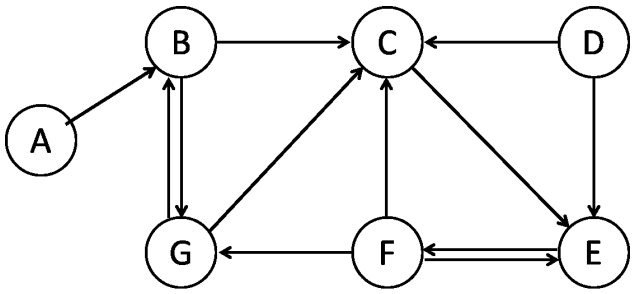
You're expected to work on the discussion problems before coming to the lab. Discussion session is not meant to be a lecture. TA will guide the discussion and correct your solutions if needed. We will not release 'official' solutions. If you're better prepared for discussion, you will learn more. TAs will record names of the students who actively engage in discussion and report them to the instructor; they are also allowed to give some extra points to those students at their discretion. The instructor will factor in participation in final grade.

1. (basic) We're given a directed graph G, along with a pair of vertices, u and v. We would like to know if there is a path from u to v. What algorithm would you like to use?
   Sol. Run either BFS or DFS with u as the starting vertex. If we use BFS, v is reachable from u if and only if BFS will eventually visit v (or equivalently, $v.d < \infty$). If we use DFS, v is reachable from u if and only if v is a descendant of u in the DFF (depth first forest).

2. (basic) To implement BFS, what data structure do you use? What about DFS (if we use no recursion)?
   Sol. We use queue for BFS, and stack for DFS.

3. (basic) What is the definition of BFT (breadth first tree)? Suppose the graph in consideration is undirected. Could there be an edge between two vertices whose depths differ by more than one?
   Sol. Def of BFT. It is tree with a source vertex (root) s. And the unique path on the tree from s to each vertex is a shortest path from s to v. No, it's not possible. There can be edges vertices of an equal depth or depths differing by one, but not more than one.

4. (basic) BFS and DFS. Consider the following directed graph.



   (a) Draw the adjacency-list representation of G as in fig. 22.2, with each list sorted in increasing alphabetical order.

   (b) Give the adjacency matrix of G.

   (c) Draw the graph, the adjacency-list representation (with each list sorted in increasing alphabetical order), and the adjacency matrix for the transpose graph G^T.

(d) Do depth-first search in G, considering vertices in increasing alphabetical order. Show the final result, with vertices labeled with their starting and finishing times, and edges labeled with their type (T/B/F/C) as in fig. 22.5(a).

(e) Based on your results, proceed to run the algorithm in p. 617 to find the strongly connected components of G (show the result of the DFS in G^T, with vertices labeled with their starting and finishing times).

(f) Draw the component graph G^{SCC} of G.

(g) Find a topological sort of G^{SCC} using the algorithm in p. 613 (label each vertex with its DFS finishing time).

(h) Run BFS with B as starting vertex. Show the tree edges produced by BFS along with $v.d$ of each vertex v as in Fig 22.3. You must draw the current tree edges in each iteration together with the queue status as in Fig 22.3. More precisley, run BFS in p. 595 with B as starting point assuming that each adjacency list is sorted in increasing alphabetical order.

5. (Intermediate/Advanced) We know that one can use DFS to detect a cycle (if it exists) – the graph has a cycle if and only if it has a back edge. Then, one student got curious if she can use BFS to detect a cycle. After a moment's thought, she claimed: Say the graph is directed, and all vertices are reachable from the source s. If we run BFS with s as the starting point, then the graph has a cycle if and only if there is an edge (u, v) such that $u.d > v.d$. Prove or disprove this claim.

   Sol. The claim is false. say 1 is the source, we have edges (1, 2), (2, 3), (3, 4), and (1, 4). This graph has no cycle, but edge (3,4) satisfies the condition stated in the claim.

6. (Intermediate/Advanced) Prove the correctness of the Topological-Sort algorithm. Recall that the algorithm orders vertices in the decreasing order of their finish times that are produced by DFS.

   Sol. See Topological Sort Correctness from the lecture slides.

7. (Basic) If u is reachable from v, u must be a descendant of v in any DFF.

   Sol. Not true. Say the graph is only one edge, (u, v). Then, the DFF could be just two vertices u and v, with no edge.