CSci 421
Introduction to Algorithms
Homework Assignment 7
Due: Tuesday, 17 Aug 2004

Reading: Chapter 11

1. 11.4. Is the original formula f satisfiable? Is the constructed 3CNF formula f_3 satisfiable? Because one is built from the other by a *reduction*, the answer to both questions should be the same. Furthermore, because of the *way* the reduction works, each assignment satisfying f (there may be several) corresponds to at least one assignment satisfying f_3 and vice versa. In particular, the assignment satisfying f_3 should be an *extension* of the one for f, i.e., they should agree on the values of all the variables they share in common. Give one satisfying assignment of f and list all extensions satisfying f_3. Take any other assignment satisfying f_3; show that it satisfies f.

2. 11.5. Is the formula satisfiable? Does the graph contain a 4-clique? Again, satisfying assignments and cliques should be related. Give one satisfying assignment and list all its corresponding 4-cliques. If possible, find a 4-clique not in that list, and give its corresponding assignment; is it a satisfying assignment?

3. 11.7. Use the definition from page 357. You may assume that Partition is NP-complete.

4. Well folks, here's the event you've all been waiting for: three, count 'em *3*, proofs that P = NP. Only you can stop Ruzzo from becoming world-famous. Find and explain the flaw in each of the proofs below.

   Let *KNAP* $= \{a_1\#a_2\#\ldots\#a_n\#C \mid a_i$ and C are integers coded in binary, and there is a set $I \subseteq \{1,\ldots,n\}$ such that $\sum_{i\in I} a_i = C\}$. Let *UKNAP* be the same, except that the integers are coded in unary, i.e., a is represented by 1^a, the string comprised of a 1's. *UKNAP* is in *P* (via the dynamic programming algorithm presented in Section 5.10), but *KNAP* is *NP*-complete.

   (a) "Proof 1:" For any string u in $\{1,\#\}^*$ we can easily produce a string v in $\{0,1,\#\}^*$ such that $u \in$ *UKNAP* $\Leftrightarrow v \in$ *KNAP*. (E.g., if $u = 11\#1\#11111$ then $v = 10\#1\#101$.) Further, the transformation can be done in time bounded by a polynomial in the length of u. Thus, P=NP.

   (b) "Proof 2:" For any string v in $\{0,1,\#\}^*$ we can easily produce a string u in $\{1,\#\}^*$ such that $v \in$ *KNAP* $\Leftrightarrow u \in$ *UKNAP*. Further, the transformation can be done in time bounded by a polynomial in the length of u. Thus, P=NP.

   (c) "Proof 3:" 1-of-3-SAT is another known NP-complete variant of the satisfiability problem: it is the set of Boolean formulas in conjunctive normal form with exactly 3 literals per clause such that the formula is satisfied by a truth assignment making exactly one literal in each clause true. See Problem 11.16.

   Let f be a formula in conjunctive normal form with exactly 3 literals per clause (3CNF). Suppose it has variables $x_1,\ldots,x_m$, and clauses $c_1,\ldots,c_q$. Suppose "x_i" occurs in clauses numbered $i_1,\ldots,i_j$ and "$\overline{x}_i$" occurs in clauses numbered $i'_1,\ldots,i'_{j'}$. Let $a_i = \sum_{k=1}^{j} i_k$, and $\overline{a}_i = \sum_{k=1}^{j'} i'_k$. Let $s = \sum_{i=1}^{q} i$. Generate the string:

   $$u = 1^{a_1}\#1^{\overline{a}_1}\#\ldots\#1^{a_m}\#1^{\overline{a}_m}\#1^s$$

   Now if f is satisfiable by an assignment that makes exactly one literal per clause true, i.e. if f is in 1-of-3-SAT, then u is in *UKNAP*: Pick a_i or $\overline{a}_i$ depending on whether x_i is true or false respectively in the 1-of-3 satisfying assignment. Every clause is satisfied by exactly one literal, so the sum of the chosen $a, \overline{a}$'s is exactly s. Thus $u \in$ *UKNAP*.

   Further the reduction can be done in time polynomial in the length of f; e.g., note that the numbers $a_i, \overline{a}_i$, and s are all of magnitude at most q^2, since each is the sum of at most q distinct numbers between 1 and q, so the length of u is $O(q^3) = O(|f|^3)$.

   Thus P=NP.

5. Optional Extra Credit: 11.24

6. Optional Extra Credit: 11.31