Analysis of Algorithms

Project 3 Graph Modeling and Graph Algorithms

30 November 2016

1. To model the problem input, I would use an undirected graph whose vertices represent cells in the maze and whose edges represent connections with other cells in the maze. Each cell can only have a maximum of six edges, as there are only six possible directions to travel towards. These edges would be stored in an adjacency list or other structure that would allow every vertex to know its neighbors. This will be the most space efficient way to store the graph, as an adjacency matrix would require each cell to know whether it is adjacent to every other cell in the graph.

2. To solve the problem, I would use a recursive backtracking algorithm. In general, this type of algorithm adds possible candidates to a working solution until it determines that this solution will not solve the problem. When this happens, it begins removing candidates and testing other possible combinations. This process is done recursively until a solution is found. In this particular problem, the recursive backtracking will take on the role of a maze solving algorithm whose candidates will be cells in the maze and whose goal is to reach the given ending cell from the given starting cell. Pseudocode for how this algorithm will work may be found below.

```
Input: cell current, cell end
Output: boolean return type, vector stores solution path
Algorithm: explore
if current and end are the same
    return true
end
if current does not exist or has been visited
    Remove last element of vector
    return false
end
Mark current as visited
for each direction (north, east, south, west, up, and down)
    if current has a neighbor in that direction
        Call explore on that neighbor
        if the algorithm returns true
            Insert the respective direction into vector
        end
    end
end
```

```
return false
```