15-251: Great Theoretical Ideas In Computer Science

Recitation 8: Boolean Circuits and NP

News Post

- No recitation this Friday; expect another large group review session on this week's material!
- Conceptual OH on Saturdays is a great time to review old material as well as gain greater understanding of new topics and ideas!
- If you have any questions about material, logistics, or anything else 251 related make sure to reach out to your mentor!

New Phrases

- We say a problem is in **NP** if there exists a polynomial time verifier TM V and a constant $k > 0$ such that for all $x \in \Sigma^*$:

  if $x \in L$, then there exists a certificate u with $|u| \leq |x|^k$ such that $V(x, u)$ accepts.

  if $x \notin L$, then for all $u \in \Sigma^*$, $V(x, u)$ rejects.

- We say there is a **polynomial-time many-one reduction** from A to B if there is a **polynomial-time** computable function $f : \Sigma^* \to \Sigma^*$ such that $x \in A$ if and only if $f(x) \in B$. We write this as $A \leq_m^P B$. (We also refer to these reductions as Karp reductions.)
- A problem Y is **NP-hard** if for every problem $X \in$ **NP**, $X \leq_m^P Y$.
- A problem is **NP-complete** if it is both in **NP** and **NP-hard**.
- A Boolean function is one of the form $f : \{0,1\}^n \to \{0,1\}$. They can be thought of as n-bit truth tables.
- A Boolean circuit with n-input variables ($n \geq 0$) is a directed acyclic graph where the vertices represent gates and the directed edges represent wires. The circuit has n input gates each with in-degree 0 and 1 output gate with out-degree 1. In our standard model we include AND, OR, and NOT gates which have in-degree 2 and correspond to their respective binary functions.
- A family of circuits is an infinite sequence $C_0, C_1, C_2, \ldots$ where C_n is a circuit with n input gates
- We say that a family $\mathcal{C}$ decides a language L if for all $n \in \mathbb{N}$, C_n decides $L_n = L \cap \{0,1\}^n$

New Point of View

Imagine there existing an untrustworthy, omnipotent (computationally unbounded) **Prover** who likes to make claims about membership in a language L. On the other hand, you are a **Verifier** who can merely compute things that run in polynomial time. You are interested in verifying if a string is in L.

The **Prover** claims to you that a certain $x \in L$. In order to convince you, the Prover uses its unlimited computational power to provide a polynomial length (with respect to x) certificate/proof to you. You then use the certificate to verify whether x is truly in L. If $L \in$ **NP** then

(a) can the **Prover** convince you for every $x \in L$ that x truly is a member of L?

(b) can the **Prover** ever fool you into thinking some $x \in L$ when really $x \notin L$?

Conversely if L is such a language so that **Prover** can always provide you with polynomial length proofs for $x \in L$, and is never able to deceive you for $x \notin L$ then is $L \in NP$?

No Privacy

3COL: Given an undirected graph, can we color the vertices with 3 colors so that no two adjacent vertices share the same color?

Show 3COL is in **NP**.

Natural Circuits

A language $L \subseteq \{0,1\}^*$ is called skinny if there is some constant $k > 0$ such that for all $n \in \mathbb{N}$, we have $L \cap \{0,1\}^n \leq n^k$.

Show that any skinny language can be computed by a polynomial size language family.