GRAPH THEORY

Prerequisites:

1. Stack
2. Queue
3. Priority Queues
4. Recursion
5. Functions

MOTIVATION:

We are going to learn about a new set of algorithms called graph algorithms. You will find these algorithms really helpful in solving a large number of problems. Let us see the following problem:

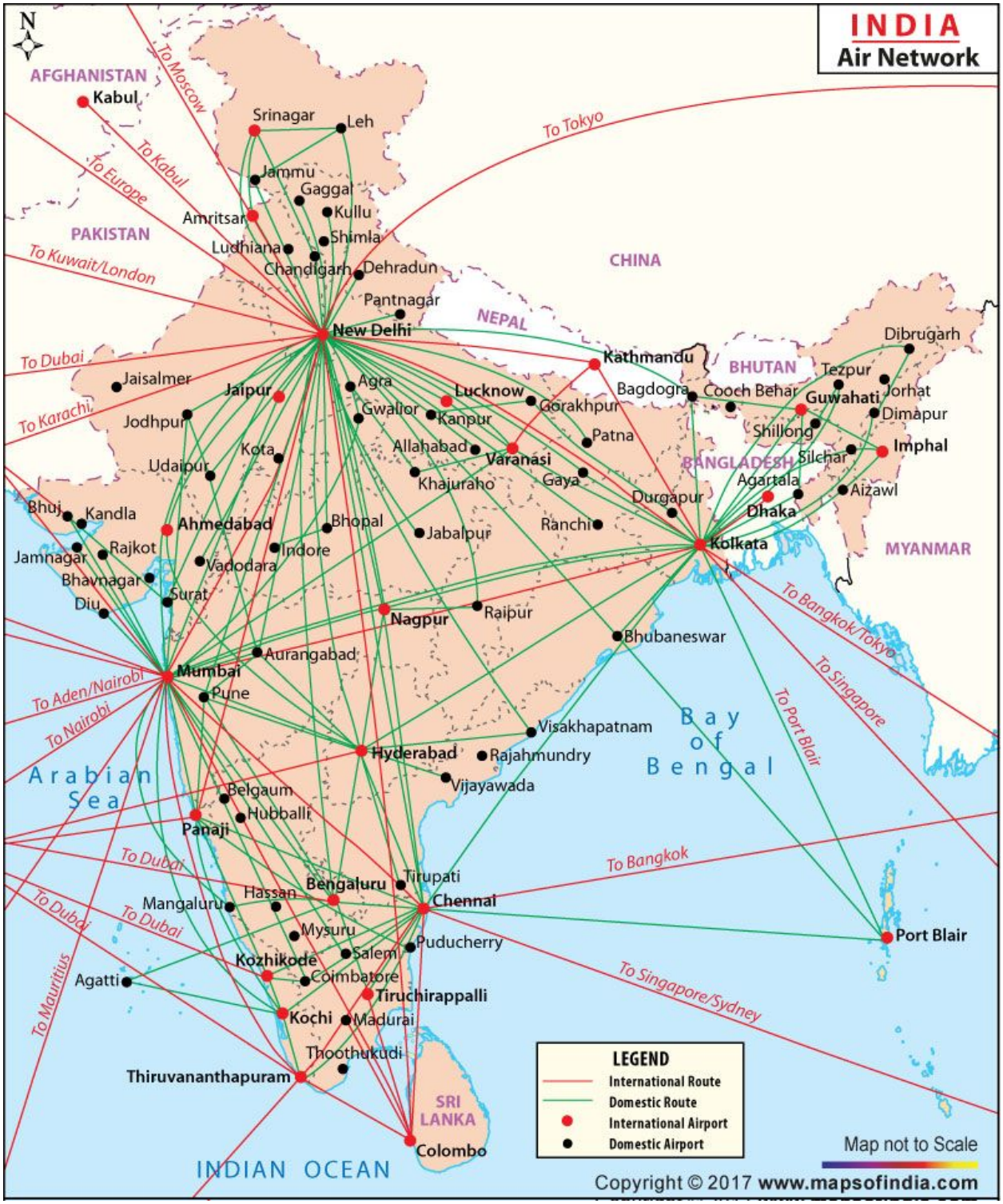
Observe the map of India given below. We have to colour the map in such a way that no two states side by side(adjacent) has the same colour. For example Maharashtra and Gujarat must be coloured using different colours. But Gujarat and West Bengal may be coloured using the same colour. This is done so that it is easy for us to distinguish between states. Now, we have to report the minimum number of colours required to colour the whole map satisfying the above criteria.

How do we solve such a problem?
Well, we need graphs.

If you think about it carefully, are the shapes of the states important to solve the problem?
No.
We just need to know whether a particular state is adjacent to some other state. So we can represent the states as small circles and if two states are adjacent in the map, then we draw a line connecting the two corresponding circles.
The lines are called edges and the small circles are called nodes. The picture we get using nodes and edges is called the graph.
So, now you can see how the map of India can be represented as a graph and later we will see how this model of graph will help us solve the mentioned problem.

Let us look at another problem.
Observe another map of India (in the next page). This map shows the cities connected by direct flights. Given two cities, we must be able to tell whether it is possible to go from that city to the other city by flights.(this can be direct or we can take multiple hopping flights as well). Again this problem looks to be a problem which can be solved using one of our graph theory algorithms. Like any graph problems, let us decide what our nodes will represent and what our edges will represent.
Hint: See the map!
The nodes are the cities and the edges between two nodes represent that there exists a direct flight between the two cities(nodes).

I hope you are slowly getting an idea about what type of problems can be solved using graph theory.


INDIA
Air Network
N
AFGHANISTAN
PAKISTAN
CHINA
NEPAL
BHUTAN
BANGLADESH
MYANMAR
SRI LANKA
Arabian Sea
Bay of Bengal
INDIAN OCEAN
To Moscow
To Kabul
To Europe
To Kuwait/London
To Dubai
To Karachi
To Tokyo
To Bangkok/Tokyo
To Singapore
To Port Blair
To Aden/Nairobi
To Nairobi
To Bangkok
To Mauritius
To Singapore/Sydney
Kabul
Srinagar
Leh
Jammu
Gaggal
Kullu
Amritsar
Shimla
Ludhiana
Chandigarh
Dehradun
Pantnagar
New Delhi
Kathmandu
Jaisalmer
Jaipur
Agra
Lucknow
Gwalior
Kanpur
Gorakhpur
Patna
Jodhpur
Kota
Allahabad
Varanasi
Udaipur
Khajuraho
Gaya
Bhuj
Kandla
Ahmedabad
Bhopal
Jabalpur
Ranchi
Jamnagar
Rajkot
Indore
Vadodara
Bhavnagar
Surat
Diu
Nagpur
Raipur
Aurangabad
Mumbai
Pune
Visakhapatnam
Hyderabad
Rajahmundry
Vijayawada
Belgaum
Hubballi
Panaji
Tirupati
Bengaluru
Hassan
Chennai
Mangaluru
Mysuru
Puducherry
Salem
Kozhikode
Coimbatore
Agatti
Tiruchirappalli
Kochi
Madurai
Thoothukudi
Thiruvananthapuram
Colombo
Dibrugarh
Tezpur
Bagdogra
Cooch Behar
Guwahati
Jorhat
Dimapur
Shillong
Imphal
Silchar
Agartala
Aizawl
Dhaka
Durgapur
Kolkata
Bhubaneswar
Port Blair
LEGEND
International Route
Domestic Route
International Airport
Domestic Airport
Map not to Scale
Copyright © 2017 www.mapsofindia.com

Let us look at one more problem:
We all have used Google Maps. What does Google Map does? Given two places it shows the shortest path to be taken from the source to the destination. This shortest path can be in terms of distance or in terms of time. Again, a problem that can be solved using Graph Theory! Infact there are classic algorithms in Graph Theory called the shortest path algorithms We will be learning about these later.
How many places are there in the whole world? Well approx. 10^6 places. So, we need efficient algorithms to compute the shortest path in linear time. Don't worry we already have such efficient algorithms which we will learn.!

BONUS: http://www.iarcs.org.in/inoi/online-study-material/topics/graphs.php
***Watch the video at the bottom of the page.

TYPES OF GRAPHS:

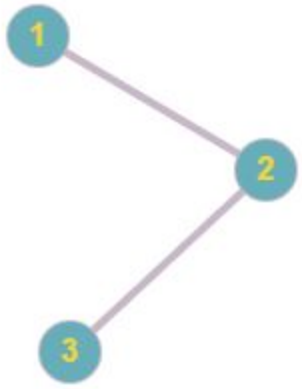
Without wasting time let us look at what types of graphs are there:
https://csacademy.com/lesson/introduction_to_graphs/

REPRESENTING GRAPHS:

Now let us look at the two different ways to represent graphs in a computer. It is very important to organise our problem in the form of graph and store it in the computer memory before proceeding to solve the problem.
Link: https://csacademy.com/lesson/graph_representation/
Also go through the following link. It basically covers almost the same things as in the previous link, but is slightly more in detail:
https://www.hackerearth.com/practice/algorithms/graphs/graph-representation/tutorial/

Exercise:
Given an undirected graph of N nodes and M edges, represent the graph in the computer using Adjacency lists. N and M are inputted by the user and so are the M edges.
Input Format:
3 2
1 2
2 3

N = 3 and M=2
Following M lines have two numbers i and j, indicating that there is an edge between nodes i and j.
The above example represents the following graph:



Hint: Use an array of vectors.
Solution: https://ideone.com/7wkmut

A more in detail introduction to graphs(Optional):
https://www.topcoder.com/community/data-science/data-science-tutorials/introduction-to-graphs-and-their-data-structures-section-1/

GRAPH SEARCH ALGORITHMS:

Let us head straight away to learn our first two of the most popular graph search algorithms called the Breadth First Search(BFS) and then the Depth First Search(DFS).
These are algorithms that explores nodes in a certain way(which you will see), making sure it does not visit any vertex twice.
These algorithms are required to solve many graph theory problems. In almost any graph theory problem you have to run a depth first search or breadth first search to extract some information(for example to count the number of nodes).
Go through the following links:

BFS :-
Document: https://www.hackerearth.com/practice/algorithms/graphs/breadth-first-search/tutorial/
Video: http://www.iarcs.org.in/inoi/online-study-material/topics/graphs-bfs.php

DFS :-
Document: https://www.hackerearth.com/practice/algorithms/graphs/depth-first-search/tutorial/
Video: http://www.iarcs.org.in/inoi/online-study-material/topics/graphs-dfs.php

Reference(Optional):
https://www.topcoder.com/community/data-science/data-science-tutorials/introduction-to-graphs-and-their-data-structures-section-2/

Visualizations:

BFS - https://www.cs.usfca.edu/~galles/visualization/BFS.html
DFS - https://www.cs.usfca.edu/~galles/visualization/DFS.html
Instructions:

1. After the radio buttons are appropriately selected, enter a vertex where you want to start your BFS/DFS from in the text field on the top left corner of the screen.
2. Click Run BFS/DFS.

At this point you should be able to implement your own BFS and DFS algorithms and be able to represent problems using graphs.
Remember, it requires a lot of practice to recognize a problem that can be solved using graphs, for a beginner. You will find that there are so many problems which can be solved just by tweaking the above two algorithms.

SHORTEST PATH ALGORITHMS(OPTIONAL):
These algorithms are slightly advanced and we do not intend to cover these in detail. So in case you are interested here are some links in no particular order.

1. http://www.iarcs.org.in/inoi/online-study-material/topics/dijkstra.php
2. http://www.iarcs.org.in/inoi/online-study-material/topics/floyd-warshall.php
3. https://www.topcoder.com/community/data-science/data-science-tutorials/introduction-to-graphs-and-their-data-structures-section-3/

Visualizations:
Dijkstra's Algorithm- https://www.cs.usfca.edu/~galles/visualization/Dijkstra.html
Floyd Warshall: https://www.cs.usfca.edu/~galles/visualization/Floyd.html

Now you know enough to solve most problems that require graph theory. I leave you to practice these algorithms on your own!