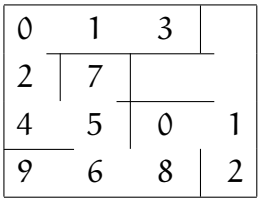
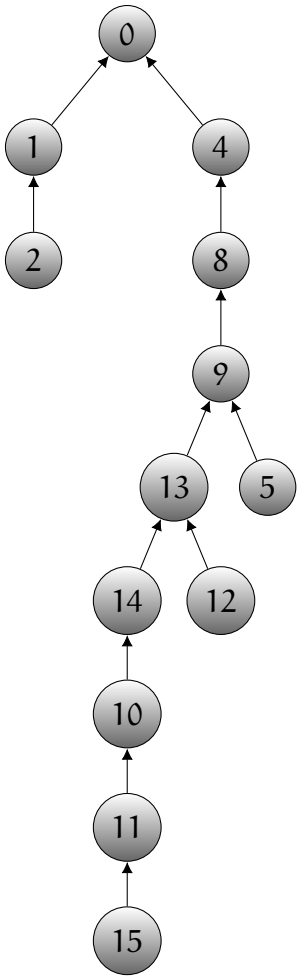
Aaron Mednick
Todor Nikolov
CS 146 - Data Structures and Algorithms
Dr. Potika
Project 3

Daedalus' Labyrinth

```
1  /**
2   * MazeSolve class contains a main method we use for testing.
3   */
4  class MazeSolve {
5      /**
6       * Pretty straightforward. We generate a maze grid, print it,
7       * turn int into a maze, print it, solve it DFS and BFS,
8       * and report the number of steps it took.
9       */
10     public static void main(String[] args) {
11         int size = 10;
12 
13         Scanner scanner = new Scanner(System.in);
14         System.out.print("Enter a maze size (10 would be a good choice): ");
15         try {
16                 size = scanner.nextInt();
17                 if (size < 1 || size > 100)
18                         throw new Exception();
19         }
20         catch (Exception e) {
21                 System.out.println("Bad size. Going with 10.\n");
22                 size = 10;
23         }
24         finally {
25                 scanner.close();
26         }
27 
28         Maze maze = new Maze(size);
29         System.out.println("Plain grid.");
30         maze.print();
31         System.out.println("Maze generated.");
32         maze.mazify();
33         maze.print();
34         System.out.println("Solving using DFS.");
35         int dfs = maze.dfs_solve();
36         System.out.println("Solving using BFS.");
37         int bfs = maze.bfs_solve();
38         System.out.println("\nDFS moves: " + dfs + "\nBFS moves: " + bfs);
39     }
40 }
```

Finding the exit of the labyrinth was a relatively easy task, but isolating the path from all the dead ends required some trickery. For this purpose, we developed an "inverse tree," where once a cell from the maze is reached, it is added to a hash map, for quick retrieval. When a node is being added, the node preceding it is its parent, and the child-parent relationship is reversed. A parent does not know its children, but children know about their parents. Here is a sample maze (from the instructions) and the associated "inverse tree." Note that the node numbers are cell numbers, not traversal order.





After having this tree, it is a matter of traversing it from the last cell (15) back to starting cell (0) to get the path to solve the maze.

Furthermore, we ran some tests to see which of the two algorithms would be faster in solving the same maze. After several thousand random 20×20 mazes, the Depth-First Search emerged as a winner by being about about thirty per cent faster than Depth-First Search. We believe this happens because, when it reaches a junction, DFS just charges forward, while BFS branches out, reducing its speed down by fifty to sixty per cent.