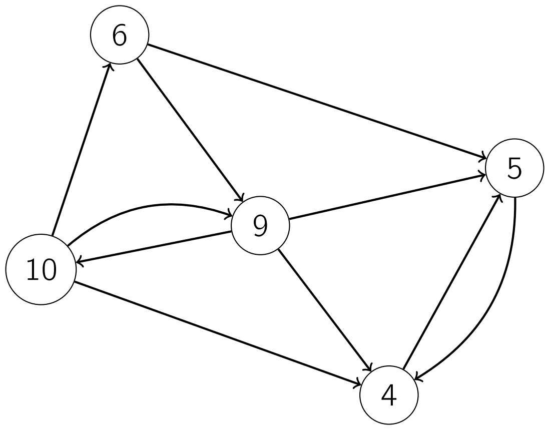
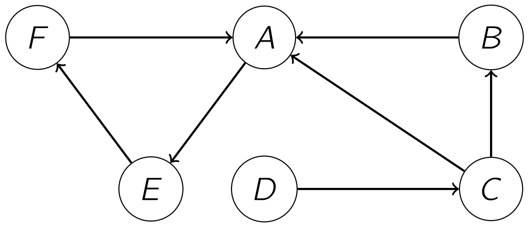
CS 161 (Stanford, Winter 2026) Section 6

1 SCC Exercises

1. In the given graph, the node labels represent the finish times from running depth-first search. Which node would the next DFS call begin from when running Kosaraju's algorithm? Perform this DFS (with edges reversed) to find the find the strongly connected components of the graph.



Consider the directed graph below for parts 2 and 3:



2. How many strongly connected components does this graph have?

3. What is the minimum number of directed edges to add to this graph to make all the vertices strongly connected?

2 SCC Concept Check

1. Assume you have two vertices u and v in a directed graph where there exists an edge from u to v. Which one of the following is incorrect about u and v?

   (A) u and v can be in the same SCC.

   (B) u and v can be in different SCCs.

(C) If u's DFS finish time is less than v's DFS finish time then u and v are in the same SCC.

(D) u's DFS finish time is always greater than v's DFS finish time.

3 Modified Kosaraju's Algorithm

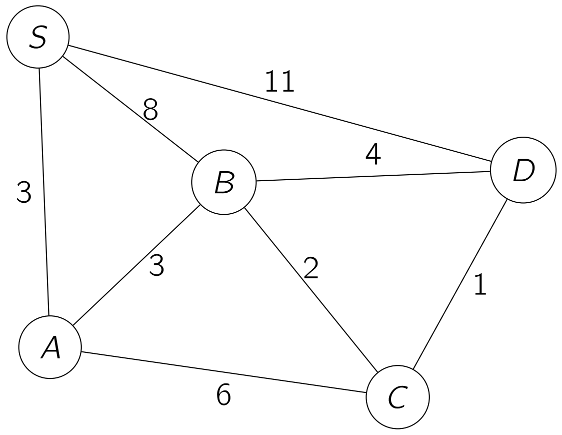
Kosaraju's algorithm for finding strongly connected components (SCCs) in a directed graph consists of two depth-first searches. First, the algorithm performs DFS on the original graph, keeping track of finishing times. Then, it performs a second round of DFS on the graph with all edges reversed, processing nodes in descending order of their finish times.

Lucky the Lackadaisical Lemur is left unconvinced by this last step. He suggests a new approach: instead of reversing the edges, why not just run the second DFS on the original graph but process nodes in **increasing** order of their finish times? This seems reasonable at first glance, but Plucky the Pedantic Penguin proclaims that something is problematic.

Help Plucky procure a counterexample that proves Lucky lacks correctness.

4 Finding Shortest Paths

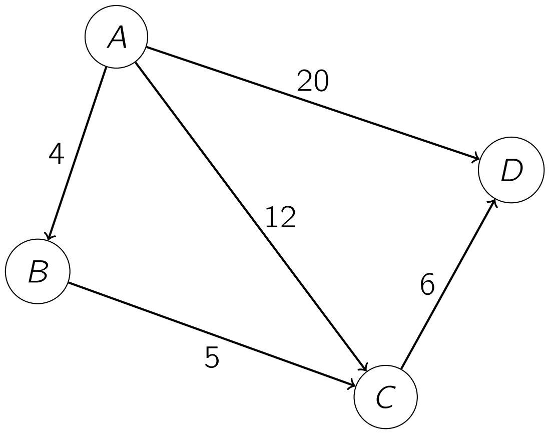
1. Perform Dijkstra's shortest path algorithm from source S on the graph below, and update the $d[v]$ values for each iteration in the table.



Vertex v	$d[v]$	$d[v]$	$d[v]$	$d[v]$	$d[v]$
S	0				
A	∞				
B	∞				
C	∞				
D	∞				

2. Given the directed graph below, run the Floyd-Warshall Algorithm, processing vertices in alphabetical order. Fill in the table below which keeps track of the shortest paths.

Ordered of vertices with no directed path (such as (B, A)) are omitted and their distance can be taken as ∞ for updates.



(u, v)	(A, A)	(A, B)	(A, C)	(A, D)	(B, B)	(B, C)	(B, D)	(C, C)	(C, D)	(D, D)
$D^{(0)}$	0	4	12	20	0	5	∞	0	6	0
$D^{(1)}$										
$D^{(2)}$										
$D^{(3)}$										
$D^{(4)}$										