15-451 Algorithms, Spring 2017
Recitation #9 Worksheet

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?

Approximation for Set Cover. We did not cover the set cover algorithm in Lecture 18 — let's do it here. The problem is simple: *Given a collection $\mathcal{F}$ of subsets $S_1, S_2, \ldots, S_m$ of some set X, find the fewest number of sets whose union is the set X.* Let k^* be the optimal number of sets.

1. Consider any subset $X' \subseteq X$. We claim there exist at most k^* sets in $\mathcal{F}$ whose union contains X'. Infer that there is some set $S_i \in \mathcal{F}$ such that $|S_i \cap X'| \geq \frac{|X'|}{k^*}$.

2. Suppose after $i-1$ sets have been picked. Call an element covered if it lies in some previously picked set.

   Suppose n_{i-1} elements of X are not yet "covered". If we pick a set from $\mathcal{F}$ that contains the maximum number of uncovered elements, explain why this set must cover at least $\frac{n_{i-1}}{k^*}$ elements.

3. Hence we get that $n_i \leq n_{i-1} - \frac{n_{i-1}}{k^*} = n_{i-1}(1 - \frac{1}{k^*})$, where $n_0 = |X|$.

4. Deduce that $n_i \leq |X|(1 - \frac{1}{k^*})^i < |X|e^{-i/k^*}$. (Recall that $(1 + x) \leq e^x$, and equality only holds when $x = 0$.

5. Show that $n_i < 1$ when $i = k^* \ln n$ sets have been picked. Hence $n_i = 0$ and all elements in X must be covered.