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?)

   Solution: Remember, we're trying to find the best (i.e., lowest) upper bound on the value of the primal LP. So it should be a *minimization*. There's one variable y_i for each constraint in the primal, so that's *three y_i variables*. We're summing up constraints of the form $blah \ge blah$, so we want each $y_i \ge 0$ to not flip the inequalities. Etc.

$$\begin{aligned} &\min(2y_1 + 6y_2 + 1y_3) \\ \text{s.t.}\quad & y_1 + 7\ y_2 + 2\ y_3 \ge 1 \\ & y_1 + 2\ y_2 + 1\ y_3 \ge 3 \\ & 2\ y_1 + 5\ y_2 + (-1)\ y_3 \ge -2 \\ & y_1, y_2, y_3 \ge 0 \end{aligned}$$

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.

   Solution: The dual's dual will be the same as the primal. Details are omitted here.

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.)*

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

Solution: To make life easy, first convert this into a nicer form. Since the dual wants to give a *upper bound* on this *maximization* problem, let's make the constraints of the form $blah \le blah$.

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

And not all variables appear in all constraints, so let's put down zeros where things are missing.

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

Finally, use a dual variable $y_i \geq 0$ for each primal constraint, and two constraints, etc.

$$\begin{aligned} &\min(-2y_1 + 5y_2) \\ \text{s.t.} \quad & -3y_1 + 0y_2 \geq 1 \\ & 0y_1 + 2y_2 \geq -3 \\ & -2y_1 - y_2 \geq 2 \\ & x_1, x_2, x_3 \geq 0 \end{aligned}$$

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:

   Solution: If the row player puts $p_1 \geq 0$ and $p_2 \geq 0$ on L, R respectively, then she wants to solve $\max_{p_1+p_2=1, p_1, p_2 \geq 0} \min(p_1 + 3p_2, 5p_1 + 2p_2)$. I.e., the LP is

   $$\begin{aligned} &\max v \\ \text{subject to} \quad & p_1 + 3p_2 \geq v \\ & 5p_1 + 2p_2 \geq v \\ & p_1 + p_2 \leq 1 \\ & p_1, p_2 \geq 0. \end{aligned}$$

   We set $p_1 + p_2 \leq 1$, but to maximize v, the LP will automatically set the sum *equal to* 1.

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.)

   Solution: First convert into inequalities *blah* $\leq$ *blah* useful to show an upper bound. (Also, since all payoffs are non-negative, we can add in non-negativity for v.

   $$\begin{aligned} &\max v \\ \text{subject to} \quad & v - p_1 - 3p_2 \leq 0 \\ & v - 5p_1 - 2p_2 \leq 0 \\ & p_1 + p_2 \leq 1 \\ & v, p_1, p_2 \geq 0. \end{aligned}$$

If the dual variables are q_1, q_2 and w, we get

$$\begin{aligned} &\min\ w \\ &\text{subject to} \quad q_1 + q_2 \geq 1 \\ &-q_1 - 5q_2 + w \geq 0 \\ &-3q_1 - 2q_2 + w \geq 0 \\ &w, q_1, q_2 \geq 0 \end{aligned}$$

Now move some variables around:

$$\begin{aligned} &\min\ w \\ &\text{subject to} \quad q_1 + q_2 \geq 1 \\ &w \geq q_1 + 5q_2 \\ &w \geq 3q_1 + 2q_2 \\ &w, q_1, q_2 \geq 0 \end{aligned}$$

And again observe that to minimize the value, any optimal solution will reduce q_1, q_2 to make their sum equal to 1. So the LP is solving:

$$\min_{q_1+q_2=1, q_1, q_2 \geq 0} \max(q_1 + 5q_2, 3q_1 + 2q_2).$$

That's the column player's strategy!!! And by strong duality, we get the minimax theorem for this particular game. Exactly the same idea holds in general, details are in the lecture notes.

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.

  Solution: What is the witness? The solution.

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

  Solution: We can reduce 3SAT to BinLP. Given an instance I of 3SAT, let the variables in ϕ be $x_1, x_2, ..., x_n$. We produce an instance $f(I)$ of BinLP as follows: we have corresponding variables $z_1, z_2, ..., z_n$ in our BinLP. First, each variable is binary (either 0 or 1):

  $$z_i \in \{0, 1\} \qquad\qquad \forall i.$$

  Assigning $z_i = 1$ in the integer program represents setting $x_i = T$ in the formula, and assigning $z_i = 0$ represents setting $x_i = F$. Now for each clause like $(x_1 \vee \bar{x_2} \vee x_3)$, we have a constraint like:

  $$z_1 + (1 - z_2) + z_3 \geq 1.$$

  To satisfy this inequality we must either set $z_1 = 1$ or $z_2 = 0$ or $z_3 = 1$, which means we either set $x_1 = T$ or $x_2 = F$ or $x_3 = T$ in the corresponding truth assignment. More generally, for each clause in the 3SAT instance, we create the constraint that the sum of literals, using z_i to represent x_i and $(1 - z_i)$ to represent $\bar{x_i}$, is at least 1.

  If the given instance I was a YES-instance of 3SAT then $f(I)$ is a YES-instance for BinLP: just take a satisfying assignment A to the variables x_i and set each z_i to 0 or 1 accordingly. Since A satisfied at least one literal in each clause, this means the associated sum is ≥ 1. In the other direction, any solution to the BinLP must set at least one of the associated literals to 1, since each is an integer 0 or 1.

  Finally, the transformation is clearly poly 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?

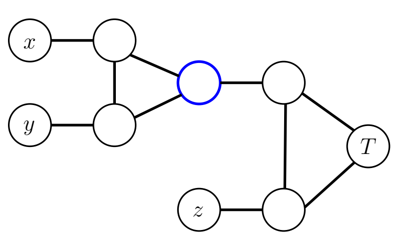
Solution: Just add the constraints $0 \le z_i \le 1$ for all i. BTW, membership in NP is a bit trickier here (how do you know that if the answer is YES, there is always a solution that can be described in a polynomial number of bits) but it follows from facts about matrices that we won't get into.

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?

   Solution:



   (Look at the triangle attached to x, y. If both $x = y = F$, then the tip of that triangle has to be F too, else it can be colored T. A similar argument now holds for the second triangle too.)
   Once all these gadgets are added, a 3-Coloring exists of G if and only if there is a satisfying assignment to the original instance I. Finally, the reduction takes linear time: putting down this gadget for each clause in I.

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?

Solution: Again k-coloring is in NP, just take the coloring and check it is valid, that each edge has distinct colors. For 4-coloring, reduce from 3-coloring. Take an instance I of 3-coloring, which is a graph. Take a new node and attach it to all the nodes of G, call this graph $H = f(I)$. This graph is 4-colorable if and only if G was 3 colorable. Hence,

$$I \text{ is a YES instance } \iff f(I) \text{ is a YES instance }.$$

Also, this reduction takes linear time: copying the graph G over, and adding a new vertex, connecting it to all other vertices.

You can use the same idea to show that k-coloring, for any constant k, is NP-complete.

2-coloring is in P: a graph is 2-colorable if and only if it is bipartite. This can be checked in linear time using DFS.