Computational Complexity

30/7/24

Ref : course webpage

Grading : Scribe (5%)
2 or 3 Assignments (20%)
Presentations (20%)
Midsem (20%)
Endsem (35%)

Pre-req : Basic Algorithm Design,
* Basic Linear Algebra,
Basic Graph Theory,
NP, NP-C,
Turing Machines

Can we show that 3-coloring problem requires atleast $\Omega(2^n)$? (1 million dollars :))

Naive : Try all colors $\rightarrow 3^n$ time

Resources : Memory (Working Memory)

Given a graph n vertices, 2 vertices u, v whether $u \sim v$

Input is in separate memory
[RAM too small for graph size]

BFS : $\Omega(n)$ memory (maintain a queue of vertices)
↳ too much if graph is big

n vertices $\Rightarrow$ $\log n$ bits to store index of vertex.

2005 Reingold det-algo. [Ambitious goal : $O(\log n)$ bits of memory randomization
using random walk $O(n^3)$ time with high probability
$\rightarrow 1$ with $n \uparrow$

Resource : Randomness

Expensive resource : seed (systime / sound) $\xrightarrow{f}$ f : complicated fn, looks random.

Not predictable $\Rightarrow$ random [view of complexity theory]

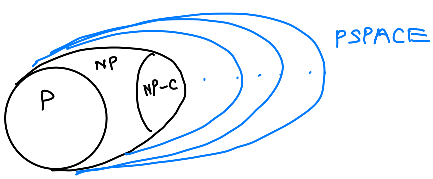
Resource : Communication

Computation b/w many parties [distributed computing].

Want to limit comm. (local computation is cheap)

Course : Connections between different concepts of complexity

Connect problems that are hard, hardness assumptions.



Proving pseudo randomness $\equiv$ Proving ckt lower bounds

Hard to compute $f^n \longrightarrow$ generate pseudo-random
↓
Make algos deterministic.

Does use of randomness give you more power ? [Open for 30 years]

E.g. Using random $O(\log n)$ bits memory w.h.p.
without random show $> \log n$

} will show ~~randomized~~ algo more powerful than deterministic .

Belief : Equal power

Interactive Proofs , Zero-Knowledge , Prob. checkable Pf

Graph : Is there a path from u to v of length ≤ 100

Yes → give path

No → give BFS pf, run algo and show

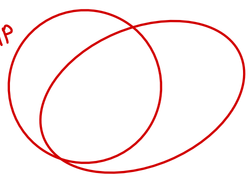
Graph : Is there a 3-coloring

Yes → give coloring scheme (easy proof)

No → Maybe some way ? (e.g. existence of 4-clique)

If a logical statement is true , then there is always a small proof ? (Open)

NP



Co-NP

(belief , Co-NP $\neq$ NP)

Allowing interaction makes it possible ! (Q = f (Ans) : Proof w.h.p.)

↓ Poly # rounds , prover unlimited power .

3-coloring has unlimited power . (90s)

Blockchains : Certain nodes compute, convince other parties of that computation, so interaction is useful here .

Probabilistic checkable proofs

← useful for blockchains : verifier reads small part of pf.

Give a proof, see 3-bits and that will convince w.h.p. (reading 3 bits of pf is enough)

Probabilistic Checkable Proofs $\Longleftrightarrow$ Hardness of Approximation

Zero-Knowledge Proofs

[Goldwasser]

Want to convince I know, without giving away information

Basic Lower Bound on Sorting

<u>Claim</u> : Sorting n numbers requires $\Omega(n \log n)$ comparisons

<u>Adversarial argument</u> :

Initially $\boxed{n!} = \boxed{A_i > A_j}\ \boxed{A_i < A_j}$ [First query $A_i > A_j$]

Adversary picks answer with bigger set from the two. (atleast half of initial size)

$n! \xrightarrow{1^{st}\text{ query}} n!/2 \xrightarrow{2^{nd}} n!/4$ } $\log_2(n!)$ queries needed to obtain permutation

$\Rightarrow$ needed for sorting

$= \Omega(n \log n)$

Above is an information theoretic lower bound $\Rightarrow$ Need x queries to obtain enough info

- Complexity Lower Bound : Given all info, how much computation needed to get answer
- Can you write a program for any given problem ?
  - Lack of understanding information
  - Lack of computational power

<u>Puzzle</u> : n numbers, exactly two of them are equal

Queries $A_i, A_j \longrightarrow \{<, >, =\}$

<u>Goal</u> : Find the pair that is equal

Computational Task :

Input $\in \{0,1\}^*$

Output $\in \{0,1\}^*$

1. Search Problem : $R \subseteq \{0,1\}^* \times \{0,1\}^*$

(OR) $f : \{0,1\}^* \longrightarrow (2^{\{0,1\}^*} - \phi)$

2. Decision Problem : $f : \{0,1\}^* \longrightarrow \{0,1\}$

For every search problem there is a natural decision problem s.t. solving the latter solves the former and vice versa [e.g. in poly time]

1. SAT : ϕ , output : a satisfying assignment ,

decision : given ϕ, is there a satisfying assignment ?

search $\xrightarrow[\text{to}]{\text{reduces}}$ decision

(self reduction)

2. Input : integer n (input size $= \log n$)

Output : prime factors of n

Decision : is there a factor of $n < k$? $\rightarrow$ Binary search .

* $f : \{0,1\}^* \longrightarrow \{0,1\}$

Counting argument,

$\#$ f^n = uncountable no bijection ($\exists$ f^n not computable by programs)

Program = $\{0,1\}^*$ is countable

Diagonalisation

Let G have a program.

Well defined f^n : 1. G : input natural no. i

output = $\begin{cases} 1 , \text{ if } i^{th} \text{ program on input } i \text{ halts} \\ 0, \text{ otherwise} \end{cases}$

Consider program P $\rightarrow$ input $i \in \mathbb{N}$

run program G on input i

If output is 1 $\rightarrow$ loop

0 $\rightarrow$ return 0

Let j = index of program P

If $G(j) = 1$: Program P halts on input j but P loops } contradiction

$= 0$: Program P doesn't halt on j but P returns 0 } contradiction.

$\Rightarrow$ G doesn't have a program.

H : input i, x

output 1 if i^{th} program halts on input x

0 otherwise

Halting problem

<u>HW Problem</u> : Given two C++ programs, do they have same behaviour?

[Show undecidable]

Hilbert's 10th problem

Input : Poly multivariable eq w/ integer coeff
Output : Is there an integer sol^n

} undecidable

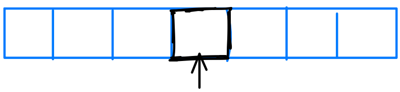
- Any C++ program can be converted into a polynomial eq^n s.t. program on this input halts iff equation has integer solution

→ Similar approach for undecidability of game of life

Turing Machine

$Q \leftarrow$ states , $\Sigma = \{0, 1\}$

- Infinite Tape , head



Transition function

$$\delta : Q \times \Sigma \longrightarrow Q \times \underset{\substack{\uparrow \\ \text{write} \\ \text{symbol}}}{\Sigma} \times \underset{\substack{\uparrow \\ \text{movement of} \\ \text{head}}}{\{\text{left}, \text{right}, \text{stay}\}}$$

Church - Turing Thesis : Every function computable by "natural", "reasonable" model of computation can be computed by a turing machine.

Abstract RAM Machine

		(unbounded)
Infinite memory cells	←	int
(finite) registers	←	int
program counter	←	int

Instructions

reset (r) (makes it zero)
dec (r)
inc (r)
load (r_1, r_2)
store (r_1, r_2)
cond goto (r, ℓ)

Universal Turing Machine (2 state, 3 symbol) ← can simulate the TM
Which takes description of any TM with input and it can simulate it

Time Complexity

$A \leftarrow$ TM which always halts

$$t_A : \{0,1\}^* \to \mathbb{N}$$

$$t_A(n) = \max_{x \in \{0,1\}^n} t_A(x) \leftarrow \text{Time complexity}$$

Efficient Computation

Cobham 1964
Edmonds 1965

DTIME$(t(n))$ = class of problems with TM running time atmost $t(n)$ $\leftarrow$ deterministic time

$$P = \bigcup_{c \geq 1} \text{DTIME}(n^c)$$

Cobham - Edmonds Thesis : Any function computable by any reasonable model can be by TM with polynomial time overhead.

Palindrome : 2 tape $\leftarrow O(n)$

1 tape $\leftarrow O(n^2)$

needs $\Omega(n^2)$ time.

reason why polytime is a universal measure.

a b c b a

Multitape TM $t(n)$
$\downarrow$
Single tape $t(n)^2$

(try!)

Eg of problems not in P

Input : description of TM and an input x for it

Output : whether it stops in $2^{|x|}$ time

Obvious : Simulate program $2^{|x|}$ time algo.

Can show : No faster algorithm $\leftarrow$ diagonalization argument

Input : A boolean ckt with 2ℓ variables.

(Defines a graph on 2^ℓ vertices)

Output : whether $s \sim t$ in this graph

Trivial algo : $2^\ell = O($? size of formula$)$

Exp-time needed in this.

NP

Decision Problem Q

Search Problem P

The two are equivalent if P has a polytime algo iff Q has a polytime algo.

Def : $L \in NP$ if $\exists$ TM M which runs in polytime and $\exists$ polynomial q s.t.

$\forall x \in L \quad \exists c \in \{0,1\}^*, \; |c| \le q(n)$ s.t. $M(x,c) = 1$

$\forall x \notin L \quad \forall c \in \{0,1\}^*, \; M(x,c) = 0$

E.g. (Independent Set)

Input : Graph G, number k

Problem : G has an independent set of size k ?

Certificate : set of vertices of size k which forms an independent set

$M \rightarrow$ verifies if c is valid indep. set in G with size k

E.g. $\#SAT = \{ \underbrace{<\phi, k>}_{|\phi| + \log k} :$ no. of satisfying assignments of $\phi \geq k \}$

$\Uparrow$ Not sure if in NP

certificate size $\leq k|\phi|$

E.g. MCSP (min. circuit size problem) $= \{ <\phi, k> :$ there is a equivalent boolean circuit ϕ with size atmost $k \}$

Not sure if in NP, since verifying if ϕ equivalent ϕ' is not known to be in P

$\overline{\text{Indset}} = \{ <G, k> :$ Graph G doesn't have an ind. set of size $k \}$

Not sure if in NP

$L \in P \iff \overline{L} \in P$

$L \in NP \overset{?}{\iff} \overline{L} \in NP$

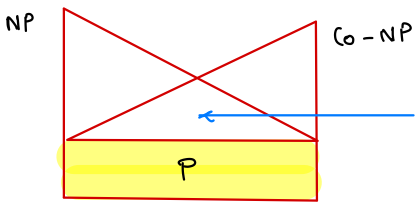
$P \subseteq NP$, Certificate : ε, TM runs the algo itself

Def (Co-NP) : We say $L \in$ co-NP if $\overline{L} \in NP$

PRIMES $\in P$: $\{ <n> : n$ is prime $\}$

Easy to see : PRIMES $\in$ CO-NP $\longrightarrow$ certificate . a, b, $n = ab$

Known to be in P before 2002



$P \subseteq NP \cap$ co-NP

Not known if $P = NP \cap$ co-NP

But if $L \in NP \cap$ co-NP then strong indication that $L \in P$

$GI = \{ <G, H> : G$ & H are isomorphic $\}$

$\overline{GI} \Rightarrow$ verification using randomisation

$GI \in$ Quasi P $(n^{\log^6 n})$

System of linear equations : Solvable / Not Solvable

SLE $\in$ NP

Solution size is poly (input)

SLE $\in$ co-NP [Give linear combination which adds upto 0]

Pf (Primes $\in$ NP) :

A number p is prime iff there is a number z s.t.

$z^{p-1} \equiv 1 \bmod p$

and for any $r < p-1$, $z^r \not\equiv 1 \bmod p$

NP

A $L \subseteq \{0,1\}^*$ is said to be NP if $\exists$ a polynomial time TM V and a polynomial fn $p : \mathbb{N} \to \mathbb{N}$ s.t.

if $x \in L$ then $\exists\, c \in \{0,1\}^{\leq p(x)}$ s.t. $V(x, c) = 1$

if $x \notin L$ then $\forall\, c \in \{0,1\}^{\leq p(x)}$ s.t. $V(x, c) = 0$

Non-deterministic TM

Two transition functions δ_0, δ_1

$\delta_0, \delta_1 : Q \times \Sigma \to Q \times \Sigma \times \{L, R, S\}$

(3 Non-det transition $\to$ 2 w/ poly running time blowup)

q_{accept}, q_{reject} $\leftarrow$ Halting states

A Language L is said to be accepted by a NTM T

if $x \in L$ iff $\exists$ computational path in TM halting in q_{accept}

A Language $L \subseteq \{0,1\}^*$ is in NP if $\exists$ polytime (all paths halt in polytime) NTM accepts L.

(accepting paths halt in P)

equivalent

(don't accept if runs for more than $p(x)$)

Claim : Two definitions are equivalent

Pf : Verifier $\Rightarrow$ NTM

1. Pick c non-deterministically
2. Run $M(x, c)$

NTM $\Rightarrow$ verifier

1. Run NTM, produce yes path as verifier TM certificate.

PRIMES $\in$ NP

A number p is prime iff there is a number z s.t.

$$z^{p-1} \equiv 1 \mod p$$

and for any $r < p-1$, $z^r \not\equiv 1 \mod p$

Pf:

If q is a prime, $\forall z$, $z^{q-1} \equiv 1 \mod q$, $q \nmid z$

$z, z^2, \ldots, z^{q-1}$ } remainder repeats at some point.

Obs: $z \times \{0, 1, \ldots, q-1\} = \{0, 1, \ldots, q-1\}$

if $z \times a_1 = z \times a_2 \Rightarrow z \times (a_1 - a_2) = 0 \mod q$

$\Rightarrow q \mid a_1 - a_2$. (contr.)

$z \times 0 = 0$. So

$(z \times 1, z \times 2, \ldots, z \times q-1) = (1, \ldots, q-1)$

$z^{q-1} \times 1 \times \cdots \times q-1 = 1 \times \cdots \times q-1$.

$z^{q-1} = 1$.

If p is not a prime, no such certificate. (trick: chinese remaindering)

$(z^{3-1})^{5-1} \equiv 1 \mod 3 \Rightarrow z^8 \equiv 1 \mod 3,\ 1 \mod 5$

$(z^{5-1})^{3-1} \equiv 1 \mod 5 \quad \Rightarrow z^8 \equiv 1 \mod 15$

$3 \mid z^8 - 1$
$5 \mid z^8 - 1$ $\Rightarrow 15 \mid z^8 - 1$

$\text{Order}_p(z) \equiv$ min power of z which is $1 \mod p$

Need to show $\exists z$ $\text{Order}_p(z) = p-1$.

HW: $\left.\begin{array}{l}\text{Order}_p(z_1) = r_1 \\ \text{Order}_p(z_2) = r_2 \\ \gcd(r_1, r_2) = 1\end{array}\right\} \Rightarrow \text{Order}_p(z_1 z_2) = r_1 r_2$

Among all elements if maximum order is r then $\forall z$, $\underbrace{z^r = 1}$.

for a field. q elements

$z^r - 1 = 0$. degree r poly. atmost r roots.

$\Rightarrow r = q-1$.

Verifying A number p is prime iff there is a number z s.t. is not simple.

$z^{p-1} \equiv 1 \bmod p$

and for any $r < p-1$, $z^r \not\equiv 1 \bmod p$

Obs : p is prime, $\text{order}_p(z) = r \Rightarrow r \mid p-1$.

Just need to check $z^r \equiv 1 \bmod p$ for maximal factors of $p-1$.

Take prime factorisation of $p-1$ in certificate.

e.g. $p = 37$ then $p-1 = 36$

2, 3 prime factors.

Check 18, 12 (all other divisors divide $\frac{z}{p_1}, \ldots, \frac{z}{p_k}$)

Prime factors of $p-1$ need certificate } Recursive scheme

HW : Overall certificate size is $O(\log p^c)$

Karp Reduction ($L' \leq_p L$)

We say L' reduces to L if $\exists$ polytime computable $f : \{0,1\}^k \to \{0,1\}^k$
s.t. $x \in L'$ iff $f(x) \in L$. (algo for $L \Rightarrow$ algo for L')

Cook Levin (NP-completeness)

L is called NP-complete if $L \in NP$ and $\forall L' \in NP$, $L' \leq_p L$

Turing Reduction (Many-one reduction)

We say L' turing-reduces to L if $\exists$ polynomial time TM for L' which uses an oracle machine for L.

NP-hard

L is called NP-hard $\forall L' \in NP$, $L' \leq_p L$

Thm : (Cook-Levin) SAT is NP-complete

(Karp 1972 — 21 Problems)

NP-compl. { SAT ↙ ⇐ reduction, transitive relation

Pf: ① SAT is in NP ⟶ SAT Assignment is certificate

② $(M, x, p, T) \rightarrow$ CNF formula $\phi_{M,x}$

s.t. p ↑ poly. T ↑ time

$\exists\, u \in \{0,1\}^{p(|x|)}$ s.t. $M(x,u) = 1$ iff $\phi_{M,x}$ is satisfiable

$\phi_{M,x}$ ↓ poly in $(|M|, |x|, c, 0)$

Tape :

Tape at $t = t_{i,*}$

$$\begin{array}{cccc|cccc} t_{00}, & \cdots & \cdot & , t_{0T} & q_{01}, & \cdots & \cdot & q_{0K} \\ \vdots & & & & \vdots & & & \vdots \\ t_{T0}, & t_{T1}, & \cdots, & t_{TT} & q_{T1}, & & \cdots & q_{TK} \end{array}$$

State

Head position

$$\begin{array}{ccc} h_{00}, & \cdots, & h_{0T} \\ \vdots & & \\ h_{T0} & \cdots & h_{TT} \end{array}$$

- At every i, only one of $q_{i,*}$ must be true
- " ______ " $h_{i,*}$ must be true

$h_{ij} \wedge t_{ij} \wedge q_{il} \Rightarrow h_{i+1,j+1}, q_{i+1,l}, \neg t_{i+1,j} \cdots$ } for all transitions time stamp ⇓ still polynomial.

Co-NP complete : Given a CNF formula, is it a tautology?

Proof of complexity :

PHP $x_{i,j} \leftarrow i^{th}$ pigeon is in j^{th} hole

n holes, n+1 pigeons

Resolution $\leftarrow$ exponential size.

Examples $\in$ Co-NP $\cap$ NP but not known in P

- Integer factoring
- Discrete log. given a, b, p find r s.t. $a^r \equiv b \mod p$

$\exists\, r \in [m, n]$ s.t. $a^r \equiv b \mod p$

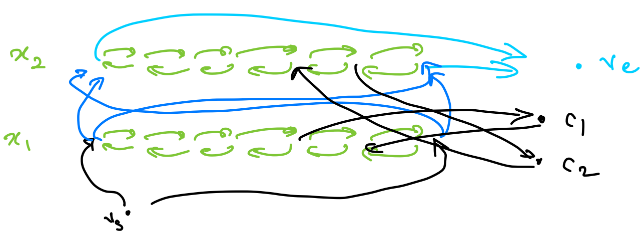
HAM. Given a directed graph, is there a path from v_1 to v_n which covers all vertices

SAT $\leq_p$ HAM

$\phi \longrightarrow G_p$ (sat iff HAM path)

- 1 vertex for every clause
- 1 chain of vertices for every variable

$c_i \rightarrow a_1 \vee a_2 \vee \neg a_3$



20/8

SAT $\leq_p$ 3-SAT (3-CNF), SAT $\leq$ HAM

Q : Longest Path : Given a directed graph and s,t. Find the longest path from s to t.

HAM $\leq$ Longest Path $\Rightarrow$ Longest path is NP-hard

HW : HAM path $=_p$ HAM cycle

HW : Dir. longest path $\leq$ Undirected longest path

$\Rightarrow$ Undirected longest path is NP-hard.

Given a directed graph and s,t is there a path of even length from s to t? $\leftarrow$ Show NP-hard.

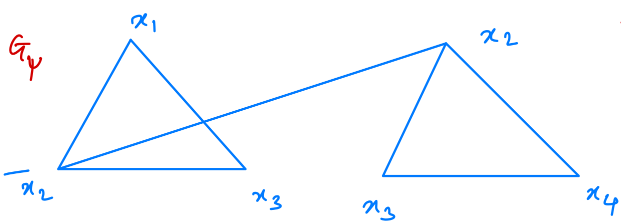
3-SAT $\leq$ IND SET (G, k, is there an independent set of size k)

Pf : Construct $\psi \rightarrow G_\psi, k_\psi$ st.

ψ is satisfiable iff G_ψ has independent set of size k_ψ

From every clause, create triangles, connect complements

$(x_1 \vee \bar{x}_2 \vee x_3) \wedge (x_2 \vee x_3 \vee x_4)$



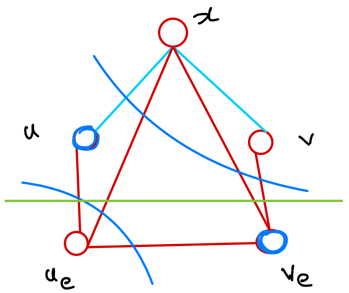
K_ψ = #clauses $\leftarrow$ in each clause pick 1 literal to satisfy clause

IND SET $\leq$ MAX-CUT

Pf : $G_i, k_i \longmapsto G_m, k_m$

G_i has independent set of size k_i iff G_m has cut of size atleast k_m

We know complement of ind-set is vertex cover.



G has ind-set of size k (I)

start with $S = I$

$\{u,v\}$ $u \in I$ v $\notin I \Rightarrow$ add u_e to S

$u, v \notin I \rightarrow$ add u_e, v_e to I

($u, v \in I$ not possible)

4 edges cut in both cases.

↓

$K_m = k + 4|E|$

#blue edges cut = k

red edges cut = 4 per edge

INTPROG : (Integer programming)

3-SAT ≤ INTPROG

$$0 \le y_i \le 1$$

$$x_1 \vee \overline{x_2} \vee x_3 \longmapsto y_1 + (1-y_2) + y_3 \geq 1$$

Quadratic Programming

Degree 2 inequalities.

3-SAT ≤ Quad-Prog (Since linear programs are quadratic)

Trick : $y_i = y_i^2$ (to set $y_i = 0$ or 1)

We don't know if it is in NP ?

Solution can be arbitrarily large ⇒ Can't say certificate is poly size.

3-SAT ≤ 3-coloring

3-SAT ≤ Subset-sum

Conjecture : 3-SAT doesn't have a polytime algorithm

$EXP = \bigcup_{c \geq 1} DTIME(2^{n^c})$

$E = \bigcup_{c \geq 1} DTIME(2^{nc})$

Notation

EXP is bigger class than E

Claim : $NP \subseteq EXP$

Pf : Iterate over all certificates

Pf : Reduce to SAT, $SAT \subseteq EXP$

Theorem : $P \subsetneq EXP$ (proper subset)

Pf : We prove it by diagonalisation

Time – Hierarchy Thm

E.g. $DTIME(n^2) \subsetneq DTIME(n^3)$

Time constructible fn : $f : \mathbb{N} \to \mathbb{N}$

If $f(n)$ can be computed in $O(f(n))$ time.

Thm : For two time constructible fn $f(n), g(n)$ s.t.

$$f(n) \log f(n) = o(g(n)) \quad \{\text{little } o\}$$

i.e. $g(n)$ is not $O(f(n) \log f(n))$

Then, $DTIME(f(n)) \subsetneq DTIME(g(n))$

E.g. $f(n) = n$, $g(n) = n^2$

Note :

$f(n) = o(g(n))$

means $\forall \alpha > 0, \exists n_0 > 0$

s.t. $\forall n \geq n_0$

$f(n) < \alpha g(n)$

$O(g(n))$: $\exists \alpha > 0$

or

$f(n) = o(g(n))$

iff $g(n)$ is not $O(f(n))$

Universal TM

Input : $\langle x_M, \alpha \rangle$

Output : $M(\alpha)$

x_M : integer coded version of M

M : a Turing Machine

If $M(\alpha)$ runs in time t

$U(x_M, \alpha)$ runs in time $O(t \log t)$

Interleave encoding of TM M, timer.

$B_0, B_1, \ldots$: copies of M

$\log t$ size blocks.

$C_0, C_1, \ldots$: computation of M.

copy of M ($\log t$ size) + $\log t(n)$ binary number = timer

Assumptions : 1. Every string represents a TM (say all invalid strings corresp. to trivial TM)

2. For any TM there are infinitely many representations ← seems crucial to pf (like adding comments to C++ code)

DTIME $O(n) \subsetneq O(n^2)$

$D : \{0,1\}^* \longrightarrow \{0,1\}$

some number < 2

$D(\langle M, x\rangle)$: Run M on input $\langle M, x\rangle$ for $|x|^{1.9}$ steps

If it stops, flip output $\begin{pmatrix} 1 \to 0 \\ 0 \to 1 \end{pmatrix}$

If doesn't stop, output 0.

Take any TM N which always halts in $O(\underbrace{|\text{input}|}_{\text{decides computation of } M \text{ in } O(n) \text{ time}})$ time (i.e. $O(n)$)

To show : N, D cannot accept same language.

Input : $\langle x_N \leftarrow \text{large}, w\rangle$

N halts before $|x|^{1.9}$

N and D will disagree on a "large enough" string representing N.

(assumption 2)

Output of D : Runs N on input $\langle N, w\rangle$ for $|x|^{1.9}$ steps $\longrightarrow$ stops $\Rightarrow$ opp N

Output of N : Runs N on input $\langle N, w\rangle \Rightarrow$ output N.

different outputs.

Space Complexity

SPACE(s(n)) : Decision problems for which there is TM working space s(n)

TM : Input tape (read only)
Work tape (read & write)

Functions (output is arbitrary)

TM : Input tape (Read only)
Output tape (Write only)
Work tape (Read / write)

Obs : $DTIME(S(n)) \subseteq SPACE(S(n))$ {can't use more cells than # steps}

$P \subseteq PSPACE$

$SPACE(s(n)) \subseteq DTIME(2^{O(S(n))})$

Configuration ← tape ← $2^{s(n)}$ config
state ← constant
head position ← $\underset{\substack{\uparrow \\ \text{work}}}{s(n)} \cdot \underset{\substack{\uparrow \\ \text{input tape}}}{n}$ positions

} $2^{O(S(n))} \cdot n = 2^{O(S(n))}$

TM always halts ⇒ Doesn't repeat configuration

We always consider $S(n) = \Omega(\underbrace{\log n})$
atleast store which cell in input tape we are looking at

NSPACE(s(n)) : On every computation path space is bounded by $O(s(n))$

DTM : Configuration graph : outdegree ≤ 1

o →
o → o → o

NDTM : Confgn graph : some directed graph

NTM : $s(n)$ ⟶ build configuration graph : $2^{O(S(n))}$ vertices
+ edges using transition function

Accepted by NTM ⇒ ∃ path Reaches an accepting state
↓
$O(|E|) = (2^{s(n)})^2 = 2^{O(s(n))}$

$DTIME(s(n)) \subseteq SPACE(s(n)) \subseteq NSPACE(S(n)) \subseteq DTIME(2^{O(S(n))})$

$\subseteq SPACE(s(n)^2)$

$NP \subseteq PSPACE \leftarrow$ go over all certificates, run checking TM.

$L \subseteq NL \subseteq P \underset{?}{\subseteq} NP \underset{?}{\subseteq} PS \underset{=}{\subseteq} NPS \underset{?}{\subsetneq} EXP$

$L \neq PS$, $P \neq EXP$

$\uparrow$

log-space   SPACE hierarchy thm

$\downarrow$ index

E.g. minimum in an array : maintain index of minimum element (comp. using traversal over bits)

Addition

(is $a+b=c$?)

$a \cdot b = c$?

$f(x) \in L$

$g(x) \in L$

} functions (output tape)

$f \circ g(x) \in L$

Pf : f only accesses few bits of $g(x)$

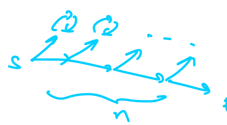
Recalculate g, output only that bit.

$g \in L \Rightarrow g \in P \Rightarrow$ output needs index space $\in L$

① Undirected graph s, t
is there a path from s to t

} Random walk $\rightarrow p > 0$ (RL)

(2005) Reingold $\in L$



: $p(s \rightarrow t) = \frac{1}{2^n}$

② Directed graph s, t is there path from s to t ? $\in NL$ (trivial)

(open) $L = NL$?   ($NL \subseteq \log^2$ $\notin L$)

$NP \subseteq PSPACE \supseteq Co-NP$

QBF : $\exists x_1 \forall x_2 \exists x_3 \forall x_4 \exists x_5 \cdots \forall x_n \; \varphi(x_1 \ldots x_n)$

QBF $\in$ PSPACE $\leftarrow$ can design recursive algo

$x_1 = 0 \; (\; \underbrace{P(x_2 \ldots)}_{x_2 = 0 \,\wedge\, x_2 = 1} \;) \vee (x_1 = 1) \; (\qquad)$ } Recursion depth = poly(n).

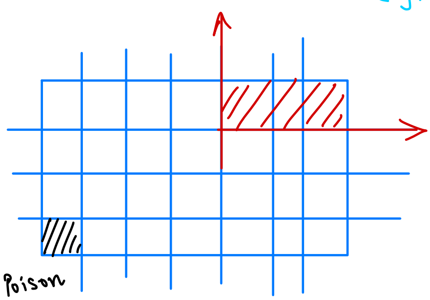
Combinatorial games : from configuration X is there a winning strategy for black?

Two player Games with perfect information, no randomness $\in$ PSPACE

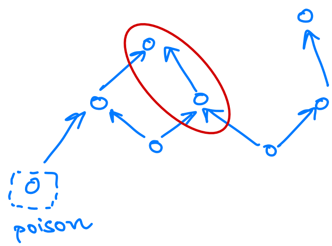
no hidden things

E.g. Poker

Many such problems are PS-complete



Poset Game :



Poset Game is PS-complete

Savich Thm

3018

$NPSPACE(S(n)) \subseteq SPACE(S(n)^2)$

$\Rightarrow NL \subseteq L^2$

A TM with space bound $O(S(n))$, and an input x.

$\longrightarrow$ Configuration graph : $2^{O(S(n))}$ size.

Def.

$\parallel$

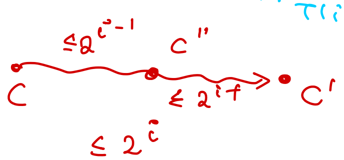
(head, state, tape)

Is there a path from starting configuration to accepting confgn?

Given C, C' how to find edge C to C' $\leftarrow$ check in $O(S(n))$ space.

Is PATH(c, c', i) : whether $\exists\, c \to c'$ of length $\leq 2^i$?

$T(i)$



iterate over c'' ($2^{O(S(n))}$ choices

$S(n) \to S(n)-1 \to \ldots$

$$\text{PATH}\underbrace{(c, c'', i-1)}_{\text{can re-use space}} \wedge \underbrace{(c', c'', i-1)}$$

$T(i) = T(i-1) + \text{store } c'' \text{ choice}$

$(\text{space}=T) = T(i-1) + O(S(n))$

$i = O(S(n)) \Rightarrow T(i) = O(S(n)^2)$

<u>Conclusion</u> : Reachability on n nodes $\in$ space $(O(\log^2 n))$, time $(n^{\log n})$

$$\text{Time} = \Big(\underbrace{2}_{\substack{\text{choices of } c\\ \text{in each step}}}{}^{O(S(n))}\Big)^{(S(n))} \leftarrow \#\text{ steps.}$$

A problem is PS – complete if every problem in PSPACE reduces polytime to Q, and $Q \in$ PS.

<u>Thm</u> : QBF is PS – complete

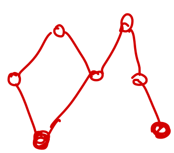
Poset Game = PS – complete.

GAMES against nature. [Papadimitriou]

$\downarrow$

PSPACE – complete

(even these hard to win)



$\longrightarrow$ Memory usage depends only on depth.

<u>Proof</u> : $\in$ PS is easy.

Is there an edge from c to c' ? $\to$ can covert to QBF

$\varphi_i(c, c')$ is true iff $\exists$ path of length $\le 2^i$ from c to c'.

$\varphi_{i+1} = \exists_{c''}\ \varphi_i(c, c'') \wedge \varphi_i(c'', c')$ ← $2\times$ blowup $s(n)$ times ⇒ Need to be better.

<u>Trick</u> :

$\varphi_{i+1}(c, c') =$

formula size is not doubling now! ← $\exists_{c''}\ \forall_{D_1}\ \forall_{D_2}$ $\boxed{\begin{array}{l}\text{If } (D_1, D_2) = (c, c'') \\ \quad (D_1, D_2) = (c'', c')\end{array}}$ $\Rightarrow \varphi_i(D_1, D_2)$

<u>Up Next</u> : Directed reach is NL - complete.

Immerman Szelepcsényi $NL = co-NL$

Unreachability has a certificate

We are going to do :

1. NL - completeness : reachability

   If reachability $\in L \Rightarrow NL = L$

2. $NL = co-NL$

NL - complete

A language $\mathcal{Q}$ is NL - complete if every language in NL <u>log-space reduces</u> to $\mathcal{Q}$.

SAT $\le_{exp}$ Reachability

Solve SAT in EXP

if Yes s—t

No s . .t

Log-space reduction

We say there is a logspace reduction from L to L' if there is a logspace computable function f s.t.

$|f(x)| \le |x|^c$, $x \in L$ iff $f(x) \in L'$

Log-space computable fn

1. Output tape – write only
2. If $|f(x)|$ can be computed in log-space and moreover, given i, i^{th} bit of $f(x)$ can be computed in log-space.

Thm: Reachability is NL-complete

Proof: Every problem $\underbrace{x \text{ in NL}}_{\text{has a NTM}}$ has to be reduced to reachability

Take any $Q \in NL$

log-space NDTM for Q, $x \longmapsto G_x$ (configuration graph)

x is accepted by NTM iff $\exists$ $s \to t$ path in configuration graph.

First, given c, c' is there an edge between them?

L' is language of (G, s, t) s.t. in G, t is reachable from s.

Is it log-space reduction?

Given NTM, we want to print graph in log-space (onto output tape)

Next pf:

NL = co-NL (readability $\in$ co-NL, i.e., unreachability $\in$ NL)

Immerman – Szelepcseny

If t is not reachable from s then there is certificate for this fact which is verifiable in logspace

(certificate is not logspace bounded!)

NL: **Defn**

$Q \in NL$ if $\exists$ NDTM M of logspace bound s.t. $x \in Q$ iff $M(x)$ accepts on some computational path.

Proof defn

$\mathcal{Q} \in NL$ if $\exists$ logspace DTM (verifier) s.t. $x \in \mathcal{Q}$ iff $\exists y \in \{0,1\}^{|x|^c}$

s.t. $M(x,y) = 1$

if NDTM is given, Non-det choices can be written on certificate

y is large but since working space limited, we can't write whole y

Hence, NDTM def $\xrightarrow{\text{conv. to}}$ $\underbrace{\text{Proof defn}}_{\text{stronger}}$ ✓ (not other way around)

Note : SAT $\in$ NL by proof def^n

- For certificate, if read once tape (only fwd) then both defn equivalent.

Now, showing reach $\in$ co-NL

Let us assume, we know no. of nodes reachable from s (say k).

Can we give a certificate ?

Yes, give node & path to node given, can verify in one go. If t not one of them, done.

$K_i \longleftarrow$ no. of nodes reachable from s using length $\leq i$ paths

Can you verify K_i if you are convinced about K_{i-1}

C_i = set of nodes reachable from s using $\leq i$ paths

$K_i = |C_i|$

if $v \in C_i \leftarrow$ easy cert. (give path)

$v \notin C_i \leftarrow$ want easy certificate

no neighbor of $v \in C_{i-1}$

So C_{i-1} given as certificate [+ path from s to $u \in C_{i-1}$]

For all vertices,

$(C_{i-1}$, its paths$)$ $\times$ # vertices $= O(n^4)$ size of certificate

Hence, unreachability $\in$ NL (NL = co-NL)

This pf also shows for $s(n) \geq \log n$

$$NSPACE(s(n)) = co-NSPACE(s(n))$$

$\{ NL = NSPACE(\log n) \}$

6/9

- HW : 2-SAT $\in$ NL
- EXACT IND SET : Given a graph, number k, is the max ind-set of size $= k$

(most likely it is out of NP and co-NP)

- min DNF : Given a DNF (or of ands) ψ, k, does $\exists$ another DNF ϕ equivalent to ψ and size $(\phi) \leq k$
- Succint Tournament Reachability (STR)

Tournament : Directed graph s.t. $\forall i,j$ exactly one of (i,j), (j,i) is an edge

Given boolean formula ψ on $2n$ variables

$$\psi(i,j) = \begin{cases} 1 & \text{if } i \to j \\ 0 & \text{if } j \to i \end{cases} \qquad i,j \in [1,n]$$

Given t, s is t reach from s.

Nothing is said about size of ψ, if exp then certificate can be path itself

Say polynomial, then it is interesting

Most likely out of NP & co-NP.

All 3 of them $\in$ PSPACE

Want : classes outside P, NP but inside PSPACE

Polynomial Hierarchy

Σ_2^P : A language L is in Σ_2^P if there is polynomial time TM and a polynomial q s.t. $x \in L$ iff

$$\exists\, u \in \{0,1\}^{q(|x|)} \;\; \forall\, v \in \{0,1\}^{q|x|} \;\; M(x,u,v) = 1$$

It is a generalisation of NP (No v is NP)

EXACT IND-SET

M takes two subsets U, V
$|U| = k$, U is ind and if $|V| > k$, V should not be ind
$\Rightarrow$ output 1.

Complement

$$\Pi_2^P = \{ \{0,1\}^k \setminus L : L \in \Sigma_2^P \}$$

$x \in L$ iff $\forall\, u \in \{0,1\}^{q(|x|)} \; \exists\, v \in \{0,1\}^{q(|x|)} \;\; M(x,u,v) = 1$

($\forall u \Rightarrow$ product, $\exists u$ = sum so may be Π, Σ resp)

STR $\in \Pi_2^P$ HW (Not easy to see).

Σ_i^P : A language $L \in \Sigma_i^P$ if $\exists$ polynomial time TM and polynomial q s.t. $x \in L$ iff $\exists\, u_1 \in \{0,1\}^{q(|x|)} \; \forall u_2 \; \exists u_3 \ldots Q_i u_i$

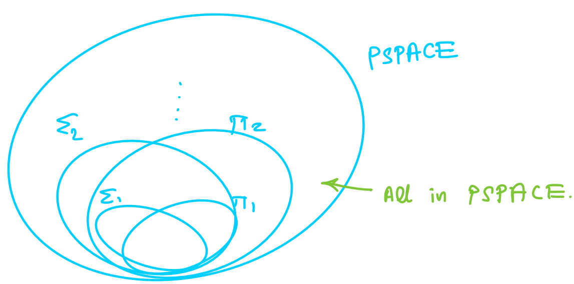
$$M(x, u_1, u_2, \ldots, u_i) = 1$$

* $\Sigma_1^P = NP$, $\Pi_1^P = $ co NP

* $\Sigma_i^P \subseteq \Sigma_{i+1}^P$, $\Pi_i^P \subseteq \Pi_{i+1}^P$

$\Sigma_i^P \subseteq \Pi_{i+1}^P$, $\Pi_i^P \subseteq \Sigma_{i+1}^P$

$PH = \bigcup_{i \geq 1} \Sigma_i^P$, $PH \subseteq PSPACE$



Theorem : If $\Sigma_{i+1}^P = \Sigma_i^P \Rightarrow PH = \Sigma_i^P$

Thm : $\Sigma_i^P = \Pi_i^P$ iff $PH = \Sigma_i^P = \Pi_i^P$

Thm : $P = NP \Rightarrow PH = P$ ($P = NP \Rightarrow$ whole hierarchy collapses to P)

Given G, given lists $L(v)$ of size k is it possible to color s.t. every maximal clique is not monochromatic ?

$\exists_{coloring} \ \forall_{max-clique}$ (check max-clique is easy)

so it is in Σ_2^P

" For any given lists "

$\forall_L \ \exists_{color} \ \forall_{max-clique}$: Π_3^P complete. (Karp-reduction, many-one).

Σ_2^P - complete :

$\varphi(x,y)$ is 3-CNF

$\exists x \in \{0,1\}^n \ \forall y \in \{0,1\}^n \ \varphi(x,y)$ $\Big\}$ Σ_2 SAT.

Thm : $P = NP \Rightarrow PH = P$

Let $P = NP$

Using induction let $\Sigma_{i-1}^P = P$ (to show $\Sigma_i^P = P$)

For any $L \in \Sigma_i^P$, $x \in L$ iff (we construct L' s.t. $L' \in \Sigma_{i-1}^P \Rightarrow L \in P$)

$\exists_{u_1} \forall_{u_2} \dots Q u_i \quad M(x, u_1, \dots u_i) = 1$

$L' = \{ (x, u_1) : \forall u_2 \exists u_3 \dots \quad M(x, u_1 \dots u_i) = 1 \}$

Then $(L')^c \in \Sigma_{i-1}^P \Rightarrow (L')^c \in P \Rightarrow L' \in P$

$(x, u_1) \in L'$ iff $M'(x, u_1) = 1$

$L = \{ x : \exists_{u_1} \ M'(x, u_1) = 1 \} \Rightarrow L \in NP = P \Rightarrow L \in P$

<u>Oracle definition</u> : NP^{SAT} (NP using SAT oracle)

$NP \subseteq NP^{SAT}$

$\exists_{u_1} \boxed{\forall u_2 \; M(x, u_1, u_2) = 1} \Rightarrow$ say comes with certificate

↳ want to do in polynomial.

$\forall_{u_2} M(x, u_1, u_2) = 1 \qquad \neg \underbrace{\exists_{u_2} \; M(x, u_1, u_2) = 0}_{\text{invert SAT outcome}}$

$NP^{SAT} \subseteq \Sigma_2^P$ (not trivial)

↳ $\exists$ certificate $\rightarrow$ can be captured by $\exists c_1$

More generally.

$$\Sigma_i^P = NP^{\Sigma_{i-1}^P} = NP^{\Pi_{i-1}^P}$$

$$\Pi_i^P = Co-NP^{\Sigma_{i-1}^P} = Co-NP^{\Pi_{i-1}^P}$$

Another thm : SAT - best known algo $O(2^n)$ time

Exp time hypothesis : SAT cannot be solved below 2^n (conjecture)

Can SAT be solved in linear time ?

Can we show $SAT \notin L$?

} open problems

We can show, (using hierarchy thms)

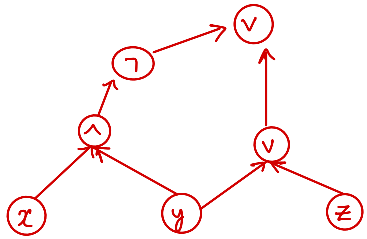
<u>Thm</u> : SAT cannot be solved in $O(n^{1.1})$ time and $O(n^{0.1})$ space

Boolean Circuits

DAG : 1 sink output gate
n source input gates

$f : \{0,1\}^n \rightarrow \{0,1\}$

Each node is AND / OR / NOT gate



$\neg(x \wedge y) \vee (y \vee z)$

Arbitrary fan-in $\longrightarrow$ fan-in 2

$(x \wedge y \wedge z) \longmapsto (x \wedge y) \wedge z$

Increase in size, depth $\longrightarrow$ manageable
(not too much blowup)

Size of circuit = no. of gates

Depth of circuit = longest path b/w output, input. (time of computation)

Boolean formula (fan out = 1 . except from input)

$((a \wedge b) \wedge c) \vee ((a \wedge b) \wedge d)$

have to
write again

Straight line Programs $\equiv$ Boolean circuits

↳ no jumps, loops allowed
every line : assignment / decl.

gates = # lines in SLP

Compl class P/poly if problem has poly sized circuit.
diff. ckts needed for diff input length $\Rightarrow$ need ckt families to define P/poly.

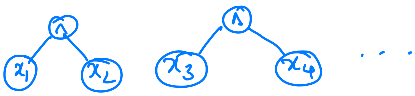
Circuit family $\{C_n\}_{n\geq 1}$

Language L is said to be in size$(T(n))$ if there is a circuit family $\{C_n\}_{n\geq 1}$ s.t. $|C_n| = O(T(n))$,

$$x \in L \text{ iff } C_{|x|}(x) = 1$$

$$P/poly = \bigcup_{c\geq 1} \text{Size}(n^c)$$

E.g. AND$(x_1, x_2, \ldots, x_n)$



Claim : $P \subseteq P/poly$.

Pf : TM running in P $\longrightarrow$ simulate that computation using circuit (gates are auxiliary variables)

Gates
1. Every cell of tape, for every time step
2. States for every time step
3. Head position for every time step.

This also shows $\underbrace{\text{CKT-SAT}}_{\text{SAT for boolean circuits}}$ is NP-complete

Is $P/poly \not\subseteq P$?

Given $\{C_n\}$, x $C_n(x)$ can be computed in polynomial time

Need to generate C_n in P time $\longleftarrow$ weaker class of circuits.

Undecidable problems in P/Poly

If no size bound, every boolean fn (including Halting) has a ckt. (could be huge). If size = Polynomial, can we compute undec. problem?

Yes!

UHALT : $\{ 1^n : n^{th}$ TM halts on $n \}$ ← exactly 1 string of length, C_n = trivial ckt (hardcode)

1^n → n = unary representation

↓

undecidable language.

$NP \subseteq P/Poly$? ← ∃ poly ckt to solve SAT/ Independent set (open)

Need to find C_n given n

$P/Poly$: Like NNs tough to find but small size

<u>Goal</u> : INDSET does not have polysize circuits $\Rightarrow P \neq NP$

$P \subseteq P/poly$ INDSET $\notin P/poly$

Karp – Lipton

$$NP \subseteq P/Poly \Rightarrow PH = \Sigma_2^P$$

(don't believe)

NP: believe : NP-c aren't sparse

$L = \{a_1, \ldots a_n\}$ ← easy circuit $\Rightarrow \in P/Poly$

small

We will show if $NP \subseteq P/Poly$ then $\Pi_2^P \subseteq \Sigma_2^P$.

Π_2^P i.e. $\forall \exists \subseteq \exists \forall$ so swapping is ok.

If $L \in \Pi_2^P \Rightarrow \exists$ TM M s.t. $x \in L$ iff $\forall_y \underbrace{\exists_z \; M(x,y,z) = 1}_{\in NP}$

y, z: poly sized strings

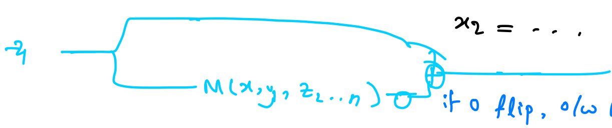
$L' = \{ (x,y) : \exists_z \; M(x,y,z) = 1 \}$

M runs in P

$L' \in NP$.

$\Rightarrow L'$ has poly sized circuit. ← input $(x,y) \Rightarrow$ compute ckt (Poly sized) to recover z.

Put $x_1 = 0$ if still SAT
$x_2 = \ldots$ } self-reduction.



if 0 flip, o/w keep same

Now, given correct z, we can verify circuit ← check if output z is correct.

even if circuit is wrong it's fine, just need correct z

$x \in L$ iff $\exists$ ckt computing z on input (x, y) ← C.

$$\underbrace{\exists_C}_{\text{independent of } x,y} \forall_y \; M(x, y, C(x,y))$$

HW Claim: $\exists$ poly sized ckt which on input x, y outputs z s.t.

$M(x, y, z) = 1$. (assuming $NP \subseteq P/Poly$)

NP : Best Lower Bound $\sim 5n$

$P \subseteq P/Poly$ (non-uniform model)

$a(n)$

Circuit lower bounds $\Rightarrow$ $P \neq NP$

TM with advice

A language L is said to be in $DTIME(T(n))/a(n)$ if $\exists$ TM running in time $T(n)$ and there is a sequence $(A_n)_n$, $|A_n| = O(a(n))$ such that

$$x \in L \text{ iff } M(x, A_{|x|}) = 1$$

Easy to show : TM w/ advice $\equiv$ $P/Poly$.

$\leftarrow$ A_n = boolean ckt

$\rightarrow$ M, A_n conv. to boolean ckt

Uniform circuits

A circuit family is called logspace uniform if given 1^n, we can compute C_n in logspace.

Thm: $L \in P$ iff there is logspace uniform family of circuits accepting the language

$(M, n) \xrightarrow{\text{logspace}}$ circuit w/ n input gates. need $\log n$ bits space. for n input gates

Note: For ckt lower bounds, we usually consider arbitrary ckt.

There is a function $f_n : \{0,1\}^n \to \{0,1\}$ which requires $\Omega(2^n/n)$ size circuits $\leftarrow$ Hint: Counting argument $\leftarrow$ actually almost all req. this much

Challenge: Construct an explicit f_n

No. of functions $= 2^{2^n}$

No. of size s circuits = # dags with s nodes ?

edges = $O(s)$

circuits = $2^{O(s \log s)}$

ckt = adjacency list

$\underbrace{O(s}_{\text{\# wires}} \underbrace{\log s)}_{\text{gate 1} \to \text{gate 2}}$ bits

$\overset{(say)}{=} 2^{10 s \log s}$

$\simeq 2^{10 \cdot \frac{2^n}{n} \log\left(\frac{2^n}{n}\right)} < 2^{2^n}$ ∎

HW: (not that difficult)

For every boolean fn there is a circuit of size $O\left(\frac{2^n}{n}\right)$ (bound is tight)

Trivial ckt: Hardcode all input-output

2^n confg. $(x_1 \cdots x_n)$ ← takes $\vee$ of those going to 1.

$(x_1 \wedge x_2 \wedge \neg x_3) \vee (x_1 \wedge \neg x_2 \wedge x_3)$

size = $n \cdot 2^n$

Can we find a function in NP or in EXP which requires exponential sized circuit?

Karp-Lipton : $NP \subseteq P/Poly \Rightarrow PH = \Sigma_2^P$ (last class)

So, we expect $f \in NP$ s.t. it doesn't have poly size circuit.

$NP \subseteq EXP$, so even better chances for EXP

} open.

Meyer : $EXP \subseteq P/Poly \Rightarrow EXP = \Sigma_2^P$

Pf: Similar to Karp-Lipton.

Bounded depth circuits : (Analogous to parallel computation, depth determines running time)

NC, AC classes.

NC^i : Class of problems which have boolean circuits of polynomial size and $O(\log^i n)$ depth (fan-in = 2)

NC^0 : " — " with constant depth

AC^i : " — " and $O(\log^i n)$ depth (fan-in = arbitrary)

↑ More powerful class.

$$NC^0 \subseteq AC^0 \subseteq NC^1 \subseteq AC^1 \subseteq NC^2 \subseteq \ldots$$

NC^0 ↑ Not interesting, const q

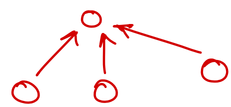
$\underbrace{AC^0 \subseteq}$ arbitrary fan-in (atmost n) → depth $*$ $\underline{\log n}$. (binary tree)

↓ trivial ckt: 2-depth, exp size

what expressed in poly? Interesting

80s – 90s : Model of $||^{el}$ computation

(Much more powerful model)



proc. = poly in input

not realistic.

Problem with no small depth ckt ⇒ no $||^{el}$ algorithm.

- Lower bounds against AC^0, NC^1, ...
- Upper bounds ← many interesting fn w/ small depth ckt i.e. potentially $||^{el}$ algorithms (usually uniform)

Addition $\in AC^0$ (uniform)

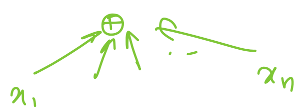
Mult / Div $\in AC^0$ (chinese remaindering)

Parity of n bits $\notin AC^0$ ('88 - '89) ← Among first circuit lower bounds.

We don't know many fns not in NC^1 !

$AC^0[2]$ ↑ const. depth with $\oplus$ gate

Parity $\in AC^0(2)$



$(x_1 + \cdots + x_n) \bmod 3 \notin AC^0[2]$

2011 NEXP $\notin ACC^0$ [Ryan Williams]

constant depth w/ any mod gate

Fast Algo $\Rightarrow$ Ckt lower bounds [started in '11]

Matrix Multiplication $\rightarrow$ n things in $\|^{el}$ add^n $\rightarrow$ binary tree log n depth

log n depth for adding/multiplying $a_i \cdot b_j$

$\in NC^2$

* SOLE, Determinant, Rank $\in NC^2$

(Basically, linear algebra $\in NC^2$)

Reachability /connectivity $\in NC^2$

$NC^1 \subseteq L \subseteq NL \subseteq NC^2$

not easy to see

Bipartite matching $\in NC$? (open)

$$NC = \bigcup_{i \geq 0} NC^i$$

(uniform) $NC \subseteq P$

$P = NC$? (open) : every problem efficiently $\rightarrow$ $\|^{el}$ w/ much faster running time.

P-complete

Google : If any depth hierarchy thm.

Q is P-complete if $Q \in P$ and every problem in P logspace-reduces to Q

E.g. Circuit valuation (TM computation $\Rightarrow$ circuit evalutation)

Ckt-val $\in NC$ $\Rightarrow$ $P = NC$

(open)

E.g. Min-cost flow

E.g. Linear Programming