Basic Algorithms, Assignment 10 SOLUTIONS

Due, TUESDAY, Nov 20

1. A positive integer n is called *banana* if it can be written as the sum of two integer squares. (E.g., $41 = 16 + 25$.) Argue that the problem class BANANA is in NP.
   Solution: The certificate is the numbers x, y with $n = x^2 + y^2$ and the we just calculate $x^2 + y^2$ and make sure it is n.

2. In 2002 Agarwal, Kayal and Saxena showed that PRIME is in P. Call a positive integer n *walrus* if it is the product of precisely three primes. Given the AKS result, argue that WALRUS is in NP. Argue (harder!) that NOTWALRUS is in NP.
   Solution: The certificate is the three primes p, q, r. Then we check (using the Agarwal,Kayal, Sexana algorithm) that they are primes and that their product is n. If n is not walrus then it is either a prime, a product of two primes p, q, or the product of four numbers a, b, c, d, not necessarily prime. In the last case the certificate is the numbers a, b, c, d and we multiply them; in the second case the certificate is the numbers p, q and we apply AKS to show they are primes and then multiply them; in the first case we apply AKS to show it is a prime.

3. Lets define `3-SATSMALL` to be the same as `3-SAT` except that no Boolean variable x_i appears (as either x_i or $\overline{x_i}$) more than 20 times. Our object is to show `3-SAT`$\leq_P$`3-SATSMALL`.

   (a) Let x, y be Boolean variables. Find a set of clauses $C_1, \ldots, C_r$ of size 3 using auxilliary Boolean variables $z_1, z_2, \ldots z_s$ so that $C_1 \wedge \ldots \wedge C_r$ can be satisfied if and only if x, y have the same truth value. (Here r, s will be small numbers.)
   Solution: $x \vee \overline{y} \vee z_1$, $x \vee \overline{y} \vee \overline{z_1}$, $y \vee \overline{x} \vee z_2$, $y \vee \overline{x} \vee \overline{z_2}$. (When x, y both true or both false the $x \vee \overline{y}, y \vee \overline{x}$ are both true so any z_1, z_2 will do. If x is true and y is false then $\overline{x} \vee y$ is false so whatever we choose for z_2 we are stuck. If y is true and x is false then $\overline{y} \vee x$ is false so whatever we choose for z_1 we are stuck.
   Solution: $x \vee \overline{y} \vee z_1$,

   (b) Let $x_1, \ldots, x_w$ be Boolean variables. Show that there is a set of clauses $C_1, \ldots, C_v$ using auxilliary variables $z_1, \ldots$ so that $C_1 \wedge \ldots \wedge C_v$ can be satisfied if and only if $x_1, \ldots, x_w$ have the same truth value. Further, we require that none of the Boolean variables (neither the original x's nor the auxilliary z's) are used more

than ten times. (Idea: Make sure x_i, x_{i+1} have the same truth value for $1 \leq i < w$.)
Solution: Just string the solution to the first problem together with different auxilliary variables. That is, to make sure x_i, x_{i+1} have the same truth value for $1 \leq i < w$ add z_{2i-1}, z_{2i} and the four clauses. Actually, no x_i appears more than eight times (four clauses with x_{i-1} and four with x_{i+1} the auxilliary z's only appear twice.

(c) Show `3-SAT`$\leq_P$`3-SATSMALL`. (Idea: When x_i appears many times replace it with copies $x_i^1, x_i^2, \ldots$, none appearing very often, that all must have the same truth value.)
Solution: Say x_i appears u times and $\overline{x_i}$ appears v times with (for the sake of argument) $v \leq u$. Replace the x_i with $x_i^1, \ldots, x_i^u$ and and $\overline{x_i}$ with $\overline{x_i^1}, \ldots, \overline{x_i^v}$. Now each x_i appears at most twice (counting appearances with negation). For each i add the auxilliary zs and the clauses to assure that $x_i^1, \ldots, x_i^u$ have the same truth value. Each x_i^j appears no more than eight times each time we do this, plus it already appeared, times two for its negation appearing, so it appears at most twenty times.

4. Suppose a graph algorithm with input a graph G takes a time polynomial in $N+M$ where N is the number of vertices and M is the number of edges in G. Show that it takes time polynomial in N. Suppose a number theoretic algorithm with input a positive integer x takes time polynomial in x. What can you say about the time it takes when the input is an n-digit number? In particular, explain why you *cannot* say that the time is polynomial in n.
Solution: As $N + M \leq N^2$ is an algorithm takes $O((N+M)^c)$ time then it takes $O(N^{2c})$ time. A different power, a big difference in the real world, but still polynomial. But if an algorithm (for example, the trial division algorithm for primality) takes time $O(x^c)$ then, as $x = \Theta(2^n)$, it takes time $O(2^{nc})$ which is *exponential* time and not polynomial time.

What I tell you three times is true
– Lewis Carroll *in* Hunting the Snark