Last time:

M has running time T, for $T: \mathbb{N} \to \mathbb{N}$, when

$T(n) = \max\{$ # of steps M takes in the worst case over all inputs $w \in \Sigma^*$ and over all computation choices if M is an NTM $\}$

$\mathrm{NTIME}(T(\cdot)) = \{A$: some multitape NTM with running time $O(T(n))$ decides $A\}$

A: language "set of strings" "yes/no question"

Thm $\mathrm{TIME}(T(n)) \subseteq \mathrm{NTIME}(T(n)) \subseteq \bigcup_{c>0} \mathrm{TIME}(2^{cT(n)})$

$\mathrm{NTIME}(2^{O(T(n))})$.

polynomial time $P = \bigcup_k \mathrm{TIME}(n^k)$ $\quad \mathrm{TIME}(n^{O(1)})$

non-deterministic polynomial time $NP = \bigcup_k \mathrm{NTIME}(n^k)$ $\quad \mathrm{NTIME}(n^{O(1)})$

Note:
- $P \subseteq NP$ (every TM is an NTM)
- P is a set of languages (decision problems)

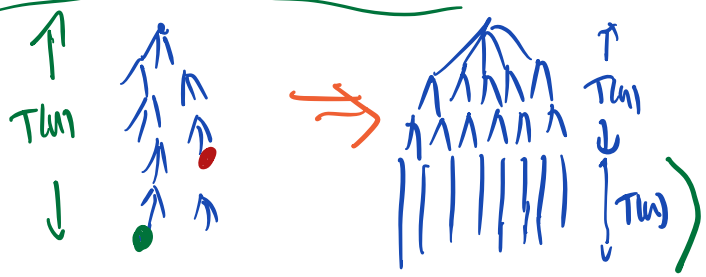
FP is the set of polynomial-time computable functions

RELPRIME $\in$ P , PATH $\in$ P

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

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

Normal Form for NTM's:



sequence of moves / "address" of leaf of NTM tree.

Guess of length TM1 that only depends on input length

Verify of length TM2

Check that following guess is consistent and leads to accepting.

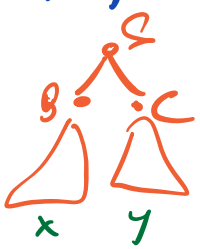
Thm For every regular language A
$A \in TIME(n)$

Proof: Use TM corresponding to DFA for A.
On input of length n takes n steps ☐

Thm For every CFL A, $A \in TIME(n^3)$

Proof Use Chomsky Normal Form and dynamic programming

parse tree is binary.
$A \to BC$



x, y strictly shorter than w

$w =$

# of parse trees possible for strings of length $\le l$ is finite. so decidable

Cocke-Kasami-Younger algorithm idea:

Every variable B in a parse tree for w generates a substring of w, defined by some indices $i \le j$.

Use dynamic programming.

See Slides

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

NP

Thm ① $A \in NP$ class

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

$\Leftrightarrow$ ③ text $\exists$ 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 def 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 $O(n^2)$

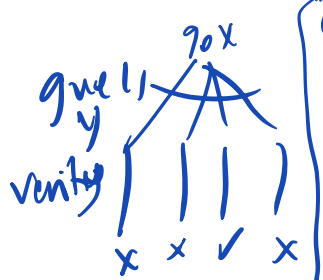
Proof of Thm

First observe that ② ⇔ ③

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

② ⇒ ① : Given verifier V for A:
Create NTM for A as follows:

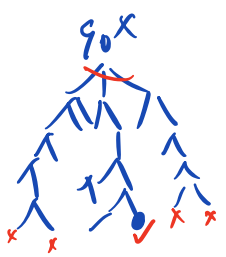


On input x:
- Use nondeterministic guesses to guess a string y of length $|x|^{O(1)}$ and create string $\langle x, y \rangle$
- Run V on input $\langle x, y \rangle$. accept iff V does.

① ⇒ ② : Total time is polynomial.

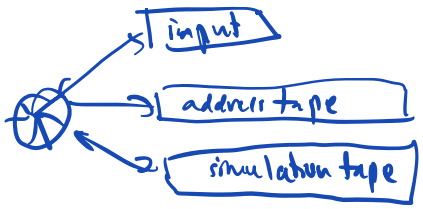
Suppose we have an NTM M for A:
In general M will have a computation tree that branches at any time

more detail on normal form



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

Recall simulation of NTM's by TM's: (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.

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

verifier V checks that simulation on input x using address string y accepts. (poly time since # of steps = length of y.)

More examples in NP:

HAMPATH = {<G> : G is a directed graph with a path that touches each vertex exactly once }

HAMPATH ∈ NP:

On input <G>: n = # vertices of G

poly time:
- Certificate: sequence of vertices of length n
- Verify:
  - all vertices are different
  - each adjacent pair of vertices u,v has a directed edge (u,v) in G

SAT $\in$ NP:

poly-length Certificate: truth assignment α

poly-time Verifier: φ evaluates to true under assignment α (and φ is a Boolean formula)

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

has a truth assignment that makes it true.

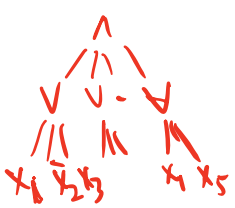
CNFSAT $= \{\langle \varphi \rangle : \varphi$ is a satisfiable formula in Conjunctive Normal Form (CNF)$\}$

Recall CNF from 311: (aka. "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 $\overline{x_i}$ (or $\neg x_i$)



CNFSAT $\in$ NP:
- same certificate as SAT
- only change is that Verifier checks that φ is a CNF in addition to checking α.

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 check kCNF format for input, too.

Relation 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 polytime computable $f: \Sigma^* \to \Sigma^*$ st. $\forall x \in \Sigma^*$, $x \in A \Leftrightarrow f(x) \in B$

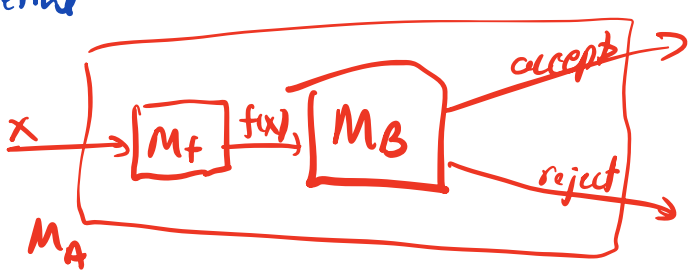
only difference from $A \leq_m B$

~

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 polytime machines 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 polynom: Let runtime of M_f be $O(n^k)$
M_B be $O(n^l)$

Total runtime for M_A on input x

$$O(|x|^k) + O(|f(x)|^l)$$

which is $$O(|x|^k) + O([O(|x|^k)]^l)$$

since size of output at most runtime

which $O(|x|^{kl})$
& is polynomial

For NTM same argument applies

Summary:

$A \leq_m^P B$
"A is at most as hard as B up to "polynomial slop""

- polynomial-time mapping reductions
  $A \leq_M^P B$ (or $A \leq_p B$)
  just like def^n of mapping reduction except f must be poly time computable

Thus
- 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 $A \in NP$

We now add defⁿs:

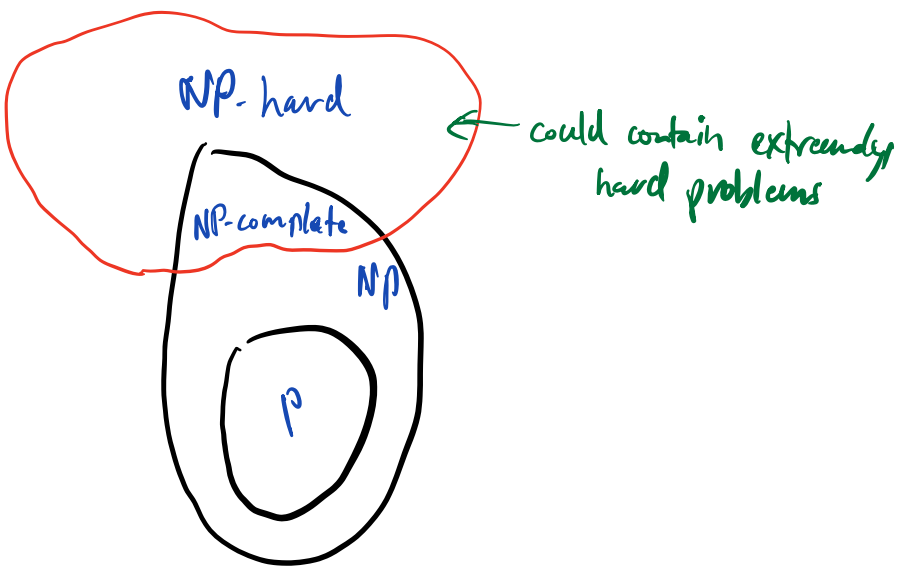
Defⁿ. B is **NP-hard** iff $\forall A \in NP.\ A \leq_m^p B$

Defⁿ. B is **NP-complete** iff

- $B \in NP$ and
- B is NP-hard

Cor If some NP-hard problem is in P then $P = NP$.

We now look at a picture of these classes



Thm [Cook, Levin] 3SAT is NP-complete

↑ special case of SAT for 3CNF formulas

We prove this via a different SAT problem:

CIRCUIT-SAT = $\{\langle C \rangle : C$ is a Boolean circuit s.t. $\exists$ input y s.t. $C(y)=1\}$

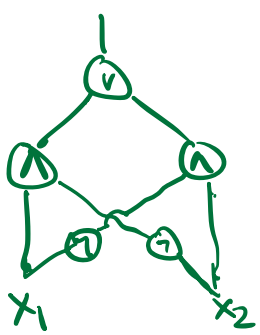
Boolean circuit: inputs $x_1, \ldots, x_n$

gates: (bottom up) 

Given by graph: output gate special marked gate.

Computes a Boolean function $f: \{0,1\}^n \to \{0,1\}$

e.g. Parity, aka. XOR
$x_1 \oplus x_2$

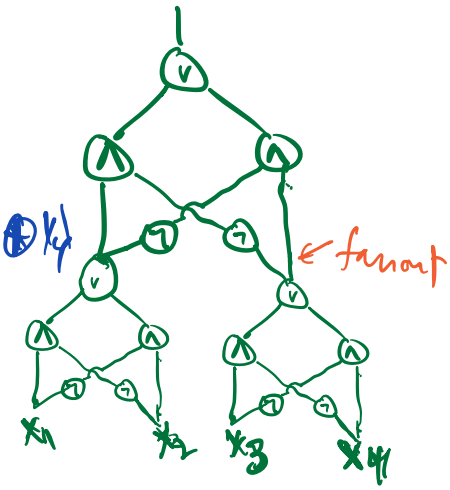


size = # gates

depth = length of longest path from input to output.

eg. Can combine these units into a larger circuit for functions like $x_1 \oplus x_2 \oplus \cdots \oplus x_n$

Circuit for $(x_1 \oplus x_2) \oplus (x_3 \oplus x_4)$



Size $O(n)$ in general

Without fan-out $\wedge, \vee, \neg$ takes $O(n^2)$ gates to compute $x_1 \oplus x_2 \oplus \cdots \oplus x_n$

Note: TMs work on inputs of all lengths
Circuits only work for a fixed input length

Next time:

Thm (Shannon) Almost all functions on n bits require circuit size $\Omega(2^n/n)$