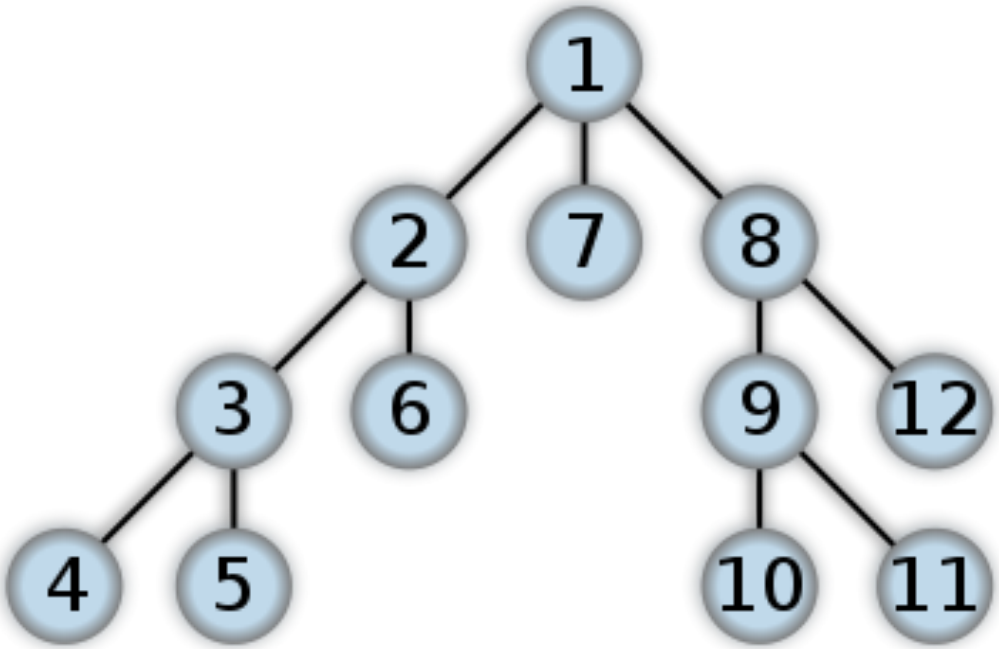
COMP 2526

A3a

Depth First Search Using Recursion

Depth First Search



Where Do We Start?

1. User select Maze > Solve Maze from menu
2. GameFrame's actionPerformed method responds to the request by invoking generateMazeSolutions
3. generateMazeSolutions asks its Maze object to reset
4. generateMazeSolutions invokes solveMaze on its MazeSolver object, and assigns the return value to a list of lists called solutions

How Does MazeSolver solveMaze()?

1. Invoke clear() to delete any old solutions
2. Check to see if [0][1] is a path and throw an exception if it's not
3. Create a new empty ArrayList of MazeSections and pass it to generatePaths

So far so good, right? No recursion yet...

Recursion Alert

The generatePaths method must be recursive. Take a deep breath. Relax. What follows is an elegant and simple algorithm for solving a maze recursively with DFS.

DFS Recursive Algorithm

generatePaths(Maze, row, column, path)

1. If the row < 0 or the row >= maze dimension or the column < 0 or the column >= maze dimension or the section at (row, column) is solid or the section at (row, column) has been visited then return (do nothing)
2. otherwise (else)
    1. if we are in the rightmost column, we solved it (yay!) so:
        1. add the maze section at row and column to the current path
        2. add the current path to the collection of maze solutions
    2. otherwise (else)
        1. mark this maze section as visited
        2. add this maze section to the current path
        3. make recursive calls to generatePaths in each of the 4 cardinal directions, passing a COPY of the current path as the final parameter
        4. mark this maze section as unvisited

DFS Recursive Algorithm

In fewer words:

When we take a step:

1. If we walk off the map, or into a wall, or into a spot we've already been, we do nothing.

2. Otherwise, if we reach the rightmost column we have escaped from the maze so we send our path to the solutions list.

3. Otherwise we clone the path (and ourself!), mark the spot we've been in as visited, take a step in each direction with a copy of the path, and then mark the spot as unvisited

The Base Case

generatePaths(Maze, row, column, path)

1. If the row < 0 or the row >= maze dimension or the column < 0 or the column >= maze dimension or the section at (row, column) is solid or the section at (row, column) has been visited then return (do nothing)
2. otherwise (else)
    1. if we are in the rightmost column, we solved it (yay!) so:
        1. add the maze section at row and column to the current path
        2. add the current path to the collection of maze solutions
    2. otherwise (else)
        1. mark this maze section as visited
        2. add this maze section to the current path
        3. make recursive calls to generatePaths in each of the 4 cardinal directions, passing a COPY of the current path as the final parameter
        4. mark this maze section as unvisited

The Recursive Call

generatePaths(Maze, row, column, path)

1. If the row < 0 or the row >= maze dimension or the column < 0 or the column >= maze dimension or the section at (row, column) is solid or the section at (row, column) has been visited then return (do nothing)
2. otherwise (else)
    1. if we are in the rightmost column, we solved it (yay!) so:
        1. add the maze section at row and column to the current path
        2. add the current path to the collection of maze solutions
    2. otherwise (else)
        1. mark this maze section as visited
        2. add this maze section to the current path
        3. make recursive calls to generatePaths in each of the 4 cardinal directions, passing a COPY of the current path as the final parameter
        4. mark this maze section as unvisited

Hint #1:

How to copy an ArrayList:

```
ArrayList<MazeSection> path
                    = new ArrayList<>();

…  fill path with MazeSections

ArrayList<MazeSection> newPath
                    = new ArrayList<>(path);
```

Hint #2:

How to make the recursive call:

```
...
generatePaths(maze, row - 1, column, new ArrayList<MazeSection>(path));
generatePaths(maze, row + 1, column, new ArrayList<MazeSection>(path));
generatePaths(maze, row, column - 1, new ArrayList<MazeSection>(path));
generatePaths(maze, row, column + 1, new ArrayList<MazeSection>(path));
...
```

We send a copy of the path in each cardinal direction

Hint #3:

Your generatePaths method will probably look a little like this:

```
public void generatePaths(Maze  maze,
                          int row,
                          int column,
                          ArrayList<MazeSection>  path) {
    if (...) {
        return;
    } else {
        if (...) {

            ... Do stuff ...

        } else {

            ... Do stuff, make recursive calls, do stuff ...

        }
}
```

FIN