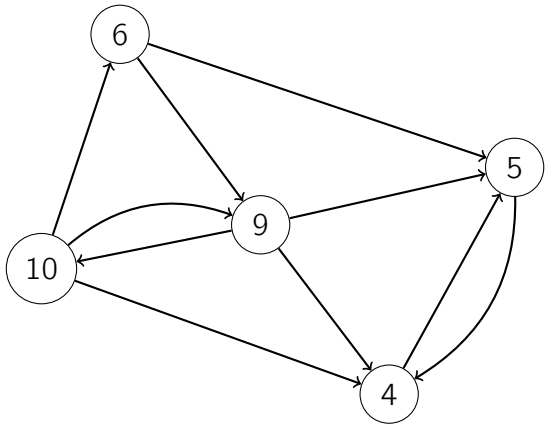
CS 161 (Stanford, Winter 2026) Section 6

1 SCC Exercises

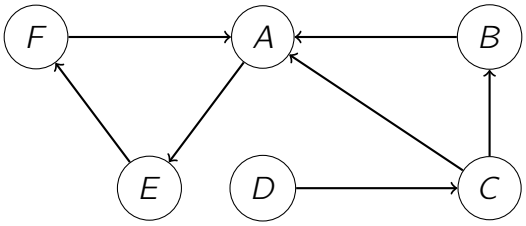
1. In the given graph, the node labels represent the finish times from running depth-first search. Which node would the next DFS call begin from when running Kosaraju's algorithm? Perform this DFS (with edges reversed) to find the find the strongly connected components of the graph.



Solution

The DFS would begin at the node labeled $\boxed{10}$ because it has the highest finishing time. Performing DFS with edges reversed, we explore 10 then 9 and then 6 before backtracking to 10. A second DFS begins at 5 and explores 4. The two SCCs found are $\boxed{\{6, 9, 10\}, \{4, 5\}}$.

Consider the directed graph below for parts 2 and 3:



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

Solution

There are $\boxed{4}$ SCCs, $\{A, F, E\}, \{B\}, \{C\}$, and $\{D\}$.

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

Solution

A directed edge from E to D makes all vertices strongly connected so only $\boxed{1}$ edge is needed.

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.

Solution

Answer $\boxed{\text{D}}$ is the only incorrect choice. The edge from A to E in the graph above gives an example of choice A, and the edge from C to A gives an example of choice B. If u and v are in different SCCs, then there cannot be a path from v to u. Therefore, when u is visited, either v has been visited (and finished since it cannot reach u) or v has not been visited (in which case it must be visited and finished before u can finish). Either way, u will have a greater finish time than v, so option C is a correct statement. Option D is not necessarily correct. For example, in a graph with just u and v and directed edges both directions, the relative finish time is based on the starting node.

3 Modified Kosaraju's Algorithm

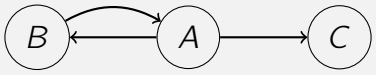
Kosaraju's algorithm for finding strongly connected components (SCCs) in a directed graph consists of two depth-first searches. First, the algorithm performs DFS on the original graph, keeping track of finishing times. Then, it performs a second round of DFS on the graph with all edges reversed, processing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

Solution

Consider the directed graph:



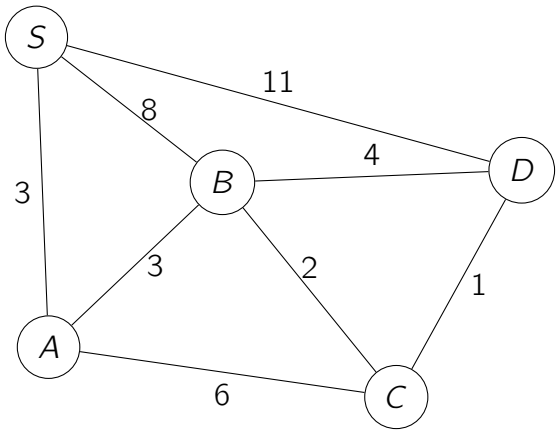
Assume we run the first DFS by processing vertices in the order A, B, C. Additionally, for Vertex A, in the neighbor exploration stage let us process B before C. This will result in the following finish times:

$$B.\text{finish} < C.\text{finish} < A.\text{finish}$$

Now, applying Lucky's approach, we run the second DFS on G without reversing edges, processing vertices in increasing order of their finishing times. The first DFS call will start with B, which reaches A, and then continues to C. This results in a single DFS tree containing {A, B, C}, which incorrectly identifies the entire set as a single SCC. This is a contradiction, since we can see that the SCCs are {a, b} and {c} in our original graph.

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.



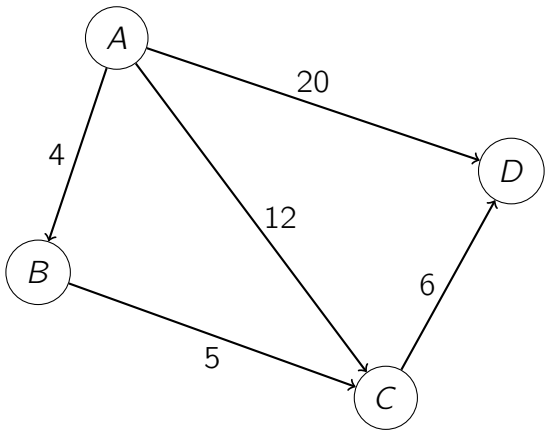
Vertex v	$d[v]$	$d[v]$	$d[v]$	$d[v]$	$d[v]$
S	0	0	0	0	0
A	∞				
B	∞				
C	∞				
D	∞				

Solution

The completed table is shown below:

Vertex v	$d[v]$	$d[v]$	$d[v]$	$d[v]$	$d[v]$
S	0	0	0	0	0
A	∞	3	3	3	3
B	∞	8	6	6	6
C	∞	∞	9	8	8
D	∞	11	11	10	9

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



(A, A)	(A, B)	(A, C)	(A, D)	(B, B)	(B, C)	(B, D)	(C, C)	(C, D)	(D, D)
0	4	12	20	0	5	∞	0	6	0
0				0			0		0
0				0			0		0
0				0			0		0
0				0			0		0

Solution

The completed table is shown below:

(A, A)	(A, B)	(A, C)	(A, D)	(B, B)	(B, C)	(B, D)	(C, C)	(C, D)	(D, D)
0	4	12	20	0	5	∞	0	6	0
0	4	12	20	0	5	∞	0	6	0
0	4	9	20	0	5	∞	0	6	0
0	4	9	15	0	5	11	0	6	0
0	4	9	15	0	5	11	0	6	0