Graph Theory - I

Lecture material:
https://drive.google.com/file/d/1OkLO91GV4M23e-1EtvFr9yeGWgk6V91q/view?usp=sharing

For the questions that ask you to propose/present an algorithm, you can write either a programmable code or a pseudocode or step-by-step instructions.

Problem #1:
Answer the following general questions regarding the basic terminology of Graph Theory:

1. In a directed graph, if there is an edge from Node A to Node B, and not the other way around, can A and B be called adjacent/neighbor to each other?
2. In an undirected graph of n vertices, how many edges can we construct without allowing loops (edge to self) and multiple edges between a pair of vertices?
3. Same question for directed graphs.
4. How many edges can we construct in an undirected graph with n vertices without introducing any cycle? Loops can also be considered as cycles.
5. Same question for directed graphs.
6. Take any graph, try to construct a path that starts from a vertex, then visits all the edges of the graph exactly once. It will not be possible in all graphs, only in some having a special property. What's that property?
7. Take any graph, try to construct a path that starts from a vertex, then visits all other vertices of the graph exactly once (some edges might be unvisited). Like the previous question, it will not be possible in all graphs. Just find some examples with both cases.
8. Analyze the efficiency of using adjacency list over adjacency matrix for different types of graphs: undirected, directed, weighted, unweighted, sparse, dense…

Problem #2:
Suppose we have the following social network, Friendsbook where:

Person A is friends with Person B, C, D, and E.
Person B is friends with Person A, C, and F.
Person C is friends with Person A, B, D, and G.
Person D is friends with Person A, C, E, G, and H.
Person E is friends with Person A, D, and H.
Person F is friends with Person B and G.
Person G is friends with Person C, D, F, and H.
Person H is friends with Person D, E, and G.

A. Draw a graph to represent the network using each user as a node or vertex and the friendships as edges between the nodes.
B. Create the adjacency list and adjacency matrix for the graph representation

C. For each pair of users, calculate the number of friends they have in mutual.
D. In Friendsbook, a user can share posts. Suppose the CEO of Friendsbook hired you for developing their post display system. You were given the task to implement a system that displays posts from users who are at most two degrees of connection away from the one who is scrolling his feed. After looking at the requirements you came up with an idea to implement the system using the BFS algorithm. Now, for person F, simulate your algorithm. Can F see the posts of all other users in his feed?

Problem #3:

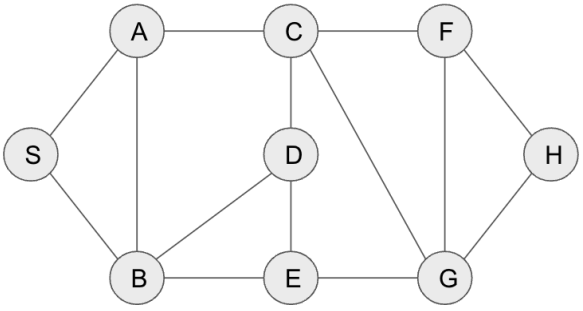
The following information is stored after implementing the DFS algorithm on an undirected graph.

Nodes	1	2	3	4	5	6	7
Parent							
Starting Time	8	11	5	6	4	1	2
Finish Time	9	12	10	7	13	14	3
Distance from Root							

A. Draw the DFS tree.
B. Fill up the empty rows of the table.

Problem #4:

Consider this graph:

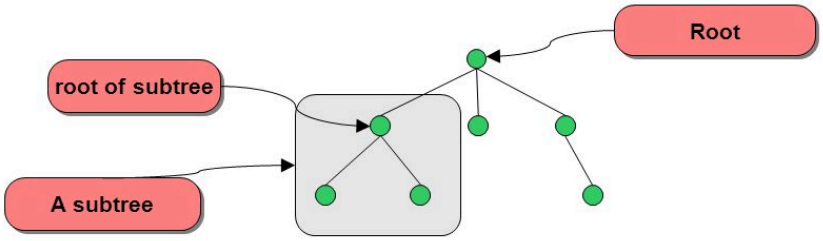


A. Justify $\sum_{v \in V} deg(v) = 2m$, where m = number of edges in the graph.
B. Calculate the number of additional edges that can be added between the nodes randomly, keeping the graph 'simple' (without adding multiple edges between any two nodes, loops in a single node).

Problem #5:

Trees and subtrees

- In a rooted tree, any node is a root of a subtree consisting of itself and the nodes "below" it
- By definition there is only one path between the root and each of the other nodes.
  - Because a root is also a node the main property applies
- The convention used in computer science is that roots are drawn on the top. It may seem strange at the beginning but you'll get used to it



Source: https://slideplayer.com/slide/7322529/

You are given a Binary Tree with N nodes. That means each of the nodes has zero to two children, and its own subtree. Nodes are numbered from 1 to N.
You will be given Q queries. Each of them tells you a node number P, and asks you to find the size of its subtree (number of nodes in the subtree including P as the root).

i. How would you build the adjacency list/matrix of this tree? Show with an example tree.
ii. You have to answer each query in constant time, *O(1)*. However, you are allowed to do a preprocessing of *O(V+E)*. Propose a suitable algorithm that satisfies this.

Problem #6:

Suppose you are given a directed graph where the number of vertices is V and the number of edges is E. The vertices are numbered from 1 to V. You are given the task to find out whether vertex '1' is reachable from all the other vertices, i.e. there exist paths to go from '2' to '1', '3' to '1', … and so on.
Propose an $O(V+E)$ algorithm to find the answer to this question.
Clarification: You cannot run DFS from each source. That would take the complexity to $O(V*E)$.

The End