Institut for Matematik og Datalogi
Syddansk Universitet

April 11, 2016
JFB

DM553 – Complexity and Computability – 2016
Lecture 11

Lecture, March 17 in U140

We started with a course evaluation of how the course has been so far. The results have been posted on the course homepage. Then, we began on NP-Completeness, introducing definitions. The definition of time complexity classes is on page 279 in Sipser's textbook, and the definition of P is on page 286. The definitions of NP and NP-Complete are in sections 7.3 and 7.4 of Sipser's textbook. Note that some of this is also in chapter 34 in the CLRS book.

Lecture, March 29

First we will show that 3-SAT is NP-Complete. To do this, we assume that CNF-SAT is NP-Complete. The proof that 3-SAT is NP-Complete combines the proofs in both the Sipser and the CLRS books. Then, we will cover the proof that SATISFIABILITY (actually CNF-SAT) is NP-Complete, from section 7.4 in Sipser's textbook. If there is time, we will do more reductions from chapter 34 in CLRS.

Lecture, April 11

We will show that CNF-SAT, CLIQUE, VERTEX COVER, INDEPENDENT SET, and SUBSET SUM are NP-Complete. See chapter 34 in CLRS.

Problems to be discussed in U14 on April 12

1. Problem 5.28 in Sipser's book.

2. The following argument is incorrect. Find the most important error.

   Consider the following algorithm:

   ```
   Input: $n \in \mathbb{N}$
   for $i = 2$ to $n - 1$ do
       check if $i$ divides $n$
       if it does then output $i$
   endfor
   output -1 if no output yet
   ```

Checking if i divides n can be done in time $O(\log^3 n)$ via binary search for an integer k such that $n = i \cdot k$ (the multiplication can clearly be done in time $O(\log^2 n)$).

Thus, the total running time is $O(n \cdot \log^3 n)$ in the worst case. Since $O(n \cdot \log^3 n) \subset O(n^2)$, and n^2 is a polynomial, this algorithm runs in polynomial time. Thus, we have an efficient algorithm for factoring, so we can break RSA, a famous cryptosystem which is believed to be secure.

3. Suppose you have a Boolean formula form, with exactly three literals per clause. Show how to add some constant number of clauses (also with exactly three literals per clause) to F to create a formula F' which is guaranteed to be false.

4. In the CLRS textbook, do the following:

   - 34.2-5, 34.2-10.
   - 34.3-6.
   - 34.4-4, 34.4-5, 34.4-6, 34.4-7.