NP and NP completeness

Efficient Certification

There is a big difference between **FINDING** a solution and **CHECKING** a solution

Independent set problem: in graph G, is there an independent set S of size at least k?

If I give you such a set S, then you can check in polynomial time, that S is independent: check all pairs in S

3-SAT: is there a truth assignment for 3-SAT CNF instance Φ ?

If I give you a truth assignment for CNF instance Φ, you can check in P time that it satisfies Φ: evaluate the expression.

Decision Problems

- Identify (the instances with a "yes" answer of) a Decision Problem with a set of binary strings X
- Instance: string s (containing the inputs)
- Algorithm A solves problem X: A(s) = `yes` iff $s \in X$.

Polynomial time. Algorithm A runs in poly-time if for every string s, A(s) terminates in at most p(|s|) "steps", where p(·) is some polynomial.
|s| is the length of s in bits.

PRIMES problem: given an int p > 1, is p prime?
PRIMES: X = { 2, 3, 5, 7, 11, 13, 17, 23, 29, 31, 37, …. }
Algorithm:
[Agrawal-Kayal-Saxena, 2002] $p(|s|) = |s|^{12}$.
Later improved to $|s|^6$ [Pomerance, Lenstra]
(See wikipedia: AKS primality test)

NP

Certification algorithms.

- Certifier doesn't determine whether $s \in X$ on its own; rather, it checks a proof that $s \in X$.

Def. Algorithm $C(s, t)$ is a certifier for problem X iff for every string s, $s \in X$ there exists a string t such that $C(s, t) =$ `yes`. Of course C will say `no` to all strings not in X.

t is called a "certificate" or "witness". It allows C to check the validity of the proof that $s \in X$.

(Think of t as a "hint" or a "clue".)

NP.

Decision problems for which there exists a polynomial-time certifier: $C(s, t)$ is a polynomial time ($p(|s|)$ algorithm.

Certifiers and Certificates: COMPOSITES

COMPOSITES. Given a positive integer s, is s composite?
i.e., s=t*q, t and q integers >1

Certificate. A nontrivial factor t (t>1, t<s) of s. Such a certificate exists iff s is composite. Moreover $|t| \leq |s|$.

Certifier.

```
boolean C(s, t) {
   if (t ≤ 1 or t ≥ s)
      return false
   else if (s is a multiple of t)
      return true
   else
      return false
}
```

Instance. s = 437,669.

Certificate. t = 541 or 809. ⟵ $437{,}669 = 541 \times 809$

Conclusion. COMPOSITES is in NP.

Why is NP called NP?

The act of searching for t can be viewed as a Non Deterministic Search over all possible certifiers.

The Non Deterministic search guesses all choices in a choice point of the search at the same time, thereby allowing us to find the right guess in one step.

This brings the search time in an exponential sized search space down to polynomial.

So NP stands for **Non Deterministic Polynomial**

Certifiers and Certificates: 3-Satisfiability

SAT. Given a Conjunctive Normal Form (ands of or-expressions) formula Φ, is there a satisfying assignment?
3_SAT: 3 variables in each or-expression

Certificate. An assignment of truth values to the n Boolean variables.

Certifier. Check that each clause in Φ has at least one true literal. Example: instance:

$$\left(\overline{x_1} \vee x_2 \vee x_3\right) \wedge \left(x_1 \vee \overline{x_2} \vee x_3\right) \wedge \left(x_1 \vee x_2 \vee x_4\right) \wedge \left(\overline{x_1} \vee \overline{x_3} \vee \overline{x_4}\right)$$

Certificate

$$x_1 = 1,\ x_2 = 1,\ x_3 = 0,\ x_4 = 1$$

Conclusion. SAT is in NP.

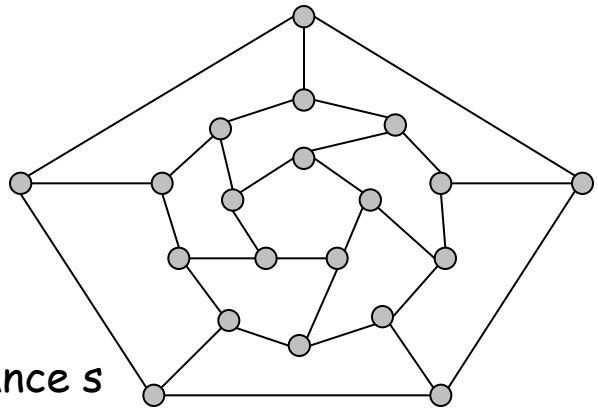
Certifiers and Certificates: Hamiltonian Cycle

HAM-CYCLE. Given an undirected graph G = (V, E), does there exist a simple cycle C that visits every node?

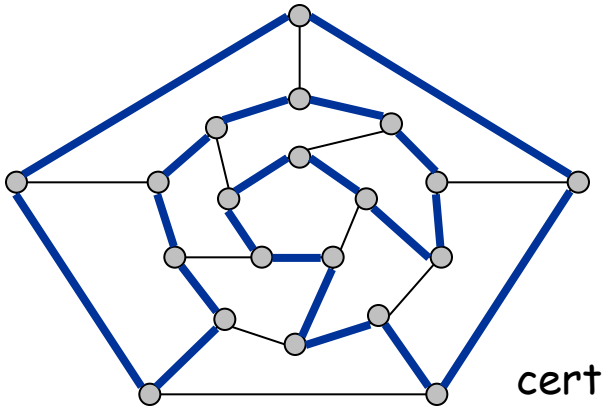
Certificate. A permutation of the n nodes.

Certifier. Check that the permutation contains each node in V exactly once, and that there is an edge between each pair of adjacent nodes in the permutation, including last and first nodes.

Conclusion. HAM-CYCLE is in NP.



instance s



certificate t

P, NP, EXP

P. Decision problems for which there is a polynomial-time algorithm.

NP. Decision problems for which there is a polynomial-time certifier.

EXP. Decision problems for which there is an exponential-time algorithm.

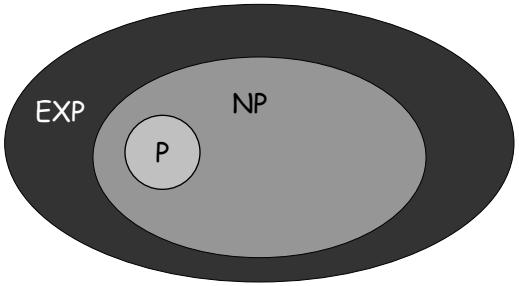
Claim. $P \subseteq NP$.
Proof. Consider any problem X in P; there exists a polynomial-time algorithm $A(s)$ that solves X. Certificate: $t = \varepsilon$, certifier $C(s, t) = A(s)$.

Claim. $NP \subseteq EXP$.
Proof. Consider any problem X in NP; there exists a poly-time certifier $C(s, t)$ for X. Run $C(s, t)$ on all strings t with $|t| \leq p(|s|)$. (There are exponentially many of these, but that is OK for class EXP.) Return `yes`, if $C(s, t)$ returns `yes` for any of these.

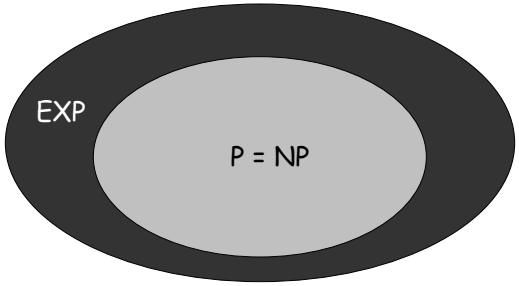
The One Million Dollar CS Question: P = NP?

P = NP?

- Is the decision problem as easy as the certification problem?
- Clay mathematics institute $1 million prize.



If P ≠ NP



If P = NP

would break RSA cryptography
(and potentially collapse economy)

If yes: Efficient algorithms for 3-COLOR, TSP, FACTOR, SAT, ...

If no: No efficient algorithms possible for 3-COLOR, TSP, SAT, ...

Consensus opinion on P = NP? Probably no.

NP-Completeness

NP-complete. A problem Y in NP with the property that for every problem X in NP, $X \leq_p Y$.

Theorem. Suppose Y is an NP-complete problem. Then Y is solvable in polynomial-time iff P = NP.

Proof.

$\Leftarrow$ If P = NP then Y can be solved in poly-time since Y is in NP.

$\Rightarrow$ Suppose Y can be solved in poly-time.

- Let X be any problem in NP. Since $X \leq_p Y$, we can solve X in poly-time. This implies NP $\subseteq$ P.
- We already know P $\subseteq$ NP. Thus P = NP. ▪

Theorem. CIRCUIT-SAT is NP-complete. [Cook 1971, Levin 1973]

Circuit SAT is NPC [Cook 1971, Levin 1973]

Take an arbitrary problem X in NP
Show it can be reduced in polynomial time to Circuit SAT

Intuitive Argument:

Any algorithm, that takes a fixed number of n input bits and produces an output bit (yes/no answer), can be represented by a circuit of **and-s**, **or-s**, and **not-s**, after all, that is how we can view any computer program in execution.

If the program takes a number of steps polynomial in n, then the circuit has polynomial size.

(The nasty details are in how an algorithm is translated into a circuit :)

3-SAT is NP-Complete

Theorem. 3-SAT is NP-complete.

Proof. Enough to show that CIRCUIT-SAT $\leq_P$ 3-SAT since 3-SAT is in NP.

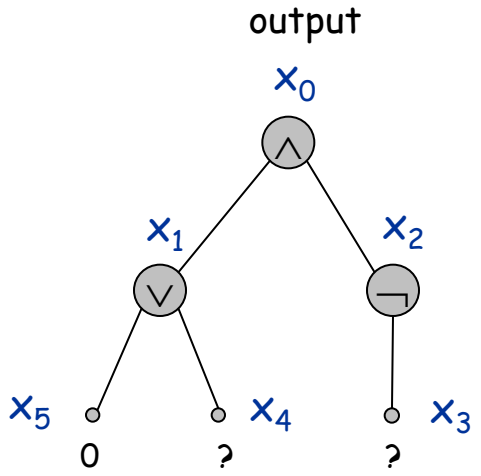
- (We already did this in lecture 07_Reductions.)
- Let K be any circuit.
- Create a 3-SAT variable x_i for each circuit element i.
- Make 3-SAT clauses compute values for each circuit-SAT node, eg.:

  $x_2 = \neg x_3$ $\Rightarrow$ add 2 clauses: $x_2 \vee x_3$, $\overline{x_2} \vee \overline{x_3}$

  $x_1 = x_4 \vee x_5$ $\Rightarrow$ add 3 clauses: $x_1 \vee \overline{x_4}$, $x_1 \vee \overline{x_5}$, $\overline{x_1} \vee x_4 \vee x_5$

  $x_0 = x_1 \wedge x_2$ $\Rightarrow$ add 3 clauses: $\overline{x_0} \vee x_1$, $\overline{x_0} \vee x_2$, $x_0 \vee \overline{x_1} \vee \overline{x_2}$

- Hard-coded input values and output value.
  - $x_5 = 0$ $\Rightarrow$ add 1 clause: $\overline{x_5}$
  - $x_0 = 1$ $\Rightarrow$ add 1 clause: x_0



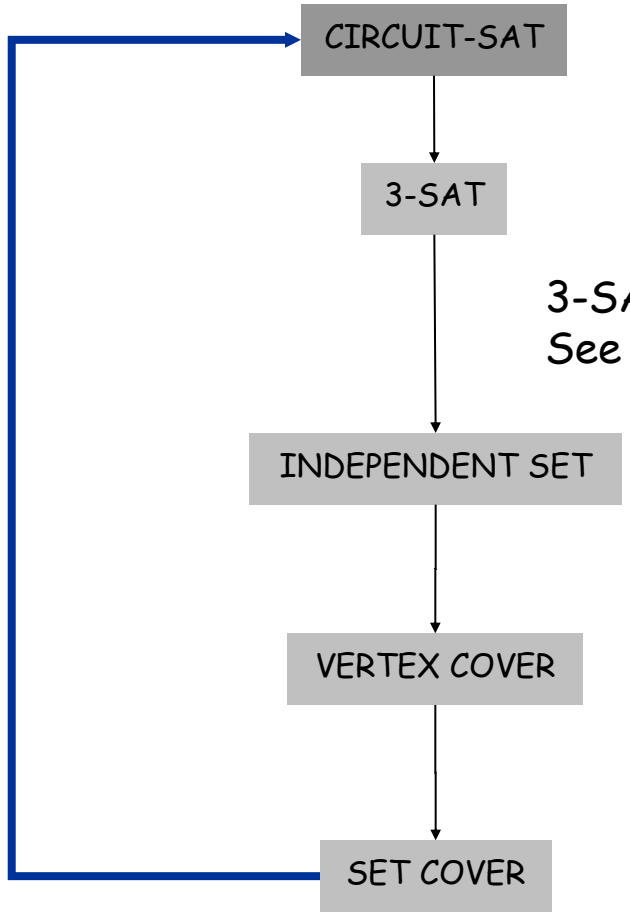
- Final step: turn clauses of length < 3 into clauses of length exactly 3. **HOW?**

NP Completeness

- Polynomial time reductions ($X \leq_P Y$)
- The class NP – problems with polynomial time certifiers
- NP complete problem – problem in NP such that every other NP problem has a reduction to it.
- Examples of NP-complete problems:
  Circuit-SAT, 3-SAT, Independent Set

NP-Completeness

Observation. All problems below are NP-complete and polynomially reduce to one another!



by NP-completeness of CIRCUIT-SAT

3-SAT reduces to INDEPENDENT SET
See lecture 07_rdc (Gadgets)

Extent and Impact of NP-Completeness

Extent of NP-completeness. [Papadimitriou 1995]

- Prime intellectual export of CS to other disciplines.
- 6,000 citations per year
- Broad applicability and classification power.
- "Captures vast domains of computational, scientific, mathematical endeavors, and seems to roughly delimit what mathematicians and scientists have been aspiring to compute feasibly."