CS2021C Homework 5

Nicholas McClorey Daniel Wood, Kyle Van Blaricom April 25, 2019

Overview

The objective of this lab was to design and implement a graph as well as explore depth first and breadth first search. Searching algorithms are used in many pathfinding task to find different routes between two points and find the optimal route. This homework was also provided great practice using pointers and managing memory.

Search Data Structures

We decided to use a stack for our depth first search. Instead of using a class to create a stack, we used recursion to take advantage of the call stack. depthFirstSearch(Vertex*) takes the vertex passed in and calls the depthFirstSearch(Vertex*) on the adjacent vertices that haven't been visited. An overloaded method, depthFirstSearch() is public and takes no arguments. It gets the first node in the list of vertices and calls the depthFirstSearch(Vertex*) method, passing in the first node. Recursion takes over from there. The method is never called on a Vertex that has already been visited. This may use a lot of memory as the call stack becomes very large.

For the breadthFirstSearch(), we used the Linked List class as a queue by adding items to the back and removing items from the front. We add the first Vertex to the queue and enter the loop. Inside, a vertex is removed from the queue, its value is printed out and all of its outgoing adjacent edges are added to the queue unless they have already been visited. This is repeated until the queue is empty. This is more efficient in its memory use. The linked list being used as a queue only stores pointers to the vertices. However, there will probably be more entries in the queue than were in the stack. This is because there will be many nodes waiting for their turn while we search alternative routes.

When trying to find a path quickly, depth first search is typically better. When trying to find the shortest path, breadth first search should be used.

Compiling

This program was compiled using the MinGw compiler v6.3.0 on Windows 10. You could also use the tdm compiler To compile this program open a command prompt in the directory containing the source code and use the command "g++ -std=c++11 main.cpp Graph.cpp" to compile the program. Alternatively, you could probably copy and paste the code into a visual studio project but we haven't compiled using visual studio yet.

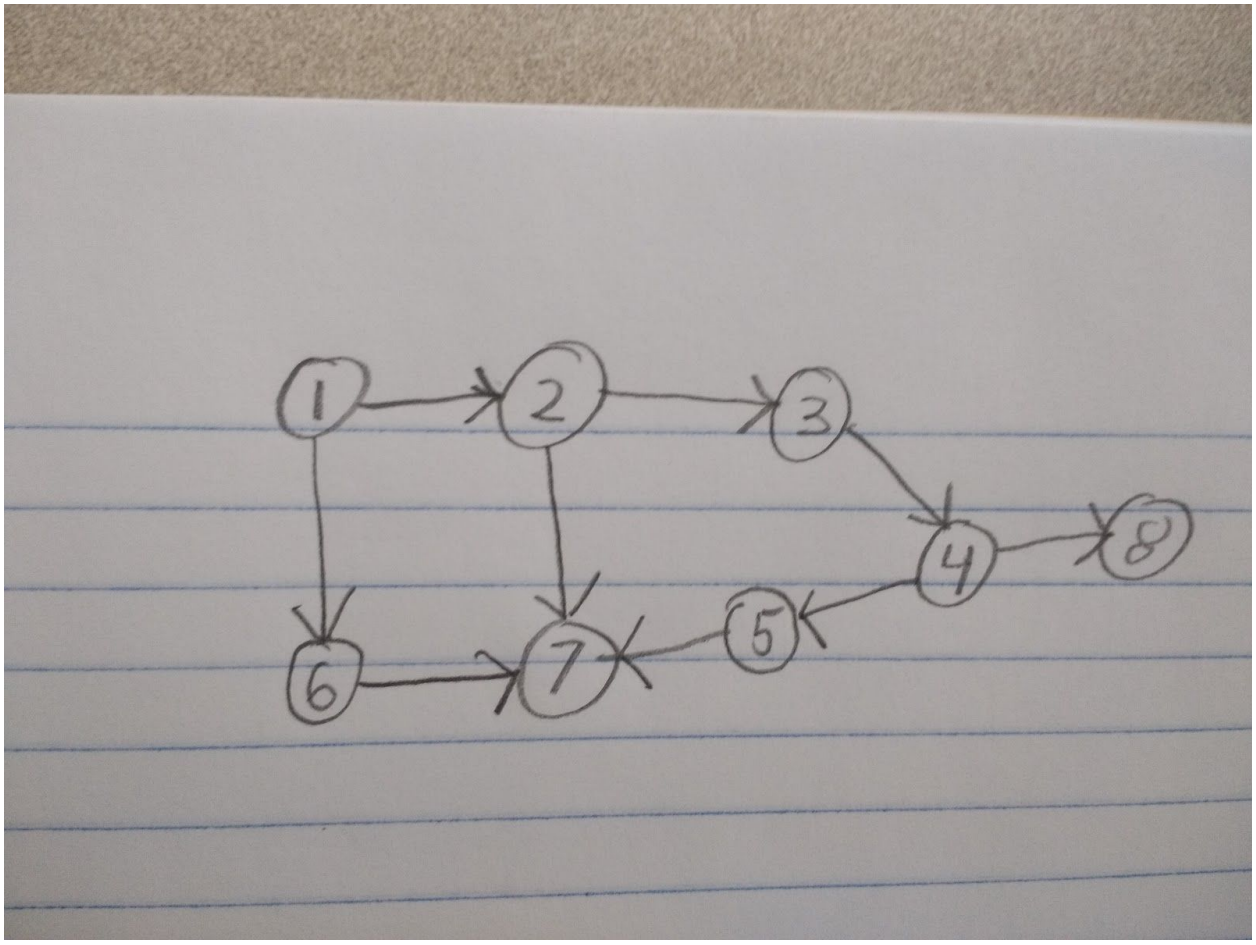
Contributions

Daniel Wood worked on the menu the user interacts with. Nick McClorey worked on Graph's methods to add, remove and find edges. Kyle Van Blaricom worked on the depth first and breadth first functions.

```
}

void testingGround() {
    Graph graph;
    graph.addEdge(1, 2);
    graph.addEdge(2, 3);
    graph.addEdge(3, 4);
    graph.addEdge(4, 8);
    graph.addEdge(4, 5);
    graph.addEdge(5, 7);
    graph.addEdge(2, 7);
    graph.addEdge(1, 6);
    graph.addEdge(6, 7);
    cout << "Breadth First Search" << endl;
    graph.breadthFirstSearch();
    cout << "Depth First Search" << endl;
    graph.depthFirstSearch();
}
```

```
Breadth First Search
1
2
6
3
7
4
8
5
Depth First Search
1
2
3
4
8
5
7
6
no runtime errors
[nick@dragon hw5]$
```



```
Enter the two vertices to connect starting with the originating vertex
2 8
Press 1 to add an edge to the graph
Press 2 to remove an edge from the graph
Press 3 to find an edge in the graph
Press 4 to find the out edges of a vertex
Press 5 to find the in edges of a vertex
Press 6 to do a depth first search
Press 7 to do a breadth first search
Press 8 to quit
2
Enter the two vertices to disconnect starting with the originating vert
2 8
The vertices have been disconnected
Press 1 to add an edge to the graph
Press 2 to remove an edge from the graph
Press 3 to find an edge in the graph
Press 4 to find the out edges of a vertex
Press 5 to find the in edges of a vertex
Press 6 to do a depth first search
Press 7 to do a breadth first search
Press 8 to quit
4
Enter the vertex
2
[ 3, 7, ]
Press 1 to add an edge to the graph
Press 2 to remove an edge from the graph
Press 3 to find an edge in the graph
Press 4 to find the out edges of a vertex
Press 5 to find the in edges of a vertex
Press 6 to do a depth first search
Press 7 to do a breadth first search
Press 8 to quit
5
Enter the vertex
7
[ 5, 2, 6, ]
```