Group A

Problem Set Question: You are given an undirected graph $G = (V, E)$. The goal of this problem is to remove a minimum number of edges such that the residual graph contains no triangles. (I.e., there is no set of three vertices (a, b, c) such that edges (a, b), (b, c), and (a, c) are all in the residual graph.)

Give a 3-approximation algorithm that runs in polynomial time. (That is, it should guarantee that there are no triangles, and that the number of edges removed is within a factor of 3 of optimal.)

Class Question: Recall the basic idea behind the Simple algorithm is that it starts at some vertex and then examines the value of the objective function at the neighboring vertices. How does the Simplex algorithm finds neighbors exactly?

In practice, this is accomplished via some linear algebraic tricks. For today, though, what is the simplest way of doing this? Imagine you have a set of constraints and a vertex. (Recall that a vertex can be defined as a linearly independent set of constraints that identify one feasible point.) How do you find a neighboring vertex? Give a simple (and inefficient) approach.

Perhaps, work out the following example:

$$\begin{aligned} &\max(x + 2y - z) \\ &\text{such that:} \quad \begin{aligned} x + y &\leq 1 \\ y + z &\leq 1 \\ x &\geq 0 \\ y &\geq 0 \\ z &\geq 0 \end{aligned} \end{aligned}$$

Assume that the Simplex algorithm starts at vertex $(0, 0, 0)$. What is a search path that Simplex might take?

Group B

Problem Set Question: Assume you are given a weighted, undirected graph $G = (V, E)$ where each edge $e \in E$ is assigned weight $w(e) \geq 0$. The goal is to remove a set of edges $D \subseteq E$ with minimum weight such that the remaining graph $G = (V, E \setminus D)$ has no triangles. Give a polynomial time algorithm for solving this problem that is a 3-approximation of optimal, i.e., that returns a set of edges D that weight at most 3-times the optimal set of edges needed to produce a triangle-free graph.

Class Question: We know that the distance function in the Euclidean plane is a distance metric. Assume we have an algorithm for solving the Metric Steiner Tree problem (or the approximate Metric Steiner Tree problem). Can we solve the Euclidean Steiner Tree Problem easily by reduction? If so, how? If not, what is the problem?

Group C

Problem Set Question: You are given a graph $G = (V, E)$ with n nodes and m edges. Each edge is colored either blue or red. You are also given a parameter k as part of the input. Give an algorithm that finds a spanning tree of G with *exactly* k blue edges (and exactly $n - k - 1$ red edges). Give the running time of your algorithm, and show that it is correct.

Class Question: Set cover and vertex cover seem very similar. Can we reduce the set cover problem to vertex cover? Can we reduce the vertex cover problem to set cover? (For the purpose of this question, asume $P \neq NP$.)

Group D

Problem Set Question: Consider the Kirkpatrick-Seidel algorithm from Problem Set 1. Assume that there are only h maximal points in the set P (but h is not known in advance). Show that the Kirkpatrick-Seidel algorithm runs in time $O(n \log h)$.

Class Question: Imagine we have a linear program with a strict inequality, e.g., $cx < b$. How do we convert that to a linear program with a non-strict inequality, e.g., $cx \leq b$?

Group A

Problem Set Question: You are given an undirected graph $G = (V, E)$. The goal of this problem is to remove a minimum number of edges such that the residual graph contains no triangles. (I.e., there is no set of three vertices (a, b, c) such that edges (a, b), (b, c), and (a, c) are all in the residual graph.)

Give a 3-approximation algorithm that runs in polynomial time. (That is, it should guarantee that there are no triangles, and that the number of edges removed is within a factor of 3 of optimal.)

Group B

Class Question: Recall the basic idea behind the Simple algorithm is that it starts at some vertex and then examines the value of the objective function at the neighboring vertices. How does the Simplex algorithm finds neighbors exactly?

In practice, this is accomplished via some linear algebraic tricks. For today, though, what is the simplest way of doing this? Imagine you have a set of constraints and a vertex. (Recall that a vertex can be defined as a linearly independent set of constraints that identify one feasible point.) How do you find a neighboring vertex? Give a simple (and inefficient) approach.

Perhaps, work out the following example:

$$\begin{aligned} &\max(x + 2y - z) \\ &\quad \text{such that:} \quad x + y \leq 1 \\ &\qquad\qquad\qquad\quad y + z \leq 1 \\ &\qquad\qquad\qquad\quad\ \ x \geq 0 \\ &\qquad\qquad\qquad\quad\ \ y \geq 0 \\ &\qquad\qquad\qquad\quad\ \ z \geq 0 \end{aligned}$$

Assume that the Simplex algorithm starts at vertex $(0, 0, 0)$. What is a search path that Simplex might take?

Group C

Problem Set Question: Assume you are given a weighted, undirected graph $G = (V, E)$ where each edge $e \in E$ is assigned weight $w(e) \geq 0$. The goal is to remove a set of edges $D \subseteq E$ with minimum weight such that the remaining graph $G = (V, E \setminus D)$ has no triangles. Give a polynomial time algorithm for solving this problem that is a 3-approximation of optimal, i.e., that returns a set of edges D that weight at most 3-times the optimal set of edges needed to produce a triangle-free graph.

Group D

Class Question: We know that the distance function in the Euclidean plane is a distance metric. Assume we have an algorithm for solving the Metric Steiner Tree problem (or the approximate Metric Steiner Tree problem). Can we solve the Euclidean Steiner Tree Problem easily by reduction? If so, how? If not, what is the problem?

Group E

Problem Set Question: You are given a graph $G = (V, E)$ with n nodes and m edges. Each edge is colored either blue or red. You are also given a parameter k as part of the input. Give an algorithm that finds a spanning tree of G with *exactly* k blue edges (and exactly $n - k - 1$ red edges). Give the running time of your algorithm, and show that it is correct.

Group F

Class Question: Set cover and vertex cover seem very similar. Can we reduce the set cover problem to vertex cover? Can we reduce the vertex cover problem to set cover? (For the purpose of this question, asume $P \neq NP$.)

Group G

Problem Set Question: Consider the Kirkpatrick-Seidel algorithm from Problem Set 1. Assume that there are only h maximal points in the set P (but h is not known in advance). Show that the Kirkpatrick-Seidel algorithm runs in time $O(n \log h)$.

Group H

Class Question: Imagine we have a linear program with a strict inequality, e.g., $cx < b$. How do we convert that to a linear program with a non-strict inequality, e.g., $cx \leq b$?