Oplossingen:

Randomized depth-first search:

Chooses a random cell next to the current cell and removes the walls between the 2 nodes. If it reaches a point where there is no node or completed nodes next to it it will backtrack until it finds unused nodes. When the algorithm is done it will backtrack to the beginning.

The depth-first search algorithm of maze generation is frequently implemented using backtracking. This can be described with a following recursive routine:

Given a current cell as a parameter

Mark the current cell as visited

While the current cell has any unvisited neighbour cells

Choose one of the unvisited neighbours

Remove the wall between the current cell and the chosen cell

Invoke the routine recursively for the chosen cell

Randomized prim’s algorithm:

This algorithm is a randomized version of Prim's algorithm.

Start with a grid full of walls.

Pick a cell, mark it as part of the maze. Add the walls of the cell to the wall list.

While there are walls in the list:

Pick a random wall from the list. If only one of the cells that the wall divides is visited, then:

Make the wall a passage and mark the unvisited cell as part of the maze.

Add the neighboring walls of the cell to the wall list.

Remove the wall from the list.

I chose for the RDFS because it is a simple but efficient algorithm that will do the job great

Sources:

<https://en.wikipedia.org/wiki/Maze_generation_algorithm>

<https://www.youtube.com/watch?v=OutlTTOm17M&t> (only used for the algorithm)