CS 341 – Algorithms

Lecture 18 – NP-completeness

23 July 2021

Today’s Plan

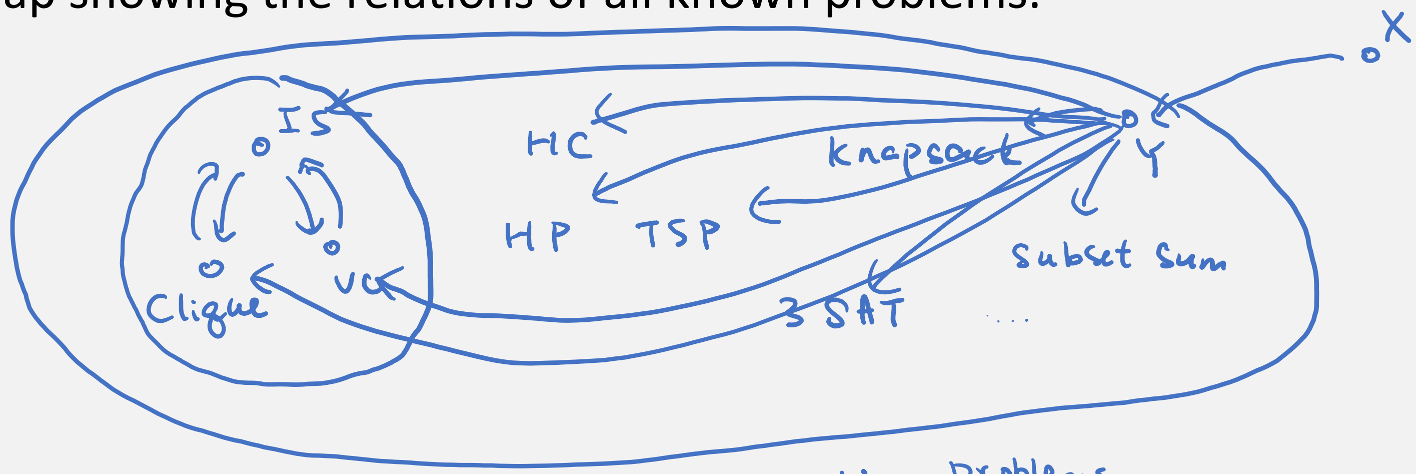
1. The Class NP

2. NP-completeness

3. Cook-Levin Theorem

The Class NP

As we discussed last time, we could do reductions between different problems and slowly build up a huge map showing the relations of all known problems.



$Y \leq_p X$

N problems

$O(N^3)$ reductions

identify "hardest" problems in a large class of problems.

general and abstract definition

Short Proofs

A general feature of the problems is that there is a **short "proof/solution"** of a YES-instance.

e.g. HC YES easy to verify
a YES-instance
if someone gives us a HC

3SAT YES satisfying assignment

Subset-sum YES subset

Formal Definition of NP

easy to check

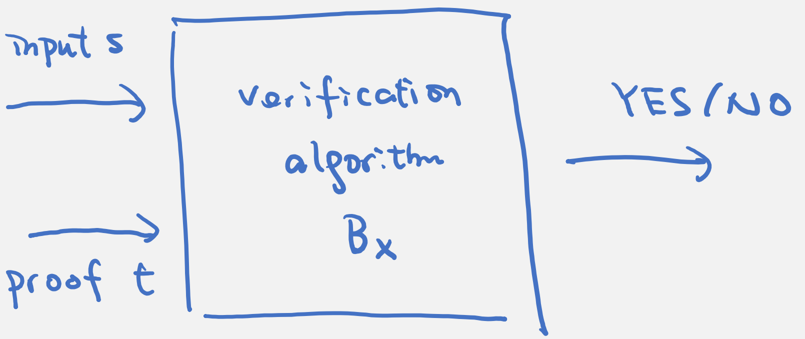
Definition (NP): For a problem X, each instance of X is represented by a binary string s.

A problem X is in the class NP if there is a polynomial time verification algorithm B_X such that

the input s is a YES-instance if and only if there is a proof t which is a binary string of length $poly(|s|)$

short proof

so that $B_X(s, t)$ returns YES.



The key points are B_X is a polynomial time algorithm and t is a short proof of length $poly(|s|)$.

In most problems, t is simply a solution and B_X is an efficient algorithm to check if t is indeed a solution.

Example

Definition (NP): For a problem X, each instance of X is represented by a binary string s.

A problem X is in the class NP if there is a polynomial time verification algorithm B_X such that

the input s is a YES-instance if and only if there is a proof t which is a binary string of length $poly(|s|)$

so that $B_X(s,t)$ returns YES.

Claim. Vertex Cover is in NP.

instance is a graph $G = (V,E)$, an integer k

s is a binary string of G and k

t is a binary string of $S \subseteq V$

B_X takes s and t as input. check if every edge is covered by S

check if $|S| \le k$

iff YES → vertex cover S accept

NO → no vertex cover of size $\le k$. reject. $\square$

More Examples

Claim. 3SAT is in NP.

proof: truth assignment ← size $\leq n$

algorithm: check if it is a satisfying asg ← runtime $O(m)$ m # clauses

clear iff

$\square$

Exercises: Clique, IS, HC, HP, Subset-Sum are all in NP.

$S \subseteq V$

Remark 1: Non-Examples

non-HC : YES iff graph has no HC.

don't know of a short proof that G has no HC

Remark 2: co-NP

$X \in$ co-NP if No-instance $\exists$ short proof
and efficient verification

bipartite matching : graph G has a matching $\geq k$?

No-instance : graph has no matching $\geq k$

$\Updownarrow$ Konig's theorem

has a vertex cover $< k$

bipartite matching $\in$ NP $\cap$ co-NP

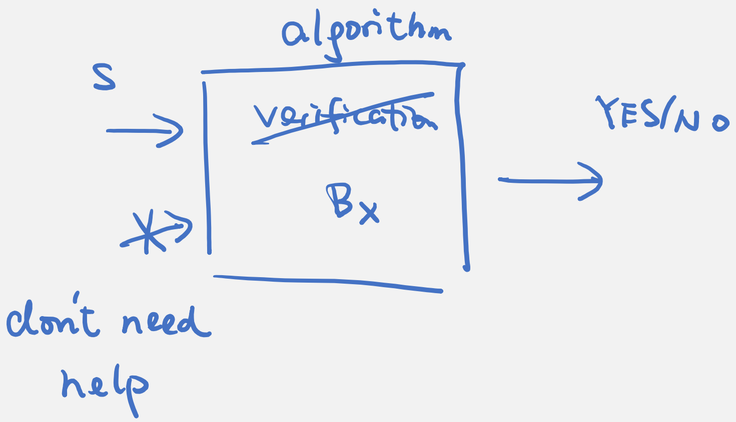
common belief NP $\neq$ co-NP

linear programming NP $\cap$ co-NP , 1984 P

graph isomorphism NP $\cap$ co-NP

Babai $O(n^{\log n})$

Remark 3: $P \subseteq NP$



$X \in P$

Remark 4: Non-Deterministic Polynomial Time

poly time

NP: class of problems solvable by a non-deterministic Turing machine

$\parallel\!\!\!|$

NP: short proof

deterministic Turing machine

(state, input) $\to$ state

non-deterministic

multiple possible states

Remark 5: P=NP?

every problem in NP can be solved in polytime?

Clay 1M prize

Common belief $P \neq NP$

Today’s Plan

1. The Class NP
2. NP-completeness
3. Cook-Levin Theorem

NP-completeness

Informally, we say a problem is NP-complete if it is a hardest problem in NP.

Definition. A problem $X \in \mathrm{NP}$ is NP-complete if $Y \leq_p X$ for all $Y \in \mathrm{NP}$.

Proposition. P=NP if and only if an NP-complete problem can be solved in polynomial time.

$\Rightarrow$ trivial $\quad\quad$ $\Leftarrow$ polytime reduction

Theorem. (Cook-Levin) 3-SAT is NP-complete.

$3SAT \leq_p IS$

Proving NP-completeness

To prove that a problem X is NP-complete, we first prove that X is in NP,

and then we find an NP-complete problem Y and prove that $Y \leq_p X$.

$Y =$ 3SAT , IS

prove hardness by giving an algorithm

problem X, usually solve it $X \leq_p Y$ algorithm

to prove hardness, assume we know how to solve X

find $Y \leq_p X$ easy to make mistake

NPc $X \leq_p Y$ NPc

Today’s Plan

1. The Class NP
2. NP-completeness
3. Cook-Levin Theorem

Cook-Levin Theorem

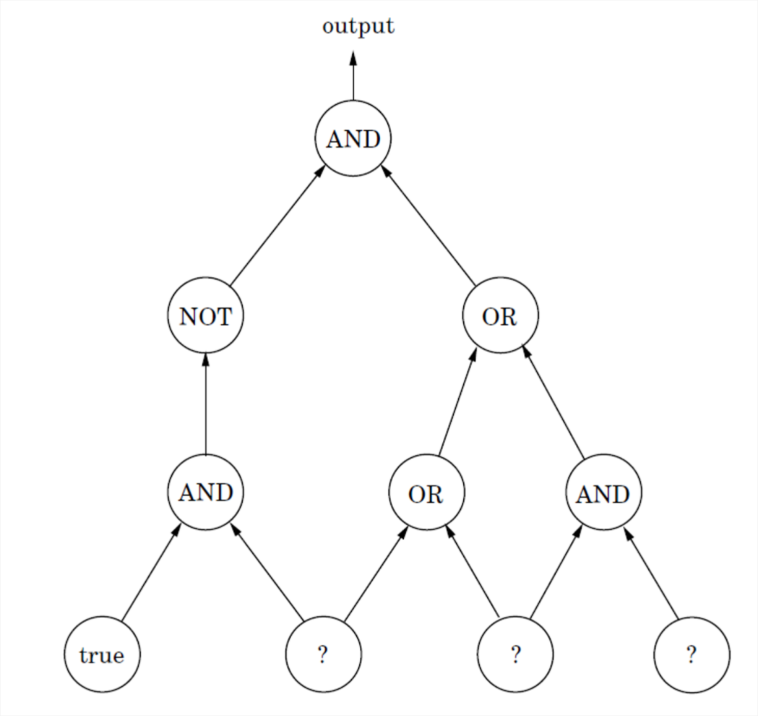
We introduce an intermediate problem in order to prove that 3SAT is NP-complete.

Circuit-SAT

Input: A circuit with AND/OR/NOT gates, some known input gates, and some unknown input gates.

Output: Is there a truth assignment on the unknown input gates so that the output is True?

We can assume that the input circuit is a directed acyclic graph,
and each AND/OR gate has only two incoming edges.

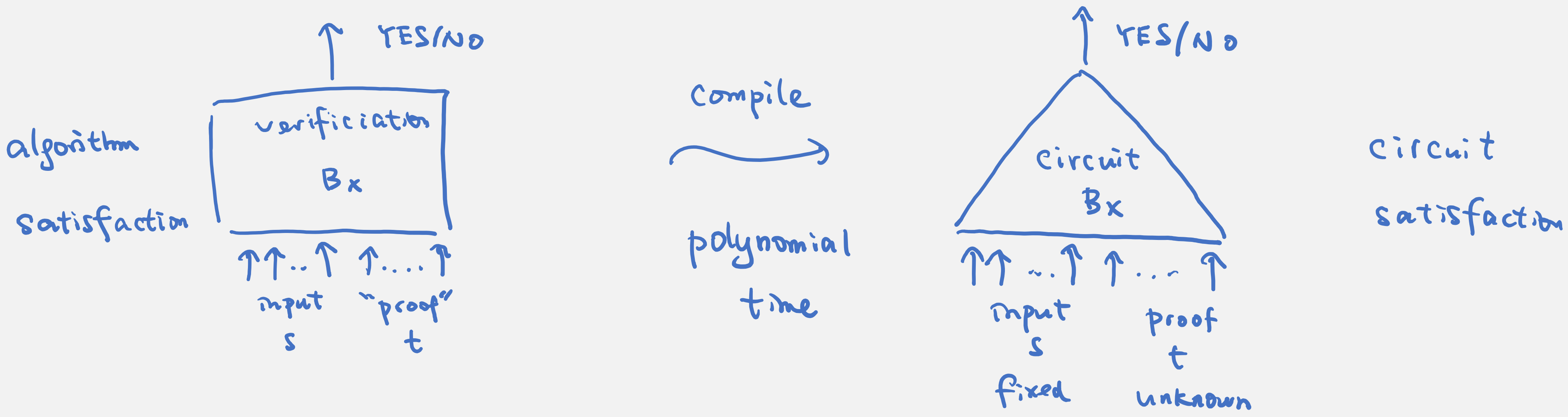


Theorem. Circuit-SAT is NP-complete.

Proof Sketch

Theorem. Circuit-SAT is NP-complete.

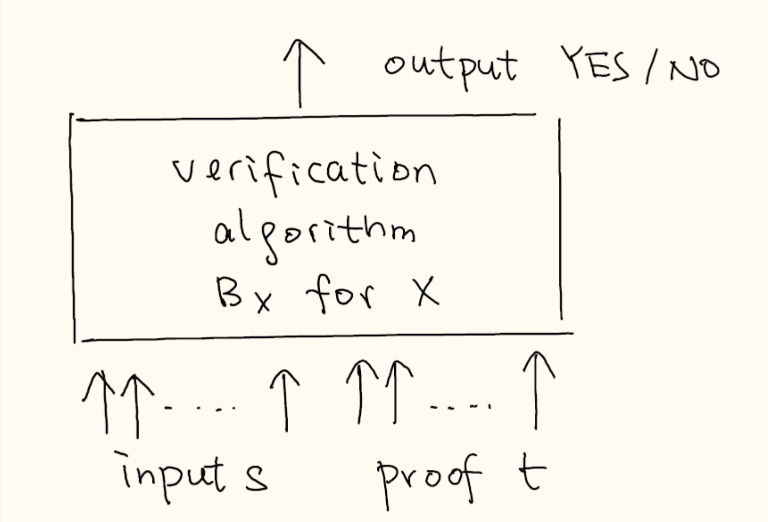
Proof sketch. We start from the abstract definition of NP to prove that $X \leq_p$ Circuit-SAT for any $X \in$ NP.

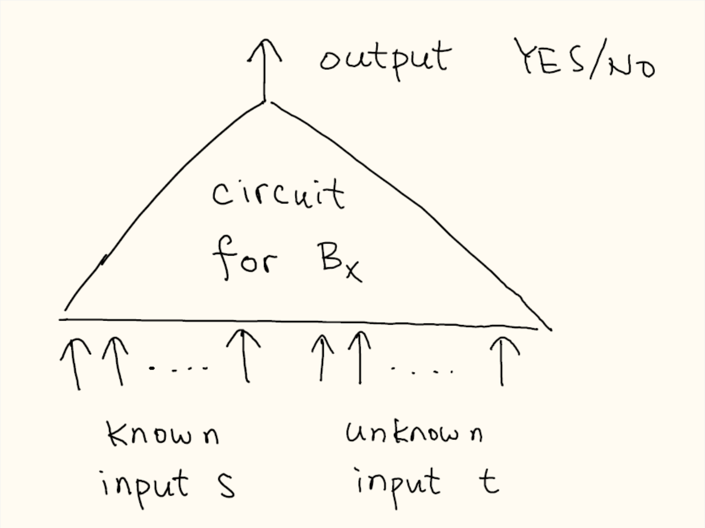


The main conceptual idea is that a circuit is as general as an algorithm.

The original proofs of Cook and Levin directly transforms a non-deterministic Turing machine into a formula.

Proof Sketch Continued





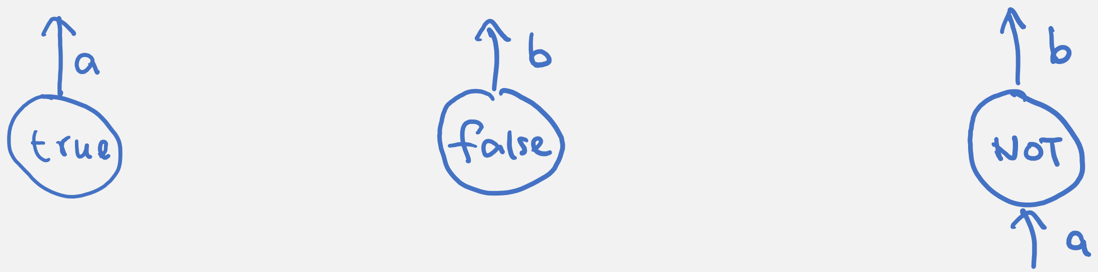
Claim. Input s is a YES-instance if and only if there is a satisfying assignment for Circuit-SAT.

From Circuit to Formula

Now we show that a Boolean formula has the same expressive power as a Boolean circuit.

<u>Theorem</u>. Circuit-SAT $\leq_p$ 3-SAT.

<u>Proof</u>. Given a circuit of n gates, we will construct a formula with $O(n)$ variables so that the circuit is satisfiable if and only if the formula is satisfiable.



$p \to q \equiv \neg p \vee q$

clause (a)

$a = T$

clause $(\bar{b})$

$b = F$

$a \leftrightarrow \bar{b}$

$= a \to \bar{b} \;\wedge\; \bar{a} \to b$

$= (\bar{a} \vee \bar{b}) \wedge (a \vee b)$

From Circuit to Formula

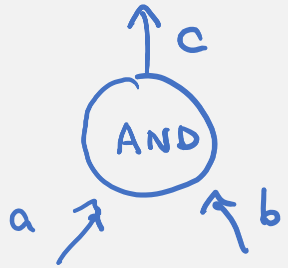
Now we show that a Boolean formula has the same expressive power as a Boolean circuit.

<u>Theorem</u>. Circuit-SAT $\leq_p$ 3-SAT.

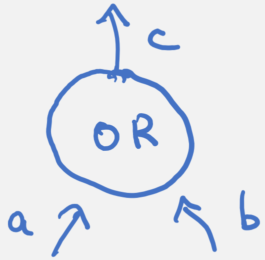
<u>Proof</u>. Given a circuit of n gates, we will construct a formula with $O(n)$ variables so that the circuit is satisfiable if and only if the formula is satisfiable.

$p \to q \equiv \neg p \vee q$

$\overline{a \wedge b} = \bar{a} \vee \bar{b}$



$(a \wedge b) \Leftrightarrow c$



$(a \vee b) \Leftrightarrow c$

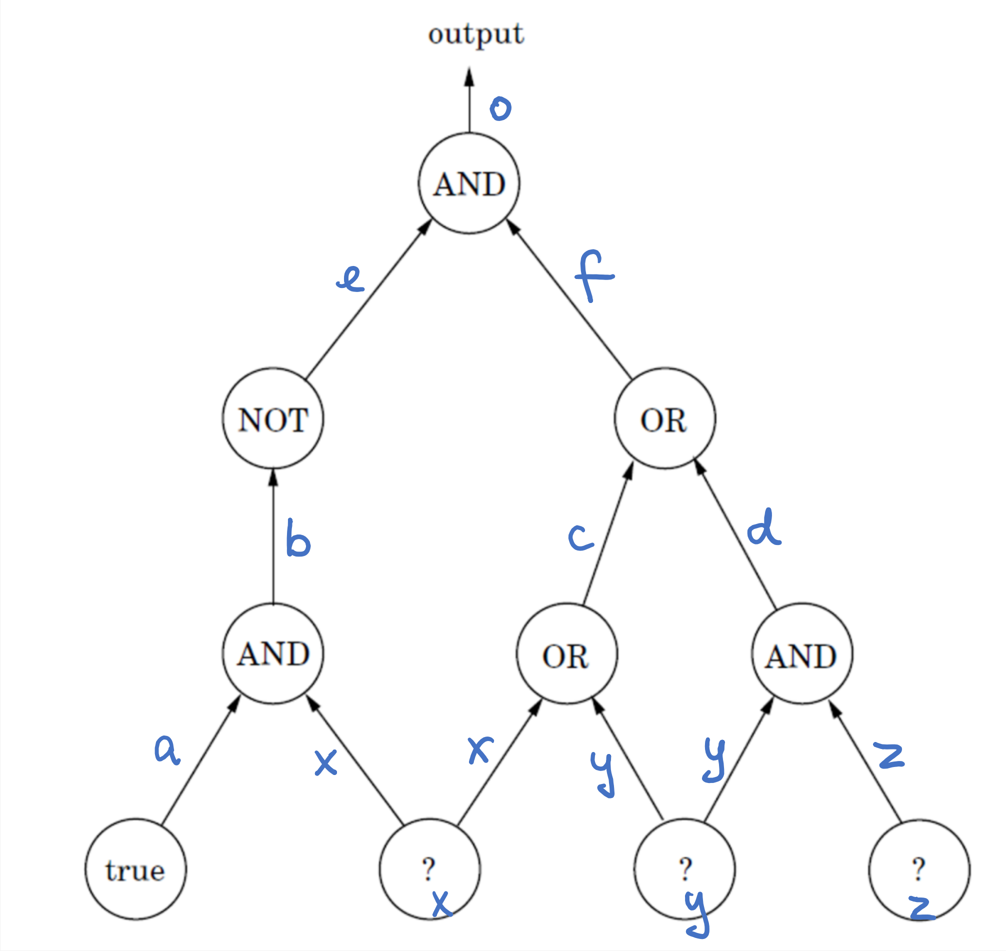
$= a \wedge b \to c \quad \wedge \quad \overline{(a \wedge b)} \to \bar{c}$

$= \overline{(a \wedge b)} \vee c \quad \wedge \quad (\bar{a} \vee \bar{b}) \to \bar{c}$

$= \bar{a} \vee \bar{b} \vee c \quad \wedge \quad \overline{(\bar{a} \vee \bar{b})} \vee \bar{c} = (\bar{a} \vee \bar{b} \vee c) \wedge ((a \wedge b) \vee \bar{c}) = (\bar{a} \vee \bar{b} \vee c) \wedge (a \vee \bar{c}) \wedge (b \vee \bar{c})$

Example

Claim. The circuit is satisfiable if and only if the formula is satisfiable.



$(o) \wedge (a)$

$\wedge\ ((a \wedge x) \Leftrightarrow b)$

$\wedge\ ((x \vee y) \Leftrightarrow c)$

$\wedge\ ((y \wedge z) \Leftrightarrow d)$

$\wedge\ (b \Leftrightarrow \bar{e})$

$\wedge\ ((c \wedge d) \Leftrightarrow f)$

$\wedge\ ((e \wedge f) \Leftrightarrow o)$

$\rightarrow$ turn it into CNF using the previous slides

polytime ✓

Concluding Remarks

With the Cook-Levin theorem, we have a firm foundation to prove that a problem is NP-complete.

We will grow our list of NP-complete problems in the next two lectures.