♣ ♣ CSCI 2300 — Introduction to Algorithms ♣ ♣
Summer 2020 Exam 2 (July 23, 2020)

- This exam is scheduled for the extended window that starts at 4:00PM EDT on Thursday 7/23 and ends at 2:00AM EDT on Friday 7/24
- This exam is a 90-minute exam, but you will have a full three-hour window to complete and submit your exam solutions; if you have extra-time accommodations, then you have either a 4.5-hour (50%) or six-hour (100%) window (and can go beyond the 2:00AM EDT end time)
- Submitty will start the "clock" for you when you first download the exam, so please plan accordingly by avoiding any distractions or interruptions; **and note that you must start your exam by 11:00PM EDT on Thursday 7/23 to have your full window of time**
- This exam is open book(s), open notes; given that you are working remotely, you may use any and all of the posted course materials, including all previous questions and answers posted in the Discussion Forum
- **Please do not search the Web for answers;** please follow the instructions carefully and only use the techniques taught in this course
- Long answers are difficult to grade; the space provided should be sufficient to answer each question; **please be brief and exact in your answers**
- **All work on this exam must be your own; do not copy or communicate with anyone else about the exam during or for 24 hours after the exam**
- Once we have graded your exam, solutions will be posted; the grade inquiry window for this exam will be one week

Submitting your Exam Answers

- Please combine all of your work into **a single PDF file called** `upload.pdf` that includes this cover page
- You **must** submit your exam file(s) within the three-hour window on Submitty (or within the extended window if you have accommodations for additional time)
- If you face any logistical problems during the exam, please email `goldschmidt@gmail.com` directly with details

Academic Integrity Confirmation

Please sign or indicate below to confirm that you will not copy and you will not cheat on this exam, which also means that you will not communicate with anyone under any circumstances about this exam:

Signature or Typed Name: _____

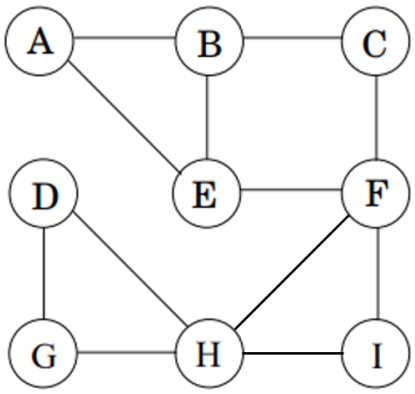
Failure to submit this page will result in a grade of 0 on the exam.

1. **(3 POINTS)** Given undirected bipartite graph $G = (V, E)$ with $|V| = 13$, what is the maximum number of edges that G can have (and still remain a bipartite graph)?
Circle the **best** answer.

(a) 25
(b) 30
(c) 42
(d) 49
(e) 56
(f) 84

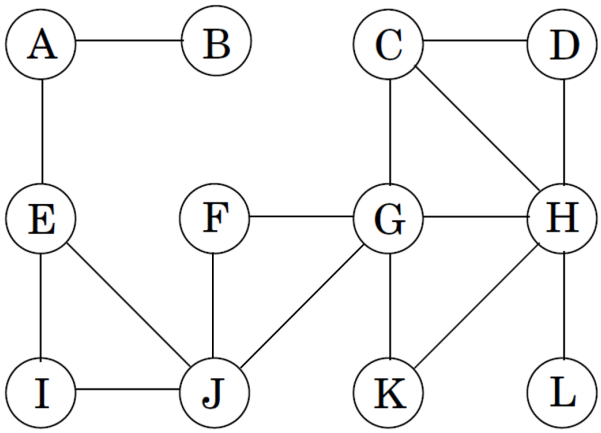
2. **(3 POINTS)** How many biconnected components are there in the given undirected graph?
Circle the **best** answer.

(a) 0
(b) 1
(c) 2
(d) 3
(e) 4
(f) 5



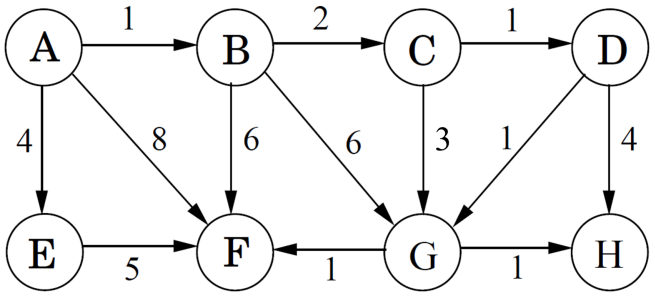
3. **(3 POINTS)** How many bridges are there in the given undirected graph?
Circle the **best** answer.

(a) 0
(b) 1
(c) 2
(d) 3
(e) 4
(f) 5



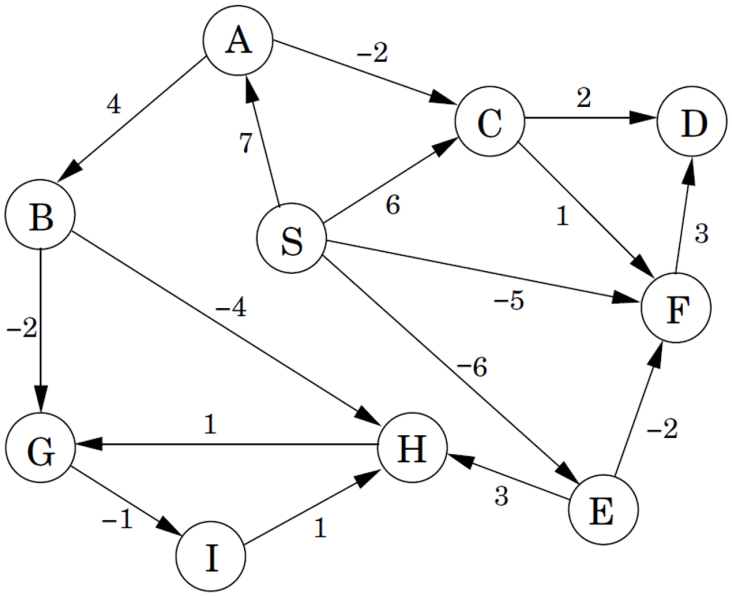
4. **(3 POINTS)** In the given directed graph, how many edges are in the shortest path from vertex A to vertex H, as determined by Dijkstra's algorithm? Circle the **best** answer.

(a) 1
(b) 2
(c) 3
(d) 4
(e) 5
(f) 6



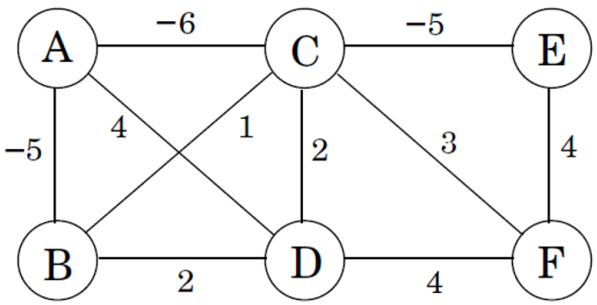
5. **(3 POINTS)** What is the weight of the shortest path from vertex S to vertex G in the graph to the right, as determined by Bellman-Ford? Circle the **best** answer.

(a) -3
(b) -2
(c) -1
(d) 7
(e) 8
(f) 9



6. **(3 POINTS)** What is the largest possible maximum degree of any minimum spanning tree of the given graph? Circle the **best** answer.

(a) 1
(b) 2
(c) 3
(d) 4
(e) 5
(f) 6



7. **(8 POINTS)** Is it possible to reduce the Huffman encoding algorithm (from the lecture notes and Dasgupta textbook) to linear time if the symbols are given to us in ascending order of frequency? Circle the **best** answer.

YES **NO**

If YES, describe this algorithm; be sure to show the runtime complexity using Big-$O()$ notation, including an explanation.

If NO, describe why it is not possible to achieve linear time for this algorithm.

8. **(8 POINTS)** Professor F. Lake has two new claims, shown below. For each claim, state whether the claim is true or false, then prove your answer. Assume that the given graphs are undirected and connected.

   (a) First, define a shortest path between two vertices s and t as path P_{st}. The claim is that for any graph $G = (V, E)$ with $s \in V$ and $t \in V$, path P_{st} is always part of at least one minimum spanning tree of G.

   Circle one: **TRUE** **FALSE**

   Proof:

   (b) First, for any $m > 0$, define an *m-path* to be a lightweight path with all of its edge weights $0 \leq w_i < m$. The claim is that if any graph $G = (V, E)$ contains an m-path from vertex s to vertex t, then every minimum spanning tree of G must also contain an m-path from vertex s to vertex t.

   Circle one: **TRUE** **FALSE**

   Proof:

9. **(6 POINTS)** Circle the correct claim below regarding complete graph K_n with $n \geq 2$, then prove the claim. Circle exactly one claim.

   Claim A: Complete graph K_n has exactly $(n-1)^n$ spanning trees.

   Claim B: Complete graph K_n has exactly n^{n-2} spanning trees.

   Claim C: Complete graph K_n has exactly $(n-1)^{n-1}$ spanning trees.

   Claim D: Complete graph K_n has exactly n^{n-1} spanning trees.

   Proof:

10. **(10 POINTS)** As an evil Uber driver, you would like an algorithm to determine the *longest* path between two points instead of the shortest path. Unfortunately, the "longest path problem" is intractable, which means it does not have a known algorithm that runs in polynomial time. Fortunately, many such problems have polynomial-time algorithms that can find *near-optimal solutions*, and in this case, a near-optimal solution would be fine.

    Therefore, for this problem, given directed graph $G = (V, E)$ with edge weights $w_e > 0$, write a greedy algorithm to try to determine the longest path from start vertex s to target vertex t, both of which are given.

    Your algorithm must be a greedy algorithm, must run in polynomial time, and must produce a near-optimal solution. Be sure you clearly describe your algorithm's inputs and outputs, detail the steps of your algorithm, and explain your algorithm's runtime using Big-$O()$ notation. Also explain why you think the solution your algorithm will produce is near-optimal.

    And be sure you make your algorithm as efficient as possible; part of your grade for this question depends on algorithm efficiency.

Use this page for any scratch or overflow work.