

LECTURE 26

Monday March 13, 2017

based on notes by Denis Pankratov

Complexity Theory

$P = \{L \subseteq \{0, 1\}^* \mid L \text{ is solvable in } \textit{polytime}\}$

P – class of decision problems that are efficiently solvable

$NP = \{L \mid L \text{ is polytime verifiable}\}$

NP – class of decision problems, so that given a solution to the problem, the solution can be verified efficiently

Definition: L is polytime verifiable if $\exists c > 0$ and $\exists A$ – polytime algorithm of (x, y) so that:

$$(\forall x \in \{0, 1\}^*)(x \in L \Leftrightarrow \exists y \text{ (certificate)} : |y| \leq |x|^c \text{ and } A(x, y) = 1)$$

Claim: $P \subseteq NP$

Proof: $L \in P$. There exists polytime B so that:

$$(\forall x)(x \in L \Leftrightarrow B(x) = 1)$$

verifier A on (x, y) :
 return $B(x)$

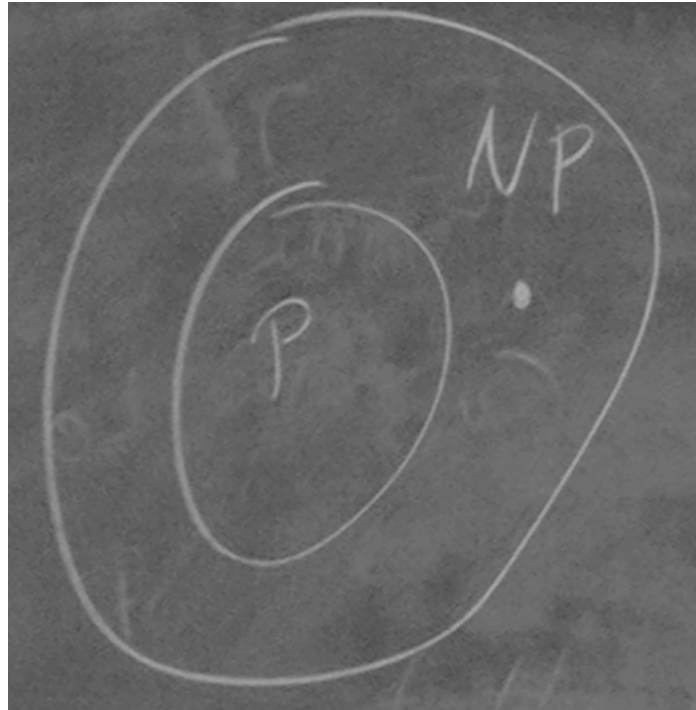
Example: Say $c = 1$

Since $A(x, y)$ ignores y :

$$\exists y : |y| \leq |x| \text{ and } A(x, y) = 1 \Leftrightarrow B(x) = 1 \Leftrightarrow x \in L$$

Question: $P = NP$

Many believe $P \neq NP$

Figure 1: Visualization of $P \neq NP$

Definition: A variable is **Boolean** if it takes values in $\{0, 1\}$

Definition: A literal is a variable or its negation. *Example:* $x_1, \neg x_1$

Definition: A *clause* is a disjunction of literals. *Example:* $(x_1 \cup \neg x_7 \cup x_3 \cup x_{15})$

Definition: A formula is in **CNF** (conjunctive normal form) if it is a conjunction of clauses. *Example:* $(x_1 \cup \neg x_7 \cup x_3) \cap (x_5 \cup \neg x_2 \cup x_1) \cap (x_3 \cup \neg x_8) \cap x_{15}$, with *clauses* 1 to 4 of different sizes

Definition: k -CNF is a formula in CNF form so that every clause *contains* exactly k literals

$$L_{k-SAT} = \{ \langle \varphi \rangle \mid \varphi \text{ is a } k\text{-CNF} \ \& \ \varphi \text{ is satisfiable} \}$$

(Note: $\langle \varphi \rangle$ is an encoding of φ over $\{0, 1\}$)

Exercise: $L_{2-SAT} \in P$

L_{3-SAT} is not known to be in P and it is widely believed not to be in P

Claim: $L_{3-SAT} \in NP$

Proof: Verifier A for L_{3-SAT} runs on inputs

$x = \langle \varphi \rangle$ where φ is 3-CNF

$y = \langle \tau \rangle$ where τ is an assignment to variables in τ

A : evaluate φ on τ , and if φ is satisfied by τ , return 1

$$\langle \varphi \rangle \in L_{3-SAT} \Leftrightarrow \exists \langle \tau \rangle \text{ so that } A(\langle \varphi \rangle, \langle \tau \rangle) = 1$$

Note: $|\langle \tau \rangle| \leq |\langle \varphi \rangle|$

$$(\forall x \in \{0, 1\}^*)(x \in L \Leftrightarrow \exists y : |y| \leq |x|, A(x, y) = 1)$$

Example 2:

$L_{SPATH} = \{ \langle G, w, s, t, k \rangle \mid \text{there exists a path from } s \text{ to } t \text{ in } G \text{ of weight } \leq k; G \text{ is a directed graph \& } w > 0 \text{ edge weights} \}$

Dijkstra's $\Rightarrow L_{SPATH} \in P$

$L_{LPATH} = \{ \langle G, w, s, t, k \rangle \mid \text{there exists a simple path from } s \text{ to } t \text{ in } G \text{ of weight } \geq k; G \text{— directed graph, } w > 0 \text{— edge weights} \}$

L_{LPATH} is not known to be in P & it is believed not to be in P .

$L_{LPATH} \in NP$

Example 3:

$L_{MST} = \{ \langle G, w, k \rangle \mid \text{there exists a spanning tree in } G \text{ of weight } \leq k \}$

Prim's, Kruskall's: $L_{MST} \in P$

Travelling Salesman Problem (TSP)

$L_{TSP} = \{ \langle G, w, k \rangle \mid \text{there exists a tour}^* \text{ of weight } \leq k \}$

* closed walk that visits each vertex exactly once

Exercise: $L_{TSP} \in NP$

L_{TSP} is not known to be in P & it is believed not to be in P .

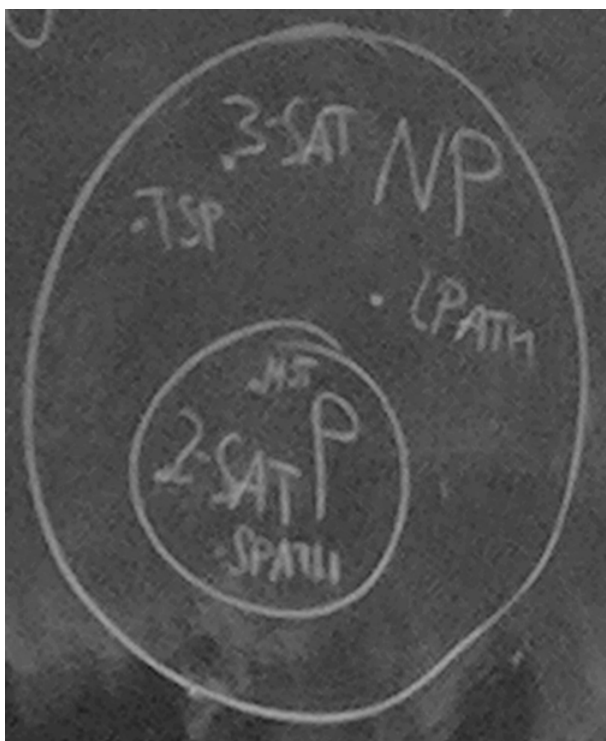
All of these are in some sense *equivalent*:

L_{3-SAT}

L_{LPATH}

L_{TSP}

Nobody has been able to solve these!

Figure 2: *Conjectured picture*

Definition: L_1 reduces to L_2 in polynomial time if there exists a polytime algorithm $A : \{0, 1\}^* \rightarrow \{0, 1\}^*$ so that:

$$(\forall x \in \{0, 1\}^*)(x \in L_1 \Leftrightarrow A(x) \in L_2)$$

Figure 3: *Reduction A*

We have done reductions before, but for single problems.

Claim: $L_1 \leq_p L_2$ & $L_2 \in P$, then $L_1 \in P$

Proof: Since $L_2 \in P$, $\exists B$ -polytime so that $(\forall x)(x \in L_1 \Leftrightarrow B(x) = 1)$

* \leq_p is reduces to, while the subscript p means it is a polynomial reduction

Since $L_1 \leq_p L_2$, $\exists A$ -polytime so that:

$$(\forall x)(x \in L_1 \Leftrightarrow A(x) \in L_2)$$

Let $C = B \cdot A$. C on input x : — $y = A(x)$
 — return $B(y)$

$$(\forall x)(x \in L_1 \Leftrightarrow A(x) \in L_2 \Leftrightarrow B(A(x)) = 1)$$

Definition: L is NP -hard if $(\forall \tilde{L} \in NP)(\tilde{L} \leq_p L)$

Definition: L is NP -complete:

1. $L \in NP$
2. L is NP -hard

(*Note:* class of NP -complete languages is NPC)

Claim: If $L \in NPC$ and $L \in P$ then $P = NP$

Proof: Let $\tilde{L} \in NP$

Since $L \in NPC$,

$\tilde{L} \subseteq L$ & $L \in P$

$\Rightarrow \tilde{L} \in P$ (by previous claim)