1. In a weighted graph with start node \( s \), there are often multiple shortest paths from \( s \) to any other node. We want to use Dijkstra’s algorithm to count them (so assume no negative edge weights). To each node \( v \), we add a field \( v.numPaths \), and initialize \( s.numPaths \) to 1. Modify the RELAX routine so that Dijkstra’s algorithm will determine both the length of the shortest path to all other nodes and the number of such shortest paths.

(sol’n:)

\[
\text{relax}(u,v) \\
\quad \text{if } (v.dist > u.dist + W(u,v)) \\
\quad \quad \text{then} \\
\quad \quad \quad v.dist = u.dist + W(u,v) \\
\quad \quad \quad v.numPaths = u.numPaths \\
\quad \text{else} \\
\quad \quad \text{if } (v.dist == u.dist + W(u,v)) \\
\quad \quad \quad \text{then} \\
\quad \quad \quad \quad v.numPaths += u.numPaths
\]

□

2. Using the graph of figure 1, 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 graph shown in figure 2 shows the start/finish times from a DFS. Listing the nodes in reverse order of finish time gives the requested topological sort: \( d, c, f, b, g, a, e \) □

3. Illustrate the Floyd-Warshall algorithm on the graph with the following weight matrix (see also figure 3):

\[
\begin{pmatrix}
0 & 5 & 8 & \infty \\
5 & 0 & 6 & 2 \\
8 & 6 & 0 & 3 \\
\infty & 2 & 3 & 0
\end{pmatrix}
\]

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 (boxed numbers indicate a change from the previous matrix).

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

1
\[
D^{(2)} = \begin{pmatrix} 0 & 5 & 8 & 7 \\ 5 & 0 & 6 & 2 \\ 8 & 6 & 0 & 3 \\ 7 & 2 & 3 & 0 \end{pmatrix} = D^{(3)}
\]

\[
D^{(4)} = \begin{pmatrix} 0 & 5 & 8 & 7 \\ 5 & 0 & 5 & 2 \\ 8 & 5 & 0 & 3 \\ 7 & 2 & 3 & 0 \end{pmatrix}
\]

4. Imagine we are on a tour of some popular tourist spots and we wish to visit each one of them. They are all along a single road at mileposts \(m_0, m_1, m_2, \ldots, m_n\), and we will stop at each one in that order. The choice we have to make is which taxi company to use to take us from one location to the next.

The choices are taxi company F (fast) and company G (good). Company F charges \(x^2\) dollars to travel \(x\) miles, so that if they are used to go from location \(i - 1\) to \(i\), the charge for that segment is \((m_i - m_{i-1})^2\) dollars. Company G charges a flat rate of $500 per segment but if chosen they must be used for three consecutive segments. For example, we could start with company G at location \(i - 3\), have them take us to \(i - 2\), \(i - 1\), and finally finish with them at location \(i\) for a total of $1500 (of course, we could decide to use them for the next three segments). Our goal is to find the cheapest cost for all segments taking us from location 0 to location \(n\).

For example, if the mileposts are at locations (0, 25, 50, 100, 150, 200) and the travel plan is (F, G, G, G, F), the travel cost is \(25^2 + 500 + 500 + 500 + 50^2 = 4625\) dollars. However the plan (F, F, G, G, G) incurs a cost of \(25^2 + 25^2 + 500 + 500 + 500 = 2750\) dollars.

Define the subproblem \(T(i)\) to be the minimum cost of a plan that starts at location 0 and ends at location \(i\), and so that either (a) company F brought us from \(m_{i-1}\) to \(m_i\), or (b) company G was used on the last three segments (from \(m_{i-3}\) to \(m_{i-2}\) to \(m_{i-1}\) to \(m_i\)).

(a) Give the base case or cases for \(T\).

(b) Provide a recurrence relation for \(T\). (No code as usual.)

(sol’n:)
Parts (a) and (b) below:

\[
T(i) = \begin{cases} 
0 & \text{if } i = 0 \\
(m_i - m_{i-1})^2 + T(i - 1) & \text{if } 0 < i \leq 2 \text{ (can only use company F)} \\
\min[(m_i - m_{i-1})^2 + T(i - 1), 1500 + T(i - 3)] & \text{if } i \geq 3 \text{ (min of choices (a) and (b) above)}
\end{cases}
\]

The minimum cost to go from location 0 (at \(m_0\)) to location \(n\) (at \(m_n\)) is thus \(T(n)\).
Figure 1: graph for question 2

Figure 2: solution for question 2, showing start/finish times
Figure 3: graph for question 3