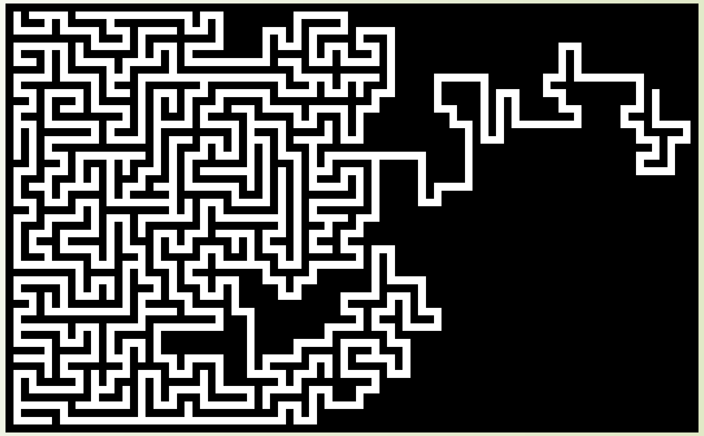
Maze Generation and Solving Algorithms

By Akarsh Kumar

Maze Generation Techniques: Recursive Backtracking Algorithm

- This method to generate mazes works by building a random path until it runs into a dead end and then backtracks to a part of the path that isn't dead.
- My version of this algorithm uses recursion and keeps building up the stack until it runs into a dead end, and then falls down to a lower stack.
- Animations:
  - Large maze generation animation: https://www.youtube.com/watch?v=6DzY9zR7qso
  - Small maze generation animation: https://www.youtube.com/watch?v=M46A4vgb4vM

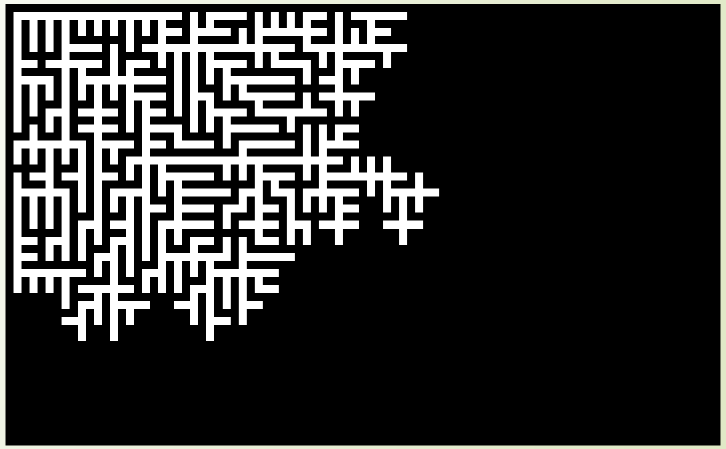


Maze Generation Techniques: Prims Algorithm

- This method works by always branching whenever possible and then choosing a random branch to continue building from a list of all the branches. This results in the maze looking sort of like a tree with lots of branches.
- My method works with a while loop running while there are available branches to continue building on.
- Animations:
  - Large maze generation animation:

    https://www.youtube.com/watch?v=nXaQzATDcnw
  - Small maze generation animation:

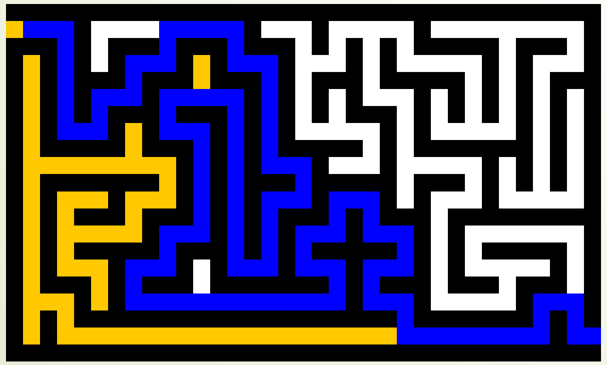
    https://www.youtube.com/watch?v=mOxGS01noo0



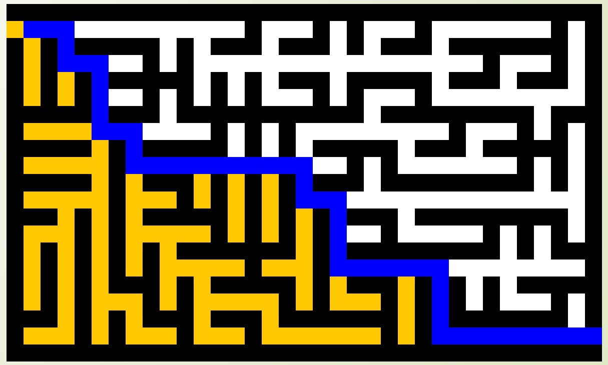
Recursive Backtracking Algorithm vs. Prims Algorithm

- Recursive Backtracking results in mazes that have long paths to the solution, but branch very rarely.
- Prims results in mazes that have short paths to the end but branch a lot, leading to more confusion when being solved.
- A combination of these two algorithms would probably generate the most optimal difficulty maze.

Recursive Backtracking Maze

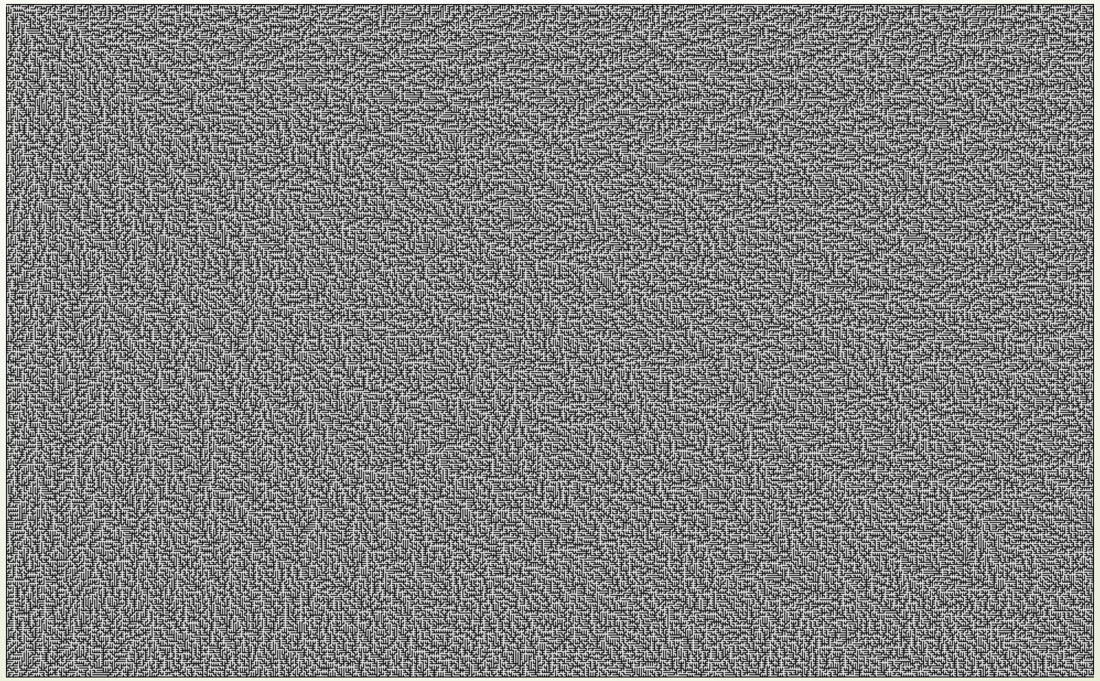


Prims Maze



Biggest Maze Generated

- The biggest maze generated by my program with the prims algorithm was with the Fibonacci number 14 resulting in the maze being 987x611.
- Fibonacci 15 was taking over 30 min to generate, so it was terminated.



Maze Solving Technique: Right Hand Rule

- All mazes can be solved by always sticking to the right wall.
- My implementation works by saving the location and which direction the "cursor" is facing and always tending toward the right side of the direction facing. This means that no memory needed to be allocated for saving which turns the cursor took and at which branch. Only a List<Coordinate> path needed to be saved.
- Animation:
  - Solving a prims maze with the right hand rule: https://www.youtube.com/watch?v=Gjo2zZBoVKQ
  - Solving a recursive backtracking maze with the right hand rule: https://www.youtube.com/watch?v=jGNP4WGKJIQ

Solving Times

- For a 377x233 maze: 0.079 sec
- For a 611x377 maze: 0.228 sec
- For a 987x611 maze: 0.712 sec
- As mentioned before, I could not get a bigger maze to test this out with.