COMPUTATION: DAY 7

BURTON ROSENBERG
UNIVERSITY OF MIAMI

Contents

1. Review of the sixth day

Definition 1.1. A language $A \subseteq \Sigma^*$ is in the class $\mathcal{P}$ of polynomially recognizable languages if there is a Turing machine M_A, in any reasonable time bounded model, and some time bound $t(n) = O(n^c)$ for some c, and

$$A = \{\, x \mid M_A(x; t(|x|)) = T \,\}$$

where $|x|$ is the string length of x.

Definition 1.2. A language $A \subseteq \Sigma^*$ is in the class $\mathcal{NP}$ of nondeterministic polynomially recognizable languages if there is a Turing machine M_A, in any reasonable time bounded model, and some time bound $t(n) = O(n^c)$ for some c, and

$$A = \{\, x \mid \exists y,\ M_A(x, y; t(|x|)) = T \,\}$$

where $|x|$ is the string length of x.

2. Reduction and NP Completeness

Wednesday, 12 April 2023

The notion of a P-time reduction was introduced, and the overview of its use to show there are universal problems in the class NP. The problem of Boolean Satisfiability was introduced,

The Cook-Levin theorem were discussed. I just referenced the book for the proof sketch and proof for the Cook-Levin theorem; however we noted that this works because the satisfiability problem that was being solved was solved in a model that can be captured by a satisfiable boolean predicate.

Friday, 14 April 2023

The problems 3SAT and k-Clique were defined. The construction of the reduction from 3SAT to k-Clique was given.

Monday 17 April 2023

The Vertex cover problem was presented. The construction of the reduction from 3SAT to Vertex Cover was given.

Wesdnesday, 19 April 2023

The Hamiltonian path problem was presented. The construction of the reduction from 3SAT to Hamiltonian Cover was given.

3. Probabilistic Turing Machines and BPP

Friday, 21 April 2023

A probabilistic Turing machine modifies the definition of the non-deterministic machine where the non-determinism is replaced with a random choice.

The computation is now a random variable for each for each input s,

$$\begin{array}{rccc} M(s): & \Omega & \to & \{\,0,1\,\} \\ & \omega & \mapsto & M(s,\omega) \end{array}$$

and the acceptance or rejection is probability that the machine will accept or reject.

Definition 3.1. A language A is BPP if there is a polynomial time probabilistic Turing machine M such that,

$$\begin{aligned} a \in A &\implies \mathcal{P}\{\omega_a\} \geq 2/3 \\ a \notin A &\implies \mathcal{P}\{\omega_a\} \leq 1/3 \end{aligned}$$

where,

$$\omega_a = \{\omega \in \Omega \mid M(a, \omega) = 1\}$$

.

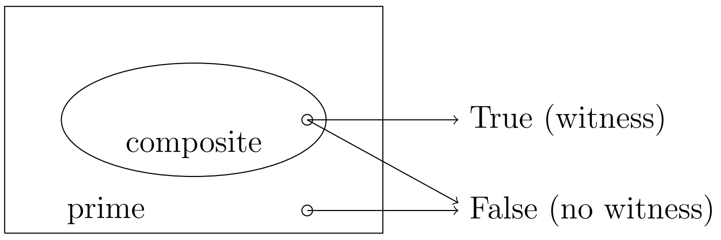
4. RP: Primality Testing

The recognition of primes by a non-probabilistic machine, as far as we know, is difficult. It requires an attempt to factor the integer. However, if even the smallest amount of error is allowed, the problem becomes easy.

Theorem 4.1. Little Fermat: If p is a prime, then $w^{p-1} = 1 \pmod p$ for all $1 \leq w < p$.

The converse is not true, but is likely. That is, if n is not a prime (and not even and not a pure power) then for about $1/2$ of the numbers between 1 and $n-1$ will contradict Fermat's theorem. That is, unless n is a certain number called a Carmichael number Let us assume we have been made a promise that we shall be presented with Carmichael numbers.

Therefore we have probabilistic algorithm that when given a prime will never proof it to be not a prime, and with probability $1/2$ when given a composite will provide a proof it is composite.



The witness can be seen as a proof. If a mathematical system does not proof false things, the system is called *sound*. If all true things can be proved, the system is called *complete*. The class RP is a more restricted version of BPP where it is sound but only probabilistically complete.

The class RP is in NP. The difference is an RP protocol is likely to find a proof and gives up quickly if it cannot.

The class BPP contains RP. This case is instructive as BPP would require the given protocol to find a proof at last 2/3 of the time, not 1/2. To satisfy this, if the first random choice does not yield a witness we try again. The probability that both numbers will not be a witness, given that the number is a (non-Carmichael) composite is only 1/4.

The relationship between RP and P is at the moment unknown.

5. Proving things in co-NP

Monday, 25 April 2023

The class RP is very conservative in that it is sound. When it can prove things it does, but never states a falsehood. If however one is will to have an un-sound system. the class co-NP becomes recognizable. The co-NP class is the class of verifiable things for which there is no witness, and is the complement of an NP problem.

The possibility to recognize co-NP sets comes from,

(1) Allowing for probabilistically sound proof system,
(2) Proving for interaction with an oracle.

While this might seem a big wish list, it does away with the case by case elimination of an exponential number of candidate witnesses.

6. An Interactive Proof system for graph non-isomorphism

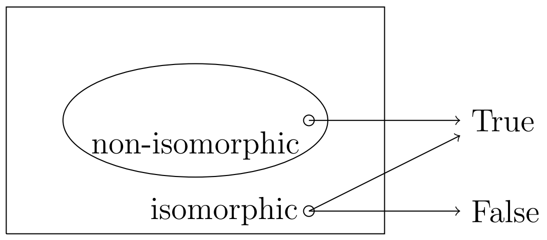
Definition 6.1. Two graphs, $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are *isomorphic* if there exists a permutation between the vertex sets, $\pi : V_1 \to V_2$, such that

$$(v_1, v_2) \in E_i \iff (\pi(v_1), \pi(v_2)) \in E_2.$$

The problem of graph isomorphism is NP complete. It is in NP because the witness can be the isomorphism π, and the polynomial time verification is checking the edge condition.

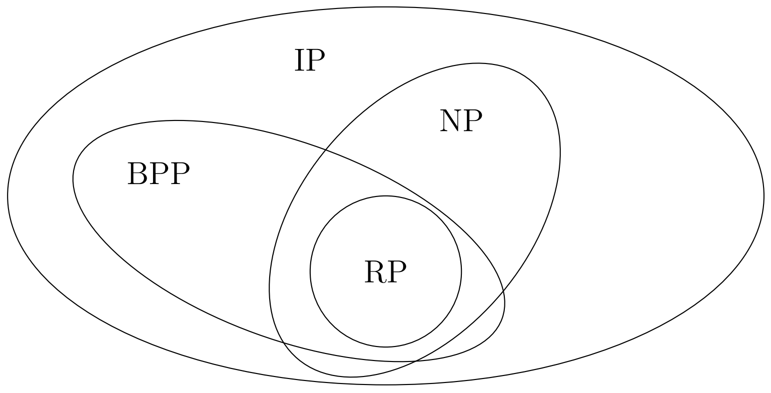
What we are about to do is to give a program to recognize the complement of this language. There will be a verifier that is a probabilistic polynomial time machine, and a prover that hasn't any bounds at all. However, it suffices here that it can solve exponential time problems. The verifier will chose a bit $b \in \{1, 2\}$ at random, and a permutation π at random. It will send the prover the graph H which is graph G_b permuted by π. The prover will send back $\tilde{b} \in \{1, 2\}$.

If $b == \tilde{b}$ the verifier concludes that the graphs are non isomorphic; else it concludes that they are isomorphic. The sense of this is that if they are isomorphic, the prover has no way of knowing if the verifier generated H from G_1 or G_2 so it simply guesses and gets it right ($b == \tilde{b}$) half the time. However, if they are non-isomorphic, it is quite clear to the prover which of G_1 or G_2 was used to create H, so it gets it right all the time.



This is the opposite situation than for RP. While RP will never bear false witness, a witness might not come forward to testify. In this protocol the witness will always testify, it it might be testifying falsely.

In general, IP does not demand perfect completeness. The definition is like BPP with errors allowed in both directions. BPP is a therefore a restriction on IP without interaction. However, the problem will drop into NP if the algorithm will work with perfect soundness. Therefore our picture is,



7. Zero Knowledge Proof Systems

Wednesday, 27 April 2023

Extending proof to only likely soundness and interactivity gives us a new technology, that of Zero Knowledge. A ZK-proof system is an interaction between a Prover and a Verifier such that the Verifier learns the truth of a fact of the form $a \in A$ but nothing more. The difficulty for the Verifier is that the fact is NP-hard, so it cannot "learn"' this fact without help from the Prover, where to know and to learn are defined as whatever is computable in polynomial time by a probabilistic Turing machine.

In an NP proof system, the prover sends the witness w and the Verifier runs a program $V(a, w, \omega)$ using perhaps some coins ω, and verifies that $a \in A$. However, the Verifier also learns w, which is something it could not have computed.

7.1. **Graph Isomorphism in Zero-knowledge.** Graph Isomorphism can be solved with an NP proof systems with perfect completeness and perfect soundness. A ZK-proof system for graph Isomorphism is the following.

(1) The Prover generates a permutation π', flips a coin b' and sends $H = \pi'(G_{b'})$ to the Verifier.
(2) The Verifier sends bit b of its choosing to the Prover.
(3) The Prover the calculates the permutation π such that $\pi(G_b) = \pi'(G_{b'})$ and sends π to the Verifier.
(4) The Verifier verifies $\pi(G_b) = H$ and accepts if it does.

Completeness: If the graphs are isomorphic, the Prover calculates the isomorphism π and the Verifier accepts.

Soundness: An important point for the soundness proof is it must be for any Prover, not just those following the protocol. If the graphs are not isomorphic, the π only exists if $b = b'$, which occurs 1/2 the time. Hence no Prover can be right more than 1/2 the time, and the Verifier rejects with probability 1/2.

7.2. **Zero-knowledge.** We need to prove that in the case where the graphs are isomorphic, the run of the protocol does not teach the Verifier anything it could not compute itself.

From the point of view of the Verifier, it is enhanced by the communication of a random triple (H, b, π). There is a consistency formula among this triple: $H = \pi(G_b)$.

Rather than receive this triple from the conversation, the Verifier can generate it itself.

Suppose that the Verifier choose b at random. Then the Verifier generates a random π and sets $H = \pi(G_b)$. *As a random variable*, this triple is identical to that achieved through the interaction. Hence any use of the run of the protocol can be replaced by a probabilistic polynomial time sampler.

Hence the protocol is zero-knowledge.

There is however a catch. The Verifier can pick b is it wishes. Hence sampling at (H, b, π) identically to a run of the protocol needs more care. For instance, a b' and π' are generated and fed to a copy of the Verifier to see what b is would choose. With probably $1/2$ it choses b' and the sample provided is (H, b', π'). Else the new b', π' is tried. In expected linear time at last a (H, b', π) is acceptable and will be a random variable with the same distribution as an authentic conversation with the Prover.

However, this is only expected linear time. If the simulator where to give up and send a $\perp$ exponentially rarely, and the original Prover modified to send a $\perp$ with the same frequency; the Verifier accept on $\perp$ with some small probability. Therefore a cheating Prover that is attempting to lie its way to getting the Verifier to accept, can only use the $\perp$ to do so very infrequently.