CS 430 – FALL 2023
INTRODUCTION TO ALGORITHMS
HOMEWORK #7

1. (5 points) The clustering coefficient $C(G)$ of a simple graph G is the probability that if u and v are neighbors and v and w are neighbors, then u and w are neighbors, where u, v, and w are distinct vertices of G.

1a) We say that three vertices u, v, and w of a simple graph G form a triangle if there are edges connecting all three pairs of these vertices. Find a formula for $C(G)$ in terms of the number of triangles in G and the number of paths of length two in the graph. [Hint: Count each triangle in the graph once for each order of three vertices that form it.]

1b) Explain what the clustering coefficient measures in each of these graphs.

- **the Hollywood graph (used for the six degrees of Kevin Bacon problem)**
- **the graph of Facebook friends**
- **the graph representing the routers and communications links that make up the worldwide Internet**

2. (5 points) Let G be a directed graph represented using an adjacency list. So, each node $G[i]$ has a list of all nodes reachable in 1 step from i (all out-neighbors of i). Suppose each node of G also has a value: e.g., node 1 might have value \$100, node 2 might have value \$50, etc.
Give an $O(|E| + |V| \log |V|)$ time algorithm that computes, for every node, the highest value reachable from that node (i.e., that you can get to by some path from that node). For instance, if it is possible to get to any node from any other node (G is "strongly-connected"), then for every node this will be the maximum value in the entire graph. Hint: one worthwhile preprocessing step is to sort nodes by value.

3. (5 points) Suppose you are implementing a spreadsheet program, in which you must maintain a grid of cells (each designated by alphabetic row and integer column). Some cells of the spreadsheet contain numbers, but other cells contain expressions that depend on other cells for their value. However, the expressions are not allowed to have a circular reference: for example, if the expression in cell $E1$ depends on the value of cell $A5$, and the expression in cell $A5$ depends on the value of cell $C2$, then $C2$ must not depend on $E1$.

3a) Describe an efficient algorithm for making sure that no circular reference exists (or finding one and complaining to the spreadsheet user if it does exist).

3b) If the spreadsheet changes, all its expressions may need to be recalculated. Describe an efficient method for sorting the expression evaluations, so that each cell is recalculated only after the cells it depends on have been recalculated.

4. **(5 points) For a sparse graph $G = \{V, E\}$, where $|E| = \Theta(V)$, is the implementation of Prim's algorithm with a Fibonacci heap asymptotically faster than the binary-heap implementation? What about for a dense graph, $|E| = \Theta(V^2)$? How must the sizes $|E|$ and $|V|$ be related for the Fibonacci-heap implementation to be asymptotically faster than the binary-heap implementation?**

5. **(5 points) Prim's and Kruskal's algorithms both "grow" a minimum spanning tree of a graph by selecting edges to add to the tree in a specified, greedy order. Design an efficient algorithm to "prune" a graph and yield a minimum spanning tree by removing edges from the graph in a specified, greedy order. Prove optimal substructure and the Greedy Choice Property.**

6. **(5 points) Let graph $G = (V, E)$ and vertices start, goal in V be given. Assuming all edges in E are of non-negative weight, describe an efficient algorithm for finding the longest acyclic path from start to goal.**

7. **(5 points) A telephone network diagram is a weighted graph where the vertices represent switching centers and the edges represent lines between two centers. The edges are labeled by their bandwidth. We define the bandwidth of a path as the smallest bandwidth of an edge in the path. Give an algorithm that computes a path of maximum bandwidth between two given vertices of a telephone network diagram. For example, in the telephone network diagram shown below, $v_1v_2v_4$ is a path of maximum bandwidth (equal to 5) between vertices v_1 and v_4, and $v_1v_2v_4v_3$ is a path of maximum bandwidth (equal to 4) between vertices v_1 and v_4**

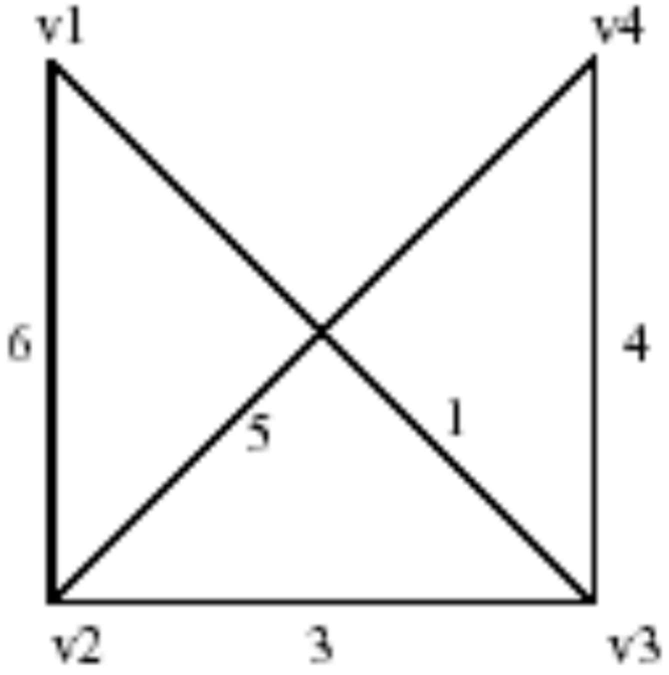

Figure 7.1:

8. **(5 points) Suppose we also wish to compute the vertices on shortest paths in the algorithms of this section, instead of keeping track of the predecessors as we**

compute the shortest paths. Show how to compute the predecessor matrix Π **from the completed matrix** L **of shortest-path weights in** $O(n^3)$ **time.**