Michael A. Forbes
miforbes@illinois.edu
2023-08-31.1

7

cs579 Computational Complexity: Lecture 4 (2023-08-31)

logistics: - pset 1 due 09-07 [[have all you read, after today]]
[[not realistic]]

last lecture: non-deterministic TMs [[models guessing]] [[verifier view]]
[[longest branch]]

NTIME, NP [[in NP, open if in P]]

HAMPATH [[clever algo: primality]]

today: reductions

Q: convincing evidence HAMPATH is hard? [[lect 1]] [[lots of study, no known algo]] [[similar problems easy (reduction)]]

[[will mostly see]] [["hardest" problem in NP]]

thm: HAMPATH $\in$ P iff P = NP [[captures NP: nondeterminism + ability to solve benchmark]]
[[amazing! infinite # problems in NP]]

prides: use [reductions]

↳ reduce solving problem A to solving problem B $\Rightarrow$ "$A \le B$"
[[view solving B as a resource, ask to solve A]]
[[eg. using programming library]]

[[formalize]]

def: TM with output is [[deterministic]] TM w/ distinguished states - q_{start} - q_{halt} [[no acc/rej]] [[only change]]

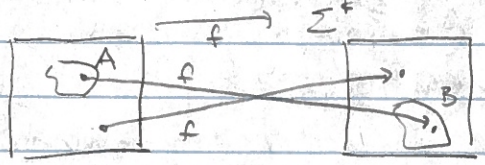
It computes by - initialization [[as before]]
- iterating transition function [[as before]]
- if ever reached q_{halt} - halt
- output tape content
↳ longest prefix of tape in Σ^* [[finite]]
[[don't output infinite # of $\sqcup$]]

the time of M on input x is # of transitions until halting.

A function $f: \Sigma^* \to \Sigma^*$ is [fully] polynomial-time computable, if exists TM M s.t. $\forall n\ \forall x \in \Sigma^n$, M on x outputs $f(x)$ in $\le t(n)$ steps.

def: $A \subseteq \Sigma^*$ is polytime mapping (/many-one) reducible to $B \subseteq \Sigma^*$, denoted $A \le_p B$ [[p for poly]], if exists polytime computable $f: \Sigma^* \to \Sigma^*$ s.t.: $\forall x \in \Sigma^*$, $x \in A$ iff $f(x) \in B$

eg:



rmk: - f not necessarily injective, nor surjective
- other notions of reduction exist, this notion is right vs [[for now]] [[seeking multiple "B questions"]]

prop: $A \le_p B$, $B \in P \Rightarrow A \in P$

pf: [[Q]]

polytime reduction $f: \Sigma^* \to \Sigma^*$
TM M for B

Michael A. Forbes
Miforbes@illinois.edu
2023-08-31.2

NTM

algo $N =$ "on input x:
poly$|x|$ - compute $f(x)$
poly$|f(x)|$ - run M on $f(x)$, accept if M does"

correctness: $x \in A \Rightarrow f(x) \in B \Rightarrow M$ acc $f(x) \Rightarrow N$ acc x
$x \notin A \Rightarrow f(x) \notin B \Rightarrow M$ rej $f(x) \Rightarrow N$ rej x

complexity: $|f(x)| \leq$ runtime of TM computing $f \leq$ poly$(|x|)$
$\Rightarrow$ poly$(|f(x)|) \leq$ poly(poly$(|x|)$) $\leq$ poly$(|x|)$ [[solve]] ∎

cor vs prop

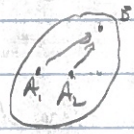
prop: $A \leq_p B$ (f), $B \leq_p C$ (g) $\Rightarrow A \leq_p C$
sketch: [[same idea]]
$x \mapsto f(x) \mapsto g(f(x))$
length $\leq$ poly$(|f(x)|) \leq$ poly$(|x|)$ □

cor: [[contrapositive]] $A \leq_p B$, $A \notin P$ [[hard]] $\Rightarrow B \notin P$ [[evidence of hardness]]

necessity?

prop: $A \leq_p B$, $B \in NP \Rightarrow A \in NP$
sketch: [[algo as before]] $x \in A \Leftrightarrow f(x) \in B \Rightarrow$ same of M acc $f(x) \Rightarrow$ same branch of N acc $\Rightarrow N$ acc
$x \notin A \Leftrightarrow f(x) \notin B \Rightarrow$ no $\Rightarrow$ no $\Rightarrow N$ rej

def: language B is NP-complete if
- $B \in NP$ [[remember this for proofs!]]
- all $A \in NP$, $A \leq_p B$ ← NP-hard



P vs NP

cor: B NP-complete, $B \in P$ iff $P = NP$ [[evidence of hardness]]
pf: $\Leftarrow$: $B \in NP \Rightarrow$ [[def]] $\Rightarrow B \in P$ [[assume]] [[apply]]
$\Rightarrow$: $A \in NP \Rightarrow A \leq_p B \in P \Rightarrow A \in P$ [[trivial]] ∎

cor: B NP-complete $C \in NP$ $B \leq_p C \Rightarrow C$ NP-complete
pf: - $C \in NP$ [[important!]]
- any $A \in NP$ $A \leq_p B \leq_p C \Rightarrow A \leq_p C$ [[B NP-C]] [[assume]] [[trivial]] ∎

cor [[Cook]]

Q: do NP-complete problems exist?

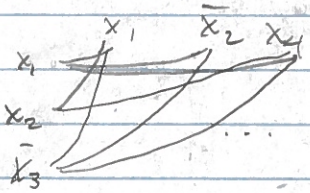
def: a boolean formula φ is an expression involving boolean variables [[over 0,1]] [[false, true]] and AND $\wedge$, OR $\vee$, NOT $\neg$ operations
φ is satisfiable if there is a $\{0,1\}$ assignment to the variables [[T,F]] that makes φ evaluate to 1 [[true]]
SAT $= \{ \langle \varphi \rangle : \varphi$ is a satisfiable boolean formula $\}$ [[encoding]]

eg: $\varphi = (\overline{x_1} \wedge y_0) \vee \overline{(x_0 \wedge \overline{z_0})} = 1$
assign $x = 0$, $y = 0$, $z = 1$

Michael A. Forbes
miforbes@illinois.edu
2023-08-31.3

prop: SAT ∈ NP
sketch: guess assignment to x (nondet), check $\varphi(x) = 1$ (efficient)
def: boolean formula φ is in conjunctive normal form (CNF) if
$\varphi = C_1 \wedge \cdots \wedge C_m$ (C_1, ..., C_m clauses) where each C_i is a clause $C = (\ell_1 \vee \cdots \vee \ell_k)$
(ℓ_i literals x_i or $\bar{x}_i$)
φ is a k-CNF if it is CNF w/ each clause having $\leq k$ literals
3SAT = $\{ \langle \varphi \rangle : \varphi$ is satisfiable boolean formula in 3CNF$\}$ (easily checkable)
lem: 3SAT ∈ NP [as before] (indep. during Cold War)
thm [Cook'71, Levin'73] There exist NP-complete problems, in particular 3SAT is NP-complete. [& hence SAT is NP-complete]
= [Q]

prop: CLIQUE = $\{ \langle G, k \rangle : G$ undirected graph w/ clique of size $\geq k\}$ [community detection]
is NP-complete (clique: $S \subseteq V$ w/ $(i,j) \in E$ $\forall i \neq j \in S$)
pf: ∈ NP: clear [guess S, verify edges]
NP-hard: show 3SAT ≤_p CLIQUE [suffices]
eg: $\varphi = (x_1 \vee x_2 \vee \bar{x}_3) \wedge (x_1 \vee \bar{x}_2 \vee \bar{x}_4) \wedge \cdots \wedge (\cdots)$
$V =$ x_1 x_2 $\bar{x}_3$ x_1 $\bar{x}_2$ $\bar{x}_4$ ... [duplicated x_1]
formally, k = #clauses of φ
$V = \{$ literals appearing in clauses, w/ repetitions $\}$
$\Rightarrow |V| \leq 3k$ [3CNF]
E = all edges except
- self loops
- between literals in same clause
- between literals that are negations of each other

eg:


correctness: clear [polynomial in output]
correctness: clm: $\langle \varphi \rangle \in$ 3SAT iff $\langle G, k \rangle \in$ CLIQUE [reduction]
pf: $\Rightarrow$: choose $\alpha \in \{0,1\}^{n = \#\text{vars}}$ s.t. $\varphi(\alpha) = 1$ [SAT]
$\Rightarrow$ clause $C = (\ell \vee \ell' \vee \ell'')$ in $\varphi = C_1 \wedge \cdots \wedge C_k$, ≥ 1 literal is true
define $S = \{$ set of literals made true by $\alpha\}$
$\rightarrow |S \cap C_i| \geq 1$ $\forall i$.

Michael A. Forbes
Miforbes@illinois.edu
2023-08-31.4

pick $T \subseteq S$ w $|T \cap C_i| = 1$, $\forall i$ arbitrarily

clm: $T \in V$ is a k-clique [k = # clauses]

pf: - $|T| = k$ by construction
- $\ell \neq \ell' \in T$, (ℓ, ℓ') not
  - self loop
  - in same clause [def T] $\Rightarrow (\ell, \ell') \in E$
  - negations of each other [α satisfies ℓ, ℓ'] $\Rightarrow$ [$\Rightarrow$ clique]

$\Leftarrow$ - $\langle G, k \rangle \in$ CLIQUE, w $S \in V$ k-clique

want α s.t. $\varphi(\alpha) = 1$

have $|S \cap C_i| \leq 1$ $\forall i$ [no edges between literals in each clause]

$\Rightarrow |S| \leq k$ [k clauses]

$\overset{|S| = k}{\Rightarrow}$ $|S \cap C_i| = 1$ $\forall i$ [pick exactly 1 literal per clause]

define $\alpha(x_i) = \begin{cases} 1 & x_i \in S \\ 0 & \bar{x}_i \in S \\ 0 & x_i, \bar{x}_i \notin S \end{cases}$

clm: $\varphi(\alpha) = 1$

pf: α well defined: cannot have $x_i, \bar{x}_i \in S$ as no such edges in G

α makes all literals in S true $\Rightarrow$ α makes each clause true

$\Rightarrow \varphi(\alpha) = 1$ ∎

fact: 3SAT $\leq_p$ HAMPATH [textbook] [$\Rightarrow$ HAMPATH NP-complete]

today: reduction [$A \leq B$, use B to solve A] [elementary p-cp] [hardest prob in NP] [def via reduction]

NP-complete [def via reduction]

SAT, 3SAT [Cook-Levin: 3SAT NP-complete] [$\Rightarrow$ CLIQUE NP-complete]

3SAT $\leq_p$ CLIQUE

next lecture: Cook-Levin theorem

logistics: pset 1 due 09-07