W12L1 P and NP

Monday, April 20, 2020 9:14

<u>Def:</u> Language $L \in P$ if L can be decided by a one-tape deterministic Turing machine in time upper-bounded by a polynomial in the length of the input.

example languages in P:

Runtime on input length n

divisibility: $L = \{ \langle x, y \rangle | y = k \cdot x \text{ for some } k \in \mathbb{Z}\}$
here's a high-level way the Turing machine could work:

1. Check formatting and read x and y — $O(n)$
2. $z = x$ — $O(n)$ to copy z to tape 2
3. while $z < y$: — $O(n)$
   set z to $z + x$ — $O(n)$ } loop runs for ~ y/x times
4. If $z = y$: halt and accept // $y = k \cdot x$ — $O(n)$
5. Reject // y is not a multiple of x — $O(1)$

other example languages in P from the textbook:

greatest common divisor $\{\langle x, y, z \rangle \mid x, y, z \in \mathbb{N}$ and z is the greatest common divisor of x and $y\}$

PATH $= \{\langle G, s, t \rangle \mid G$ is a graph and s and t are nodes, and there is a path from s to t along edges of graph $G\}$

every context-free language

most problems you've encountered in CS (BFS, DFS, sorting, etc.)

<u>NOTE:</u>
Inputs must be **reasonably** encoded, for example binary/decimal/base k for $k \geq 2$.
Unary encoding is **NOT** ok.

<u>Def:</u> Language $L \in NP$ if L can be decided by a one-tape nondeterministic Turing machine in time upper-bounded by a polynomial in the length of the input.

With this definition:

- if there is some branch that accepts, then overall we accept
- if every branch rejects, we reject overall
- the runtime of a nondeterministic TM is the worst-case runtime of the longest-running branch

<u>An alternate definition of NP:</u>
A **verifier** for a language L is a one-tape deterministic Turing machine where inputs are of the form $\langle w, v \rangle$, and we say that
$L = \{w \mid$ there exists some string v where the verifier accepts the pair $\langle w, v \rangle\}$

A polynomial-time verifier must:

- have some certificate v which is of length polynomial in $|w|$
- run in time polynomial in $|w|$

<u>Theorem:</u> $L \in NP$ if and only if L has a polynomial-time verifier.

In order to prove this, we need to show that "has a verifier" is equivalent to "has a nondeterministic decider" (both are polynomially-bounded in runtime).

one direction:
S'pose that L has a one-tape nondeterministic polytime TM N.
Then the verifier will be:
$V =$ "on input $\langle w, v\rangle$:

1. Run N, but use the characters in v to make the nondeterministic choices at each step.
2. If N accepted, accept. Else, reject."

Need to check:

- V is deterministic
- V is polynomial-time in the length of the input
- $w \in L$ if and only if there is some v such that V accepts $\langle w, v\rangle$

other direction:
S'pose that L has a polytime verifier V.
Then we build a nondeterministic TM N:
$N =$ "on input w:

1. Nondeterministically guess a string v.
2. Run V on input $\langle w, v\rangle$.
3. If V accepted, accept. Else, reject."

Need to check:

- N is nondetermistic
- N is polynomial-time in the length of w
- $w \in L$ if and only if N accepts w

So an easy verifier for divisibility:
ask for $\underline{\underline{k}}$

"On input $\langle x, y, k\rangle$:
If $y = k \cdot x$ accept.
Else reject."