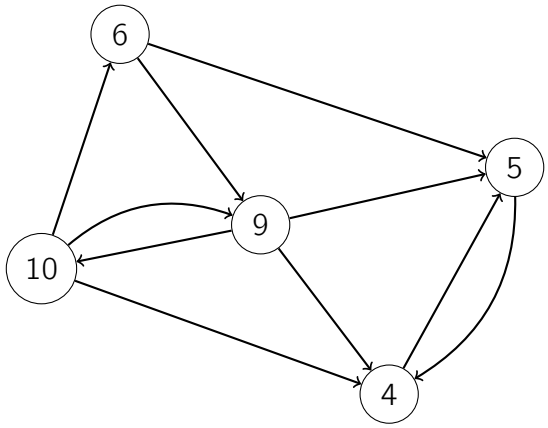
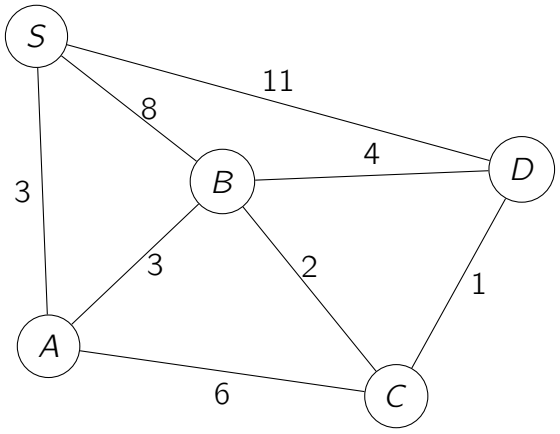
CS 161 (Stanford, Winter 2024) Section 6

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.

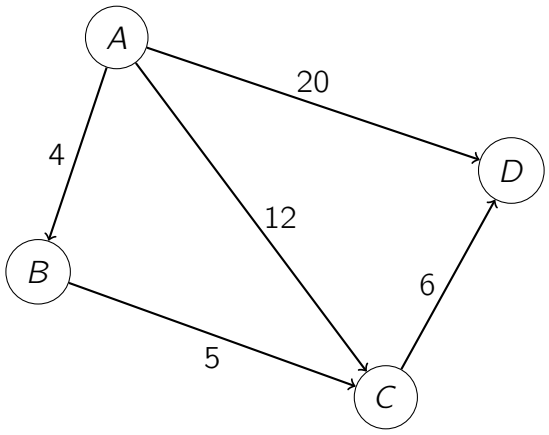


2. Perform Dijkstra's shortest path algorithm from source S on the graph below, and update the $d[v]$ values for each iteration in the table.



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

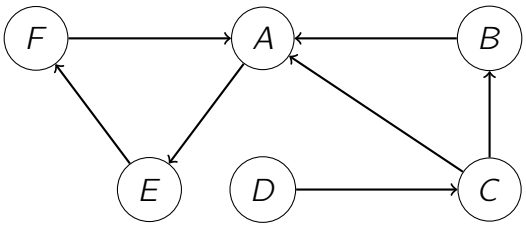
3. Given the directed graph below, run the Floyd-Warshall Algorithm, processing vertices in alphabetical order. Fill in the table below which keeps track of the shortest paths. Ordered of vertices with no directed path (such as (B, A)) are omitted and their distance can be taken as ∞ for updates.



(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.

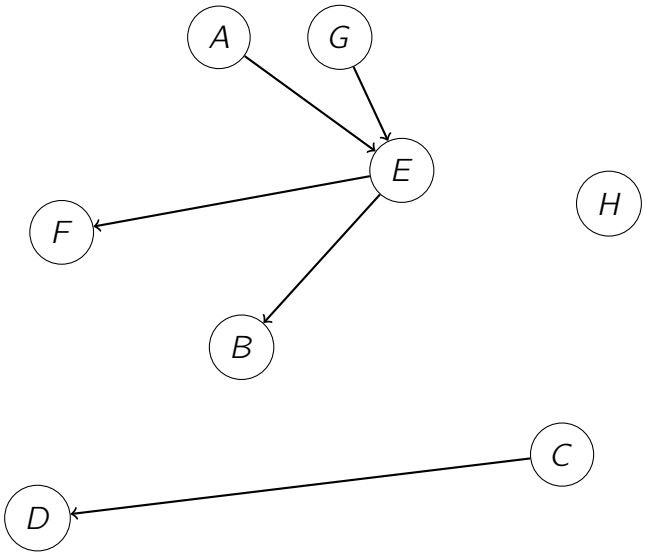
(C) If u's DFS finish time is less than v's DFS finish time then u and v are in the same SCC.

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

3 Hyperlinks can go backward?

On the internet, many pages have links pointing to other pages, but sometimes it's not possible to reach a site you were on previously without clicking the "back" button in your browser. Elgoog can model their website as a directed graph G with n pages, and each page has some number of links to other pages. There are a total of m links over all these pages. Currently, it's not possible to get from some pages to some other pages without clicking the back button, and sometimes not possible at all! They want your help in designing an algorithm which can output the **minimum** total number of extra links they need to add so that every page is reachable from every other page.

1. In the given graph, find the minimum number of links that Elgoog must add.



2. Suppose G is a directed, acyclic graph (DAG) where every sink is reachable from every source. Define $S \subseteq V$ as the set of *source* nodes: those vertices with no incoming edges and $T \subseteq V$ as the set of *sink* nodes: those vertices with no outgoing edges. **Prove** that the minimum number of links which have to be added is $\max(|S|, |T|)$.

3. Suppose G is a *weakly connected* directed, acyclic graph (DAG) with at least two vertices. Weakly connected means that every node is reachable from every other node by traversing edges in some direction (not necessarily the direction they point). In other words, replacing directed edges with undirected ones will produce a connected graph. ABEFG are weakly connected and CD are weakly connected, for example. Write an algorithm which runs in time $O(m+n)$ and computes the minimum number of links to be added in G.

4. Suppose G is a general directed graph which is not necessarily connected. **Prove** an expression for the minimum number of links which must be added.

   Hint: You will have to case on whether G has exactly one vertex or not.

4 Edsger's Apfelstrudel

You are eating at a cozy little restaurant which serves a *prix fixe* menu of $k + 1$ courses, with several available choices for each course. Each dish belongs to exactly one course (e.g., risotto can only be ordered as an appetizer, not a main), and you are effectively indifferent between most of the items on the menu (because they are all so tasty), but the main draw of this particular restaurant is that they serve a delicious 'bottomless' dessert: their world-famous Viennese-style apple strudel. They have an unlimited supply of this apple strudel, but each serving will still cost you \$1. The restaurant also has a few interesting rules:

1. You must finish your current dish before ordering another.
2. Each dish after the first course depends on what you ordered in the previous course, e.g., you can only order salmon for your main if you ordered a Caesar salad or chicken noodle soup for the previous course. You are told on the menu exactly what these restrictions are before you order anything.
3. Most importantly, you are not allowed to have their unlimited dessert unless you finish one dish from each of the first k courses!

You are told the cost of each item in each course on the menu, and you plan your meal with a twofold goal: to be able to order the strudel, but also to save as much money as possible throughout the first k courses so that you have more money to spend on the unlimited dessert. Design an algorithm to find the smallest amount of money you can spend on the first k courses and still order the 'bottomless' strudel. If you would like, you may assume the very first course has exactly one choice (e.g., a single complimentary leaf of spinach that costs 0 dollars).

5 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

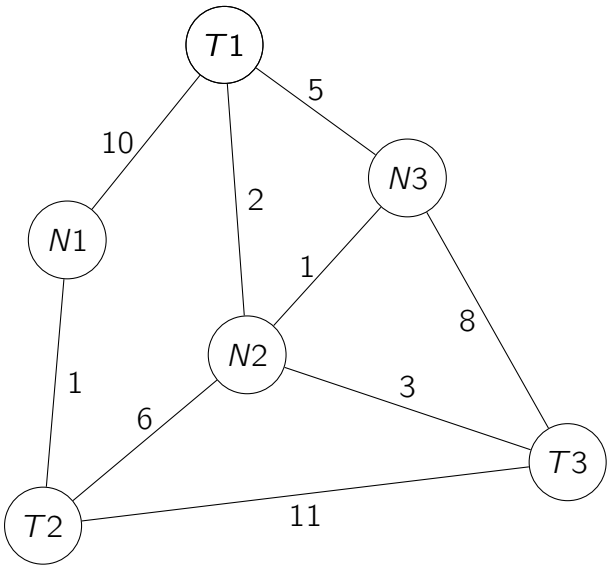
the highest profit, creating a set $T \subset V$ of terminals which includes the neighborhood with Algorithmia's headquarters along with the high demand neighborhoods.

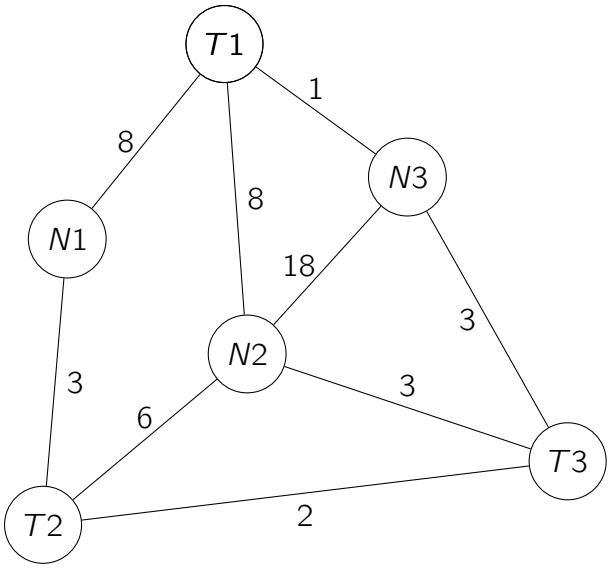
To connect all these neighborhoods, we can model this with a *Steiner tree*, a tree (aka graph with no cycles) that contains all of the terminals and possibly some other vertices. Algorithmia wants to find the minimum weight Steiner tree to find the lowest cost to install new cables that connect Algorithmia's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

6 Currency conversion

Suppose the various economies of the world use a set of currencies $C_1, C_2, \ldots, C_n$ – think of these as dollars, pounds, bitcoins, etc. Your bank allows you to trade each currency C_i for any other currency C_j, and finds some way to charge you for this service (in a manner to be elaborated in the subparts below). We will devise algorithms to trade currencies to maximize the amount we end up with.

6.1 Flat fees

Suppose that for each ordered pair of currencies (C_i, C_j) the bank charges a flat fee of $f_{ij} > 0$ dollars to exchange C_i for C_j regardless of the quantity of currency being exchanged). Devise an efficient algorithm which, given a starting currency C_s, a target currency C_t, and a list of fees f_{ij} for all $i, j \in \{1, 2, \ldots, n\}$, computes the cheapest way (that is, incurring the least in fees) to exchange all of our currency in C_s to currency C_t. Justify the correctness of your algorithm and its runtime.

6.2 Exchange rates

Consider the more realistic setting where the bank does not charge flat fees, but instead uses exchange rates. In particular, for each ordered pair (C_i, C_j), the bank lets you trade one unit of C_i for r_{ij} units of C_j, i.e. you receive r_{ij} units of C_j in exchange of one unit of C_i. Devise an efficient algorithm which, given starting currency C_s, target currency C_t, and a list of rates r_{ij}, computes a sequence of exchanges that results in the greatest amount of C_t. Justify the correctness of your algorithm and its runtime.

6.3 Making money

Due to fluctuations in the markets, it is occasionally possible to find a sequence of exchanges that lets you start with currency A, change into currencies, B, C, D, etc., and then end up changing back to currency A in such a way that you end up with more money than you started with—that is, there are currencies $C_{i_1}, \ldots, C_{i_k}$ such that

$$r_{i_1 i_2} \times r_{i_2 i_3} \times \ldots \times r_{i_{k-1} i_k} \times r_{i_k i_1} > 1.$$

Devise an efficient algorithm that finds such an anomaly if one exists. Justify the correctness of your algorithm and its runtime.