1 Nondeterministic Time

1.1 Time Bounded Classes

Time Bounded Nondeterministic Computation

Definition 1. A nondeterministic Turing machine is said to run in *time* $t(n)$ if on any input u, *every* computation of M on u takes at most $t(|u|)$ steps

Definition 2. $L \in \text{NTIME}(t(n))$ iff there is a nondeterministic TM M that runs in time $t(n)$ and $L = \mathbf{L}(M)$

Linear Speedup
Nondeterministic TMs

Theorem 3. *Let M be a k-tape nondeterministic TM running in time $t(n)$. For any constant $c > 0$, there is a $k+1$-tape nondeterministic TM M' such that $\mathbf{L}(M') = \mathbf{L}(M)$ and M' runs in time $ct(n) + n$. (Note, here we think of c being less than 1.)*

Proof. It is the same as the proof for deterministic TMs □

Corollary 4. *If $n = o(t(n))$ and (nondeterministic) M runs in time $t'(n)$ such that $t'(n) = O(t(n))$ then $\mathbf{L}(M) \in NTIME(t(n))$.*

1.2 Relationship with Deterministic Classes

Nondeterministic and Deterministic Classes

Proposition 5. *For any $t(\cdot)$, $DTIME((t(n)) \subseteq NTIME(t(n))$*

Proof. Follows from the fact that a deterministic TM is special kind of nondeterministic TM. □

Proposition 6. *$NTIME(t(n)) \subseteq DTIME(2^{t(n)})$*

Proof. Let M be a nondeterministic TM running in $t(n)$ time, and let d be an upper bound on the number of choices at any step.

- Given a sequence of nondeterministic choices σ, a deterministic machine can simulate M on that sequence.
- On input w, the deterministic machine tries out all sequence of nondeterministic choices of length upto $t(|w|)$, and simulates M, and checks if M accepts w on any such sequence.
- Total time is $\sum_{k=1}^{t(n)} kd^k \leq d^{2t(n)+2} = O(2^{t(n)})$. □

---

2 NP

Nondeterministic Polynomial Time

Definition 7. $\mathrm{NP} = \cup_k \mathrm{NTIME}(n^k)$

Proposition 8. *1.* $P \subseteq NP$

2. $NP \subseteq \bigcup_k DTIME(2^{n^k}) = EXPTIME$

Proof. Follows from the observations relating deterministic and nondeterministic classes. □

2.1 Efficiently Verifiable Languages

Polynomially Verifiable Languages

Definition 9. A *polynomial time verifier* for a language L is a (deterministic) Turing machine V such that

$$L = \{w \mid \exists p.\ V \text{ accepts } \langle w, c\rangle\}$$

and V on input $\langle w, p\rangle$ takes at most $|w|^k$ steps, for some k.

If L has a polynomial verifier then L is said to be *polynomially verifiable*.

Remark
Since a polynomial verifier V for L runs in time $|w|^k$ (for some k) on input $\langle w, p\rangle$, it must be the case that $|p| \leq |w|^k$.

Examples

Example 10. Let COMPOSITES $= \{n \in \mathbb{N} \mid \exists p, q.\ n = pq \text{ and } p, q > 1\}$. COMPOSITES are polynomially verifiable. The "proof" that n is composite are factors p and q such that $n = pq$. Observe that $|p|, |q| \leq |n|$ (so the proof is polynomially bounded), and the product pq can be computed in time that is bounded by $|p||q|$.

Example 11. A *Hamiltonian path* in a directed graph G, is a path that visits every vertex (in G) exactly once. Let

$$\text{HAMPATH} = \{\langle G, s, t\rangle \mid \begin{array}{l} G \text{ is a directed graph that has a} \\ \text{Hamiltonian path from } s \text{ to } t\} \end{array}$$

HAMPATH is polynomially verifiable. The proof that $\langle G, s, t\rangle \in$ HAMPATH is a Hamiltonian path π from s to t. Observe that $|\pi|$ is equal to the number of vertices in G, and given a path π, one can check in linear time if it is Hamiltonian path from s to t.

Non-Example (?)

Example 12. $\overline{\text{HAMPATH}}$ which is the complement of HAMPATH may not be polynomially verifiable. It seems like the only "proof" that a graph G does not have a Hamiltonian path, would be to go through all permutations on the vertices of G and check that thet are not valid paths.

Note, the above is just an argument for why $\overline{\text{HAMPATH}}$ may not be polynomially verifiable. Nobody knows of a precise proof that establishes this fact.

NP and efficient verifiers

Theorem 13. *$L \in NP$ iff L has an efficient verifier.*

Proof. ($\Rightarrow$) If M recognizes L in time n^k then the "proof" for a string $w \in L$ is a string p that lists the sequence of nondeterministic choices that leads M to accept the input w. Thus, the verifier for L, on input $\langle w, p\rangle$, simulates M on input w taking the symbols of p as the nondeterministic choices to be taken at each step. V accepts $\langle w, p\rangle$ if the simulation of M accepts.

($\Leftarrow$) Let V be a verifier for L that runs in n^k time. The nondeterministic TM for L will be as follows

```
On input w
    Nondeterministically pick a string p of length |w|^k
    Run V on ⟨w,p⟩ and accept only if V does
```

$\square$

2.2 P versus NP

$$\text{Is P} = \text{NP?}$$

Can the collection of problems that have short, efficiently checkable proofs, be the same as the collection of problems for which you can *find* short, efficiently checkable proofs, *efficiently*?
