Programming Assignment 6

Syed Abeer Hasan Zaidi

April 29, 2019

Program Description and Purpose of Assignment

The purpose of this assignment was to reinforce and test our knowledge of the Graph Data structure by having us implement a graph in the form of a adjacency matrix. We then had to print this graph in the form of an adjacency list. In the second part of the assignment we had to determine whether the graph met the conditions for a Eulerian Path, and if it did we had to print the path. The program I made is compiled using a makefile in the terminal of my macbook, and executed using the command **./run-graph**.

The Data Structure

The data structure is implemented as a adjacency matrix of size nxn, where n is the number of elements in the graph. The matrix was a boolean 2D array, and the elements adjacent to a certain element in the graph was denoted by whether their coordinate was marked as true or not. The data structure had 3 main functions, addEdge, deleteEdge, and isEdge. I also implemented a clear function for use in the copy assignment operator and the destructor for the Graph. The Graph had two private data members, numVertex, and the Adjacency matrix. numVertex stored the number of vertices in the graph. My implementation of the graph also had an input operator and an output operator to help with reading data, and printing the adjacency list. The oneStroke() function determined whether the graph was a eulerian path. PrintPath() printed the path if it was a eulerian path.

Conditions for Eulerian Path

1. All vertices with and non-zero degree are connected.
2. A maximum of two vertices can have an odd-degree.

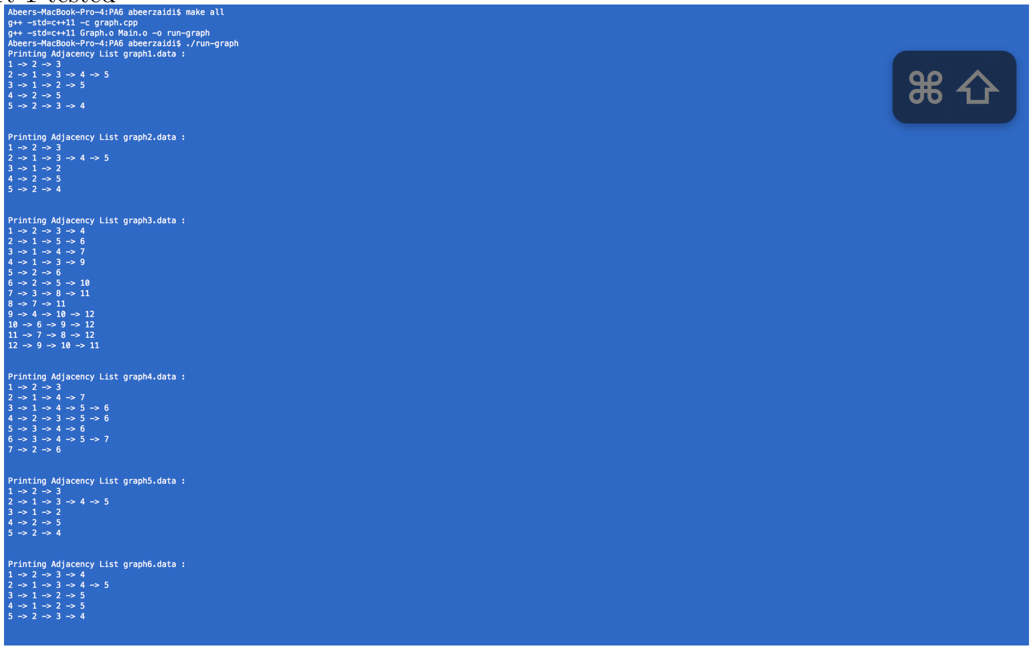
By odd degree I'm referring to an odd number of edges, and non-zero degree means the element has a positive number of edges.

Running Time of functions

1. AddEdge: O(1)
2. DeleteEdge: O(1)
3. isEdge: O(1)
4. oneStroke: O(V+E), V(number of vertices), E(number of edges)
5. PrintPath: O(E), E(number of edges).

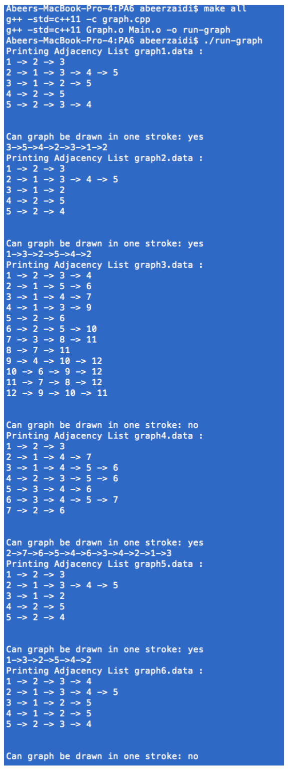
Testing Evidence

Part 1 tested



```
Abeers-MacBook-Pro-4:PA6 abeerzaidi$ make all
g++ -std=c++11 -c graph.cpp
g++ -std=c++11 Graph.o Main.o -o run-graph
Abeers-MacBook-Pro-4:PA6 abeerzaidi$ ./run-graph
Printing Adjacency List graph1.data :
1 -> 2 -> 3
2 -> 1 -> 3 -> 4 -> 5
3 -> 1 -> 2 -> 5
4 -> 2 -> 5
5 -> 2 -> 3 -> 4


Printing Adjacency List graph2.data :
1 -> 2 -> 3
2 -> 1 -> 3 -> 4 -> 5
3 -> 1 -> 2
4 -> 2 -> 5
5 -> 2 -> 4


Printing Adjacency List graph3.data :
1 -> 2 -> 3 -> 4
2 -> 1 -> 5 -> 6
3 -> 1 -> 4 -> 7
4 -> 1 -> 3 -> 9
5 -> 2 -> 6
6 -> 2 -> 5 -> 10
7 -> 3 -> 8 -> 11
8 -> 7 -> 11
9 -> 4 -> 10 -> 12
10 -> 6 -> 9 -> 12
11 -> 7 -> 8 -> 12
12 -> 9 -> 10 -> 11


Printing Adjacency List graph4.data :
1 -> 2 -> 3
2 -> 1 -> 4 -> 7
3 -> 1 -> 4 -> 5 -> 6
4 -> 2 -> 3 -> 5 -> 6
5 -> 3 -> 4 -> 6
6 -> 3 -> 4 -> 5 -> 7
7 -> 2 -> 6


Printing Adjacency List graph5.data :
1 -> 2 -> 3
2 -> 1 -> 3 -> 4 -> 5
3 -> 1 -> 2
4 -> 2 -> 5
5 -> 2 -> 4


Printing Adjacency List graph6.data :
1 -> 2 -> 3 -> 4
2 -> 1 -> 3 -> 4 -> 5
3 -> 1 -> 2 -> 5
4 -> 1 -> 2 -> 5
5 -> 2 -> 3 -> 4
```

Part 2 and 3 tested



```
Abeers-MacBook-Pro-4:PA6 abeerzaidi$ make all
g++ -std=c++11 -c graph.cpp
g++ -std=c++11 Graph.o Main.o -o run-graph
Abeers-MacBook-Pro-4:PA6 abeerzaidi$ ./run-graph
Printing Adjacency List graph1.data :
1 -> 2 -> 3
2 -> 1 -> 3 -> 4 -> 5
3 -> 1 -> 2 -> 5
4 -> 2 -> 5
5 -> 2 -> 3 -> 4


Can graph be drawn in one stroke: yes
3->5->4->2->3->1->2
Printing Adjacency List graph2.data :
1 -> 2 -> 3
2 -> 1 -> 3 -> 4 -> 5
3 -> 1 -> 2
4 -> 2 -> 5
5 -> 2 -> 4


Can graph be drawn in one stroke: yes
1->3->2->5->4->2
Printing Adjacency List graph3.data :
1 -> 2 -> 3 -> 4
2 -> 1 -> 5 -> 6
3 -> 1 -> 4 -> 7
4 -> 1 -> 3 -> 9
5 -> 2 -> 6
6 -> 2 -> 5 -> 10
7 -> 3 -> 8 -> 11
8 -> 7 -> 11
9 -> 4 -> 10 -> 12
10 -> 6 -> 9 -> 12
11 -> 7 -> 8 -> 12
12 -> 9 -> 10 -> 11


Can graph be drawn in one stroke: no
Printing Adjacency List graph4.data :
1 -> 2 -> 3
2 -> 1 -> 4 -> 7
3 -> 1 -> 4 -> 5 -> 6
4 -> 2 -> 3 -> 5 -> 6
5 -> 3 -> 4 -> 6
6 -> 3 -> 4 -> 5 -> 7
7 -> 2 -> 6


Can graph be drawn in one stroke: yes
2->7->6->5->4->6->3->4->2->1->3
Printing Adjacency List graph5.data :
1 -> 2 -> 3
2 -> 1 -> 3 -> 4 -> 5
3 -> 1 -> 2
4 -> 2 -> 5
5 -> 2 -> 4


Can graph be drawn in one stroke: yes
1->3->2->5->4->2
Printing Adjacency List graph6.data :
1 -> 2 -> 3 -> 4
2 -> 1 -> 3 -> 4 -> 5
3 -> 1 -> 2 -> 5
4 -> 1 -> 2 -> 5
5 -> 2 -> 3 -> 4


Can graph be drawn in one stroke: no
```