CSCI 3333 Homework MZ1: Maze Solving

1 Introduction

Many films and television shows, including Westworld, The Shining, The Maze Runner, and Labyrinth, involve literal or metaphorical mazes and the disorienting complexity of navigating them. In the real world, tens of thousands of robots plan efficient delivery routes through Amazon's vast labyrinthine warehouses. Also, Schweller has a maze-solving homework.

In this homework you'll implement a maze-solving C++ function. Both the input maze and output solution use an ASCII art "map" format (see Sections 4 and 5). Your code should solve large mazes in a fraction of a second using a breadth-first search.

2 Instructions

The following files have been given to you:

1. A C++ header file (`solve.h`) declaring the `solve` function.
2. A C++ header file (`vertex.h`) declaring and implementing the `Vertex` class.
3. A C++ source file (`main.cpp`) containing a `main` function with tests.

Create a new C++ source file named `solve.cpp` that implements the function declared in `solve.h`, so that `solve.cpp` and the provided files compile into a program that runs with no failed tests. Submit the source file `solve.cpp`.

3 Submission and Grading

Submit the aforementioned source file(s) via Blackboard as attached file(s). In the case of multiple submissions, the last submission before the deadline is graded.

For grading, each submission is compiled with the provided files and run. Submissions that do not run to completion (i.e. fail to print "Assignment complete.") receive no credit. Submissions that take an unreasonable amount of time (e.g. more than a minute or so) to run and do not meet the asymptotic efficiency requirements receive no credit. All other submissions receive full credit.

See the course late work policy for information about receiving partial credit for late submissions.

4 Maze String Format

Each maze is represented as a string with the following format:

- The maze consists of several lines of the same length, each terminated by the `'`
  `n'` (*newline*) character.
- Excluding these newline characters, all characters are either `'#'` (*hash/pound*), or `' '` (*space*).
- The hashtag character denotes a wall.
- The space character denotes an empty location.
- Exactly two locations on the *boundary* of the maze (first and last row and column) are spaces. All other locations are hashes. These two spaces are called *exits*.

For instance, the maze represented as the string

```
"# ####\n#    #\n# ## #\n#    #\n### ##\n"
```

which prints as:

```
# ####
#    #
# ## #
#    #
### ##
```

5 Maze Solution Format

A maze solution is represented as a string identical to that of a maze, but with a path of `'o'` (lowercase letter O) symbols denoting a shortest path between the two exits of the maze. A *shortest path* consists of a minimum-length sequence of locations, each above, below, left, or right of the previous, beginning and ending with the locations of the exits. A solution to the maze above is

```
"#o####\n#o   #\n#o## #\n#ooo #\n###o##\n"
```

which prints as:

```
#o####
#o   #
#o## #
#ooo #
###o##
```

Note that valid solutions must be shortest paths. Thus the following is **not** a solution:

```
#o####
#oooo#
# ##o#
#  oo#
###o##
```