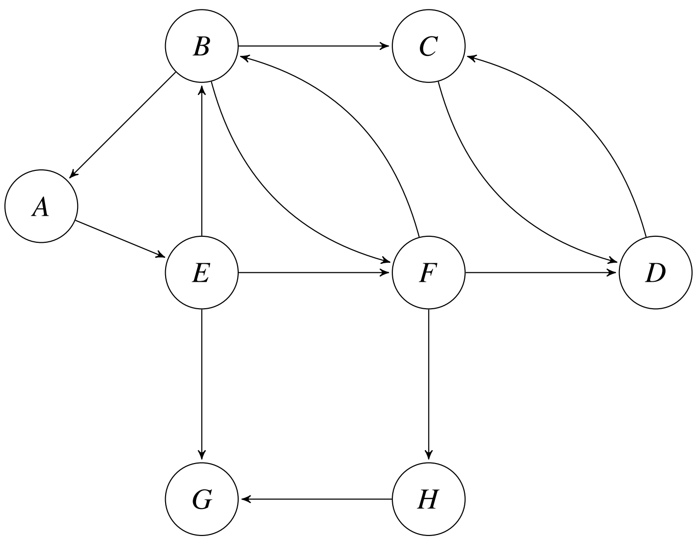
CMPSC 465 Fall 2025	Data Structures & Algorithms Ke Chen and Yana Safonova	Worksheet 5

Monday, Oct 06, 2025

1. **SCC.** Run the strongly connected components algorithm on the following directed graph G. When doing DFS on G^R: whenever there is a choice of vertices to explore, always pick the one that is alphabetically first.



   (a) Give the pre and post number of each vertex in the reverse graph G^R.

   (b) In what order are the connected components found?

   (c) Which are source connected components and which are sink connected components?

   (d) Draw the "metagraph" (each meta-node is a connected component of G).

2. **Edges for SCC.** Let $G = (V, E)$ be a finite directed graph. Determine the minimum number of edges that must be added to G so that the resulting graph becomes strongly connected.

3. **Multi Source Shortest Path** Let $G = (V, E)$ be an unweighted graph (directed or undirected) and let $S \subseteq V$ be a set of source vertices. For every vertex $v \in V$ compute

$$d(v) = \min_{s \in S} \text{dist}_G(s, v),$$

   the minimum number of edges from v to the closest source.

4. **Dijkstra with Edge Budget** Let $G = (V, E)$ be a directed graph with nonnegative edge weights, a source vertex $s \in V$, and an integer $k \leq |V| - 1$. The task is to compute the shortest path distance from s to every vertex $v \in V$ such that no path uses more than k edges. Design an algorithm. Prove that it is correct and analyse the runtime complexity.

5. **Dijkstra with Maximum Edge Weight** Let $G = (V,E)$ be a directed graph with nonnegative edge weights. Given a source vertex $s \in V$ and a threshold $W_{\max} \geq 0$, a path is called *feasible* if all its edges have weights at most $W_{\max}$. Modify Dijkstra's algorithm to compute the shortest distances from s to all vertices along *feasible* paths only. Explain why your modification is correct and analyze its time complexity.

6. **Dijkstra with a Pinned Node** You are given a strongly connected directed graph $G = (V,E)$ with positive edge weights along with a particular node $v_0 \in V$. Give an efficient algorithm for finding the shortest path between *all pairs of nodes*, with the one restriction that these paths must all pass through v_0. Make your algorithm as efficient as you can, perhaps as fast as Dijkstra's algorithm. Explain why your algorithm is correct and justify its running time.