CSCE 350 – Data Structures and Algorithms

Project 2

Depth-first Maze Running

50 Points

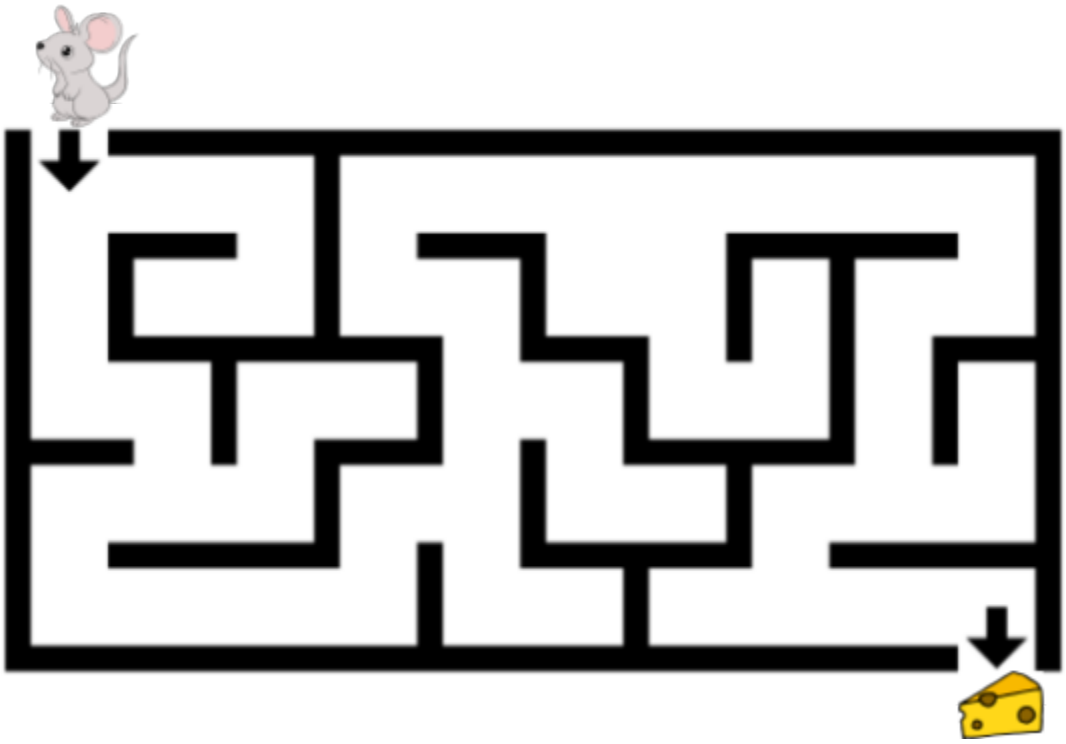
Assigned on: October 9th, 2019
Due: October 24th, 2019 @ 11:59 pm

Topics Covered:

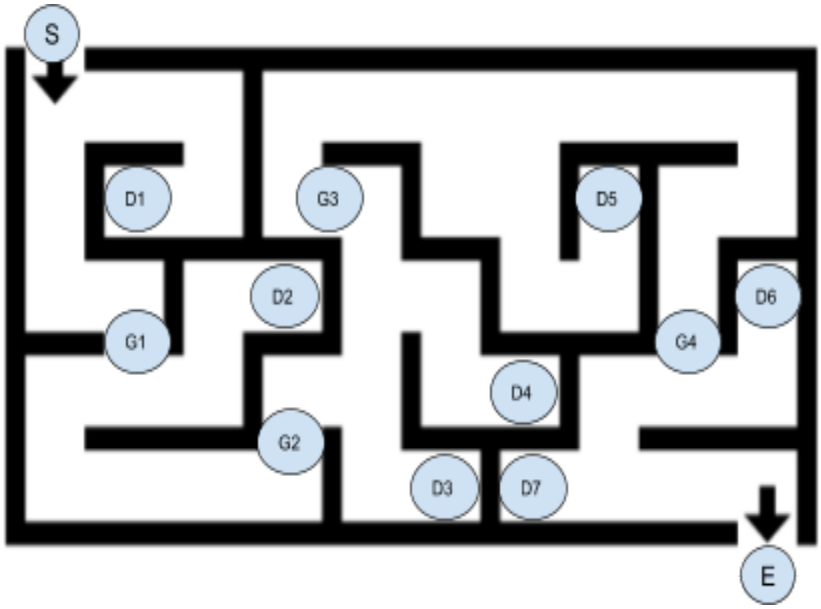
- Exhaustive search
  - Depth-first search
- Counting operations

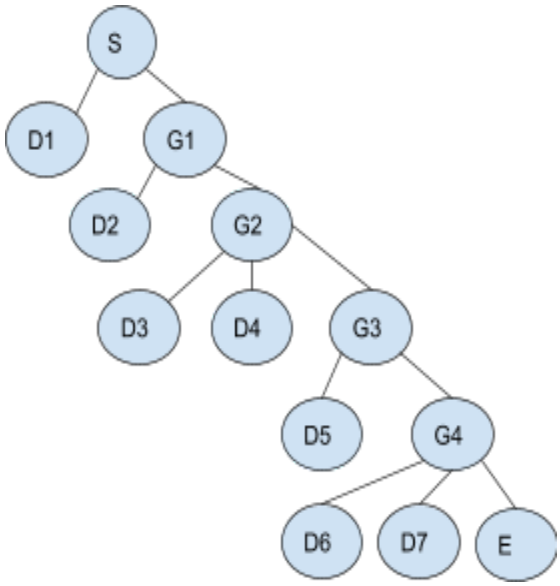
Background:

One application of the search algorithms that we have discussed is path planning. To illustrate, you will write an algorithm using several of the techniques discussed in class to path plan for Luey, who really wants to get to his cheese.



Any maze can be converted to a graph by defining a vertex for every start point, endpoint, dead-end and all the points in the maze where more than one path can be taken, and then connecting the vertices according to the paths in the maze. An example annotated maze and the corresponding graph is shown below:





Alternatively, the graph above can be represented with the following adjacency matrix where 1 indicates a connection between two nodes:

	S	D1	D2	D3	D4	D5	D6	D7	G1	G2	G3	G4	E
S	1	1	0	0	0	0	0	0	1	0	0	0	0
D1	1	1	0	0	0	0	0	0	0	0	0	0	0
D2	0	0	1	0	0	0	0	0	1	0	0	0	0
D3	0	0	0	1	0	0	0	0	0	1	0	0	0
D4	0	0	0	0	1	0	0	0	0	1	0	0	0
D5	0	0	0	0	0	1	0	0	0	0	1	0	0
D6	0	0	0	0	0	0	1	0	0	0	0	1	0
D7	0	0	0	0	0	0	0	1	0	0	0	1	0
G1	1	0	1	0	0	0	0	0	1	1	0	0	0
G2	0	0	0	0	1	0	0	0	1	1	0	0	0
G3	0	0	0	0	0	1	0	0	0	1	1	1	0
G4	0	0	0	0	0	0	1	1	0	0	1	1	1
E	0	0	0	0	0	0	0	0	0	0	0	1	1

Description:

You should write a C++ program that does the following:

1. First, your program should read in a set of integers and a set of characters from “standard in” via file redirection. The first number of the set of integers will be the size (n) of your n by n adjacency matrix. The rest of the integers you will use to populate the elements of the adjacency matrix (you will need to use either a double pointer array or a vector<vector<int>>). The next line of the input file will be a list of characters denoting the vertex names. You should store these somewhere in your program for the final output. For example, an input file will have this form:

```
Maze1.txt
----------
3
1 1 0
1 1 1
0 1 1
S G1 E
```

2. Using the stack class in C++ (http://www.cplusplus.com/reference/stack/stack/) implement a depth-first search on the maze to find a legitimate path from S to E (when S is at the bottom of your stack and E is at the top). You should also keep track of the number of operations that it takes to find that path (number of pushes and pops). At the end of execution, your search should output the path of the nodes to travel (use the character labels) and the number of operations.

Example input/output:

```
For Maze1.txt the output would be:
Order of nodes: S, G1, E
Number of operations: 3
```

Additional Specifications:

- I have provided a main.cpp that you may use as a starting point for your code.
- Your program should consist of a long comment that contains the following information:
    - First name and last name of the programmer.
    - Email
    - Date and time of the program completion.
    - A brief description of the program function.
    - Input requirements and format.
    - The output of the program.
    - Any additional needed comments (e.g. related to compilation, execution or other requirements).
- Make sure your program compiles and runs on one of the Linux machines in the Linux lab before you submit.
- **Submit the source code (.cpp) not the executable.**

NAME YOUR FILE: Project2_<last_name>.cpp (replace <last_name> with your own last name.)