You will submit your solution to this assignment to the Curator System (as `HW04`). Your solution must be either a plain text file (e.g., NotePad++) or a typed MS Word document; submissions in other formats will not be graded.

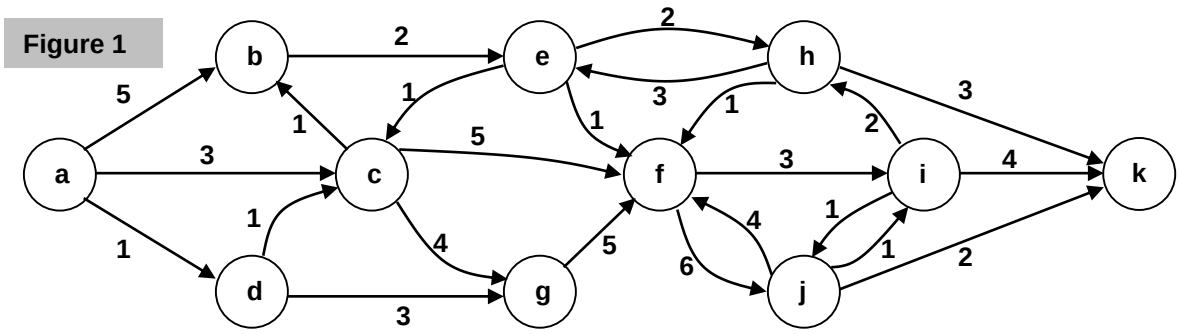
Credit will only be given if you show relevant work.

---

1. **[20 points]** Apply Dijkstra's SSAD algorithm to find the shortest distance from vertex a to every other vertex in the graph shown in Figure 1 below. For uniformity, when choosing which node to visit next, take them in increasing alphabetic order. You must show supporting work in the form of a table; see the course website for an acceptable format. You do not need to list the paths in your answer, just the minimum distances.

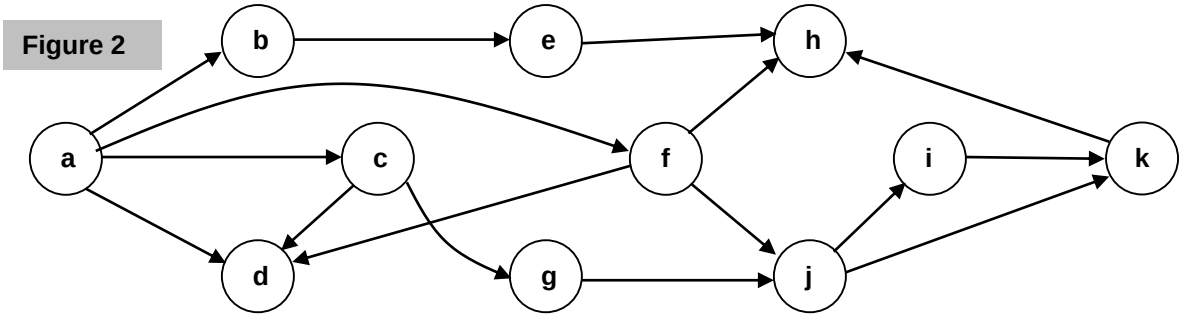
Note: the example in the course notes shows an undirected graph, but the algorithm applies to directed graphs as well, and in the obvious manner.

Figure 1



2. **[20 points]** Using a depth-first traversal, find a topological ordering of the nodes in the graph shown in Figure 2 below. For uniformity, when choosing which node to visit next, take them in increasing alphabetic order. You must show supporting work; see the course website for an acceptable format.

Figure 2



3. Suppose you are given a collection S of $N = 2^{20}$ different strings. You are not given any further information about the strings in the collection, and you must not depend on any assumptions about them.

Explain whether each of the following search problems could be solved more efficiently if the elements in S were sorted in some useful order, describing (in words, not code) the most efficient algorithm for solving the problem. You must describe how the strings would be ordered, the algorithm you would use to solve the given problem, and state the cost of the algorithm in Θ terms. Do not consider the cost of sorting the values in S as part of the analysis.

a) **[15 points]** Determine whether there is some string Z such that Z and the reversal of Z are both in S. (E.g., "evil" and "live".)

b) **[15 points]** Given a specific string X, which may or may not be in S, determine whether there is a string Z in S such that Z is a suffix of X. (E.g., "inaction" and "action".)

c) **[15 points]** Given a specific string X, which may or may not be in S, determine whether there are two strings in S whose concatenation is X. (E.g., "foolproof" and the pair "fool" and "proof".)

d) **[15 points]** Given a specific string X, which may or may not be in S, determine whether there is a string Z in S such that X is a substring of Z (E.g., "act" and "inaction".)