Examples of problems in NP

SAT $= \{ \langle \varphi \rangle \mid \varphi(x_1, \dots, x_n)$ is a satisfiable propositional formula $\}$

HamPath $= \{ \langle G, s, t \rangle \mid G$ is a digraph with a Hamiltonian path from s to t $\}$

Clique $= \{ \langle G, k \rangle \mid G$ is a graph with a clique of size k $\}$

SubsetSum $= \{ \langle a_1, a_2, \dots, a_n ; t \rangle \mid \exists S \subseteq \{1, \dots, n\}$ s.t. $\sum_{i \in S} a_i = t$ $\}$

Exercise: Check that all the problems above $\in$ NP.

Search-to-Decision Reductions

SAT : Given $\varphi(x_1, \dots, x_n)$, decide if φ is satisfiable.

SAT-Search : Given $\varphi(x_1, \dots, x_n)$, find a satisfying assignment, if it exists.

Claim: If SAT $\in$ P, then there is a

Claim: If SAT $\in$ P, then there is a deterministic polytime algorithm for SAT-Search.

Proof: Given $\varphi(x_1, \dots, x_n)$,

if $\langle \varphi \rangle \notin$ SAT **then** return "No"
endif
for $i = 1..n$
  $a_i = 0$
  if $\langle \varphi(a_1, \dots, a_i, x_{i+1}, \dots, x_n) \rangle \notin$ SAT
  then $a_i = 1$
  endif
endfor
return $(a_1, a_2, \dots, a_n)$

Binary Search on $\{0,1\}^n$ using SAT-algorithm

Suppose $\varphi(x_1, \dots, x_n) \in$ SAT
Then $\varphi(0, x_2, \dots, x_n) \in$ SAT <u>OR</u>
$\varphi(1, x_2, \dots, x_n) \in$ SAT.

Set $a_1 \in \{0,1\}$ so that $\varphi(a_1, x_2, \dots, x_n) \in$ SAT
& recurse on $x_2, \dots, x_n$.
Return $(a_1, a_2, \dots, a_n)$ as a satisfying assignment.

Exercise: Design Search-to-Decision reductions for all the example NP problems above.

Caution: Sometimes no obvious "search-to-decision" reduction is known.

Example:

Composite: Given $\langle n \rangle$, decide if n is composite (i.e., has a non-trivial factor).

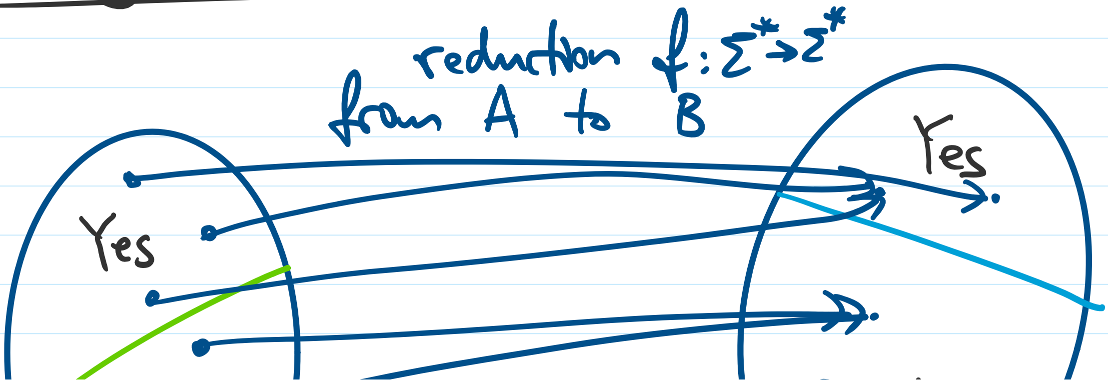
Composite-Search: Given $\langle n \rangle$, find a non-trivial factor of n, if it exists.

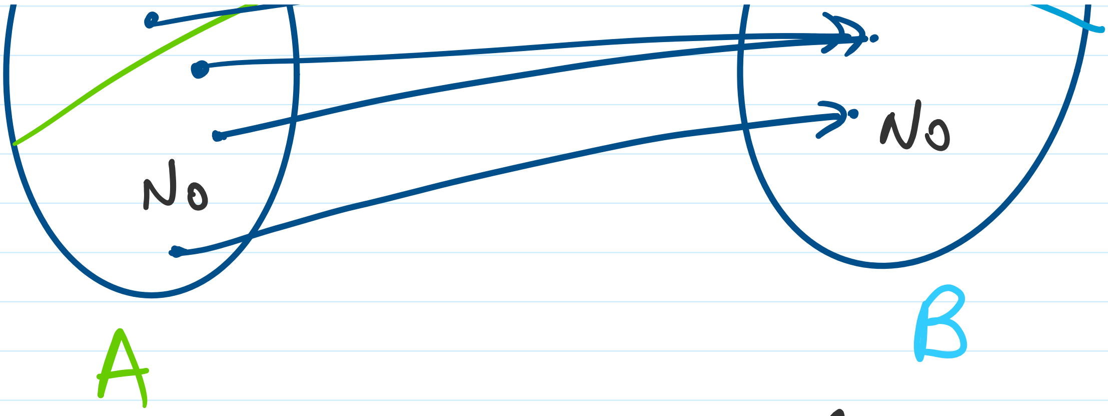
Fact: Composite $\in$ P [Agrawal, Kayal, Saxena 2003]

But, Composite-Search $\equiv$ Factoring is not known to be in PolyTime!

(Yet, Factoring is in Quantum PolyTime.)

Polytime Reductions





f is a polytime reduction from A to B
if (1) f is computable in polytime, and
(2) $\forall x$, $x \in A \iff f(x) \in B$.

(Same as the reductions in Computability Theory, except require "polytime-computable".)

Notation: $A \leq_p B$

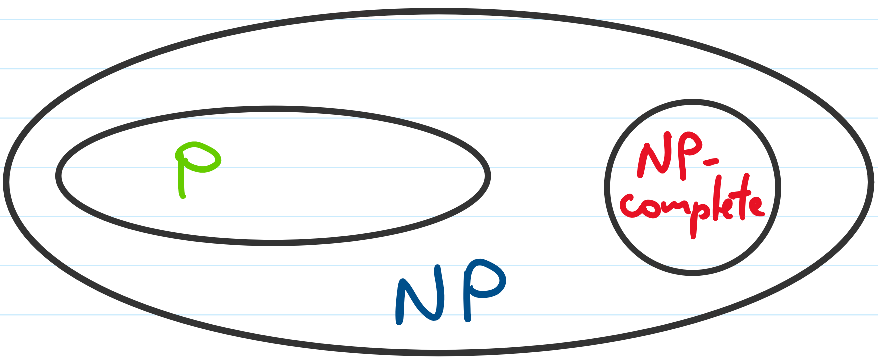
Thm: $A \leq_p B$ & $B \in P \Rightarrow A \in P$.

NP- Completeness

a language B is NP- complete if
(1) $B \in NP$
(2) $\forall A \in NP$, $A \leq_p B$.

A language B is NP-hard if
$\forall A \in NP$, $A \leq_p B$.

Thus, B is NP-complete if
B is NP-hard AND $B \in NP$.



1. Each NP-complete problem B is a <u>hardest</u> problem in NP (every other NP problem is reducible to B).
2. For every two NP-complete problems A, B, we have

$$\left.\begin{array}{l} A \leq_p B \\ B \leq_p A \end{array}\right\} \Rightarrow A \equiv_p B$$

polytime-equivalent

polytime-equivalent

3. Let A be any NP-complete problem.
Then $A \in P \iff P = NP$.

4. $\forall A, B \in NP$,

$$\underbrace{A \text{ is NP-complete } \& \ A \le_p B}$$

$$\Downarrow$$

B is NP-complete.

(Exercise: Verify 3 & 4.)

"Trivial" NP-complete problem

$$A^p_{NTM} = \{ \langle M, w, 1^t \rangle \mid \text{NTM } M \text{ accepts } w \text{ within } t \text{ steps} \}$$

Claim: A^p_{NTM} is NP-complete.

Proof: (1) $A^p_{NTM} \in NP$:

Just simulate M on w, non-det.-ly,

Just simulate M on w, non-det-ly, for t steps.

(2) $\forall L \in NP$, show $L \leq_p A^{p}_{NTM}$:

L has NTM M, deciding L, in time n^c (some const $c>0$).

<u>Need</u>: a polytime - reduction f s.t.

$\forall x$, $x \in L \iff f(x) \in A^{p}_{NTM}$

$f(x) := \langle M, x, 1^{|x|^c} \rangle$. ✓

Natural NP-complete problems

<u>Thm (Cook-Levin Thm)</u>: SAT is NP-complete.

<u>Proof</u>: (1) SAT $\in NP$. ✓

(2) $\forall L \in NP$

$L \leq_p SAT$.