CSCE 221 Cover Page

Programming Assignment #6

Due **December 6th** by midnight to CSNet

First Name: Quinn Last Name: Nguyen UIN: 524002419

User Name:quinnminh_nguyen

E-mail address: quinnminh_nguyen@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, read more: Aggie Honor System Office

Type of sources			
People	Peer TA		
Web pages (provide URL)	http://www.cplusplus.com/		
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: Quinn Nguyen Date: 12/04/2017

Report

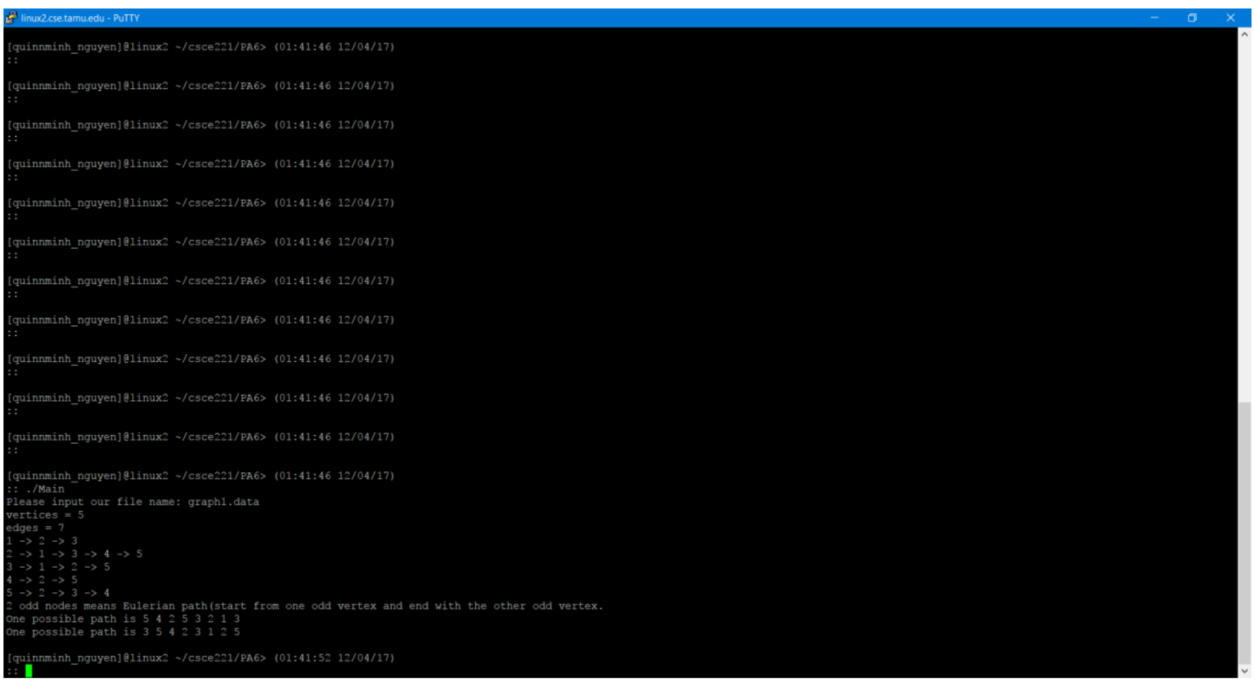
Data Structure: I use 2D vector to represent an adjacent matrix and a stack in order to keep track path that I draw.

Sufficient conditions for drawing one-stroke:

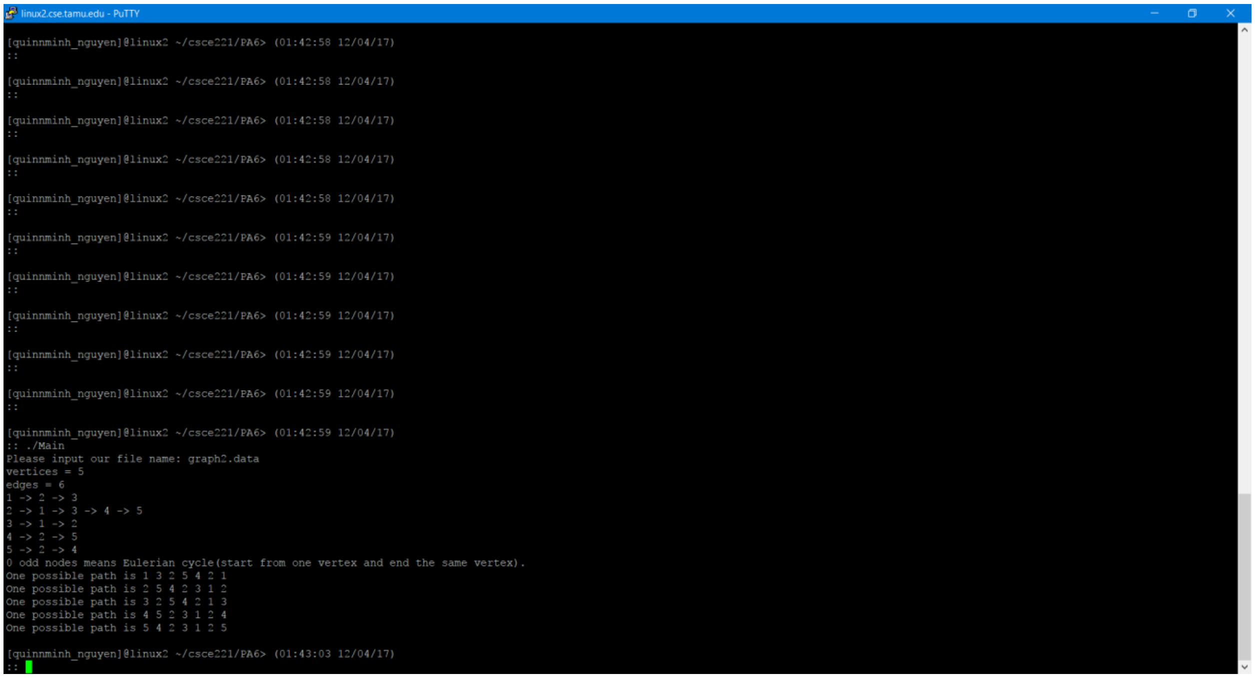
1) If there is no edge that has odd number of vertices, there must exist at least one solution that the starting point and ending point are the same (Eulerian circuit).
2) If there is exactly 2 edge that has odd number of vertices, there must exist at least one solution that starts from one odd vertex and ends with the other odd vertex.
3) Any other conditions will fail to have a solution.

The first algorithm determines if a solution exists. It has to iterate 2D adjacent matrix to determine if there are vertices having odd edges. Therefore, this algorithm should take $O(V^2)$ where V is the number of vertices.

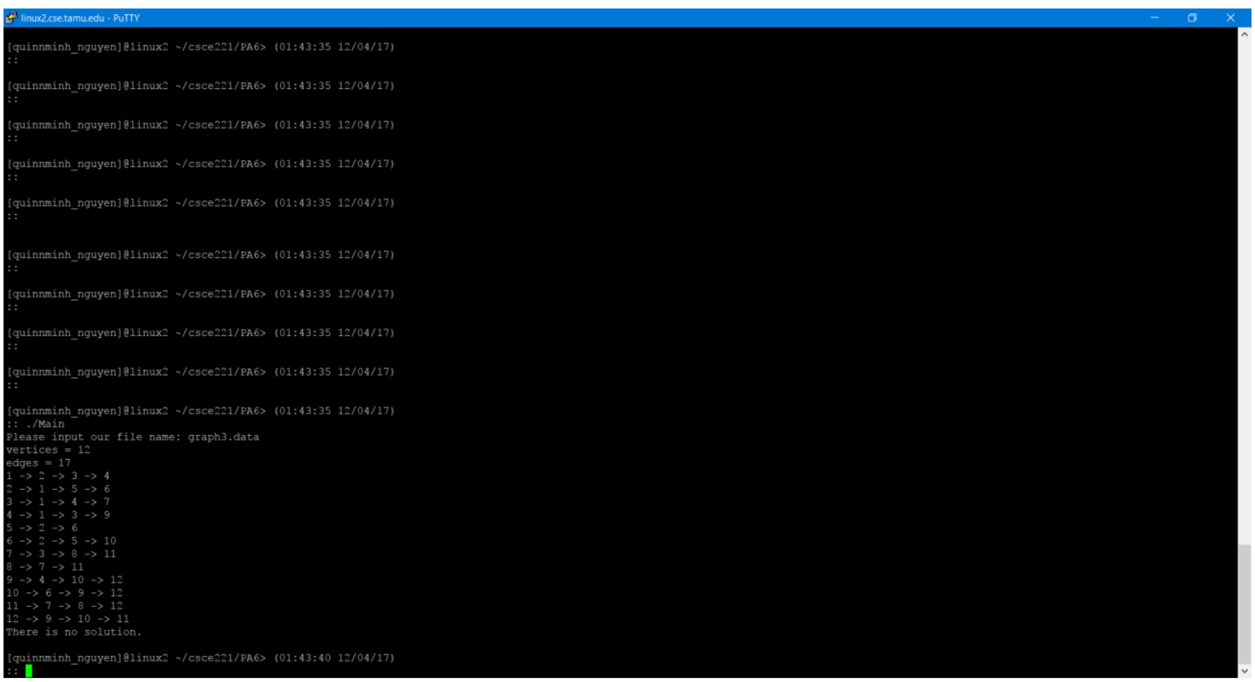
The second algorithm determines how to draw such a solution. It works like a DFS search. This means that it goes through to the next lowest adjacent node and deletes the edges between the nodes. It does this until there is no way to go; Then, if all vertices and edges are visited, then there exists a solution. If not, it re-creates the edge connection so that it can find another path. Repeat all of the above steps until it finds one. If there is no possible way to find a path, then there is NO solution. The running time of this algorithm Is $O(|V|^2)$ where $|V|$ is the number of vertices.



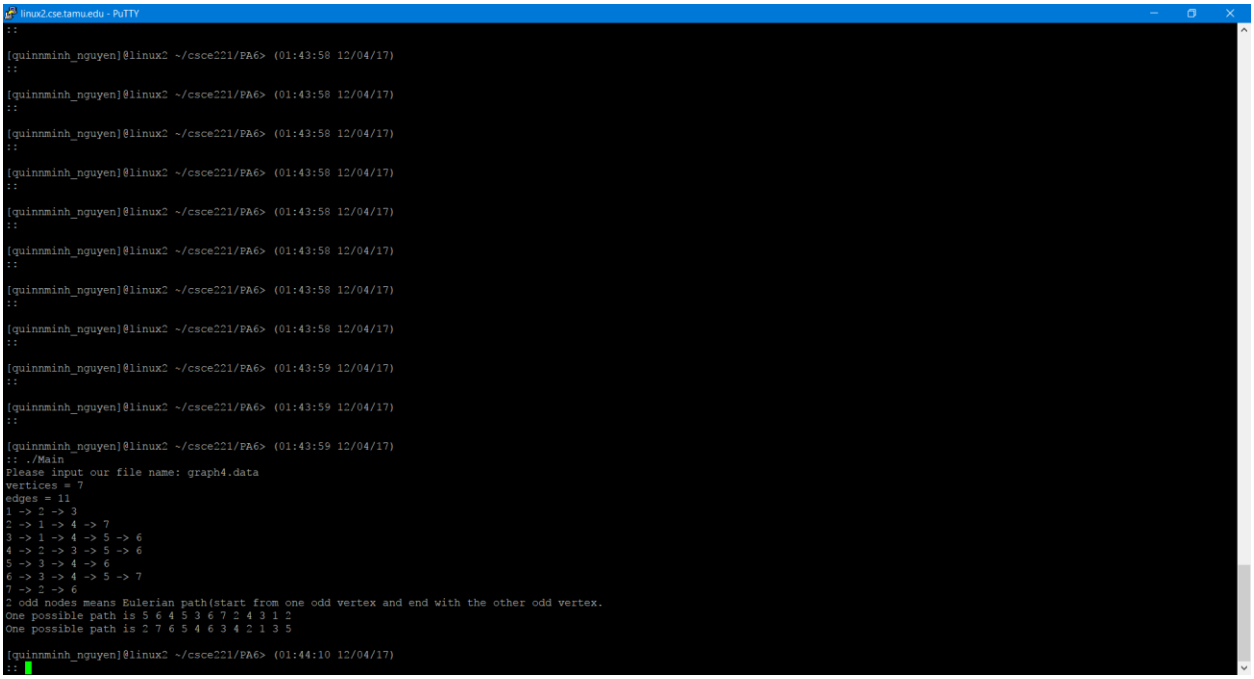
Graph1.data



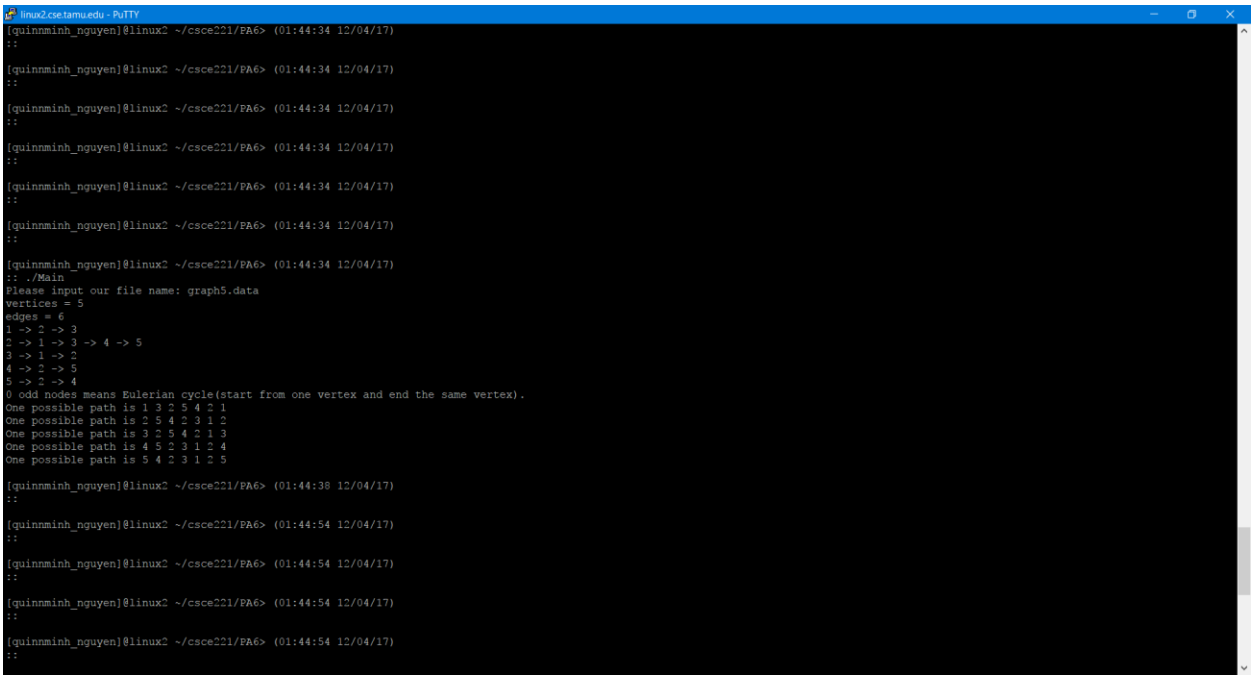
Graph2.data



Graph3.data



Graph4.data



Graph5.data

linux2.cse.tamu.edu - PuTTY

```
::

[quinnminh_nguyen]@linux2 ~/csce221/PA6> (01:44:54 12/04/17)
::

[quinnminh_nguyen]@linux2 ~/csce221/PA6> (01:44:54 12/04/17)
::

[quinnminh_nguyen]@linux2 ~/csce221/PA6> (01:44:54 12/04/17)
::

[quinnminh_nguyen]@linux2 ~/csce221/PA6> (01:44:54 12/04/17)
::

[quinnminh_nguyen]@linux2 ~/csce221/PA6> (01:44:54 12/04/17)
::

[quinnminh_nguyen]@linux2 ~/csce221/PA6> (01:44:54 12/04/17)
::

[quinnminh_nguyen]@linux2 ~/csce221/PA6> (01:44:54 12/04/17)
::

[quinnminh_nguyen]@linux2 ~/csce221/PA6> (01:44:54 12/04/17)
::

[quinnminh_nguyen]@linux2 ~/csce221/PA6> (01:44:54 12/04/17)
::

[quinnminh_nguyen]@linux2 ~/csce221/PA6> (01:44:55 12/04/17)
::

[quinnminh_nguyen]@linux2 ~/csce221/PA6> (01:44:55 12/04/17)
::

[quinnminh_nguyen]@linux2 ~/csce221/PA6> (01:44:55 12/04/17)
:: ./Main
Please input our file name: graph6.data
vertices = 5
edges = 8
1 -> 2 -> 3 -> 4
2 -> 1 -> 3 -> 4 -> 5
3 -> 1 -> 2 -> 5
4 -> 1 -> 2 -> 5
5 -> 2 -> 3 -> 4
There is no solution.

[quinnminh_nguyen]@linux2 ~/csce221/PA6> (01:45:01 12/04/17)
::
```

Graph6.data