CSCI 670 Final

December 10, Friday, 2021: 2-4pm

Total points: 105

Name:	
Student ID:	

Don't panic!

Print your name, student id, and email in the box above, and print your name on top of every page.

Please write your answers on the front of the exam pages. Use the back of the page as scratch paper. Let us know if you need more paper.

Read the entire exam before writing anything.
Make sure you understand what the question is asking.
If you give a beautiful answer to a wrong question, you'll get no credit. If any question is unclear, please ask us for clarification.

Don't spend too much time on any single problem (unless you solved all but one). If you get stuck, move on to something else and come back later.

Work on your own! Plagiarism and other anti-intellectual behavior will be dealt with severely. This includes the possibility of failing the course or being expelled from the University.

Don't panic!

Problem 1: [It's a Theory Class: 35 Points total]

1.A: [Duality Principle and Approximation: 10 Total points]

The CSCI 670 students have collectively developed a new algorithm, called SCALABLECUT, that has the following property: For each instance $[G = (V, E, \mathit{capacity}), s, t]$ of the s-t maximum-flow/minimum-cut problem, the execution of

$$(S, T, f) = \text{SCALABLECUT}([G := (V, E, \mathit{capacity}), s, t])$$

always returns an s-t partition S and T of V (i.e., $s \in S$, $t \in T$, and $S \cup T = V$) as well as a feasible s-t-flow f for G such that

$$\mathit{cut}(S, T) \leq 1.01 \mathit{value}(f) \tag{1}$$

where $\mathit{cut}(S, T)$ denotes the total capacity of the edges connecting nodes in S to nodes in T and $\mathit{value}(f)$ denotes the value of flow f.

Prove the following statement: Suppose $(S^{\text{OPT}}, T^{\text{OPT}})$ is a minimum s-t-cut of instance $[G = (V, E, \mathit{capacity}), s, t]$. Then:

$$\mathit{cut}(S, T) \leq 1.01 \mathit{cut}(S^{\text{OPT}}, T^{\text{OPT}}).$$

PROBLEM 1.B: [BISECTION AND SECOND EIGENVALUE: 15 POINTS]

Given an undirected graph $G = (V, E)$ with $|V|$ being an even integer, a bisection of G is a partition of V into $(S, V - S)$ such that $|S| = |V|/2$.

Recall that the Laplacian matrix of G is $L_G = D_G - A_G$.

1. **(5 Points)**: Prove that L_G satisfies the following: for all vector $x = (x_1, ..., x_n) \in \mathbb{R}^n$

$$(x_1, ..., x_n) L_G (x_1, ..., x_n)^T = \sum_{(u,v) \in E} (x_u - x_v)^2. \tag{2}$$

2. **(10 Points)**: Therefore, by linear algebra, L_G is a symmetric and positive semi-definite matrix, and hence its second smallest eigenvalue $\lambda_2(G)$ satisfies:

$$\lambda_2(G) = \min_{(x_1,\ldots,x_n): \sum_{u\in V} x_u = 0} \frac{\sum_{(u,v)\in E}(x_u - x_v)^2}{\sum_{u\in V} x_u^2}. \tag{3}$$

   Recall also:

$$cutsize(S) = |E(S, V-S)|.$$

   Use Equation (3) to prove the following bound:

$$\lambda_2(G) \leq \min_{S:(S,V-S) \text{ is a bisection}} \frac{4 \cdot cutsize(S, V-S)}{n} \tag{4}$$

Hint: To apply Equation (3), using each bisection S to define a vector $(x_1, ..., x_n)$ satisfying $\sum_{u\in V} x_u = 0$.

Problem 1.C: [Integer Linear Programming for Maximum Independent Set: 10 Points]

In graph theory, an independent set of a graph $V = (V, E)$ is a subset of vertices $S \subseteq V$ such that no two of which are adjacent. Formulate the problem of computing a maximum sized independent set of a graph as an Integer Linear Program. [**Hint**: Following the example we had in the class of encoding the Vertex Cover Problem as an Integer Linear Program.]

Problem 2: Computing Without the Whole Data: Calling an Election [15 Total points]

Suppose there is an election with n voters in which everyone is a candidate and each can cast a single vote. Suppose further, the election commission is counting the ballots in a uniform random order.

1. **(6 points)**: First, consider the following definition: for $0 < \epsilon \leq 1/2$, we say a candidate is an ϵ-decisive winner if they receive $(0.5 + \epsilon)n$ votes in the election. Clearly, for any $\epsilon > 0$, any election has either zero or one decisive winner.

   Conditioning upon in the election, there exists an ϵ-decisive winner for some $\epsilon > 0$, answer the following question:

   For a parameter $\delta < 1$, after counting how many votes (the order of the smallest possible number) can the election commission identify, with confidence at least $1 - \delta$, the winner of the election? How should the election commission identify the winner in this case?

   (**Hint**: Please provide the best possible bound in terms of δ and ϵ, don't worry about constant).

2. **(6 points)**: Answer the following question, conditioning upon that in the election, there are three candidates each receiving more than 30 percent of the total votes:

   After counting how many votes (smallest possible number) can the election commission identify, with confidence at least $1 - \delta$, all three winners (in order to run a subsequent runoff election)? How should the election commission identify these first-round winners in this case?

   (**Hint**: Again, provide the best possible bound in terms of δ and ϵ, don't worry about constant).

3. **(3 points)**: Is counting ballots in a uniform random order a good method for analyzing the election result(s)? Why or why not?

(Blank page for Problem 2)

Problem 3: Network Routing [15 points]

USC will have a new CS building in two years! Boosted by this exciting development, the USC Network Group won a large grant for designing the routing protocols for the next generation Internet. So, they need to recruit talented students who have solid background in algorithm design and analysis. They post the following algorithmic problem on the whiteboard of the CSCI 670 classroom as an interviewing problem to attract our students. You are asked to solve the following networking problem:

Problem: Suppose a directed graph $G = (V, E)$ models a routing network, where nodes in V represent routers and edges in E represent the links between routers. In the network, there are two special nodes $s, t \in V$, representing the information source and destination. Suppose all links have infinity capacity. However, each router $v \in V$ has an integer capacity $c(v) \in [1 : n^2]$, where $n = |V|$, which defines the upperbound limit on how much flow can pass through v (i.e., on the maximum in-flow at the router).

Task: Design a polynomial-time algorithm for computing the maximum flow from s to t (80% points) and prove that your algorithm is correct (20% points).

(Blank page for Problem 3)

Problem 4: Active Learning [20 points]

Fall semester ended, and you are invited to Google for a possible summer intern position. During your interview, they ask you to consider the following binary classification problem. Assume X is the set of points over an n by n grid (thus, $|X| = n^2$), and each point has a binary label which is $-$ or $+$ defined by an unknown hypothesis $H_{a,b}$ with parameters (a, b) according to the following rule: For any a point $(x, y) \in X$, its label is $+$ if and only if $x > a$ and $y > b$. In other words, there are at most n^2 possible hypotheses, one of which is the correct one for classifying points.

The goal is to find the correct hypothesis (i.e., determining a and b) using active learning: in each round, the learning algorithm queries a point in the grid and observes its label.

Give an active-learing algorithm that can learn the correct hypothesis as well prove the correctness and analyze the query complexity of your algorithm.

- If your algorithm is correct with a correct proof, and its sample complexity (the number of queries it needs) is $O(\log n)$, then you can earn all 20 points.
- If your algorithm is correct with a correct proof, and its sample complexity (the number of queries it needs) is $O(n)$, then you can earn all 10 points.
- If your algorithm is correct and its sample complexity is quadratic in n, then you will earn 5 points.

(Blank page for Problem 4)

Problem 5: Back to School [20 Total points]

The success (or setback) of your industrial interviews further strenghen your resolve to pursue your Ph.D. So you decide to talk to David Kempe, one of the most serious yet student-friendly professors at USC CS Department, who is also an leading expert at the intersection between theoretical computer science and the emerging network science.

To bring your mind "back to school," Prof. Kempe asks you to solve the following simple and elegant algorithmic problem, inspired by the Divide-and-Conquer algorithms behind FFT.

Background: The family of *Hadamand matrices* $\{H_0, H_1, ...,\}$ is recursively defined as the following:

- H_0 is the 1×1 matrix $[1]$
- For $k > 0$, H_k is the $2^k \times 2^k$ matrix

$$H_k = \begin{bmatrix} H_{k-1} & H_{k-1} \\ H_{k-1} & -H_{k-1} \end{bmatrix}$$

Algorithmic Questions:

1. (**20 points**) Show that if v is a column vector of length $n = 2^k$, then the matrix-vector product $H_k \cdot v$ can be computed using $O(n \log n)$ arithmetic operations.
2. If you can't figure out the $O(n \log n)$ time algorithm, you can still earn **5 points** by giving any polynomial-time algorithm for computing this matrix-vector product.

   Give a short proof of correctness, and a short analysis of the running time of your algorithm.

(Blank page for Problem 5)