William Briggs

Data Structures

Heather Amthauer

READ ME (Puzzle Project)

The following project is two implemented search algorithms that will solve for a correct solution to a puzzle. This 3x3 puzzle should have six tabs facing out and six tabs facing inwards along the border. The following search algorithm it uses is depth first search and breadth first search. Breadth first search algorithm uses a queue data structure to store intermediate results as it traverses the graph in order to find a solution. While depth first search algorithm extends the current path as far as possible before backtracking to the last choice point and trying the next alternative path until it finds a solution.

In order to run this project you will need to compile the three files Board.java, Puzzle.java, and PuzzlePiece.java. When the puzzle.java file is run, it will use the two other files to find a solution to the puzzle. The file calls for arguments that are passed into the command prompt as well when you try and run this file. In order to call upon the depth first search algorithm, you will need to pass the arguments (dfs) in order to find a solution. Or another alternative is using two arguments (dfs v) which will show the user how the program is running and the search method. In order to run the breadth first search algorithm you will need to pass the arguments (bfs) or (bfs v). With the second argument ‘v’ the breadth first search will show the user how the algorithm is running and the search method.

**The expected execution time to find a valid solution:**

* dfs 1 second
* dfs v 2 seconds(This will display the search method traversals)
* bfs 1 second
* bfs v 20 minutes(This will display the search method traversals)

**The solution will be displayed as a 3x3 grid along with each puzzle piece’s description:**

A D G

B E H

C F I

A: (TOP, RIGHT, BOTTOM, LEFT)

B: (TOP, RIGHT, BOTTOM, LEFT)

C: (TOP, RIGHT, BOTTOM, LEFT)

D: (TOP, RIGHT, BOTTOM, LEFT)

E: (TOP, RIGHT, BOTTOM, LEFT)

F: (TOP, RIGHT, BOTTOM, LEFT)

G: (TOP, RIGHT, BOTTOM, LEFT)

H: (TOP, RIGHT, BOTTOM, LEFT)

I: (TOP, RIGHT, BOTTOM, LEFT)