Maria Deslis

csci406 - Mehta - spring 2013

Project 3: Maze

1. Problem Modeling[50 pts]:

- [10] Explain how you modeled the problem as a graph.

  I modeled the problem as a graph by making each vertex in the graph represent an edge in the maze.

  This allowed me to also include the possibility of going backwards in a maze. For example, one vertex can contain the edge B to E, but there is also a vertex that contains the possibility of E to B.

  I could also compare whole edges of the maze to each other rather than each village. Making it easier to compare colors, transit types, and to ensure I wasn't looping and returning to the same village over and over.

- [15] Draw enough of the resulting graph to convince us that you have modeled the graph correctly.

  (see attached papers)

- [10] Identify the graph algorithm needed to solve the problem.

  Depth First Search (DFS)

- [15] Argue that this algorithm will actually solve the problem.

  For both directed and undirected graphs, DFS investigates the graph from the start until all vertices reachable from the start are visited. During its execution, DFS traverses the edges of the graph, computing the information such that it reveals the inherent structure of the graph. It maintains a counter that is incremented when a vertex is first visited, coloring it gray, and when DFS is done with the vertex, coloring it black.

  DFS only needs to store a color (either white, gray, or black) with each vertex as it traverses the graph. Thus DFS requires minimal overhead in storing information while it explores the graph starting at the initial vertex.

  DFS can store its processing information in arrays (or in my case, vectors) separately from the graph. The only requirements DFS has on the graph is that one can iterate over the vertices that are adjacent to a given vertex. This feature makes it easy to perform DFS on complex information, since it accesses the original graph as a read-only structure.

  The recursive DFS function is called once for each vertex in the graph, and the loop is executed no more than n times. Within DFS, every neighboring vertex must be checked; in this case where we have an undirected graph, they are traversed once and are seen one other time. Total performance cost is O(vertex + edges).

  Thus, we know that DFS actually solves the problem because it exhaustively searches through the graph and all choices have been considered.

2. Code [30 pts]:

(see attached papers)

3. Results [45 pts]:

```
 $ ./main
Villages: 36 Transit Lines: 70
Depth First Search:
Path Length: 48
[A, B, E, O, U, V, a, f, Z, U, O, E, B, C, D, J, N, T, S, R, Q, P, K, F, G, L, Q, W, b, a, Z, V, W, X, S, Y, e,
d, S, M, I, D, G, K, O, V, b, h, i, j]
```