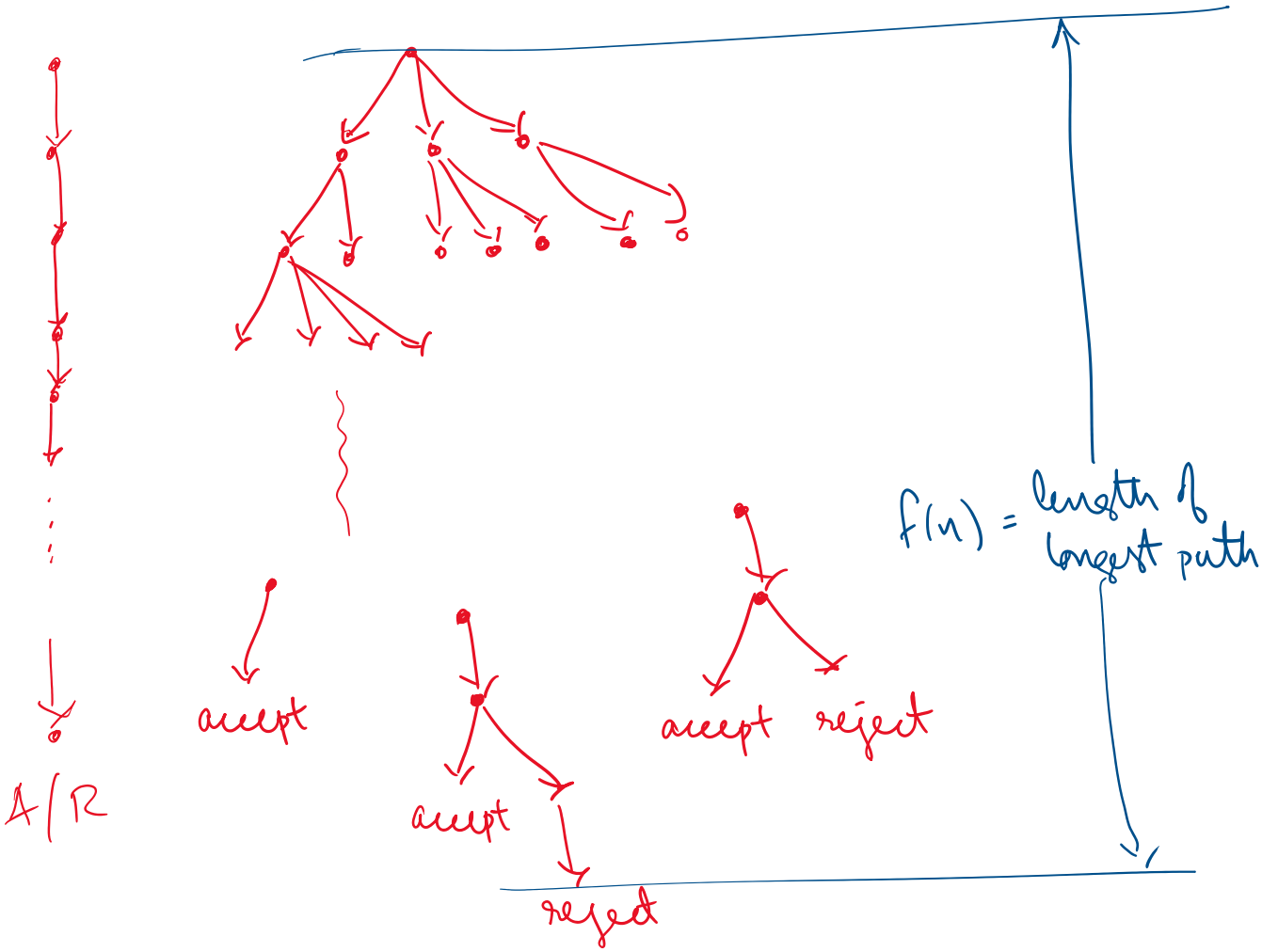
Nondeterministic Polynomial Time

31 March 2021 01:44

Def 7.9: Let N be a non-det TM that is a decider. The running time of N is $f: \mathbb{N} \to \mathbb{N}$ where $f(n)$ is the maximum no. of steps that N uses on any branch of its computation on an input of length n.



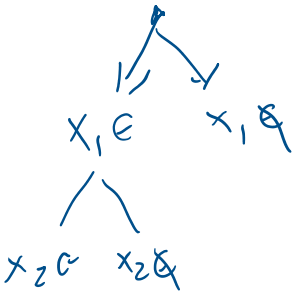
Def 7.21: $NTIME(t(n)) = \{L \mid L$ is decided by an NTM in $O(t(n))$ time$\}$

$\ldots \in NTIME(n^k)$

Def 7.22: $NP = \bigcup_{k=1}^{\infty} NTIME(n^k)$.

This is equivalent to the guess and verify model.

$$SUBSET\text{-}SUM = \{ \langle S, t \rangle \mid S = \{x_1, x_2 \ldots x_k\}, \exists T \subseteq \{1, 2, \ldots k\} \text{ s.t } \sum_{i \in T} x_i = t \}$$



Example: SUBSET-SUM $\in NP$.

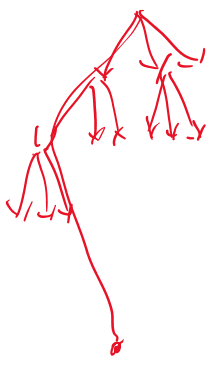
On input $\langle S, t \rangle$:

1. Non deterministically select / reject each of $x_1, x_2, x_3 \ldots x_k$.
2. Add all the selected x_i and verify if they add up to t.
3. If sum $= t$, then accept. Else, reject.

3-COLORABLE $= \{ \langle G \rangle \mid G$ is 3-colorable $\}$.

On input $\langle G \rangle$:

1. Go through the vertices $1, 2, \ldots n$, non-deterministically assigning colors R, G, B.



deterministically assigning colors R, G, B.

2. Go through each edge of G and check whether it is properly colored.

3. Accept if coloring is valid. Else reject.

Accept

Notice that a DTM can also be considered as an NTM.

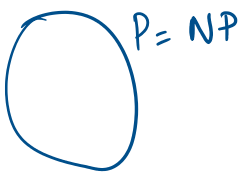
So $DTIME(t(n)) \subseteq NTIME(t(n))$

$DTIME(n^k) \subseteq NTIME(n^k)$

$P \subseteq NP$

P vs. NP question: Is the above containment strict?

Is $P = NP$ or Is $P \subsetneq NP$?



This is a big open question.

SAT: Short for Satisfiability

Π ... in a Boolean formula with

There is a Boolean formula with

Variables x_i, Negated variables $\bar{x}_i$

Literals

AND $\wedge$ OR $\vee$

SAT = $\{ \langle \phi \rangle \mid \phi$ is a Boolean formula that is satisfiable $\}$

There is an assignment of True/False to the Boolean variables such that the whole formula evaluates to True.

CNF formula: Conjunctive Normal Form.

$(x_1 \vee \bar{x}_2 \vee x_3 \vee x_4) \wedge (x_2 \vee x_3 \vee \bar{x}_5) \vee (\bar{x}_3 \vee \bar{x}_6)$

AND's of OR's.

AND's of clauses. Each clause is an OR of literals.

$\{ \langle \phi \rangle \mid \phi$ is a CNF formula

CNFSAT = $\{\langle\phi\rangle \mid \phi$ is a CNF formula that is satisfiable$\}$

3-CNFSAT or 3-SAT $= \{\langle\phi\rangle \mid \phi$ is a CNF formula where each clause has exactly 3-literals, ϕ is satisfiable$\}$

SAT, CNFSAT, 3-SAT are all in NP.

We saw that SAT $\in$ NP, 3-COLORABLE $\in$ NP and SUBSET-SUM $\in$ NP. In each of these three, we "guess" a solution and then verify it. This makes one ask the question: Can we convert every nondeterministic machine in this format?

A <u>verifier</u> 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

For x to be in L, there should exist y (y is called proof / certificate / witness) 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

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

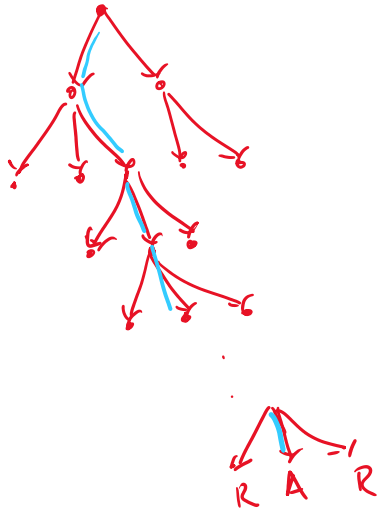
Contrast this with P.

$L \in P \iff \exists$ a polynomial time decider alg. A such that $x \in L \iff A(x) = 1$.

Proof: Two directions.

(1) NP ⇒ Poly time verifiers.

Suppose $L \in NP$. Then there is a 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 choice, V is guided by y.

2. Accept iff N accepts.

---

(2) Poly time verifier ⇒ NP.

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

y ... n^k.

NTM machine 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.