CS121 Lecture 18 October 31, 2019

What if P = NP?

What do we know?

Fastest algo for 3SAT: $> 2^{0.35n}$

Best lower bound on SAT: $< 10n$

Algorithmia: P = NP

Cryptomania: P ≠ NP

Search to Decision Reduction

Def. NP = class of decision problems that are efficiently verifiable

If P = NP, we can efficiently test if a solution exists and find the solution

Thm 1. Suppose P = NP. Then ∃ poly-time alg that given 3CNF φ, finds x satisfying

φ if such x exists. ← for every NPC problem

Given. Alg to decide if 3CNF φ is satisfiable

Goal. Alg to find given φ a satisfying assignment for φ.

$$\varphi : (x_0 \vee \bar{x}_7 \vee x_3) \wedge (\bar{x}_0 \vee \bar{x}_1 \vee x_7) \wedge (x_2 \vee x_5 \vee \bar{x}_{12}) \dots$$

replace $x_0 = 0$

$$\varphi_0 : (0 \vee \bar{x}_7 \vee x_3) \wedge (1 \vee \bar{x}_1 \vee x_7) \wedge (x_2 \vee x_5 \vee \bar{x}_{12}) \dots$$

$$= (\bar{x}_7 \vee x_3) \wedge (x_2 \vee x_5 \vee \bar{x}_{12}) \dots$$

check if φ_0 satisfiable. If so, $x_0 = 0$. Else, $x_0 = 1$ and check again. If satisfiable, continue with next x_i.

Polynomial Time?

$$T(n) = 2 \cdot \text{cost(oracle)} + T(n-1)$$

3SAT to everything

Theorem 2. Suppose that $P = NP$. Then $\exists$ poly-time algo that given G and number k, finds k-sized independent set in G if such set exists.

$$G, k \xrightarrow{\text{reduction}} \varphi$$

$$\text{IJET}(G, k) = 1 \iff \text{3SAT}(\varphi) = 1$$

In particular: if $\exists x$ satisfying φ then $\exists$ k-sized ind set S in G

Proof of Cook-Levin theorem:

$S \xleftarrow{\text{efficient algorithm}} X$ ← if $P = NP$

General Search to Decision Reduction

Thm. If $P = NP$ then for every poly-time V there is a poly-time FINDv that on input $x \in \{0,1\}^*$ and 1^m.

$$FINDv(x, 1^m) = \begin{cases} w, & \exists w \in \{0,1\}^m \text{ s.t. } V(x,w) = 1 \\ \text{"n"}, & \text{otherwise} \end{cases}$$

Optimization 1.

Q. Prove that if $P = NP$ there is a poly-time algo that on input G finds an independent set in G of maximum size

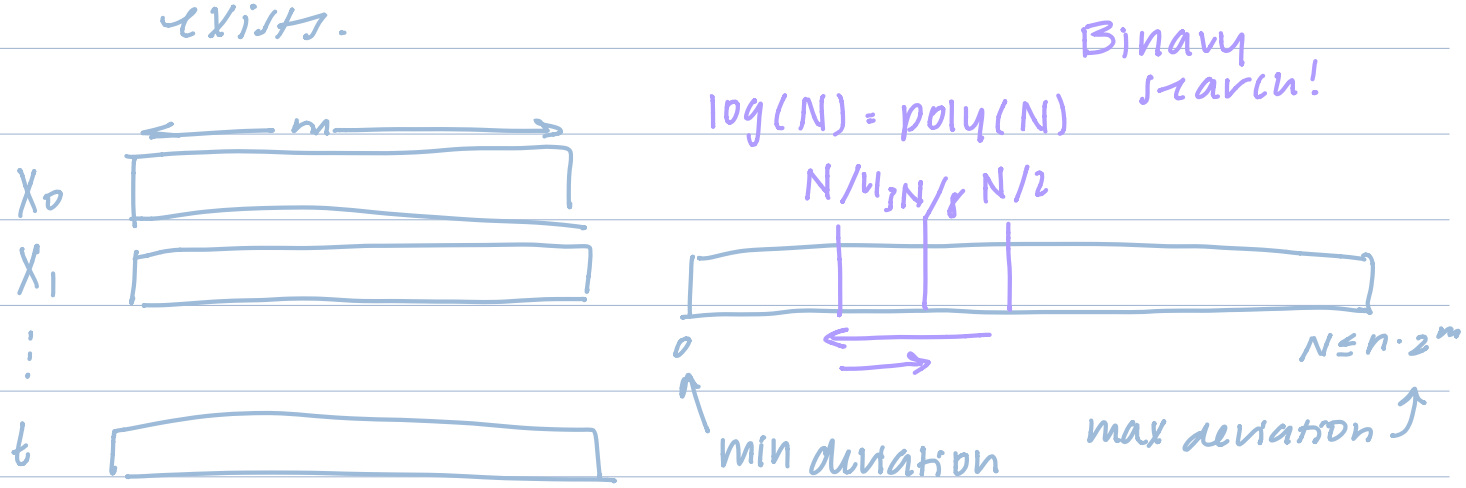
From SAT, poly-time algo to check if set exists. Simply run on $k = n$, $k = n-1$, $k = n-2$... until true.

Optimization 2.

Q. Prove that if $P = NP$ there is poly-time algo (in # of bits) that on input $x_0, x_1, \ldots, x_{n-1}, t$ finds set S minimizing $|\sum_{i \in S} x_i - t|$

From $\exists \mathbb{R}$, poly time algo to see if J exists.

Lemma: If $P = NP$, there is a poly-time algo that finds J s.t. $|\sum_{i \in J} x_i - t| \leq v$ if such J exists.



Optimization (General)

If $P = NP$ then for every poly-time computable $F: \{0,1\}^* \to \mathbb{N}$ there is a poly-time algo OPT_F such that on input $x \in \{0,1\}^*, 1^m$:

$$OPT_F(x, 1^m) = \arg\max_{y \in \{0,1\}^m} F(x,y)$$

Quantifier Elimination

If $P = NP$, can efficiently compute

$$3SAT(\varphi) = \begin{cases} 1 & \exists x \in \{0,1\}^m \text{ s.t. } \varphi(x) = 1 \\ 0 & \text{o/w} \end{cases}$$

what about $F(\varphi) = \begin{cases} 1 & \exists x \in \{0,1\}^{n/2} \forall y \in \{0,1\}^{n/2} \text{ s.t. } \varphi(x,y) = 1 \\ 0 & \text{o/w} \end{cases}$

Thm. if $P=NP$, $F \in P$.

Proof. Let S = algo solving $\overline{3SAT}$ in poly time.

For every φ, X, $\exists y \in \{0,1\}^{n/2}$ s.t. $\varphi(X,y)=0$ iff $S(\varphi(X\cdot))=1$

Define $V(\varphi, X)=1$ iff $S(\varphi(X\cdot))=0$

Since S is poly-time, V is poly-time:

if $P=NP$ can determine in poly-time if $\exists X$ s.t. $V(\varphi,X)=1$:

$$\exists X \text{ s.t. } V(\varphi,X)=1 \iff \exists X \neg(\exists y \neg \varphi(Xy))$$

Generalization (Polynomial Hierarchy collapse)

$$SOLVESAT(1^n, 1^m) = \begin{cases} 1 & \exists \text{ circuit of } \leq m \text{ gates solving 3SAT on length } n \text{ input} \\ X & X \geq 0 \end{cases}$$

if $P=NP$, then $SOLVESAT \in P$ and moreover can find minimal circuit

Can use slower algorithm for SAT to find optimal algorithm for SAT.

if $P=NP$, $P=NP=$ Polynomial Hierarchy $=$ PSpace

Counting and Sampling

If $P = NP$, can do the following in polynomial time:

Given $F: \{0,1\}^* \to \{0,1\}^*$ poly-time computable and y do:

- get $1 \pm \epsilon$ approx to # solutions to $F(x) = y$
- Sample (close to) random solution x to $F(x) = y$

Doing Science

Occam's Razor: Simplest explanation is correct one.