1. (15 points) **DFS and strongly connected components (SCCs) in directed graphs**

Consider the graph in Figure 1.
![Graph Image]

(a) (6 points) In what order are the SCCs found? Break any ties lexicographically (i.e., according to alphabetic order).

(b) (4 points) Which are source SCCs and which are sink SCCs?

(c) (5 points) The graph can be made strongly connected by adding edges. Draw the "metagraph" with a smallest possible set of such additional edges. If there are multiple solutions, any one will do.

We draw the reverse graph and run DFS to find pre and post numbers for each vertex.
Starting from node with highest post number, we get the SCC
\[ \{ D, E, F, G, H, I \} \]
of the remaining nodes, C has the highest post number. This gives the SCC \[ \{ C \} \]
After removing the nodes from above SCCs, node J has highest post number. This again has the singleton SCC \[ \{ J \} \]
The remaining two nodes form the SCC \[ \{ A, B \} \]
Thus, the SCCs are found in the following order
\[ \{ D, E, F, G, H, I \}, \{ C \}, \{ J \}, \{ A, B \} \]
(b) The SCC \[ \{ D, E, F, G, H, I \} \] is a **sink SCC** since it is the first one to be found. \[ \{ C \} \] has edges going out and coming into it, hence it is neither source nor sink SCCs and \[ \{ J \} \] is a source SCC since it has only edges going out of it. \[ \{ A, B \} \]
Metagraph

ADD edge from D to A and D to J
2. (15 points) **Most reliable path**

Consider the graph in Figure 2.

```
A -- 0.1 -- C
     |       |
     0.5    0.05
     |       |
     0.2    0.2
E     D     B
```

*Figure 2*

Given a graph $G(V,E)$ where each edge in the graph represents a road and the corresponding weight gives the probability of accident on that road. The most reliable path between two nodes is defined as the (simple) path with the lowest probability of accident. In Figure 2, \textbf{ACDB} is the most reliable path from \textbf{A} to \textbf{B}, as it has the lowest probability of accident among the three possible (simple) paths (\textbf{AB} and \textbf{AEB} are the two other paths).

The probabilities of accident for these three paths are:

\[
\begin{align*}
\text{Prob}\_\text{Accident}\{\text{ACDB}\} &= 1 - (1 - 0.1) \times (1 - 0.05) \times (1 - 0.2) = 0.316 \\
\text{Prob}\_\text{Accident}\{\text{AB}\} &= 0.5 \\
\text{Prob}\_\text{Accident}\{\text{AEB}\} &= 1 - (1 - 0.2) \times (1 - 0.2) = 0.36
\end{align*}
\]

(a) \textit{(4 points)} Given that all roads have nonzero probability of accident, show that the most reliable path between two nodes cannot include any cycles.

(b) \textit{(11 points)} Modify Dijkstra's algorithm to find the most reliable path and the probability of accident along this path.

\[a) \quad \text{If we had a cycle on a most reliable path between some nodes } U, v \in V \]

\[\text{take the path without cycle} \]

Thus, instead of going around the cycle, one could just one part of the cycle to go to $v$ giving a path with a higher reliability $\Rightarrow$ a contradiction!
(b) We have to come up with an update method for labels of nodes that correctly modify the probability of accident to a node.

Consider the following situation:

\[ PA[A] = 1 - T(1 - p_e) \]

where \( T \) is on path from \( s \) to \( A \)

\[ Probability \ of \ accident \ on \ path \ to \ node \ A \]

Then

\[ PA[B] = 1 - T(1 - p_e) \times (1 - p_{AB}) \]

where \( T \) is on path from \( s \) to \( A \)

\[ = 1 - (1 - PA[A])(1 - PA[B]) \]

This becomes update equation for probability of accident.
DIJKSTRA = Lowest Prob. of Accident

Step 0: all nodes v ∈ V, v ≠ A
receive temporary labels

\[ PA[v] = \rho_{av} \]

(\( \rho_{av} = 1 \)) if no edge (A, v) exists
\[ PA[A] = 0 \quad R = \{ A \} \]

Step 1: Among all temporary labels, pick \( \min_{v \in V \setminus R} PA[v] \)
and change k's label to permanent and add k to R

Step 2: Replace all temporary labels of neighbors of k using

\[ PA[v] = 1 - (1 - PA[k]) (1 - Pr_v) \]
3. (15 points) **Divide and Conquer**
   (a) (4 points) Find the big-$O$ complexity for the following recurrence relations:
   
   i. $T(n) = T(n-1) + n^2$
   
   ii. $T(n) = 16T(n/2) + n^3$

   (b) (5 points) In 3-way merge sort, we divide the array into 3 equal parts, recursively sort each part and then combine. For an array of $n$ elements, the combine part of this algorithm takes $(n-1)$ constant-time operations. Show the recurrence relation for this algorithm and deduce its time complexity.

   (c) (6 points) The na"ive way to compute $3^{27}$ involves multiplying by 3, 26 times. Show that it is possible to compute $3^{27}$ by using not more than 7 multiplications. How many multiplications will it require to compute $3^{127}$ using the same technique?

4. (15 points) **Bellman-Ford, Negative cycle**

   (a) (3 points) Show an example where Dijkstra's algorithm may not find the shortest path in the presence of negative-cost edges (but no negative cycle).

   (b) (7 points) Describe how the Bellman-Ford algorithm can be used to detect the presence of negative cycles in a directed graph. (Give a brief description and write out the pseudocode.)

   (c) (5 points) Show that if iteration $k$ of the Bellman-Ford algorithm does not improve the shortest cost of any node, then iteration $k+1$ will also not improve the shortest cost of any node.

\[ 3(a) \quad (i) \quad T(n) = T(n-1) + n^2 \]

\[ = T(n-2) + (n-1)^2 + n^2 \]

\[ = T(n-3) + (n-2)^2 + (n-1)^2 + n^2 \]

**In general**

\[ T(n) = T(n-k) + (n-k+1)^2 + \ldots + n^2 \]

**Substituting** $k = n-1$ gives

\[ T(n) = T(1) + 2^2 + \ldots + n^2 = \frac{T(1) + n(n+1)(2n+1)}{6} = \Theta(n^3) \]
\( T(n) = 16^{\log_2 n} + n^3 \)
3(b) 3-way Merge Sort

Let $T(n)$ be running time for this algorithm.

\[
\begin{align*}
\text{n} & \quad \leftarrow \quad T(n) \\
\frac{n}{3} & \quad \frac{n}{3} & \quad \frac{n}{3} & \quad T\left(\frac{n}{3}\right)
\end{align*}
\]

Merge takes $(n-1)$ operations.

\[
T(n) = 3T\left(\frac{n}{3}\right) + (n-1)
\]

Using Master Theorem.

- $a = 3$, $b = 3$, $d = 1$
- $\log_b^a = 1 = d$

Therefore, $T(n) = O(n \log n)$.
3)(c) 

\[ 3^{27} = (3^{13})^2 \times 3 \rightarrow \text{one square + one multiply = 2 multiply} \]

\[ 3^{13} = (3^6)^2 \times 3 \rightarrow " \ + " = 2 " \]

\[ 3^6 = (3^3)^2 \rightarrow " \ + " = 1 " \]

\[ 3^3 = (3^2)^2 \times 3 \rightarrow " \ + " = 2 " \]

\text{Total \# of multiplicatives = } 2 + 2 + 2 + 2 + 2 = 7

\[ 3^{127} = (3^{63})^2 \times 3 \rightarrow \text{one square + one product = 2 multiplications} \]

\[ 3^{63} = (3^{31})^2 \times 3 \rightarrow " \ + " = 2 " \]

\[ 3^{31} = (3^{15})^2 \times 3 \rightarrow " \ + " = 2 " \]

\[ 3^{15} = (3^7)^2 \times 3 \rightarrow " \ + " = 2 " \]

\[ 3^7 = (3^4)^2 \times 3 \rightarrow " \ + " = 2 " \]

\[ 3^3 = (3^2)^2 \times 3 \rightarrow " \ + " = 2 " \]

\text{Total \# of multiplicatives = } 2 + 2 + 2 + 2 + 2 + 2 + 2

\[ = 12 \]
4) Bellman-Ford, Negative Cycle

(a) 

Node B given permanent label 1
but shortest path has length \( 5 - 100 = -95 \)

(b) All shortest paths in a graph with no negative cycles will have at most \((n-1)\) many edges on them.

Thus, if after \((n-1)\) many iterations of B-F, if another iteration changes any shortest path lengths, then there is a negative cycle.

Pseudo code

```plaintext
for each \( v \in V \)
    \( \text{dist}[v] = \infty \)
    \( \text{dist}[s] = 0 \)

for \( i = 1 \) to \( |V| - 1 \)
    for each edge \((u, v) \in E\)
        if \( \text{dist}[u] + w_{uv} < \text{dist}[v] \)
            \( \text{dist}[v] = \text{dist}[u] + w_{uv} \)

for each edge \((u, v) \in E\)
    if \( \text{dist}[u] + w_{uv} < \text{dist}[v] \) \( \rightarrow \text{Negative Edge Cycle Exists} \)

Checking for \(-ve\) cycles
```
We will prove this by induction on the iteration number.

**Base Case:** Given costs do not change from iterate \( k \) to \( k+1 \).

**Inductive Hypothesis:** Costs do not change from iterate \( k+1 \) to \( k+2 \).

This means extra hops not needed.

\[
\ell^{(k+2)}(u) = \begin{cases} 
\ell^{(k+2)}(v) & \text{this case selected for all nodes} \\
\min \left( \min_{(v,u) \in E} \ell^{(k+1)}(v) + w_{vu}, \ell^{(k+1)}(u) \right) & 
\end{cases}
\]

Thus,

\[
\ell^{(k+2)}(u) = \ell^{(k+1)}(u) \text{ for all } u \in V
\]

**Induction Step:**

Going from iterate \( (k+1) \) to iterate \( (k+2) \),

\[
\ell^{(k+3)}(u) = \begin{cases} 
\ell^{(k+2)}(v) & \text{this case selected for all nodes} \\
\min \left( \min_{(v,u) \in E} \ell^{(k+1)}(v) + w_{vu}, \ell^{(k+2)}(u) \right) & 
\end{cases}
\]

From \( \ell^{(k+2)}(u) = \ell^{(k+1)}(u) \) and Induction Hypothesis,

\[
\ell^{(k+3)}(u) = \ell^{(k+2)}(u) \]

Thus, distances do not change from Induction Hypothesis.