# CS 163 – Fall 2022 – Midterm exam

Name:

UCInet id:

This is a closed book, closed note exam. Calculators or other electronic devices are not allowed.

Do not open this exam until told to start by the instructor.

Please write your answers ONLY on the front side of each page.
Answers written elsewhere will not be scanned and will not graded.

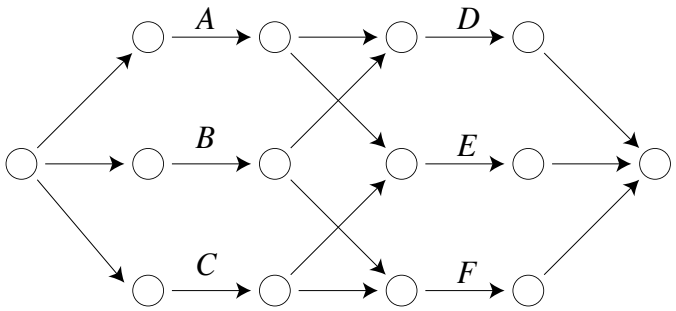
You may use the back sides of the pages as scratch paper.
Do not unstaple the pages.

1. (20 points) The lecture on strongly connected components included an algorithm for testing whether an entire directed graph is strongly connected, by performing a reachability test in both the given graph and another graph obtained by reversing the edges of the first graph. The following Python expression describes a directed graph that could be given as input to this algorithm.

   {0: [1, 2], 1: [2, 3], 2: [3, 4], 3: [1], 4: [ ]}

   Write a Python expression for the reversed graph.

2. (20 points) In the activity-on-edge graph shown below, there are six activities, labeled as $A$, $B$, $C$, $D$, $E$, and $F$. The other edges constrain the order of the milestones, but are not activities. Suppose that activities $A$, $B$, $C$, $D$, $E$, and $F$ will take time 9, 11, 7, 6, 10, 4 (in days), respectively.




   (a) In the class, we described a method for constructing optimal schedules based on longest paths. If we use this method to schedule this graph, with tasks $A$, $B$, and $C$ all starting on day 0, what day do tasks $D$, $E$, and $F$ start on?

   (b) Which activities are on the critical path for this graph? What is the length of the path?

3. (20 points) Let $G$ be an undirected complete graph with three vertices. Find weights for the edges of $G$ for which every optimal graph traveling salesperson tour has at least one repeated edge. Show the optimal tour for your graph. (Hint: Your weights should not obey the triangle inequality.)

4. (20 points) The algorithms we described in class for minimum spanning trees include Borůvka's algorithm, the Prim–Dijkstra–Jarník algorithm with Fibonacci heaps, and Kruskal's algorithm. For graphs with a small number of edges (meaning that $m = O(n)$), all three of these algorithms are slower than linear time by a logarithmic factor. When $m$ is large enough, one of these algorithms takes linear time ($O(m)$ time) while the other two are always nonlinear.

   (a) Which one takes linear time for large $m$?

   (b) How large does $m$ need to be, as a function of $n$, for its time to be linear?

5. (20 points) Suppose that we perform a depth-first search of a directed graph, and we discover that vertex $v$ has DFS number $x$ (using a numbering that starts from 1), and that it is in position $y$ of a postorder traversal of the depth-first search forest (again, starting from 1). As a function of $x$ and $y$, how many vertices are descendants of $v$ in the depth-first search forest? (Note: Every vertex has at least one descendant, itself.)