Finding the faces

Mudit Rathore[1], Ishani Mishra[2], and Gugloth Lavanya[3]

[1]Indian Institute of Information Technology Allahabad, ICM2015502
[2]Indian Institute of Information Technology Allahabad, IWM2015008
[3]Indian Institute of Information Technology Allahabad, ISM2015003

***Abstract*— The digital age has set in, all tasks are performed by machines computers, robots etc. All problems being solved can be seen as graph problems, therefore it is the need of the hour to aid our machines with graph solving skills. Planarity is a concept in graph theory. A graph is said to be planar when it can be drawn with no edges crossing over each other. A planar graph divides the plane into regions (bounded by the edges), called faces. Planar graphs are frequently used to analyze morphological properties of networks occurring in complex systems. In these networks, an important task is to successfully extract the faces. Thus finding faces can be helpful in solving the above mentioned problems. This paper deals with an algorithm to find out the faces of a graph given its adjacency matrix representation.**

I. INTRODUCTION

A Graph [1] ***G*** is defined as an ordered pair of a finite set ***V*** of vertices and a finite set ***E*** of edges, where a vertex is a node and an edge is a connection between any two vertices. A graph can be represented using Adjacency Matrix and Incidence Matrix.

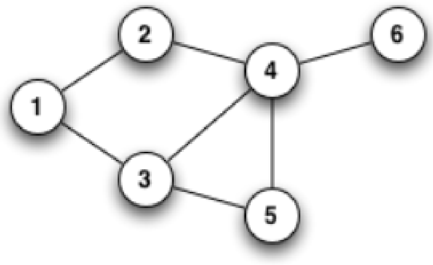
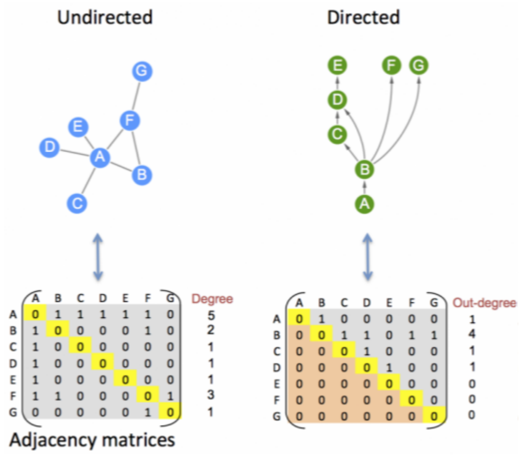


Fig. 1. A simple undirected graph

Adjacency is a relation between two vertices of a graph. A vertex v is adjacent to vertex u if there is an edge from vertex u to vertex v i.e. edge(u,v) $\in$ E.

An *adjacency matrix* is a matrix holding this relation. It is a square matrix A used to represent a finite graph. The elements of the matrix indicate whether pairs of vertices (v_i, v_j) are adjacent or not, in the graph. If vertex vi is adjacent to vertex v_j, then a_{ij} = 1 and a_{ji} = 1, if they are not connected then a_{ij} = 0 and a_{ji} = 0. In the special case of a finite simple graph, the adjacency matrix is a (0,1)-matrix of order $|V|$ with zeros on its diagonal a_{ii} = 0 which represents loop condition.



	A	B	C	D	E	F	G	Degree
A	0	1	1	1	1	1	0	5
B	1	0	0	0	0	1	0	2
C	1	0	0	0	0	0	0	1
D	1	0	0	0	0	0	0	1
E	1	0	0	0	0	0	0	1
F	1	1	0	0	0	0	1	3
G	0	0	0	0	0	1	0	1

	A	B	C	D	E	F	G	Out-degree
A	0	1	0	0	0	0	0	1
B	0	0	1	1	0	1	1	4
C	0	0	0	1	0	0	0	1
D	0	0	0	0	1	0	0	1
E	0	0	0	0	0	0	0	0
F	0	0	0	0	0	0	0	0
G	0	0	0	0	0	0	0	0

Fig. 2. Adjacency Matrix

When a connected graph can be drawn without any edges crossing, it is called planar [2]. When a planar graph [3] is drawn in this way, it divides the plane into regions called *faces*. Hence faces of a graph may be defined as the regions within the graph, divided by the edges of a planar graph.

In the graph shown below(Fig. 3.), there are three faces:

1) 2-5-4-3-2

2) 2-5-4-1-2
3) 0-1-4-3-0

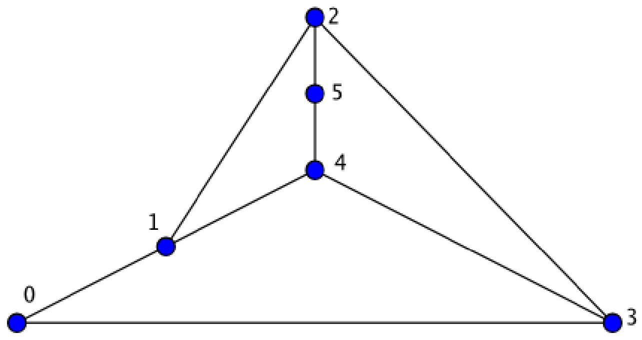


Fig. 3.

The graph shown below does not look planar because edges at the diagonals seem to be crossing each other.

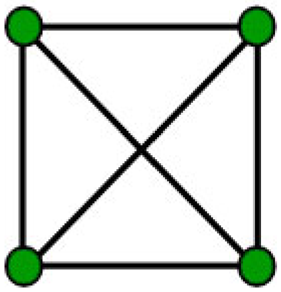


Fig. 4. A graph

But the graph is planar, just as we can see below:

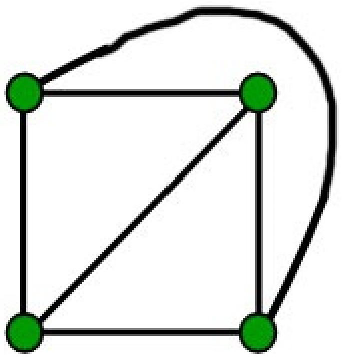


Fig. 5. Planar Graph

Since the point of intersection of the diagonals is not a vertex, the edge can be stretched out of the 4 edges bounds of the square and can be drawn from outside. Note that the edge shifted still connects the same pair of vertices.

Faces can now be computed easily. If we name the four vertices as A B C and D, taken clockwise from top left, the faces are:

- A B D A
- A B C A
- B C D A
- A B C D A

Euler's Formula for planar graphs
For any connected planar graph G = (V, E), the following formula holds [4]:

$$V + F - E = 2$$

where F stands for the number of faces.

For any disconnected planar graph G = (V, E) with k components, the following formula holds:

$$V + F - E = 1 + k$$

II. METHODOLOGY

The paper deals with finding the faces of a graph given its adjacency matrix.

A. *Input*

The user is made to input the number of nodes and the adjacency matrix of the graph.

B. *Proposed Algorithm*

The first algorithm finds all possible circuits [5] from all vertices one by one, finally considering non redundant minimal circuits or cycles from all vertices.

1) Select a vertex.
2) Find all cycles from the vertex. Once all cycles from a vertex have been found out, remove the vertex. So now cycles are found from other vertices will not include the previously removed vertices.
3) Keep storing cycle in a vector.
4) Repeat step 2 - 3 for all vertices.
5) Sort the cycles in accordance to cycle length.
6) Consider the shortest cycle.
7) Add it to face vector if it hasn't appeared before in the vector.
8) Consider the next cycle from the sorted list of cycles.
9) Repeat steps 7 – 8 till all cycles have been considered.
10) At the end, non redundant unique cycles will be in the face array and Those will be all the faces.

11) The overall face can be added which is actually the longest cycle containing all vertices.

- *Finding all possibles cycles*

The algorithm to find all possible cycles from all vertices. It is a recursive approach to identify all possible cycles present in the graph from every vertex to every other vertex. A vector declared globally stores all cycles.

1) Choose a vertex.
2) Choose adjacent vertices of a vertex.
3) Find the path between the vertex and above chosen adjacent vertex.
4) Store it in cycle vector along with the start vertex at the end of the above found path. This completes the cycle and stores it.

We repeat the same above approach for all vertices. In order to reach from start vertex to the destination vertex, we change the start vertex to the next reachable vertex in every function call while keeping the destination fixed in order to trace the path. Towards the end of the algorithm, when all cycles have been calculated and stored. Further computing the minimal unique cycle can be done as stated in the algorithm above.

1) Time Complexity: Time complexity of the above algorithm is *O(V^V)* because the maximum adjacency of a vertex can be V - 1. Thus finding cycles for each vertex, comparing all cycles with previously occurred faces involve V x V x V .... (V-1) times computations, hence the time complexity is *O(V^V)*.

2) Space Complexity: The space complexity for the adjacency matrix is *O(V x V)* and the final face matrix will be of *O(F x V)*, where F is the number of faces. Also we will need vectors to store cycles and paths but they are declared, used and destroyed in the local scope of the functions.

III. Results

The proposed algorithm is run on the graph shown above (Fig. 6.) and the following results were obtained:

```
Enter the total number of vertices: 6
Enter the (v x v) adjacency matrix of the graph
0 1 0 1 0 0
1 0 1 0 1 0
0 1 0 1 0 1
1 0 1 0 1 0
0 1 0 1 0 1
```

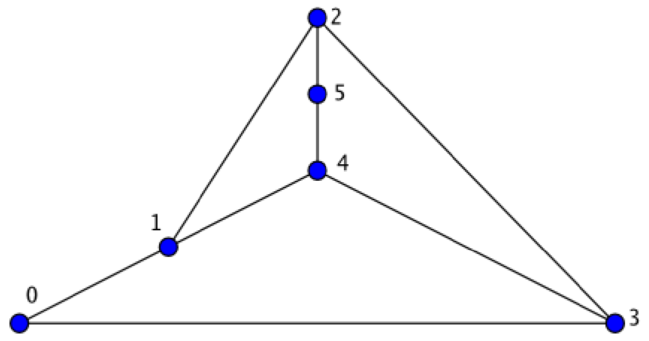


Fig. 6.

```
0 0 1 0 1 0
The faces in the graph are:
0 1 2 3 0
0 1 4 3 0
1 2 5 4 1
2 5 4 3 2
```

IV. Discussion

We can verify the above obtained results using the Euler's formula

$$V + F - E = 2$$

. In the graph taken above for results, There are 6 vertices, 8 edges. Total number of faces obtained are, as we can count, 4. Hence,

$$E = 8, V = 6, F = 4$$

$$V + F - E = 6 + 4 - 2 = 2 = RHS$$

Hence verified.

References

[1] J. Cook, M. Wootters, V. Williams, R. Verma, S. Hildick-Smith, and G. Valiant, *Graphs.* [On-line]. Available: https://web.stanford.edu/class/cs161/Lectures/Lecture9/CS161 Lecture09.pdf. [2017]

[2] V. Adamchik *Graph Theory* [On-line]. Available: http://www.cs.cmu.edu/adamchik/21-127/lectures/graphs.pdf [2005]

[3] R. J. Trudeau. *Introduction to graph theory.* Courier Corporation [2013].

[4] *Planar Graphs and Euler's Formula* [On-line]. Available: https://www.math.upenn.edu/ mlazar/math170/notes05-3.pdf [2016]

[5] S. Schneider, I. F. Sbalzarini *Finding faces in a planar embedding of a graph* [2015]