1. Using the graph below, show the start and finish times determined by DFS. Using those finish times, provide a topological sort of the nodes of the graph. As on the homework, visit nodes in alphabetical order.

(sol’n:)
The input graph is figure 1. The start and finish times are shown on the graph of figure 2. The derived topologic sort is $g, a, e, f, c, b, d$.

2. We are given a weighted (positive weights) undirected graph $G = (V, E)$, and a set $F \subseteq V$ of special nodes. The nodes represent all locations in a city, and the set $F$ represents the locations of a firehouse. For planning purposes, we want to know, for each location in the city, the minimum distance to some firehouse. Describe (briefly) an $O((V + E) \lg V)$ algorithm to do this. Refer to known algorithms and describe the (simple) modification.

(sol’n:)
The simplest approach is to change the initialization step of Dijkstra’s method: for every $v \in F$, set the initial distance of $v$ to 0, and to $\infty$ for all $v \notin F$. Then put them all in a priority queue and run the rest of Dijkstra’s method, otherwise unchanged.

Alternatively, create a new node $v'$ and create new edges $(v', v)$, for all $v \in F$, with all the new edges having weight 0. Then run Dijkstra’s method on the new graph with start vertex $s = v'$.

3. Illustrate the Floyd-Warshall algorithm on the graph with the following weight matrix

\[
\begin{pmatrix}
0 & 8 & \infty & \infty \\
8 & 0 & 5 & 3 \\
\infty & 5 & 0 & 6 \\
\infty & 3 & 6 & 0
\end{pmatrix}
\]

Show the intermediate matrices for $k = 1, 2, 3, 4$. Note that the graph is undirected (and hence the matrix is symmetric).

(sol’n:)
The input matrix above is the initial $D^{(0)}$. The remaining computed matrices are below (boxes indicate a change from the previous matrix).

\[
D^{(1)} = D^{(0)}
\]

\[
D^{(2)} = \begin{pmatrix}
0 & 8 & 13 & 11 \\
8 & 0 & 5 & 3 \\
13 & 5 & 0 & 6 \\
11 & 3 & 6 & 0
\end{pmatrix} = D^{(3)} = D^{(4)}
\]
4. The Oregon Department of Transportation (ODOT) has decided to place a series of warning signs along a section of a major road. This is a one-way road and the dangerous sections are at mileposts \((m_1, m_2, \ldots, m_n)\). They wish to cover each of these locations, meaning that for each milepost there should be a sign at that point or at most \(k\) miles before it. Formally, for each milepost \(m_i\), it is covered if there is a sign at some \(m_j\) \((j \leq i)\) with \(m_i - m_j \leq k\).

Signs can only be placed at a milepost and there is a cost \(c_i\) to place a sign at \(m_i\). The input consists of \([m_1, m_2, \ldots, m_n]\), \([c_1, c_2, \ldots, c_n]\), and \(k\). You can assume that the \(m_i\) are sorted \((m_{i-1} < m_i)\) and that \(c_i > 0\). The goal here is to determine the minimum total cost of a placement of signs at mileposts that covers all locations.

To start a dynamic programming solution, we define subproblem \(WS(i)\) to be the minimum placement cost of warning signs such that \((i)\) there is a sign at location \(m_i\) and \((ii)\) locations \(m_1, m_2, \ldots, m_{i-1}\) are also covered. The overall cost that we can report to ODOT is \(\min\{WS(i) \mid 1 \leq i \leq n, \ m_n - m_i \leq k \}\).

**Give a recurrence relation for WS. Include the base case(s).**

\((sol'n:)\)

**base case:** \(WS(1) = c_1\), since we must place a warning sign at \(m_1\).

**recurrence:** When \(i > 1\) we place a sign at \(i\), then look for the minimum cost to cover location \(i - 1\) (at \(m_{i-1}\)). To do this, look at all locations at \(i - 1\) or up to \(k\) miles before it.

\[WS(i) = c_i + \min\{WS(j) \mid 1 \leq j < i, \ m_{i-1} - m_j \leq k\}\].

\(\square\)
Figure 1: graph for question 1

Figure 2: solution for question 1