This week we will study a very general and powerful tool for algorithm development. *Linear programing* deals with the problem of finding solutions to a system of linear constraints defined over real-valued variables that minimize (or maximize) some linear objective function. Many optimization problems can be expressed as linear programs. However, in computer science, we often encounter *combinatorial problems*—problems over discrete and finite objects—that don't necessarily have *real-valued* or numerical solutions. In this case, we turn to *integer programs*—linear programs where the variables take on integer values instead of real values. Often the variables encode some sort of combinatorial structure so we can use a solution to the integer program to create a solution to the combinatorial problem it encodes.

Of course, there is some bad news: in general, solving integer programs is NP-complete. But there is some good news too. There are polynomial time algorithms for solving linear programs. Moreover, since linear programs are relaxations of integer programs (by relaxation, I mean that we don't restrict the solutions to be integral), you can always solve the relaxed integer program to obtain a lower bound (or upper bound for maximization problems) on the optimal solution to the integer program. This is a great tool for approximation algorithms since often the bottleneck to a good approximation algorithm is finding a good lower bound.

We will focus on linear and integer programs for NP-hard problems. Begin by reading the scribe notes from Shuchi Chawla's lectures on LP and IP from her University of Wisconsin Advanced Algorithms course. The notes are available on the website.

Question 1. *Why is it easy to show that Integer Programming is NP-complete?*

Question 2. *Can you think of a problem that doesn't easily lend itself to an integer program? If you can't, try checking out the list of problems on the NP optimization website:*
`https://www.nada.kth.se/~viggo/problemlist/compendium.html`.
Defend your selection.

Problem 1. *Consider the following* party problem: *You want to throw a dinner party for at most k people (excluding yourself). Since you're going to be busy hosting, you want every other person at the party to have at least one friend present. We identify the friendship relation using a directed graph $G = (V, E)$ which we call a friendship graph. You should think of V as the set of people you know. If $u, v \in V$ and u considers v a friend, then $(u, v) \in E$. Notice that friendship is not necessarily symmetric. Let $V' \subseteq V$ be a set of people. We say $u \in V'$ is happy if and only if there exists $v \in V'$ such that $(u, v) \in E$. If every person in V' is happy, we call V' a happy subset. Write down an integer program for this version of the party problem where your goal is to find the largest happy subset that does not exceed k people.*

Question 3. *Is the following a theorem: An approximation algorithm designed using an LP-relaxation cannot achieve a better approximation guarantee than the integrality gap of the relaxation.* Hint: In principle it may be possible to show, using additional structural properties, that whenever an instance has a bad gap, the cost of the solution found by the algorithm is much less that αOPT, where α is the integrality gap of the relaxation. (Observe that if the instance has a bad gap, the cost of the solution found cannot be much less than αOPT_f where OPT_f is the cost of the fractional solution to the LP-relaxation minimization problem).

Problem 2. *We know from the reading that the dual of the dual of a linear program is the original problem itself. Be prepared to present this result during our meeting.*

Problem 3 (Vazirani 12.10)**.** *Show that the following is an integer programming formulation for the minimum spanning tree (MST) problem. Assume we are given a graph $G = (V, E)$, $|V| = n$, with cost function $c : E \to \mathbb{Q}^+$. For $A \subset E$, we will denote by $\kappa(A)$ the number of connected components in graph $G_A = (V, A)$.*

$$
\begin{aligned}
\textit{minimize} \quad & \sum_e c_e x_e \\
\textit{subject to} \quad & \sum_{e \in A} x_e \leq n - \kappa(A), \qquad A \subset E \\
& \sum_{e \in E} x_e = n - 1 \\
& x_e \in \{0, 1\}, \qquad\qquad e \in E
\end{aligned}
$$

The rest of this exercise develops a proof that the LP-relaxation of this integer program is exact for the MST problem.

1. *First, it will be convenient to change the objective function of the IP to* $\max \sum_e -c_e x_e$. *Obtain the LP-relaxation and dual of this modified formulation.*

2. *Consider the primal solution produced by Kruskal's algorithm. Let* $e_1, \ldots, e_m$ *be the edges sorted by increasing cost,* $|E| = m$. *This algorithm greedily picks a maximal acyclic subgraph from this sorted list. Obtain a suitable dual feasible solution so that all complementary slackness conditions are satisfied.* Hint: Let $A_t = \{e_1, \ldots, e_t\}$. Set $y_{A_t} = e_{t+1} - e_t$, for $1 \leq t < m$, and $y_E = -c_m$, where y is the dual variable.

3. *Show that* $x = \{x_1, \ldots, x_{|E|}\}$ *is a feasible solution to the above-stated primal program if and only if it is a feasible solution to the following LP. That is, prove that this is also an exact relaxation for the MST problem.*

$$\begin{aligned} \textit{minimize} \quad & \sum_e c_e x_e \\ \textit{subject to} \quad & \sum_{e \in S} x_e \leq |S| - 1, \qquad S \subset V \\ & \sum_{e \in E} x_e = n - 1 \\ & x_e \geq 0, \qquad e \in E \end{aligned}$$

   Note that $e \in S$ *means edges that have both endpoints in* S.

4. *Denote the set of all feasible solutions to the LP in the previous part by* P_{sub}. *Now consider the following LP formulation of the MST problem, where* $\delta(S)$ *is the set of edges having* exactly one *vertex in* S*:*

$$\begin{aligned} \textit{minimize} \quad & \sum_e c_e x_e \\ \textit{subject to} \quad & \sum_{e \in \delta(S)} x_e \geq 1, \qquad S \subset V \\ & \sum_{e \in E} x_e = n - 1 \\ & x_e \geq 0, \qquad e \in E \end{aligned}$$

   (a) *Show that this ILP version of the LP above also describes the MST problem.*

   (b) *Denote the set of all feasible solutions to this LP by* P_{cut}. *Argue that* $P_{sub} \subseteq P_{cut}$. *It turns out that for some graphs,* $P_{sub} \subset P_{cut}$, *which shows that different ILP formulations of the same combinatorial problem can give rise to LPs with different solution sets!*