1. Floyd’s Algorithm of All Pairs Shortest Path in a Graph
For the directed graph shown below trace Floyd’s algorithm for computing all-pair shortest distance. Compute the associated matrix (P) to determine the shortest path between each pair. From the P matrix derive the actual path between every pair (if there exists any).

\[
\begin{array}{c|cccc}
\hline
 & A & B & C & D \\
\hline
A & 0 & 2 & \infty & \infty \\
B & \infty & 0 & \infty & \infty \\
C & 3 & 7 & 0 & 1 \\
D & 6 & \infty & \infty & 0 \\
\end{array}
\]

\[
\begin{array}{c|cccc}
\hline
 & A & B & C & D \\
\hline
A & 0 & 2 & \infty & \infty \\
B & \infty & 0 & \infty & \infty \\
C & 3 & 5 & 0 & 1 \\
D & 6 & 8 & \infty & 0 \\
\end{array}
\]

\[
\begin{array}{c|cccc}
\hline
 & A & B & C & D \\
\hline
A & 0 & 2 & \infty & \infty \\
B & \infty & 0 & \infty & \infty \\
C & 3 & 5 & 0 & 1 \\
D & 6 & 8 & \infty & 0 \\
\end{array}
\]
Valid Paths:
Path from A to B: A—B
Path from C to A: C—A
Path from C to B: C—A A—B
Path from C to D: C—D
Path from D to A: D—A
Path from D to B: D—A A—B

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>2</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>B</td>
<td>∞</td>
<td>0</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>C</td>
<td>3</td>
<td>5</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>6</td>
<td>8</td>
<td>∞</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>0</td>
<td>null</td>
<td>null</td>
</tr>
<tr>
<td>B</td>
<td>null</td>
<td>0</td>
<td>null</td>
<td>null</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td>1</td>
<td>null</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>2</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>B</td>
<td>∞</td>
<td>0</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>C</td>
<td>3</td>
<td>5</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>6</td>
<td>8</td>
<td>∞</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>0</td>
<td>null</td>
<td>null</td>
</tr>
<tr>
<td>B</td>
<td>null</td>
<td>0</td>
<td>null</td>
<td>null</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td>1</td>
<td>null</td>
<td>0</td>
</tr>
</tbody>
</table>
Q2. Apply Prim-Jarník’s, Kruskal’s and Baruvka’s MST algorithms to the graph given below to compute the minimum spanning tree. Show the computation in each case.

Show trace of Prim’s Algorithm
Show Trace of Kruskal’s Algorithm
Show Trace of Baruvka’s Algorithm

A → B
5
C → D
3

A → B → C → D
5
C → D
3

A → B → C → D
5
C → D
3