Lab 1

Assigned: Sep 13
Due: Sep 30, 10pm

Assignment

Write a program that, using the selected algorithm, finds a path in a graph from a start to goal vertex.

Grading

out of 100:

- 90 points for correct behavior (30 per algorithm)
- 10 points for well written solver code: clear data structures, methods, control flow, etc.

Expect us to run your program against a number of different input graphs.

Running your program

A program run should look like: (may be slightly different depending on language, note this in your README)

```
path [-v] -start $start-node -goal $goal-node -alg $alg graph-file
```

- -v is an optional flag for verbose mode (more later)
- -start is followed by the name of the node to start from
- -end is followed by the name of the node which is the goal
- -alg is followed by one of: BFS, ID, ASTAR
- -depth used only for iterative-deepening, that indicates the initial search depth, with a default increase of 1 after that.
- a graph file (next section)

Your program should exit gracefully and indicate as best as possible the problem in case of bad arguments (e.g. -start referencing a vertex not in the graph-file) or a bad file (e.g. an edge referencing a vertex not in the file).

Graph File Input

Input should be a text file with the following contents.

- Each line in the file should be either a comment, vertex or an edge.
- A line that starts with # is a comment line and should be skipped
- A vertex consists of a node label followed by two ints: an x and y coordinate.
- An edge (always undirected) is simply two node labels.
- Note: A node label should be any string with alphanumeric characters
- Note: there is no guaranteed order. Edges might reference vertices not yet defined, and that is fine as long as they appear somewhere in the file.

The length of any edge is computed from the 2 vertices (x,y) coordinates using standard Euclidean distance for p, q of = sqrt((p_x - q_x)^2 + (p_y - q_y)^2).

Algorithms

You are to implement 3 algorithms discussed in class:

- Breadth First Search using a visited list to avoid duplicate vertices
- Iterative Deepening, using the -depth parameter for initial depth, then increasing by 1. (You may also use a visited list: not required)
- A*, using an h function of Euclidean distance (see above) from potential expansion node to goal. You are free to pre-compute this, but it is not required.

Output

You should print to console the path from start to goal. (this does mean for BFS, you will have to remember which edge was taken to which vertex)

Additionally, in verbose mode (-v flag) the following [also see examples]

- You should print out each node searched.
- For Iterative Deepening, print each time the depth parameter increases
- For A*, you should print out each expansion choice, with distances rounded to 2 decimal places

Examples

Files

- Similar to class presentation: Example 1 BFS Iterative Deepening A*
- Example 2 BFS Iterative Deepening A*

Submission

On Brightspace, a zip file containing:

- Code for the programs
- Makefile/BUILD/pom.xml/etc needed to build your code
- README indicating how to compile and run your code
- In your README include any classmates you consulted with
- As previously indicated, the code should build on department linux machines