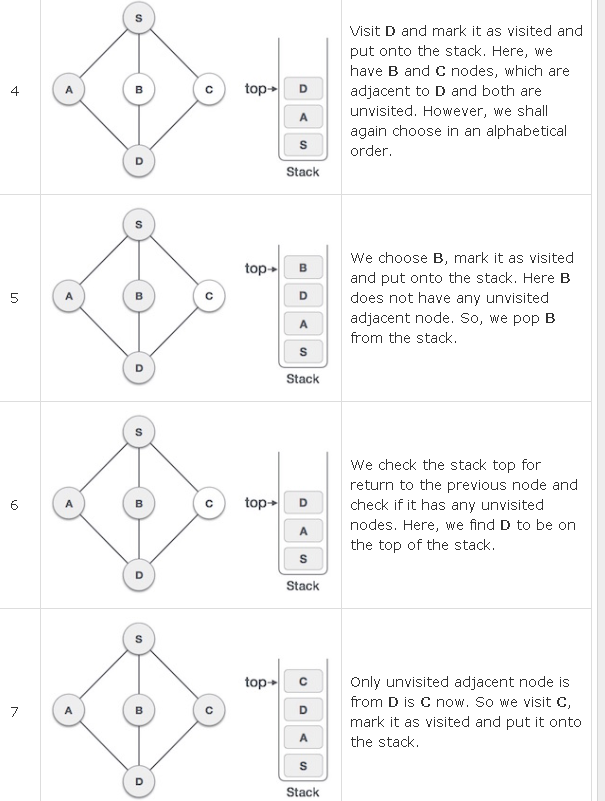


* 1. What are the rules for DFS?

As in the example given above, DFS algorithm traverses from S to A to D to G to E to B first, then to F and lastly to C. It employs the following rules.

* **Rule 1** − Visit the adjacent unvisited vertex. Mark it as visited. Display it. Push it in a stack.
* **Rule 2** − If no adjacent vertex is found, pop up a vertex from the stack. (It will pop up all the vertices from the stack, which do not have adjacent vertices.)
* **Rule 3** − Repeat Rule 1 and Rule 2 until the stack is empty.
  1. Steps:





As **C** does not have any unvisited adjacent node so we keep popping the stack until we find a node that has an unvisited adjacent node. In this case, there's none and we keep popping until the stack is empty.

* 1. Visualizing the DFS:

<https://www.cs.usfca.edu/~galles/visualization/DFS.html>

* 1. Visualizing the DFS in maze generation:

<http://www.migapro.com/depth-first-search/>

***Pair – sharing Activity:***

Continue working on your Maze Runner project. Due: November 30th.

***Class Presentation:***

Garden defender game.