Verifier Model for NP

We saw that SAT $\in$ NP, 3-COLORABLE $\in$ NP and SUBSET-SUM $\in$ NP. In each of these proofs, we "guess" a solution and then deterministically verify it. The natural question, at this point, is:

Can we connect every nondeterministic machine to this format?

A **verifier** for a language L is a decider (algorithm) DTM V such that

$$L = \{ x \mid \exists y, V \text{ accepts } \langle x, y \rangle \}$$

For x to be in L, there should exist y (y is called **proof** | **witness** | **certificate**) such that we can use y to verify that x is in the language.

Note that the existence of y is a lesser requirement than independently being able

to test if $x \in L$.

Theorem 7.20: NP is the class of languages that have polynomial time verifiers.

$L \in NP \iff$ $\exists$ a polynomial time verifier algorithm V such that

$$x \in L \iff \exists y, |y| = \text{poly}(|x|), V(x,y) = 1$$

$$x \notin L \Rightarrow \forall y, |y| = \text{poly}(|x|) \; V(x,y) = 0$$

Contrast this with P.

$L \in P \iff$ $\exists$ a polynomial time decider algorithm A such that

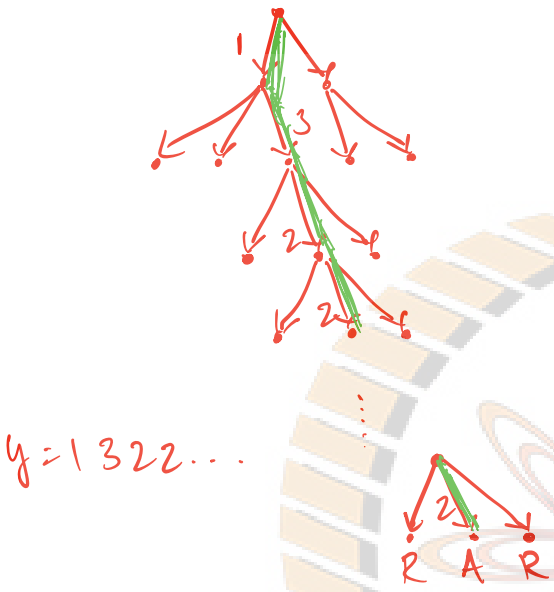
$$x \in L \iff A(x) = 1$$

$$x \notin L \Rightarrow A(x) = 0$$

Proof: Two directions

(1) NP $\Rightarrow$ Poly time verifiers.

Suppose $L \in NP$. Then there is an NTM N that runs in poly time, say n^k time, that decides L.



If $x \in L$, there is an accepting computation path. This path can be verified. The proof/certificate can be the encoding of this path.

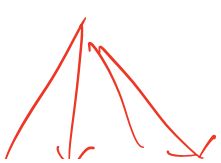
Verifier machine: V on input x, y.

1. Simulates N on input x. When N has to make a non-det. choice, V is guided by y.
2. Accept iff N accepts.

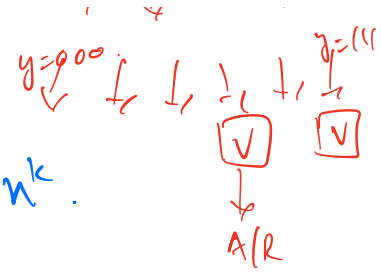
---

(2) Poly time verifier $\Rightarrow$ NP

Suppose there is a poly time verifier V for L. Let us say V runs in time n^k.



NTM decider N:



(1) Guess a string y of length n^k.
(2) Run V on $\langle x, y \rangle$.
(3) Accept if and only if V accepts.

What is the proof / certificate ?

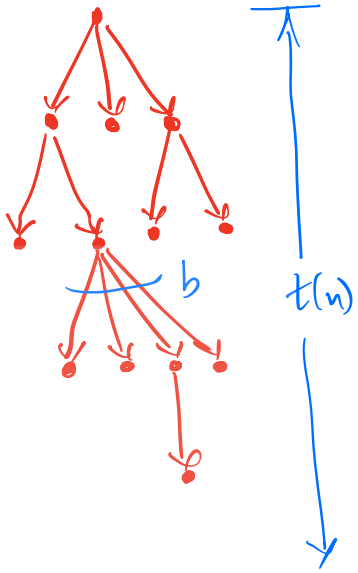
SUBSET-SUM	: The subset itself
CNF-SAT	: The satisfying assignment.
3-COLORING	: The valid 3-coloring.

Theorem 7.11: Let $t(n)$ be such that $t(n) \geq n$. Every NTM which runs in $t(n)$ time has an equivalent DTM that runs in $2^{O(t(n))}$ time.

Proof: Quantifying the proof of Theorem 3.16 seen in lecture 31, where we showed that every NTM has an equivalent DTM.

We simulate all computation paths and check if any one leads to an accept.

Suppose b is the largest number of children of any configuration. Note that b is a constant that depends only on the NTM and not on the input to the NTM.



Then we have $\leq b^{t(n)}$ computation paths. Total time needed to check all these paths using the DTM is upper bounded by

$$\leq b^{t(n)} * t(n)$$

$$= 2^{\log b * t(n)} * 2^{\log t(n)}$$

$$= 2^{[t(n) \log b + \log t(n)]}$$

$$= 2^{O(t(n))}$$

Note: Any NTM is equivalent to an NTM that makes ≤ 2 choices at every step, not $\leq b$. This can be accomplished with only a constant blow-up in time.