15-451/651 Algorithms, Spring 2019
Recitation #8 Worksheet

Taking Duals I. Consider this maximization linear program:

$$\begin{aligned} &\max(x_1 + 3x_2 - 2x_3) \\ \text{s.t.}\quad & x_1 + x_2 + 2x_3 \le 2 \\ & 7x_1 + 2x_2 + 5x_3 \le 6 \\ & 2x_1 + x_2 - x_3 \le 1 \\ & x_1, x_2, x_3 \ge 0 \end{aligned}$$

1. Write down its dual LP. (Is it a maximization or minimization problem? What are the variables? Constraints?)

2. Now write down the dual of this dual LP. Remember, since the dual will be a minimization LP, this dual's-dual will give a best lower bound on the dual.

Taking Duals II. Write down the dual of this minimization LP: *(Be careful, some inequalities are greater-than, some are less-than. And not all constraints have all variables.)*

$$\begin{aligned} &\max(x_1 - 3x_2 + 2x_3) \\ \text{s.t.}\quad & 3x_1 + 2x_3 \ge 2 \\ & 2x_2 - x_3 \le 5 \\ & x_1, x_2, x_3 \ge 0 \end{aligned}$$

Minimax from Duality (by Example). Let the row-player's payoffs be given by this (non-negative) matrix

	L	R
L	1	5
R	3	2

1. If the probabilities on the two rows are p_1 and p_2, write down an LP for the row player's optimal strategy:

2. Now take the dual of this LP. Show this dual is an LP computing the column player's optimal strategy. (And hence strong duality implies the minimax theorem.)

NP-Completeness Reductions (general). To show that a problem B is NP-Complete, we take a *known NP-Complete* problem A, and then we reduce A to B. I.e., we show that $A \leq_p B$. We do this by coming up with a polynomial-time procedure f for taking instances x of problem A and converting them to instances $f(x)$ of problem B such that $f(x)$ is a YES-instance of B *if and only if* x is a YES-instance of A. Make sure you understand:

- Why do we reduce this way, and not the other way around?
- Why is the *if and only if* condition important? Why wouldn't this work if f only satisfied the "if" or "only if"?

Binary LPs. Binary linear programming (BinLP) is like linear programming, with the additional constraint that all variables must take on values either 0 or 1. The decision version of binary linear programming asks whether or not there exists a point satisfying all the constraints. (For the decision version there is no objective function).

Show that BinLP is NP-complete.

- Show that BinLP is in NP.

- Reduce a NP-hard problem to BinLP. (Remember, you should use a Karp reduction, and the reduction should take polynomial time.)

Integer LPs. Integer linear programming (ILP) is like linear programming, with the additional constraint that all variables must take on values in the integers $\mathbb{Z}$. The decision version of integer programming asks whether or not there exists a point satisfying all the constraints. (Again for the decision version there is no objective function). Note that the above reduction, with a small tweak, immediately shows that ILP is NP-hard. Do you see why?

3-Coloring is NP-complete.

Some of you have seen a slightly different reduction from Circuit-SAT to 3-Coloring in 15-251. Here we'll reduce from 3SAT.

1. Step I: Why is 3-Coloring in NP?

2. Step II: We want to reduce 3SAT to 3-Coloring. Given a 3-CNF formula I, and we to produce a graph $G = f(I)$ such that G is 3-Colorable if and only if I is satisfiable.

   (a) Let's call the three colors R (red), T and F, and add three special nodes in a triangle called R, T, and F that we can assume without loss of generality are given the corresponding colors.

   (b) For each x_i, we have one node called x_i and one node called $\neg x_i$. Add a triangle between R, x_i, and $\neg x_i$ for each i. This forces the coloring to make a choice for each variable of whether it should be T or F.

   (c) Now, we need to add in a "gadget" for each clause. Say for $(x \vee y \vee z)$, we want to make it impossible to color all three of x, y, z with color F, but all other settings of $\{T, F\}$ are OK.

   Can you create such a gadget?

***k*-Coloring is NP-complete.** Can you show that 4-coloring is NP-complete? k-coloring for constant $k \geq 3$? What about 2-coloring?