CSCI 3333 Homework MZ2:
Maze Solving with Portals

1 Introduction

Many films, televisions shows, and video games, including Donnie Darko, Stargate, Rick & Morty, and Portal, involve literal or metaphorical portals that enable teleportation between distant locations. In the real world, there are no such things.

In this homework you'll implement a maze-solving C++ function using your `MinPriorityQueue` template class implementation from hwPQ. Both the input maze and output solution use an ASCII art "map" format similar to that in hwMZ1 (see Sections 4 and 5).

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 (`minpriorityqueue.h`) declaring the `MinPriorityQueue` template class.[1]
3. A C++ header file (`vertex.h`) declaring and implementing the `Vertex` class.
4. 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`. Also correct the implementation of the template class in `minpriorityqueue.h` so that the provided files and `solve.cpp` compile into a program that runs with no failed tests.[2] Submit the source files `solve.cpp` and `minpriorityqueue.h`.

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*), `' '` (*space*), or digits (`'0'-'9'`).
- The hashtag character denotes a wall.

[1] This is the same file as in hwPQ.
[2] Feel free to use your solution to hwPQ.

- 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*.
- A digit denotes a portal *opening*. Each such digit appears exactly twice, denoting the two locations connected by the portal. The distance between these two locations via the portal is equal to the digit's value (e.g., the distance between the `'3'` portal openings is 3).

For instance, the maze represented as the string

```
"# ######\n#  1  7#\n########\n#1    7#\n### ####\n"
```

which prints as:

```
# ######
#  7  1#
########
#17    #
### ####
```

contains two portals: the `'1'`-portal and the `'7'`-portal. The distance between the openings of the `'1'`-portal via the portal is 1. The distance between the openings of the `'7'`-portal via the portal is 7.

5 Maze Solution Format

The maze solution format is identical to that in hwMZ1, but with the addition of portals potentially changing the

A maze solution is represented as a string identical to that of a maze, but with `'o'` (lowercase letter O) symbols denoting a shortest path between two *used* portal locations and other locations, respectively, visited along 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, right or portal-adjacent to the previous, beginning and ending with the locations of the exits. A solution to the maze above is

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

which prints as:

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

Note that valid solutions must be shortest paths. Thus the following, while it is a valid path connecting the exits, is **not** a solution:

```
#o######
#ooo  1#
########
#1oo   #
###o####
```