CS760 Topics in Computational Complexity **2024-25 Sem I**

Lecture 3: 09-08-2024

Scribe: Navya, Aditya Agrawal *Lecturer: Rohit Gurjar*

1 Decision and Search Problems

A decision problem is characterized by a specific response to each $s \in \{0,1\}^*$. Hence every decision problem can be represented by a function f of following kind

$$f : \{0,1\}^* \to \{0,1\}$$

Such a function can also be represented by a language over $\{0,1\}^*$ which consists of only those elements which are mapped to 1. We can also represent this function using a infinitely large bit string by using a one - one and onto mapping from $\{0,1\}^*$ to $\mathbb{N}$. (Onto will ensure that every infinitely large bit string is representing a unique function)

Informally speaking, we call a decision problem Q to be equivalent to a search problem P, if having a polynomial time algorithm for P implies a polynomial time algorithm and also vice-versa. (The definition of polynomial time is already defined in previous notes)

2 Class NP

Note: There are two ways two define the NP class. Out of them, only one is defined here. The other one and its equivalence to the one here is the subject matter of next lecture.

We say that a language $L \in$ NP if there exists a turing machine (the verifier, call it M for now) which runs in polynomial time and a polynomial $q\colon \mathbb{N} \to \mathbb{N}$ such that $x \in L$ iff $\exists C \in \{0,1\}^{\leq q(|x|)}$ s.t $M(x, C) = 1$. By Turing Machine which runs in polynomial time, we mean that it runs in polynomial time over the input size where input size will include both x and C. In other words, there exist a Turing machine such that for every valid x (and only for those), there exist a certificate which is polynomial in size of x and on which the Turing machine will halt with an approval in polynomial time.

Remark: It is not necessary for the machine to stop always in polynomial time. But more important is existence of polynomial size certificate which makes the machine to stop in polynomial time with acceptance. Because we have the guarantee that there is a certificate on which the machine will stop in time $t(n)$, then we can create another machine that simulates the first one, but we make it stop in time $t(n)$ (if not certificate is not accepted so far, just reject it).

2.1 Examples of problems in NP

1. **Independent Set Problem**
   L = $\{\langle G, k\rangle$: Graph G has an independent set of size $k\}$.

   A valid certificate would be the indices of an independent set of size k. This is of course polynomially bounded in size of graph. And verifying it also will take polynomial time. Verifier will check if the size of set is indeed k and if it is indeed independent.

2. **Traveling Salesman Problem**
   L = $\{\langle G, \text{distance metric}, T\rangle$: Graph G has tour covering all the vertices in length at most $T\}$

   Certificate will be a sequence of vertices. Now whether it meet the required criteria or not can be checked in polynomial time.

2.2 Examples of problems which are not known to be in NP

1. **Boolean Circuit Satisfiability (SAT Modified)**
   L = $\{\langle \phi, k\rangle$: Number of satisfying assignments of ϕ is at least $k\}$

   Here the size of input is size(ϕ) $+ \log k$. Simply providing the k assignments won't work (because that will be exponential in the input size).

2. **MCSP**
   L = $\{\langle \phi, k\rangle$: there is a Boolean circuit with size at most k which is equivalent to given circuit $\phi\}$.

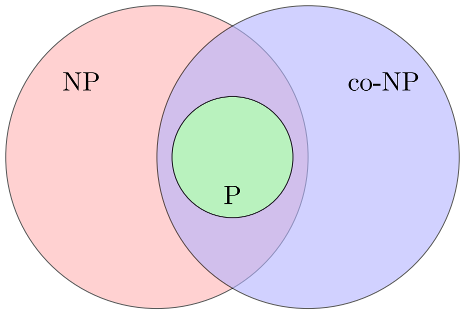
   Not known to be in NP because we don't know how to check equivalence of two circuits in polynomial time.

3 Class co-NP

We now discuss another interesting class of problems known as **co-NP**. A problem is said to be in **co-NP** if the complement of the problem is in **NP**. In other words, there is an easily verifiable certificate for all the inputs for which the answer to the problem is NO.

Some other interesting comments are as follows:

- All problems in P are in both **NP** and **co-NP**, simply because the verifier can ignore the proof and run his own P algorithm to check the answer for both the YES and NO cases.
- The converse is not true or rather we do not know at the moment. However, a problem being in both **NP** and **co-NP** gives a strong indication that the problem can be in **P**



3.1 Problems both in NP and co-NP

- **SLE** Problem - The System of Linear Equations Problem asks for whether given a system of linear equations, there exists an assignment of variables that satisfies all the linear equations.
  - The problem is very clearly in **NP** as the certificate is simply the values of the variables which the verifier can verify in the equation by substituting in the equations (we need to show that there is an assignment where values are polynomially bounded in size).
  - The problem is also in co-NP as when there does not exist a satisfying assignment, the certificate can be the coefficients which the equations need to be multiplied and then added to lead to some sort of mathematical contradiction (i.e., $0 = 1$).
- **Convex Optimization** - Given a convex function f and domain D, we want $x \in D$ such that $\forall y \in D$ $f(x) \leq f(y)$. Rather is x the minima of this convex function f? The decision problem is: given x, is x a minima of f?
  - If the answer is YES, then the gradient can be given as the certificate, which should be zero.
  - If the answer is NO, then another point y can be given as the certificate such that $f(y) < f(x)$

- **Graph Isomorphism** $\langle G, H\rangle$, G and H are isomorphic
  - Though not really in **co-NP** there is sort of a game which the verifier can play to ensure that the answer can be extracted.
  - It is in **NP** as if the answer is YES, then the permutation of vertices can be presented as the certificate.
  - If answer is **NO**, then the verifier can play a sort of a game. They flip an unbiased coin to choose one of the graphs and then apply a random permutation to the vertices of the graphs and then supply this graph to the all powerful prover.
    - If the graphs were indeed isomorphic, then the all powerful prover would not be able to tell from which initial graph this new graph has come since they both have an equal probability of having being picked.
    - If the graphs were not isomorphic, then the prover knows for sure which graph this graph comes from.
  - Via this game, the verifier can verify the answer to this problem.
  - It is now known that **Graph Isomorphism** is in class **QuasiP** – i.e., class of problems which have algorithms running in $O(n^{\log^c n})$ time for some $c \geq 1$.

3.2 Primality

The complexity of this problem was not known for a long time but in 2002, a **P** time algorithm was discovered by AKS. We study and look at ideas that might have led to this development by showing that Primality is both in **NP** and **co-NP**.

Primality Problem - $\langle n\rangle$, n is prime.

- If the answer is **NO**, then a factor of n apart from 1 and n can be provided as a certificate.
- If the answer is **YES**, then things get interesting. We use ideas from number theory to provide a simple certificate.

Theorem 3.1. *A number p is prime iff there exists a number z such that $z^{p-1} \equiv 1 \pmod p$ and $\forall r < p-1$, $z^r \not\equiv 1 \pmod p$.*

Hence it seems like this number z can be provided as a certificate for the same. Now, we run into a new problem. Even if we are given such a z we cannot check for all $r < p-1$ whether $z^r \not\equiv 1$, as the number of choices for r is not polynomially bounded in the input size. There is a clever workaround for this as well.

Claim 3.2. *If there exists $r < p-1$ such that $z^r \equiv 1 \pmod p$, then there exists $b < p-1$ such that $z^b \equiv 1 \pmod p$ and b is a factor of $p-1$.*

This is pretty simple to see. If $z^r \equiv 1 \pmod p$ and $z^{p-1} \equiv 1 \pmod p$ then $z^{\gcd(r,p-1)} \equiv 1 \pmod p$. We can simply choose b to be $\gcd(r, p-1)$.

Claim 3.3. *It is sufficient to check the maximal factors of $p-1$ to verify whether there exists $r < p-1$ such that $z^r \equiv 1 \pmod p$.*

This is also simple to see. If such an r does exist then by Claim 3.2, $z^b \equiv 1 \pmod p$ where $b = gcd(r, p-1)$. Now b divides $p-1$. Let us denote $(p-1)/b$ as c where $c \neq 1$. Hence we consider a prime factor of c say p_1. Then $(p-1)/p_1$ is a maximal factor of $p-1$ and is a multiple of b. Hence if such an r did exist, then atleast 1 maximal factor of $p-1$, say q would also satisfy $z^q \equiv 1 \pmod p$.

Now our task has simplified. The number $p-1$ has at most $O(\log p)$ prime factors so the number of maximal factors is also $O(\log p)$ in number for which this fact has to be checked.

However, the verifier can't compute the prime factorization of $p-1$, so this also needs to be included in the certificate. But then we also need to check whether the prime factors provided are really prime or not recursively. It turns out the recursively built certificate remains polynomially bounded, the reader can do this as an exercise.

For example, see the certificate for a specific prime number here `https://www.theoremoftheday.org/LogicAndComputerScience/Pratt/TotDPratt.pdf`