Lecture 16

CSE 431
Intro to Theory of Computation

Last time:

RELPRIME $\in P$

FACTOR $= \{\langle N, k, \ell \rangle :$ integer $N$ has a factor $m$ with $k \le m \le \ell \}$

FACTOR $\in$ NP $\quad \in P$? open

Textbook: COMPOSITES $= \{\langle N \rangle : N$ is an integer s.t. $\exists p, q$ s.t. $pq = N$, $1 < p, q < N \}$

$\langle N \rangle \in$ COMPOSITES $\Leftrightarrow \langle N, 2, N-1 \rangle \in$ FACTOR

$\in$ NP

$\forall A \in$ CFL, $A \in P$

---

Alternate characterizations of NP

Recall that $\quad NP = \bigcup_k NTIME(n^k)$

set of languages

We will use notation $n^{O(1)}$ to mean $n^k$ for some $k$

"polynomial"

**Thm** ① $A \in NP$ class

$\Longleftrightarrow$ ② $\exists$ deterministic polytime TM $V$ (a "verifier") s.t. $\forall x$. $x \in A \Leftrightarrow \exists y$. $|y|$ is $|x|^{O(1)}$ and $V$ accepts $\langle x,y \rangle$.

③ text

$\Longleftrightarrow$ $\exists$ deterministic TM $V'$ s.t. $V'$ on input $\langle x,c \rangle$ runs in time polynomial in $|x|$ and $\forall x$. $x \in A \Leftrightarrow \exists c$ s.t. $V'$ accepts $\langle x,c \rangle$

"certificate", or "proof" that $x \in L$

Example using defn ②:

COMPOSITES $\in NP$:

For input $N$ — $n$ bits

Certificate: integers $p, q$ — each at most $n$ bits

Verify: check that $N = p \cdot q$ — time at most $O(n^2)$

**Proof of Thm**

First observe that ② $\Leftrightarrow$ ③

② $\Rightarrow$ ③: We can use $c = y$ and $V' = V$

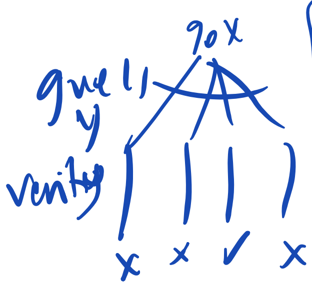
easy — $|y|$ is only $|x|^{O(1)}$ so total running time of $V'$ is $|x|^{O(1)}$ as required

③ $\Rightarrow$ ②: Suppose there is such a $V'$. We use $V = V'$. Since $V'$ runs in $|x|^{O(1)}$ time, it only can look at 1st $|x|^{O(1)}$ bits of $c$. Let $y$ be those bits.

a bit harder

② ⇒ ①: Given verifier V for A:

Create NTM for A as follows:




On input x:

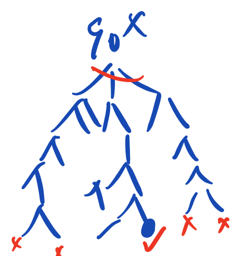
- Use nondeterministic queries to guess a string y of length $|x|^{O(1)}$ and create string $\langle x,y \rangle$
- Run V on input $\langle x,y \rangle$. accepting iff V does.

Total time is polynomial.

① ⇒ ②:

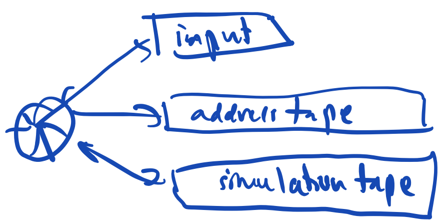
Suppose we have an NTM M for A:

In general M will have a computation tree that branches at any time




address of node: sequence of move choices starting from root.

Recall simulation of NTMs by TMs: (BFS of tree)




steps:
- copy input to tape 3
- use address tape on tape 2 to deterministically simulate on tape 3. If M accepts then accept.
- increment address & repeat.

poly length

Idea here: certificate y is a possible address of polynomial.

poly time since # of steps = length of y.

verifier V checks that simulation on input x using address string y accepts. ∎

## More examples in NP:

HAMPATH = $\{\langle G\rangle$ : G is a directed graph with a path that touches each vertex exactly once $\}$

**HAMPATH $\in$ NP:**

On input $\langle G\rangle$: $n$ = # vertices of G

poly length [ Certificate : sequence of vertices of length $n$

poly time [ verify : • all vertices are different
• each adjacent pair of vertices $u,v$ has a directed edge $(u,v)$ in G

SAT = $\{\langle\varphi\rangle$: $\varphi$ is a Boolean (propositional logic) formula that is satisfiable $\}$

has a truth assignment that makes it true.

**SAT $\in$ NP:**

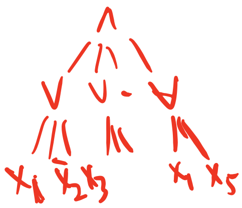
poly length [ Certificate : truth assignment $\alpha$

poly time [ Verify : $\varphi$ evaluates to true under assignment $\alpha$ (and $\varphi$ is a Boolean formula)

CNFSAT = $\{\langle\varphi\rangle$ : $\varphi$ is a satisfiable formula in Conjunctive Normal Form (CNF)$\}$

Recall CNF from 311: (a.k.a. "product-of-sums")

Recall: A CNF is an $\wedge$ of clauses
a clause is an $\vee$ of literals
a literal is a propositional logic variable or its negation, $x_i$ or $\overline{x_i}$
$\neg x_i$




CNFSAT$\in$NP: • same certificate as SAT
• only change is that Verifier checks that $\varphi$ is a CNF in addition to checking $\alpha$.

Def$^n$ kCNF formula is a CNF formula where each clause has length $\leq k$ ($\leq k$ literals)

kSAT = $\{\langle\varphi\rangle : \varphi$ is a satisfiable kCNF$\}$

kSAT$\in$NP: as before but checks kCNF format for input, too.

---

Relations among problems

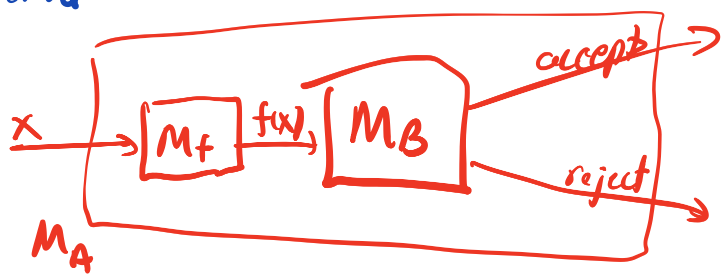
Def Given $A, B \subseteq \Sigma^*$, A is polynomial-time mapping reducible to B
written $A \leq_p B$ (text), or $A \leq_m^p B$ (more standard)
iff There is a poly time computable $f: \Sigma^* \to \Sigma^*$
s.t. $\forall x \in \Sigma^*$, $x \in A \Leftrightarrow f(x) \in B$

only difference from $A \leq_m B$

Thm. • If $A \leq_m^p B$ and $B \in P$ then $A \in P$

• If $A \leq_m^p B$ and $B \in NP$ then $A \in NP$

Proof. As usual given poly time machine $M_B$ for $B$ and $M_f$ for reduction $f$ define




This is correct as always. Only need to check run time

Since both are polytime: Let runtime of $M_f$ be $O(n^k)$
" " $M_B$ be $O(n^\ell)$

Total runtime for $M_A$ on input $x$

$$O(|x|^k) + O(|f(x)|^\ell)$$

which is $O(|x|^k) + O([\underline{O(|x|^k)}]^\ell)$

since size of output at most runtime

which $O(|x|^{k\ell})$ & is polynomial

For NTM same argument applies