# 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)

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

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

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

n vertices ⇒ $\log n$ 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) $\xrightarrow{f}$ : complicated fn, looks random.

Not predictable ⇒ random [view of complexity theory]

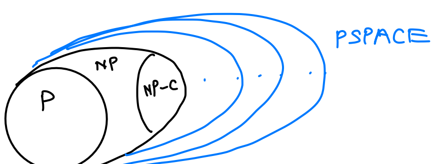
Resource : Communication

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 $\leq 100$

Yes → give path

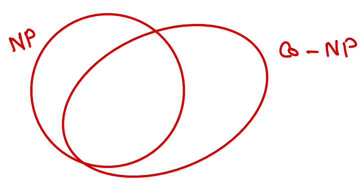
No → give BFS pf, run algo and show

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 $\neq$ 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



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

**Adversarial argument**:

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

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

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 : $\phi$ , output : a satisfying assignment ,
   decision : given $\phi$ , 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

Well defined $f^n$ : 1. $G$ : input natural no. $i$ — Let $G$ have a program.

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 $j$ } 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 $x$

0 otherwise

Halting problem

HW Problem : Given two C++ programs, do they have same behaviour ?

[Show undecidable]

## Hilbert's $10^{th}$ problem

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

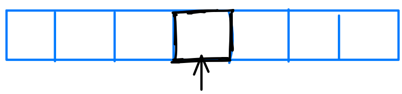
- Any C++ program can be converted into a polynomial eq$^n$ s.t. program on this input halts iff equation has integer solution

→ Similar approach for undecidability of game of life

## Turing Machine

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

- Infinite Tape , head



Transition function

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

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

Abstract RAM Machine

| | | |
|---|---|---|
| Infinite memory cells | ← | int (unbounded) |
| (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

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

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

## 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 $t(n)$

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 $2\ell$ variables.

(Defines a graph on $2^\ell$ vertices)

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

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

EXP – time needed in this.

---

NP

10/8

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.

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

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

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 $\phi$ with size atmost $k \}$

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

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

Not sure if in NP

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

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

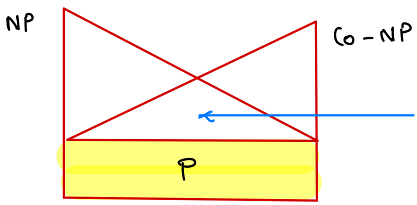
$P \subseteq NP$ , Certificate : $\varepsilon$ , TM runs the algo itself

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

PRIMES $\in P$ : $\{ \langle n \rangle : n \text{ is prime} \}$

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 $L \in P$

$GI = \{ \langle G, H \rangle : G \,\&\, H \text{ are isomorphic} \}$

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

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

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.

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

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