CS2134 Data Structures and Algorithms

Extra Credit Assignment A: The Maze

Due 11:00 PM, November 26 2016

Core Instructions

This assignment is extra credit, and therefore completely *optional*.

This assignment is worth up to **100 extra credit points**, to be added back to one homework where you lost the most points on.[1] If you decide to submit this EC assignment, you cannot submit the Maze EC assignment. (That is, you can only select *one* of the two for extra credit.)

In order to receive credit for this assignment, you must code up a working solution *and* explain your code to one of the TAs. You will first need to submit this assignment to NYU Classes. You will be able to receive partial extra credit for this assignment, based on how much you have completed. This assignment is purposely slightly open-ended so that you have the freedom to approach it in a manner that is interesting or educationally valuable to you.

1 Programming Part

1. Write a recursive function to find a location[2] in a maze.

   You may only walk left, right, up or down (no diagonals).

   The maze will be implemented as a two dimensional character vector (use `vector<vector<char> >`). Passages are marked with '.', walls

[1] That is, these extra credit points **do not** add directly on top of all of your homework scores together. Rather, the extra credit will only count for **points back on the one homework assignment** where you lost the most points.

[2] Perhaps my office? Has anyone been on the 10th floor of building 2? This might be a useful program to have.

are marked by 'x'. The start position will be marked by 's', and the location to find by 'e'. External walls are not marked (you will have to do bound checking in the two dimensional vector.) You function will first need to find the location of 's' by exhaustive search. You might find it convenient to modify the maze to show where you have been.

Here is a sample maze:

```
xxxxxxxxxxxxxxxxx
s.x...x....x...exx
x.x.x.x.xx.x..xxxx
x...x...xx.x..x.xx
xxx.x..xxx...xx.xx
xxx.xx..xx......xx
xxx...x.xxxxxxxxxx
xxxxx.x........xxx
xxxxxxxxxxxxxxxxx
```

2. Solve programming problem 1 this time using a stack to help you traverse the maze instead of recursion.

3. Solve programming problem 1 this time using a queue to help you traverse the maze instead of using a stack or recursion (and assume you can teleport to any location specified by it's coordinates.)

4. Combine the previous three programming problems into a class that also contains:

   - a method `print_maze()` which prints the maze
   - a method to load a maze from a file given an `istream`
   - collect information when traversing that graph, so that you can print the path (with no wrong turns) that goes from 's' to 'e'.

2 Written Part

1. Under what condition will programming problem 1 traverse fewer locations in the map than programming problem 3? And vice versa? Demonstrate using a maze.

2. Will programming problem 1 traverse fewer locations in the map than programming problem 2?