15-455: UNDERGRADUATE COMPLEXITY THEORY, SPRING 2017, RYAN O'DONNELL

A PREPRINT

Yang Li
Faculty of Information Technology
Monash University
Clayton, VIC, Australia
yang.kelvinli@monash.edu

March 26, 2019

1 Lecture 1 Overview

2 Lecture 2 Turing Machine

3 Lecture 3 Simulations and Turing Machine Variants

4 Lecture 4 Time Complexity and Universal Turing Machine

5 Lecture 5 Time Hierarchy Theorem

6 Lecture 6 Problems in P

7 Lecture 7 SAT

7.1 CIRCUIT-SAT

Definition 7.1. Boolean circuit C is a DAG, in which nodes are called gates, and edges are called wires... usually has n inputs and 1 output... gates are and $\wedge$, or $\vee$, negation $\neg$ with fan-in 2, 2, 1 respectively and arbitrary number of fan-out.

A circuit C computes a Boolean function which maps n bits strings into one bit $\{0,1\}^n \to \{0,1\}$.

You can think of circuit as a computational model.

One circuit can only operations on inputs of a fixed length, because number of sources in the DAG is fixed. This is one of the main reason we don't use it as the first choice of our computational model.

When the input to algorithm is a circuit $< C >$, we measure run time in terms of $n :=$ number of input gates and $m :=$ the number of total gates. These parameters are polynomially related to the input size, so if we only care if the algorithm can run in P time or not, whether the run time is measured by the size of the input or n, m doesn't matter. $n \leq m \leq | < C > | \leq O(m \log m)$.

Definition 7.2 (circuit-eval)**.** given $< C, x = \{0,1\}^n >$, what is $C(x)$? This can also be written into a language $\{< C, x >: C(x) = 1\}$.

This can be computed in P time. It's the same as checking an answer of a circuit. Roughly speaking, you can do it in $O(m)$.

Definition 7.3 (circuit-sat)**.** given $< C >$, is there an x s.t. $C(x) = 1$? That is, $\{< C >: \exists x s.t. C(x) = 1\}$.

Brute force, try all 2^n xs, find the one that gives the true answer. This is probably the best we can do. We think that it can be done in P iff $P = NP$.

7.2 FORMULA-SAT

Definition 7.4. A formula is a circuit where all the fan-outs are 1.

eg, $(x_1 \wedge x_2) \wedge (x_2 \vee x_3)) \vee (\neg(x_1 \wedge x_2))$. A circuit can be represented by a formula.

Formula-sat is a special case of a circuit-sat. Any formula is an example of a circuit. It's potentially easier, but we still don't have a smarter way of computing it.

7.3 CNF-SAT

Definition 7.5. Special case of formula-sat, where the formula is a conjunction of clauses, and a clause is a disjunction of literals, a literal is a variable or its negation.

For a cnf problem, n is the number of variables, and m is the number of clauses. It's a potential even easier problem. It is in EXP time $O(2^n m)$. It is NP-complete.

Definition 7.6. k-sat is a special case of cnf-sat, where all clauses have at most k literals.

it's potential easier than cnf.

7.4 3-sat

A popular example is 3-sat. WLOG, in 3-sat, $m \leq O(n^3)$, this is over counting a lit, but it's ok. We know it's NP-complete. It's in time 1.34^n.

4-sat is in 1.5^n, 5-sat in 1.6^n, 6-sat 1.667^n,..., k-sat in tim $(2 - 2/k)^n poly(n)$. 2-sat is in polynomail time.

Theorem 7.1. *2-sat is in P.*

e.g. input: $(x_1 \vee x_2) \wedge (x_1 \vee \bar{x_3}) \wedge (\bar{x_1} \vee x_2) \wedge (x_2 \vee x_3) \wedge (\bar{x_3})$

Proof. observation: $(x_1 \vee x_2) = (\bar{x_1} \to x_2) = (\bar{x_2} \to x_1)$ because of the truth table for disjunction $(x_1 \vee x_2)$ is the same as the truth table for the implication $(\bar{x_1} \to x_2)$. See `https://en.wikipedia.org/wiki/Truth_table`

rewrite the above expression using the $\to$ formulas, using both directions. ie., $(\bar{x_1} \to x_2) \wedge (\bar{x_2} \to x_1) \wedge \ldots$

draw a directed graph with $2n$ vertices, one for each literal and $2m$ directed edges, one for each implication.

key observation of the directed graph: if $l_i \to l_j$ is an edge, l is a literal, then there is also an edge $\bar{l_j} \to \bar{l_i}$.

There is $\bar{x_2} \to x_1 \to x_2$, i.e, there is a path $\bar{x_2} \to x_2$. In any satisfiable assignment, x_2 better be TRUE.

Suppose exists a path $x_2 \to \bar{x_2}$, then x_2 better be FALSE.

If fomr some i, there exists a path $x_i \to \bar{x_i}$ and $\bar{x_i} \to x_i$, then the instance is unsatisfiable. call them contradictory path.

In poly(n) time, can check whether exists contradictory paths.

It remains to show that if there is no such contradictory path, then it is satisfiable.

assume for al i, we don't have contradictory path. □

8 NP

3-sat is in P iff circuit-sat is in P iff 3-colorable is in P iff k-clique is in P iff every language in NP is in P.

Conjecture: $P \neq NP \iff 3 - sat \notin P$.

Conjecture Exponential time hypothesis (ETH): $\exists \delta > 0$ s.t. 3-sat $\notin Time((1 + \delta)^n)$.

Conjuecture Strong ETH: $\forall \delta > 0$, $\exists k$ s.t. k-sat $\notin TIME((2 - \delta)^n)$. This is almost the same as saying that CNF with no upperbound on k has no better algorithm than just 2^n.

The first conjecture is weaker than the 2nd that is weaker than the last. The last conjecture also implies that $\forall \epsilon > 0$, cannot solve the longest common subsequence problem in time $O(n^{2-\epsilon})$.

8.1 NP

idea: for many problems there exists brute force algorithms enumerate (exponentially many) candidate solutions and in p-time check if each one is a genuine solution.

Two features: candidate solutions are encodable by strings y with polynomial length $|y| \leq poly(|x|)$. That is, candidate solutions are not extremely long comparing with the input size; there exists an efficient algorithm to check if a candidate solution is a genuine solution.

Roughly: NP is the class of decision problems/languages with those two features.

Un 3-colorable problem is highly believed to not be in NP.

Definition 8.1. An algorithm (TM) V is a verifier for language L if

- V takes as input a pair $< x, y >$;
- $\forall x, x \in L \iff \exists y$ s.t. $V(< x, y >=$accept.

Definition 8.2. Verifier V is said to polynomial time if $V < x, y >$ runs in time $O(|x|^k)$ for some $k \in N$

Sublty: V's running time is measured in terms of $|x|$.

Definition 8.3. $NP = \{L \mid L$ has a P-time verifier$\}$.

Example: proving squares $= \{x \mid x =< B >, B \in N$ is a perfect square $\}$ is in NP. Proof is omitted by me...

3-colorable is in NP, proof is omitted... two main steps. Firstly, there exists a p-time verifier V. Secondly, if G is 3-col then V accepts. If V accepts then G is 3-col.

Roughly, P is all the decision problems where deciding if there is a genuine solution can be done efficiently. NP is the set of problems where checking if a candidate solution is a genuine solution can be done efficiently.

$P \subseteq NP \subseteq EXP$. Recall $P \neq EXP$ by THT, so $P \neq NP$ and/or $NP \neq EXP$. We can't prove $P \neq NP$ or $NP \neq EXP$, but we can prove $P \neq EXP$.

Proof. Let $L \in P$, wts $L \in NP$. Proof omitted... □

Proof. P = NP iff 3-sat is in P by Cook-Levin theorm... □