

Homework 7

Nguyen Tien Duc ITITI18029

December 15, 2019

Page 608-609

6

Consider each person at the party a vertex. If that person shakes hands with another person, an edge is created between 2 persons.

The degree of each vertex is the number of people the person represents that vertex shakes hands with. By Theorem 1, the sum of the degrees is twice the amount of edges, so the sum of the number of handshakes each person makes is twice the total number of handshakes.

Therefore, the total number of people a person has shaken hands with, over the set of people at the party, is even.

10

Exercise 7

$$\text{In-degrees} = 3 + 1 + 2 + 1 = 7$$

$$\text{Out-degrees} = 1 + 2 + 1 + 3 = 7$$

And the number of edges is 7

Exercise 8

$$\text{In-degrees} = 2 + 3 + 2 + 1 = 8$$

$$\text{Out-degrees} = 2 + 4 + 1 + 1 = 8$$

And the number of edges is 8

Exercise 9

$$\text{In-degrees} = 6 + 1 + 2 + 4 + 0 = 13$$

$$\text{Out-degrees} = 1 + 5 + 5 + 2 + 0 = 13$$

And the number of edges is 13

12

Degree of a vertex represents the number of acquaintanceship a person has, or the number of people a person knows.

Isolated vertices represents people who knows nobody else.

Pendant vertices represents people who knows only one other person.

If the average degree of a vertex is 1000, then an average person knows 1000 different people.

Page 618-619

6

$$\begin{bmatrix} & a & b & c & d & e \\ a & 0 & 1 & 0 & 1 & 0 \\ b & 1 & 0 & 0 & 1 & 1 \\ c & 0 & 0 & 0 & 1 & 1 \\ d & 1 & 1 & 1 & 0 & 0 \\ e & 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$

8

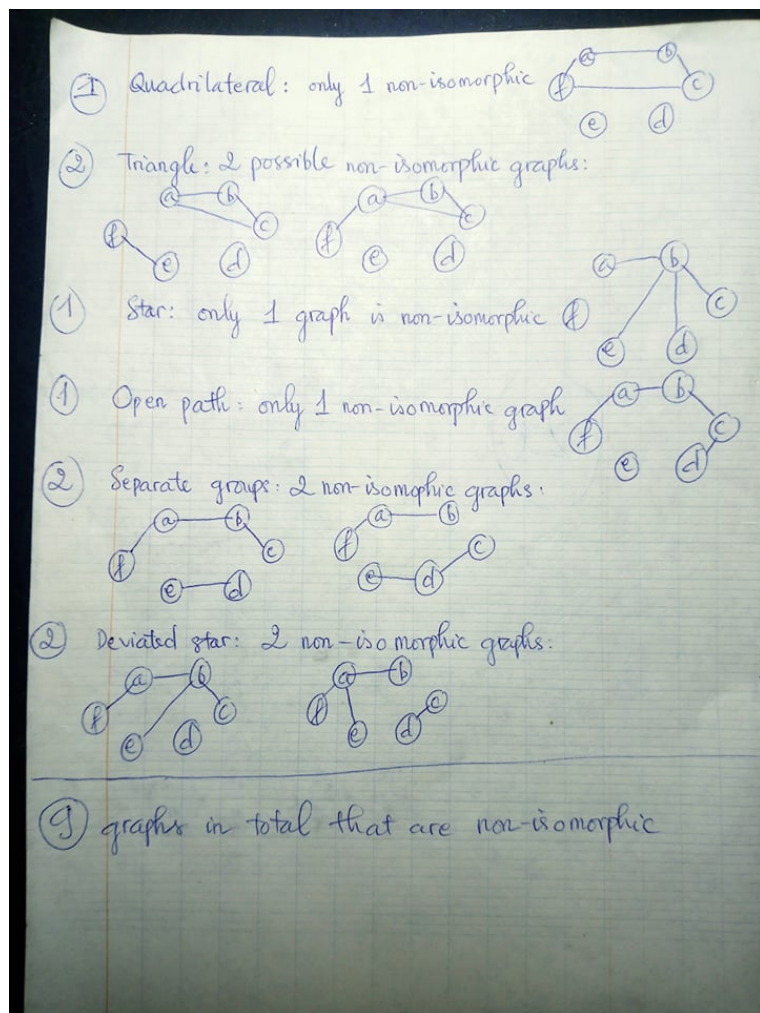
$$\begin{bmatrix} & a & b & c & d & e \\ a & 0 & 1 & 0 & 1 & 0 \\ b & 1 & 0 & 1 & 1 & 1 \\ c & 0 & 1 & 1 & 0 & 0 \\ d & 1 & 0 & 0 & 0 & 1 \\ e & 0 & 0 & 1 & 0 & 1 \end{bmatrix}$$

Page 644-645

10

We can model this problem using graph theory, the vertices are the lands:

- Vertice N: northern bank
- Vertice S: southern bank
- Vertice L: left island
- Vertice R: right island



$$\begin{bmatrix} & a & b & c & d \\ a & 0 & 3 & 0 & 1 \\ b & 3 & 0 & 1 & 0 \\ c & 0 & 1 & 0 & 3 \\ d & 1 & 0 & 3 & 0 \end{bmatrix}$$

The bridges are the edges between vertices.

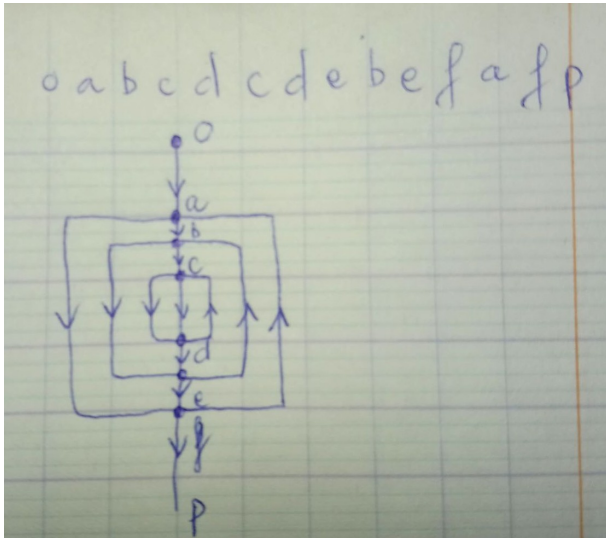
$$\begin{bmatrix} & N & S & L & R \\ N & 0 & 1 & 1 & 0 \\ S & 1 & 0 & 2 & 1 \\ L & 1 & 2 & 0 & 1 \\ R & 0 & 1 & 1 & 0 \end{bmatrix}$$

As shown in the matrix, each of the piece of land is connected to an even number of other lands.

Therefore, there must be an Euler circuit for this graph.

In conclusion, a person can cross all bridges exactly once and return to the starting point.

14



We can represent the picture as a graph. As shown the above figure, there are 8 vertices.

All of the vertices have equal number of in-degrees and out-degrees except for only 2 vertices, o and p.

Therefore, there is no Euler circuit for this graph but there is still an Euler path.

Therefore, we can draw this picture in a continuous motion without lifting the pencil.

16

Suppose the directed multigraph has an Euler circuit. Then, the graph must be strongly connected, thus,

also weakly connected.

We can count the in-degrees and out-degrees by following the circuit: as the circuit pass through a vertex, it adds one to the count of both the in-degrees and out-degrees as it comes in and out of every vertex. Therefore, number of in and out degrees of each vertex are equal.

Reversely, if the go against what have been stated, the number of in and out degrees are different, then the circuit can just go in but cannot leave a vertex or cannot even go in a vertex.

Therefore, the statement is proved.

22

Since vertex b has 4 in-degrees and 3 out-degrees and vertex c has 2 in-degrees, 3 out-degrees, this graph firstly does not satisfy the condition in problem 16.// However, all the other vertices have the same number of in-degrees and out-degrees, thus making the graph satisfy the condition in problem 17.// Therefore, the graph has an Euler path but not an Euler circuit.// As c has more outs than ins while b has more ins than outs, the Euler path must go in the direction from c to b.

One Euler path is: c e b d c b f d e f e a f a b c b

Euler Circuit Algorithm

Code:

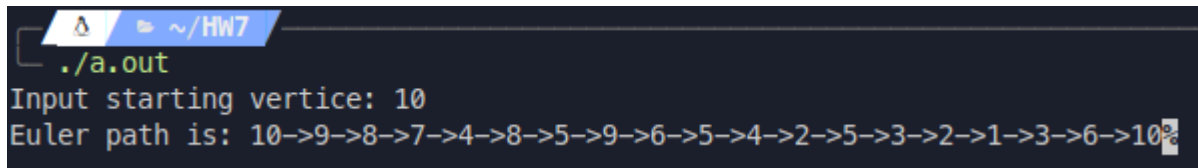
```
1 #include <iostream>
2 #include <vector>
3 using namespace std;
4
5 void findEulerPath(int vertices, int* edges, bool directed, int start,
6                   vector<int>& answer) {
7     for (int next = 0; next < vertices; next++) {
8         if ((*((edges + start * vertices) + next) > 0) {
9             ((*((edges + start * vertices) + next) -= 1;
10             if (!directed) ((*((edges + next * vertices) + start) -= 1;
11             findEulerPath(vertices, edges, directed, next, answer);
12         }
13     }
14     answer.push_back(start);
```

```

15 }
16
17 int main(int argc, char const* argv[]) {
18     const int vertices = 10;
19     int edges[vertices][vertices] = {
20         {0, 1, 1, 0, 0, 0, 0, 0, 0, 0}, {1, 0, 1, 1, 1, 0, 0, 0, 0, 0},
21         {1, 1, 0, 0, 1, 1, 0, 0, 0, 0}, {0, 1, 0, 0, 1, 0, 1, 1, 0, 0},
22         {0, 1, 1, 1, 0, 1, 0, 1, 1, 0}, {0, 0, 1, 0, 1, 0, 0, 0, 1, 1},
23         {0, 0, 0, 1, 0, 0, 0, 1, 0, 0}, {0, 0, 0, 1, 1, 0, 1, 0, 1, 0},
24         {0, 0, 0, 0, 1, 1, 0, 1, 0, 1}, {0, 0, 0, 0, 0, 1, 0, 0, 1, 0}};
25
26     vector<int> EulerCircuit;
27     int start_vertice;
28     cout << "Input starting vertice: ";
29     cin >> start_vertice;
30     findEulerPath(vertices, (int*)edges, false, start_vertice - 1, EulerCircuit);
31
32     cout << "Euler path is: ";
33     for (int i = 0; i < EulerCircuit.size(); i++) {
34         cout << EulerCircuit[i] + 1;
35         if (i < EulerCircuit.size() - 1) cout << "->";
36     }
37
38     return 0;
39 }

```

Demo:



```

~/HW7
./a.out
Input starting vertice: 10
Euler path is: 10->9->8->7->4->8->5->9->6->5->4->2->5->3->2->1->3->6->10%

```