You're expected to work on the discussion problems before coming to the lab. Discussion session is not meant to be a lecture. TA will guide the discussion and correct your solutions if needed. We will not release 'official' solutions. If you're better prepared for discussion, you will learn more. TAs will record names of the students who actively engage in discussion and report them to the instructor; they are also allowed to give some extra points to those students at their discretion. The instructor will factor in participation in final grade.

1. (Basic) Run the Floyd-Warshall algorithm on the weighted, directed graph of Figure 25.2 (CLRS). Show the matrix $D^{(k)}$ that results for each iteration of the outer loop.

   Sol. Omitted.

2. (Intermediate) The Floyd-Warshall algorithm requires $O(n^3)$ space, since we compute $d_i j^{(k)}$ for $i, j, k = 1, 2, \ldots, n$. Show that the following procedure, which simply drops all the superscripts, is correct, and thus only $O(n^2)$ space is required.

```
FLOYD-WARSHALL'(W)
1  n = W.rows
2  D = W
3  for k = 1 to n
4    for i = 1 to n
5      for j = 1 to n
6        d_ij = min(d_ij, d_ik + d_kj)
7  return D
```

   Sol.
   With the superscripts, the computation is ${d_{ij}}^{(k)} = \min({d_{ij}}^{(k-1)}, {d_{ik}}^{(k-1)} + {d_{kj}}^{(k-1)})$. If, having dropped the superscripts, we were to compute and store d_{ik} or d_{kj} before using these values to compute d_{ij} , we might be computing one of the following:

   ${d_{ij}}^{(k)} = \min({d_{ij}}^{(k-1)}, {d_{ik}}^{(k)} + {d_{kj}}^{(k-1)})$,
   ${d_{ij}}^{(k)} = \min({d_{ij}}^{(k-1)}, {d_{ik}}^{(k-1)} + {d_{kj}}^{(k)})$,
   ${d_{ij}}^{(k)} = \min({d_{ij}}^{(k-1)}, {d_{ik}}^{(k)} + {d_{kj}}^{(k)})$.

   In any of these scenarios, were computing the weight of a shortest path from i to j with all intermediate vertices in $1, 2, \ldots, k$. If we use ${d_{ij}}^{(k)}$, rather than ${d_{ij}}^{(k-1)}$, in the computation, then were using a subpath from i to k with all intermediate vertices in $1, 2, \ldots, k$. But k cannot be an *intermediate* vertex on a shortest path from i to k, since otherwise there would be a cycle on this shortest path. Thus, ${d_{ik}}^{(k)} = {d_{ik}}^{(k-1)}$. A similar argument applies to show that ${d_{kj}}^{(k)} = {d_{kj}}^{(k-1)}$. Hence, we can drop the superscripts in the computation.

3. (Intermediate) How can we use the output of the Floyd-Warshall algorithm to detect the presence of a negative-weight cycle?

Sol.
Here are two ways to detect negative-weight cycles:

(a) Check the main-diagonal entries of the result matrix for a negative value. There is a negative weight cycle if and only if $d_{ij}^{(n)}$ ¡ 0 for some vertex i:

- $d_{ij}^{(n)}$ is a path weight from i to itself; so if it is negative, there is a path from i to itself (i.e., a cycle), with negative weight.
- If there is a negative-weight cycle, consider the one with the fewest vertices.
  - If it has just one vertex, then some w_{ii} ¡ 0, so d_{ii} starts out negative, and since d values are never increased, it is also negative when the algorithm terminates.
  - If it has at least two vertices, let k be the highest-numbered vertex in the cycle, and let i be some other vertex in the cycle. $d_{ik}^{(k-1)}$ and $d_{ki}^{(k-1)}$ have correct shortest-path weights, because they are not based on negative weight cycles. (Neither $d_{ik}^{(k-1)}$ nor $d_{ki}^{(k-1)}$ can include k as an intermediate vertex, and i and k are on the negative-weight cycle with the fewest vertices.) Since $i \to k \to i$ is a negative-weight cycle, the sum of those two weights is negative, so $d_{ii}^{(k)}$ will be set to a negative value. Since d values are never increased, it is also negative when the algorithm terminates.

In fact, it suffices to check whether $d_{ii}^{(n-1)}$ ¡ 0 for some vertex i . Heres why. A negative-weight cycle containing vertex i either contains vertex n or it does not. If it does not, then clearly $d_{ii}^{(n-1)}$ ¡ 0. If the negative-weight cycle contains vertex n, then consider $d_{nn}^{(n-1)}$. This value must be negative, since the cycle, starting and ending at vertex n, does not include vertex n as an intermediate vertex.

(b) Alternatively, one could just run the normal FLOYD-WARSHALL algorithm one extra iteration to see if any of the d values change. If there are negative cycles, then some shortest-path cost will be cheaper. If there are no such cycles, then no d values will change because the algorithm gives the correct shortest paths. This is the easiest method that we covered in the lecture for both Bellman-Ford and Floyd-Warshall.

4. (Advanced) Suppose that we can compute the *transitive closure* (see CLRS page 697-699 for more details) of a directed acyclic graph in $f(|V|, |E|)$ time, where f is a monotonically increasing function of $|V|$ and $|E|$. Show that the time to compute the transitive closure $G^* = (V, E^*)$ of a general directed graph $G = (V, E)$ is then $f(|V|, |E|) + O(V + E^*)$.

Sol.
Start by computing the component graph $G^{SCC} = (V^{SCC}, E^{SCC})$, which takes $\Theta(V+E)$ time. While computing the component graph, keep track of which vertices are in each strongly connected component by making a linked list of vertices for each component. Because G^{SCC} is a directed acyclic graph, we can compute its transitive closure G^{SCC*} in $f(|V^{SCC}|, |E^{SCC}|)$ time, which is $O(f(|V|, |E|))$ since $|V^{SCC}| \leq |V|$ and $|E^{SCC}| \leq |E|$.
The edges in E^* are the ordered pairs of vertices (u, v) such that either

- u and v are in the same strongly connected component, or
- u is in the strongly connected component represented in G^{SCC} by u', v is in the strongly connected component represented in G^{SCC} by v and the edge (u', v') is in G^{SCC*}.

To compute the edges in E^*, therefore, first add all pairs of vertices that are in the same strongly connected component of G. Then, for each edge (u', v') in G^{SCC*}, find all pairs of vertices for which the first vertex is in the strongly connected component for u' and the second vertex is in the strongly connected component for v', and add each pair into E^*. There are $|E^*|$ such edges, and so this step takes $\Theta(E^*)$ time.
The total time, therefore, is $O((V+E)+f(|V|, |E|)+E^*)$. Since the transitive closure of a directed graph must have at least as many edges as the graph, we have $|E^*| \leq |E|$, and so the running time is $O(f(|V|, |E|) + V + E^*)$.