Complexity Theory
Lecture 9

Anuj Dawar

University of Cambridge Computer Laboratory
Easter Term 2013

http://www.cl.cam.ac.uk/teaching/1213/Complexity/

Prime Numbers

Consider the decision problem PRIME:

Given a number x, is it prime?

This problem is in co-NP.

$$\forall y(y < x \to (y = 1 \vee \neg(\mathrm{div}(y, x))))$$

Note again, the algorithm that checks for all numbers up to $\sqrt{n}$ whether any of them divides n, is not polynomial, as $\sqrt{n}$ is not polynomial in the size of the input string, which is $\log n$.

Primality

In 2002, Agrawal, Kayal and Saxena showed that PRIME is in P.

If a is co-prime to p,

$$(x - a)^p \equiv (x^p - a) \pmod{p}$$

if, and only if, p is a prime.

Checking this equivalence would take to long. Instead, the equivalence is checked *modulo* a polynomial $x^r - 1$, for "suitable" r.

The existence of suitable small r relies on deep results in number theory.

Factors

Consider the language Factor

$$\{(x, k) \mid x \text{ has a factor } y \text{ with } 1 < y < k\}$$

Factor $\in$ NP $\cap$ co-NP

Certificate of membership—a factor of x less than k.

Certificate of disqualification—the prime factorisation of x.

Optimisation

The Travelling Salesman Problem was originally conceived of as an optimisation problem

to find a minimum cost tour.

We forced it into the mould of a decision problem – TSP – in order to fit it into our theory of NP-completeness.

Similar arguments can be made about the problems CLIQUE and IND.

This is still reasonable, as we are establishing the *difficulty* of the problems.

A polynomial time solution to the optimisation version would give a polynomial time solution to the decision problem.

Also, a polynomial time solution to the decision problem would allow a polynomial time algorithm for *finding the optimal value*, using binary search, if necessary.

Function Problems

Still, there is something interesting to be said for *function problems* arising from NP problems.

Suppose

$$L = \{x \mid \exists y R(x, y)\}$$

where R is a polynomially-balanced, polynomial time decidable relation.

A *witness function* for L is any function f such that:

- if $x \in L$, then $f(x) = y$ for some y such that $R(x, y)$;
- $f(x) =$ "no" otherwise.

The class FNP is the collection of all witness functions for languages in NP.

FNP and FP

A function which, for any given Boolean expression ϕ, gives a satisfying truth assignment if ϕ is satisfiable, and returns "no" otherwise, is a witness function for SAT.

If any witness function for SAT is computable in polynomial time, then $\mathsf{P} = \mathsf{NP}$.

If $\mathsf{P} = \mathsf{NP}$, then for every language in NP, some witness function is computable in polynomial time, by a binary search algorithm.

Under a suitable definition of reduction, the witness functions for SAT are FNP-complete.

Factorisation

The *factorisation* function maps a number n to its prime factorisation:

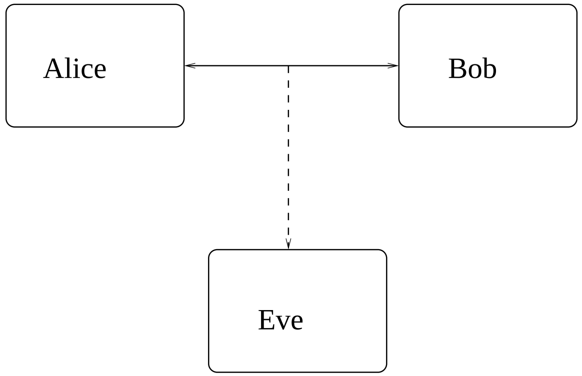
$$2^{k_1} 3^{k_2} \cdots p_m^{k_m}.$$

This function is in FNP.

The corresponding decision problem (for which it is a witness function) is trivial - it is the set of all numbers.

Still, it is not known whether this function can be computed in polynomial time.

Cryptography



Alice wishes to communicate with Bob without Eve eavesdropping.

Private Key

In a private key system, there are two secret keys

e – the encryption key
d – the decryption key

and two functions D and E such that:

for any x,

$$D(E(x, e), d) = x$$

For instance, taking $d = e$ and both D and E as *exclusive or*, we have the *one time pad*:

$$(x \oplus e) \oplus e = x$$

One Time Pad

The one time pad is provably secure, in that the only way Eve can decode a message is by knowing the key.

If the original message x and the encrypted message y are known, then so is the key:

$$e = x \oplus y$$

Public Key

In public key cryptography, the encryption key e is public, and the decryption key d is private.

We still have,

for any x,

$$D(E(x, e), d) = x$$

If E is polynomial time computable (and it must be if communication is not to be painfully slow), then the function that takes $y = E(x, e)$ to x (without knowing d), must be in FNP.

Thus, public key cryptography is not *provably secure* in the way that the one time pad is. It relies on the existence of functions in FNP − FP.

One Way Functions

A function f is called a *one way function* if it satisfies the following conditions:

1. f is one-to-one.
2. for each x, $|x|^{1/k} \leq |f(x)| \leq |x|^k$ for some k.
3. $f \in \mathsf{FP}$.
4. $f^{-1} \notin \mathsf{FP}$.

We cannot hope to prove the existence of one-way functions without at the same time proving P $\neq$ NP.

It is strongly believed that the RSA function:

$$f(x, e, p, q) = (x^e \bmod pq, pq, e)$$

is a one-way function.

UP

Though one cannot hope to prove that the RSA function is one-way without separating P and NP, we might hope to make it as secure as a proof of NP-completeness.

Definition
A nondeterministic machine is *unambiguous* if, for any input x, there is at most one accepting computation of the machine.

UP is the class of languages accepted by unambiguous machines in polynomial time.

UP

Equivalently, UP is the class of languages of the form

$$\{x \mid \exists y R(x, y)\}$$

Where R is polynomial time computable, polynomially balanced, *and* for each x, there is *at most one* y such that $R(x, y)$.

UP One-way Functions

We have

$$\mathsf{P} \subseteq \mathsf{UP} \subseteq \mathsf{NP}$$

It seems unlikely that there are any NP-complete problems in UP.

One-way functions exist *if, and only if,* $\mathsf{P} \neq \mathsf{UP}$.

One-Way Functions Imply $\mathsf{P} \neq \mathsf{UP}$

Suppose f is a *one-way function*.

Define the language L_f by

$$L_f = \{(x, y) \mid \exists z (z \leq x \text{ and } f(z) = y)\}.$$

We can show that L_f is in UP but not in P.