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
→ 1 with n ↑

Resource : Randomness

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

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

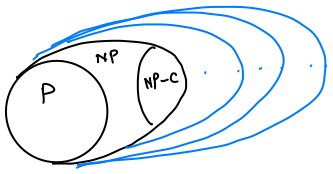
Resource : Communication

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 ≡ Proving ckt lower bounds

Hard to compute f^n → 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. } will show randomized algo more powerful than deterministic.
Without random show $> \log n$

→ 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

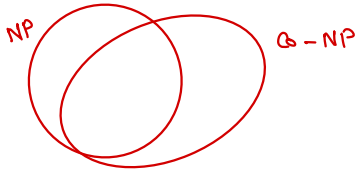
No → give BFS pf, run algo and show

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)



(belief, co-NP ≠ NP)

Allowing interaction makes it possible ! ($\Phi = f(\text{Ans})$: 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 ⟺ Hardness of Approximation

Zero-Knowledge Proofs [Goldwasser]

Want to convince I know, without giving away information

Basic Lower Bound on Sorting

Claim: Sorting n numbers requires $\Omega(n \log n)$ comparisons

Adversarial argument:

Initially $n!$ = $A_i > A_j$ | $A_i < A_j$ [First query $A_i > A_j$]

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

$n! \xrightarrow{1^{st}\ query} n!/2 \xrightarrow{2^{nd}} n!/4$ } $\log_2(n!)$ 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

Puzzle: n numbers, exactly two of them are equal

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

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

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

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 $n < k$? → Binary search.

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

Counting argument,

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

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

Diagonalisation

Well defined f^n : 1. Let G have a program.

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 → input $i \in \mathbb{N}$

run program G on input i

If output is 1 → loop

0 → return 0

Let j = index of program P

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

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

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

H : input i, x

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

Halting problem 0 otherwise

HW Problem : 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

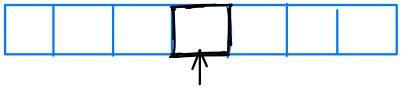
- Any C++ program can be converted into a polynomial eqn s.t. program on this input halts iff equation has integer solution

→ Similar approach for undecidability of game of life

Turing Machine

Q ← states , $\Sigma = \{0, 1\}$

- Infinite Tape , head



Transition function

$$\delta : Q \times \Sigma \longrightarrow Q \times \Sigma \times \{\text{left, right, stay}\}$$

(Σ: write symbol; $\{\text{left, right, stay}\}$: movement of head)

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 (makes it zero)
- reset (r)
- 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

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

$t_A(n) = \max_{x \in \{0,1\}^n} t_A(x)$ ← Time Complexity

Efficient Computation

Cobham 1964
Edmonds 1965

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

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

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

Palindrome : 2 tape ← $O(n)$

1 tape ← $O(n^2)$

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

↓ reason why polytime is a universal measure.

Multitape TM $t(n)$

↓

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 ← 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 : $2^l = O(?$ size of formula$)$

Exp-time needed in this.

NP

Decision Problem $\mathcal{Q}$

Search Problem P

The two are equivalent if P has a polytime algo iff $\mathcal{Q}$ 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

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

e.g. $\#SAT = \{ \langle \phi, k \rangle$: no. of satisfying assignments of $\phi \ge k \}$

$|\phi| + \log k$; Not sure if in NP ; certificate size $\le k|\phi|$

e.g. MCSP (min. circuit size problem) $= \{ \langle \phi, k \rangle$: 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

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

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

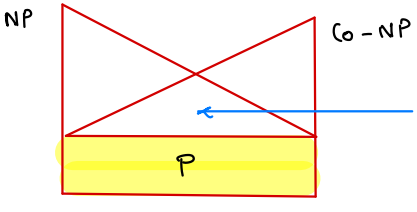
$P \subseteq NP$, Certificate : ε , TM runs the algo itself

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

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

Easy to see : PRIMES $\in$ Co-NP $\rightarrow$ certificate - a, b, $n = ab$

Known to be in P before 2002



$P \subseteq NP \cap co\text{-}NP$

Not known if $P = NP \cap co\text{-}NP$

But if $L \in NP \cap co\text{-}NP$ 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 $\neq$ s.t.

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

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

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

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

q_{accept}, q_{reject} ← Halting states

(3 Non-det transition → 2 w/ poly running time blowup)

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 ⇒ NTM

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

NTM ⇒ Verifier

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

PRIMES ∈ 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

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

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

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

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

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

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

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

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

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

$\gcd(r_1, r_2) = 1$

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

↓ 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.

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

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

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

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

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

2, 3 prime factors.

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

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

↓

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) → 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

↓ poly in $(|M|, |x|, c, 0)$

Tape :

Tape at $t = t_{i,*}$ { $t_{00}, \ldots, t_{0T}$; $t_{T0}, t_{T1}, \ldots, t_{TT}$; $q_{01}, \ldots, q_{0K}$; $q_{T1}, \ldots, q_{TK}$ } State

Head position { $h_{00}, \ldots, h_{0T}$; $h_{T0} \ldots h_{TT}$

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

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

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

Proof of complexity :

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

n holes, n+1 pigeons

Resolution ← exponential size.

Examples ∈ Co-NP ∩ 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. $a^r \equiv b \mod p$

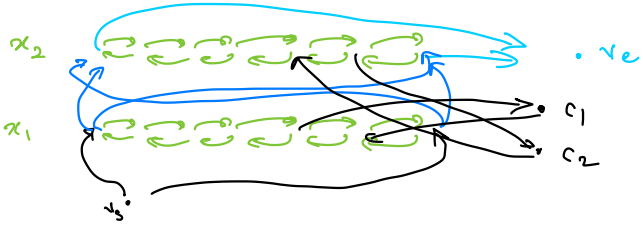
HAM · Given a directed graph, is there a path from v_1 to v_n which covers all vertices

$SAT \leq_p HAM$

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

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

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



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? ← Show NP-hard.

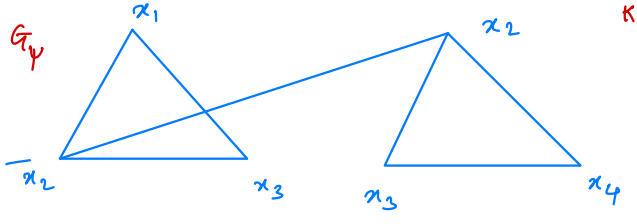
3-SAT $\leq$ IND SET (G, k, is there an independent set of size k)

Pf: Construct $\psi \rightarrow G_\psi, k_\psi$ s.t.

ψ is satisfiable iff G_ψ has independent set of size k_ψ

From every clause, create triangles, connect complements

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



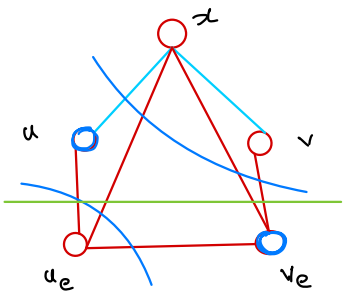
k_ψ = #clauses ← in each clause pick 1 literal to satisfy clause

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 $S = I$

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

$u, v \notin I$ → add u_e, v_e to I

4 edges cut in both cases.

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

$K_m = k + 4|E|$

#blue edges cut = k

red edges cut = 4 per edge

INTPROG : (Integer programming)

3-SAT ≤ INTPROG

$0 \le y_i \le 1$

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

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.

$f(n) \log f(n) = o(g(n))$ {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 : $\langle \chi_M, \alpha \rangle$

Output : $M(\alpha)$

χ_M : integer coded version of M

M : a Turing Machine

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

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

Interleave encoding of TM M, timer.

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

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

(copy of M (log t size)) + (log t(n) binary number = timer) ← (log t size blocks.)

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

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

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

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

$D(\langle M, x\rangle)$: Run M on input $\langle M, x\rangle$ for $|x|^{1.9}$ steps ← some number < 2

If it stops, flip output ($1 \to 0$, $0 \to 1$)

If doesn't stop, output 0.

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

To show : N, D cannot accept same language.

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

N and D will disagree on a "large enough" string representing N. → N halts before $|x|^{1.9}$ steps (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 $\leftarrow$ tape $\leftarrow 2^{s(n)}$ config
state $\leftarrow$ constant
head position $\leftarrow$ s(n) . n positions (work, input tape)

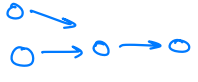
$2^{O(s(n))} \cdot n$
$= 2^{O(s(n))}$

TM always halts $\Rightarrow$ Doesn't repeat configuration

We always consider $s(n) = \Omega(\log n)$
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



NDTM : Confgn graph : some directed graph

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

Accepted by NTM $\Rightarrow$ $\exists$ path reaches an accepting state

$O(|E|) = (2^{s(n)})^2 = 2^{O(s(n))}$

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

$\subsetneq SPACE(S(n)^2)$

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

$L \subseteq NL \subseteq P \subseteq NP \subseteq PS \subseteq NPS \subseteq EXP$

$\neq$ (over P to EXP)

$\subsetneq$ (over L to NL) ; $=$ (over PS to NPS) ; ? (over NP to PS, over NPS to EXP)

↑ log-space

↓ index

SPACE hierarchy thm

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)

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

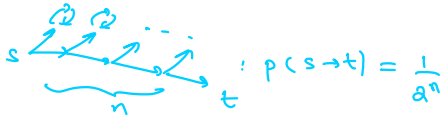
Pf : f only accesses few bits of $g(x)$

Recalculate g, output only that bit.

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

① Undirected graph s, t is there a path from s to t } Random walk → $P > 0$ (RL)

(2005) Reingold $\in L$



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

(open) $L = NL$?

($NL \subseteq \log^2 \not\in L$)

$NP \subseteq PSPACE \supseteq co\text{-}NP$

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 ← can design recursive algo

$x_1 = 0 \; (\; P(x_2 \ldots) \;) \; \vee \; (x_1 = 1) \; (\quad)$

$x_2 = 0 \wedge x_2 = 1$

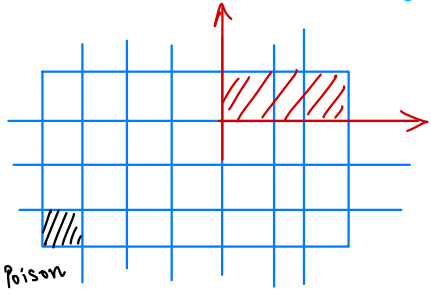
} Recursion depth = poly(n).

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

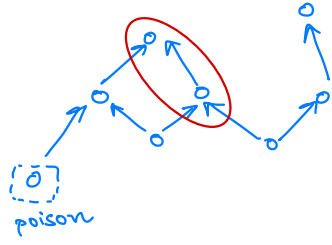
Two player Games with perfect information , no randomness $\in$ PSPACE

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.

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

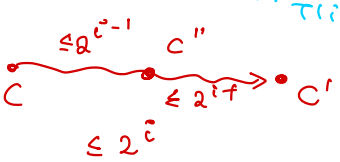
Def. (head, state, tape)

Is there a path from starting configuration to accepting confgn?

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

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

$\Leftarrow T(i)$



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

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

PATH $(C, C'', i-1)$ $\wedge$ $(C', C'', i-1)$

can re-use space

$T(i) = T(i-1) +$ store C'' choice

(space=T) $= T(i-1) + O(S(n))$

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

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

Time $= \left(2^{O(S(n))}\right)^{(S(n))}$ ← # steps.

choices of C in each step

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

Thm : QBF is PS-complete

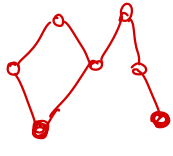
Poset Game = PS-complete.

GAMES against nature. [Papadimitrou]

↓

PSPACE-complete

(even these hard to win)



→ Memory usage depends only on depth.

Proof : $\in$ PS is easy.

Is there an edge from C to C' ? → can convert to QBF

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

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

Trick :

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

formula size is not doubling now !

$\leftarrow \exists_{c''} \forall_{D_1, D_2}$ [If $(D_1, D_2) = (c, c'')$ $(D_1, D_2) = (c'', c')$] $\Rightarrow \psi_i(D_1, D_2)$

Up Next : Directed reach is NL - complete.

Immerman Szelepcsényi $NL = co\text{-}NL$

Unreachability has a certificate