NAME: _____

CMPSCI 611
Advanced Algorithms
Final Exam Fall 2020

A. McGregor

8pm, 12/2/20 to 8pm, 12/3/20

DIRECTIONS:

- **Honesty Policy:** No communicating with anyone (except the instructor and TAs via private posts on Piazza) about the exam during the 24 hours the exam is open. You are not allowed to use any resources except from the slides and scribed notes linked in Moodle.
- Answers to Questions 1 and 2 should be entered directly into Gradescope.
- You should type or handwrite the answers to Questions 3, 4, and 5 on separate pages and upload into Gradescope. It should be possible to answer each question using a single page. Most parts of each question can be answered in a few sentences. Remember that the best answers are those that are clear and concise (and of course correct).
- Once you are finished submit your solutions in Gradescope by 8pm Thursday. If you are using a camera to "scan" your answers, please take care to make the picture as legible as possible. If you handwrite, please write as clearly as possible.

1	/10
2	/10
3	/10
4	/10
5	/10
Total	/50

Question 1. For each of the following statements, enter in Gradescope whether they are TRUE or FALSE. No justification is required.

1. I have read and understood the honesty policy on the front page of the exam. I agree to follow this policy.

   TRUE FALSE

2. Given a list of n integers, it is possible to find the largest element in $O(n)$ time assuming that any two elements can be compared in $O(1)$ time.

   TRUE FALSE

3. If every capacity in an instance of network flow is an odd integer then the size of the maximum s-t flow is always an odd integer.

   TRUE FALSE

4. If $E = \{a, b, c\}$ and $\mathcal{I} = \{\{\}, \{a\}, \{c\}\}$ then $(E, \mathcal{I})$ is a subset system.

   TRUE FALSE

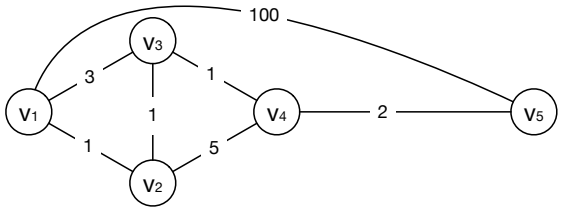
5. If X is random variable then $E[X + X^2] = E[X] + E[X^2]$.

   TRUE FALSE

6. Every tree with n nodes (assume n is even) has a matching of size $n/2$.

   TRUE FALSE

Question 2. Enter in Gradescope the numerical answers in these questions. Consider the following graph with weights on the edges.



1. Let $d_{i,j}$ be the length of the shortest path between v_i and v_j What is the value of $d_{1,5}$ in the above graph?

2. In the Floyd-Warshall algorithm, $d_{i,j}^{(k)}$ is the length of the shortest path between v_i and v_j which does not include any nodes in the set $\{v_{k+1}, v_{k+2}, \ldots, v_n\}$ as *intermediate* nodes. n is the number of nodes in the graph. What is the value of $d_{1,5}^{(2)}$ in the above graph.

3. Suppose we compute the shortest paths from v_1 to all other nodes using Dijkstra's algorithm. This algorithm maintains an array D such that for each $i \in [n]$, $D[i]$ refers to the current distance from v_1 to v_i. For the above graph, the algorithm updates this array as follows:

   After Step 1: $D = (0, 1, 3, \infty, 100)$
   After Step 2: $D = (0, 1, 2, 6, 100)$
   After Step 3: $D = (0, 1, 2, ?, 100)$

   What is the missing value in the last line?

4. Seidel's Algorithm only works when all edge lengths are 1. However, you could get around this if all edge weights are strictly positive integers: replace every edge $e = (u, v)$ by a path that has end points u and v and $w_e - 1$ intermediate nodes. In the resulting graph, we set all edge weights to 1. If we transformed the above graph as suggested, how many nodes would be in the resulting graph?

5. What is the length of the shortest tour (a path that visits every node exactly once and returns to starting point) in the above graph?

Question 3. In this question, you should assume that $P \neq NP$ and that any linear program can be solved in polynomial time.

1. Let $a(G)$ be the size of the minimum vertex cover of graph G. Does there exist a polynomial time algorithm to compute $a(G)$? Justify your answer.

2. If T is an arbitrary tree with at least three nodes, show that there exists a vertex cover for T of minimum size that does not include any nodes of degree 1.

3. Using the fact in Question 3.2, design a polynomial time algorithm that compute $a(T)$ for any tree T. No need to prove correctness or running time. Describing the algorithm in words (rather than pseudo-code) is fine.

Given a graph $G = (V, E)$ with nodes $\{v_1, \ldots, v_n\}$, we say the vector $(x_1, x_2, \ldots, x_n)$ is a *fractional vertex cover* of size $\sum_{i=1}^{n} x_i$ if

$$x_i + x_j \geq 1 \text{ for every edge } (v_i, v_j) \in E \qquad \text{and} \qquad x_i \geq 0 \text{ for every node } v_i \in V \ .$$

Let $b(G)$ be the size of the minimum fractional vertex cover. For example, if G is a triangle then $a(G) = 2$ and $b(G) = 1.5$.

4. Does there exist a polynomial time algorithm to compute $b(G)$ for an arbitrary graph G? Justify your answer.

5. Prove that $b(G) \leq a(G)$.

Question 4. Let S be a set of n different integers. We say $x \in S$ is an *almost-median* if at least a third of the elements in S are strictly less than x and at least a third of the elements in S are strictly greater than x. For example, if $S = \{2, 5, 13, 1, 8, 3, 10, 6, 11\}$ then 5, 6 and 8 are almost-median. Throughout the question you may ignore rounding issues, e.g., you may assume that the sizes of all sets encountered are divisible by 3.

1. How many elements in a set of size n are almost-median?

2. Consider the algorithm FindAM(S) for finding an almost-median of a set S:
   (a) Pick a random element x from S (each element is equally likely to be picked and note that when we pick an element we *do not* remove it from S)
   (b) Check if x is an almost-median by counting the elements in S strictly less than x.
   (c) If x is an almost-median, output x. If not, go back to Step (a).

   What is the expected running time of this algorithm? Justify your answer.

3. What is the probability FindAM takes at least twice as long as expected?

4. Consider the following sorting algorithm AM-Sort(S):
   (a) If $|S| \leq 4$, return the list of the elements of S in sorted order.
   (b) $x \leftarrow$ FindAM(S)
   (c) Compute $L = \{y \in S : y < x\}$ and $H = \{y \in S : y > x\}$
   (d) Return the list that concatenates AM-Sort(L) with x and then AM-Sort(H).

   What is the expected running time of AM-Sort? Justify your answer.

Question 5. Given graph $G = (V, E)$ and any subset of nodes U, let $\delta(U)$ be the number of edges with exactly one endpoint in U. The *k-constrained max-cut problem* is to find the subset $S \subseteq V$ of *at most* k nodes that maximizes $\delta(S)$. Note the optimum may contain $< k$ nodes.

1. If $k = 10$, design a polynomial time algorithm that solves this problem exactly. Describing the algorithm in words (rather than pseudo-code) is fine. **Hint:** For any $k \leq n/2$, $\binom{n}{0} + \binom{n}{1} \ldots + \binom{n}{k} = O(n^k)$.

In what follows, we analyze the following approximation algorithm: 1) Let S be the set of k nodes with the highest degree. 2) For every subset $S' \subseteq S$, calculate $\delta(S')$ and return the subset $S^* \subseteq S$ that gives the maximum value.

2. Show that the running time of this algorithm is polynomial in n if $k \leq \log n$.

3. Prove that $\delta(S^*) \geq (d_1 + \ldots + d_k)/4$ where $d_1 \geq d_2 \geq \ldots \geq d_k$ are the k highest degrees. **Hint:** Recall that the max cut of any graph includes at least half the edges. Furthermore, if there are m_1 edges with one endpoint in S and m_2 edges with two endpoints in S then $m_1 + 2m_2 = d_1 + \ldots + d_k$.

4. Prove that $\delta(T^*) \leq \delta(S^*) + kd_k$ where T^* is the optimum solution for the k-constrained max-cut problem.

5. Using the inequalities in part 3 and part 4 (whether or not you proved them), prove that $\delta(T^*) \leq 5\delta(S^*)$, i.e., that the above algorithm is a 5 approximation.