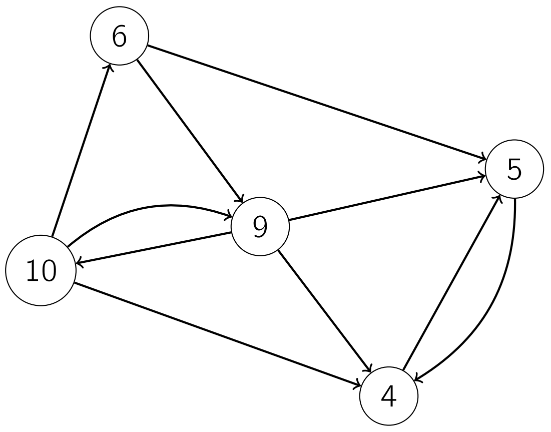
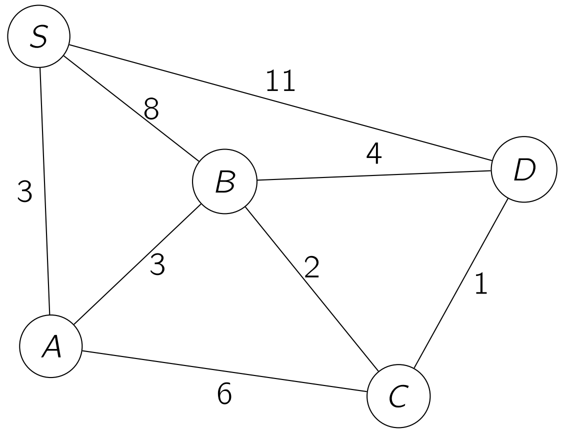
CS 161 (Stanford, Winter 2025) Section 6

1 Algorithm Practice

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.

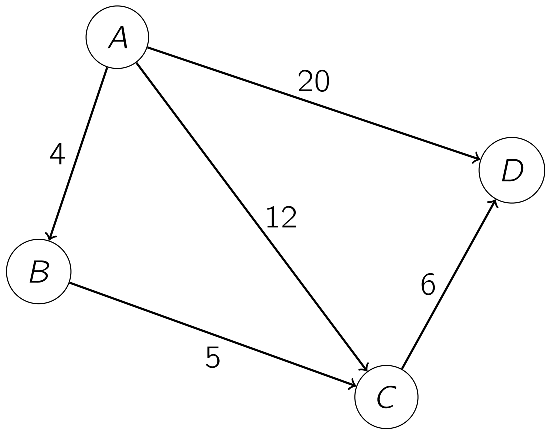


2. Perform Dijkstra's shortest path algorithm from source S on the graph below, and update the $d[v]$ values for each iteration in the table.



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

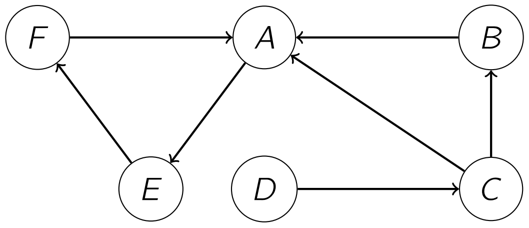
3. Given the directed graph below, run the Floyd-Warshall Algorithm, processing vertices in alphabetical order. Fill in the table below which keeps track of the shortest paths. Ordered of vertices with no directed path (such as (B, A)) are omitted and their distance can be taken as ∞ for updates.



(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
$D^{(1)}$										
$D^{(2)}$										
$D^{(3)}$										
$D^{(4)}$										

2 Strongly Connected Components

Consider the directed graph below for parts 1 and 2:



1. How many strongly connected components does this graph have?
2. What is the minimum number of directed edges to add to this graph to make all the vertices strongly connected?
3. 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 High Speed Cable Internet

Algorithmia, an internet service provider, has a new high speed cable internet technology that will require new cable installation. They will install these new cables on the currently existing network of cables but it will be costly.

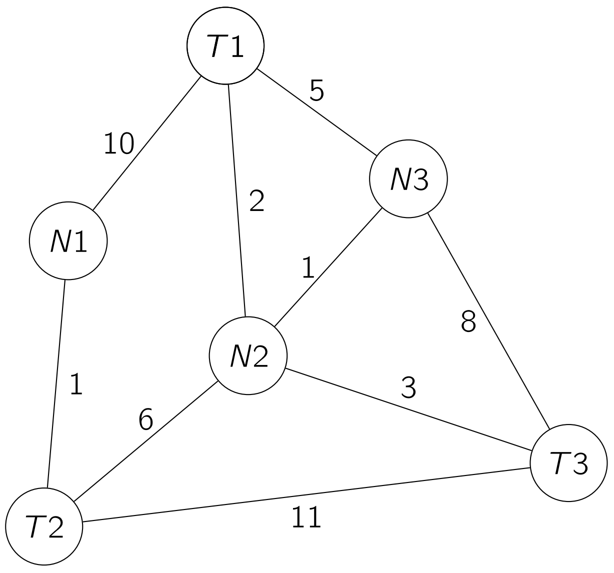
This can be modelled with a weighted undirected graph $G = (V, E)$ with non-negative edge weights. The nodes represent neighborhoods, edges represent the existing cables between the neighborhoods, and edge weights represent the cost to install a new cable.

Because of limited resources and to minimize costs, Algorithmia has chosen to start with the neighborhoods with highest demand for this high speed internet access that will lead to the highest profit, creating a set $T \subset V$ of terminals which includes the neighborhood with Algorithmia's headquarters along with the high demand neighborhoods.

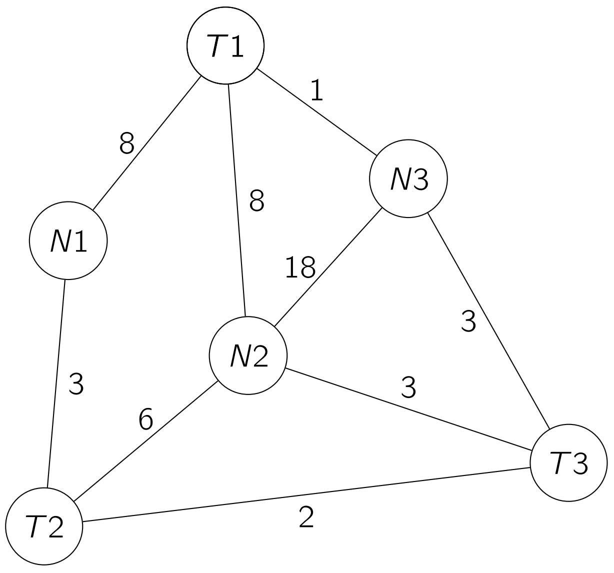
To connect all these neighborhoods, we can model this with a *Steiner tree*, a tree (aka graph with no cycles) that contains all of the terminals and possibly some other vertices. Algorithmia wants to find the minimum weight Steiner tree to find the lowest cost to install new cables that connect Algorithmia's headquarters and the high demand neighborhoods.

1. If there are only two terminals (Algorithmia's headquarters and another high demand neighborhood), give an $O(n\log(n) + m)$-time algorithm for finding a minimum weight Steiner tree. [**We are expecting:** *English description and brief running time analysis*]
2. For terminals $T1, T2, T3$, **draw** the minimum weight Steiner tree in each of the following graphs: [**We are expecting:** *A drawing of the Steiner trees.*] (You can use

copy+paste the tikz code on tex and delete the lines corresponding to edges that do not participate in the Steiner tree.)



(a)



(b)

3. Give an $O(n^2 \log(n) + nm)$-time algorithm for finding a minimum weight Steiner tree with **three terminals**. [**We are expecting:** *English description, pseudocode, and running time analysis*]

4. **Bonus:** Give an $O(n^2 \log(n) + nm)$-time algorithm for finding a minimum weight Steiner tree with **four terminals**. [**We are expecting:** *English description and brief running time analysis*]