CSCE 221 Cover Page
Programming Assignment #6

First Name: Pratik Last Name: Patel UIN: 527004337

User Name: p.pratik99 E-mail address: p.pratik99@tamu.edu

Please list all sources in the table below including web pages which you used to solve or implement the current homework. If you fail to cite sources you can get a lower number of points or even zero. According to the University Regulations, Section 42, scholastic dishonesty are including: acquiring answers from any unauthorized source, working with another person when not specifically permitted, observing the work of other students during any exam, providing answers when not specifically authorized to do so, informing any person of the contents of an exam prior to the exam, and failing to credit sources used. Disciplinary actions range from grade penalties to expulsion, read more: Aggie Honor System Office

Type of sources			
People			
Web pages (provide URL)			
Printed material			
Other Sources			

I certify that I have listed all the sources that I used to develop the solutions/codes to the submitted work.

“On my honor as an Aggie, I have neither given nor received any unauthorized help on this academic work”

Your Name: Pratik Patel Date: 04/28/2019

Report

(a) The description of the data structures implemented in your program.

I have implemented an undirected graph data structure using adjacency matrix. It has n number of vertices and m number of edges between them. I have also used a stack to keep track of the order of the vertices in which a one stroke drawing might be possible.

(b) The necessary and sufficient conditions for drawing one-stroke pictures.

Assume that there are n vertices in the undirected graph. If n or $(n-2)$ number of vertices have even number of edges, then there is a possible Eulerian train. Otherwise, there is no Eulerian trail possible.

(c) Description of the algorithm and its running time.

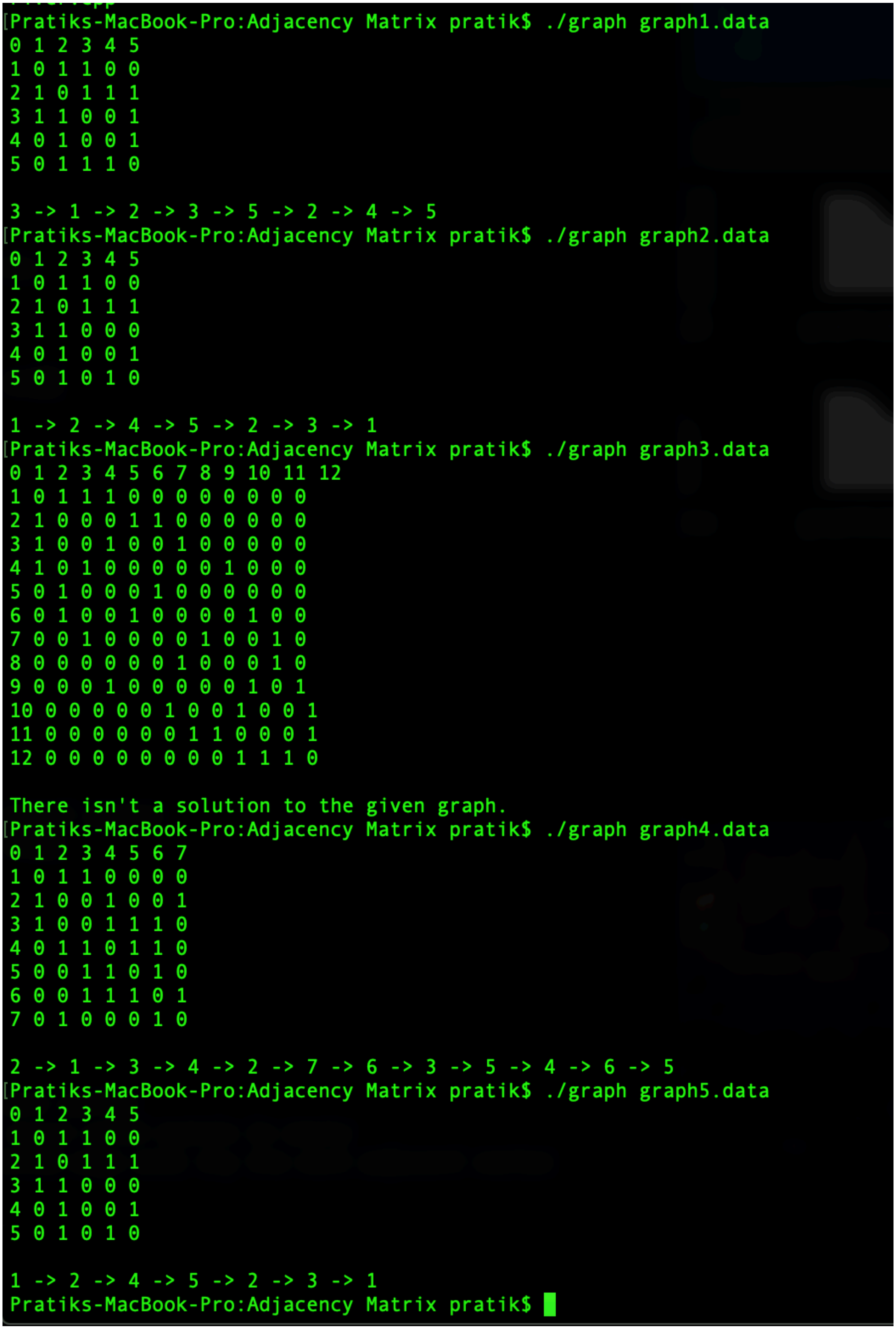
Following are the running times of various functions implemented in the Graph class.

$$Assume\ that\ there\ are\ n\ vertices\ and\ m\ edges.$$

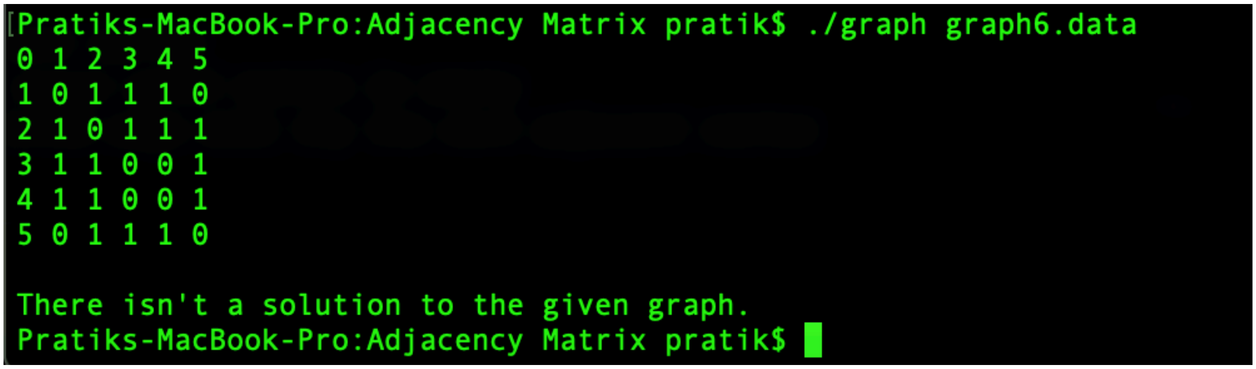
I. searchSolution(int u) – $O(mn^2)$
II. printGraph() – $O(n^2)$
III. printSolution() – $O(m)$
IV. isThereASolution() – $O(n)$
V. makeGraph(ifstream& inFile) – $O(n^2)$

(d) The evidence of testing your program for correctness.

I have attached some screenshots of the output on the following page.



```
[Pratiks-MacBook-Pro:Adjacency Matrix pratik$ ./graph graph1.data
0 1 2 3 4 5
1 0 1 1 0 0
2 1 0 1 1 1
3 1 1 0 0 1
4 0 1 0 0 1
5 0 1 1 1 0

3 -> 1 -> 2 -> 3 -> 5 -> 2 -> 4 -> 5
[Pratiks-MacBook-Pro:Adjacency Matrix pratik$ ./graph graph2.data
0 1 2 3 4 5
1 0 1 1 0 0
2 1 0 1 1 1
3 1 1 0 0 0
4 0 1 0 0 1
5 0 1 0 1 0

1 -> 2 -> 4 -> 5 -> 2 -> 3 -> 1
[Pratiks-MacBook-Pro:Adjacency Matrix pratik$ ./graph graph3.data
0 1 2 3 4 5 6 7 8 9 10 11 12
1 0 1 1 1 0 0 0 0 0 0 0 0
2 1 0 0 0 1 1 0 0 0 0 0 0
3 1 0 0 1 0 0 1 0 0 0 0 0
4 1 0 1 0 0 0 0 0 1 0 0 0
5 0 1 0 0 0 1 0 0 0 0 0 0
6 0 1 0 0 1 0 0 0 0 1 0 0
7 0 0 1 0 0 0 0 1 0 0 1 0
8 0 0 0 0 0 0 1 0 0 0 1 0
9 0 0 0 1 0 0 0 0 0 1 0 1
10 0 0 0 0 0 1 0 0 1 0 0 1
11 0 0 0 0 0 0 1 1 0 0 0 1
12 0 0 0 0 0 0 0 0 1 1 1 0

There isn't a solution to the given graph.
[Pratiks-MacBook-Pro:Adjacency Matrix pratik$ ./graph graph4.data
0 1 2 3 4 5 6 7
1 0 1 1 0 0 0 0
2 1 0 0 1 0 0 1
3 1 0 0 1 1 1 0
4 0 1 1 0 1 1 0
5 0 0 1 1 0 1 0
6 0 0 1 1 1 0 1
7 0 1 0 0 0 1 0

2 -> 1 -> 3 -> 4 -> 2 -> 7 -> 6 -> 3 -> 5 -> 4 -> 6 -> 5
[Pratiks-MacBook-Pro:Adjacency Matrix pratik$ ./graph graph5.data
0 1 2 3 4 5
1 0 1 1 0 0
2 1 0 1 1 1
3 1 1 0 0 0
4 0 1 0 0 1
5 0 1 0 1 0

1 -> 2 -> 4 -> 5 -> 2 -> 3 -> 1
Pratiks-MacBook-Pro:Adjacency Matrix pratik$ █
```



```
[Pratiks-MacBook-Pro:Adjacency Matrix pratik$ ./graph graph6.data
 0 1 2 3 4 5
 1 0 1 1 1 0
 2 1 0 1 1 1
 3 1 1 0 0 1
 4 1 1 0 0 1
 5 0 1 1 1 0

 There isn't a solution to the given graph.
 Pratiks-MacBook-Pro:Adjacency Matrix pratik$ █
```