14/10/17

Post Mid-2

Lec-18

Algorithm:

Four facets of Algos:

1> Where to start?

2> How to flow?

3> Dealing with ADVERSITY?

4> How & where to End?

Start → Flow → Stop

Not knowing Input

Start well

Digital signal (0, 1) can never happen.

Sathvik Sanjeev

Stop Well:

TV resolution → increased until decievable

Adversity:

Password of ∞ security ⇒ ∞ length.

1> But a password that gets changed before cracking.

@ time t0

Info(R) = Info(Eve) dropper

Eve

S ——— R

At every subsequent time period

Info_Rec(R) = Info(Eve)

Randomised Quick Sort

affects → Adv: $O(n^2)$ input

Adv: A randomized quick sort. Don't know what will happen

Greedy, Divide & Conquer, D.P, L.P are just methods of flows.

The Start:

What can machines do?

1> Machines are not OMNIPRESENT

(Else Einstein - $v \leq c$ is broken)

2> Machines are NOT OMNISCIENT

Sathvik S

Sathvik

(They are not HERMOINE's ip) : ∞ info can't be stored in finite space
bcoz in finite space: Only finite symbols can be written

5) MACHINES ARE NOT OMNIPOTENT:
You can give only a finite control (code) to a MACHINES.

Turing Machines:
One instruction only.

Algorithm: is a turing machine:

Turing Machines:-

Isse
7-tuple
Elements are $\langle Q, \Sigma, \Gamma$
Finite Tape set Alphabet
Finite set of States
Finite Input Alphabet
,

$\Gamma \geq \Sigma \cup \{\$\}$
End of file

$\delta : Q \times \Gamma \longrightarrow Q \times \Gamma \times \{L, R\}$

eg: $\delta(q, a) = (q_2, b, L)$

q_{accept} : Out true & Stop

q_{reject} Reject Input

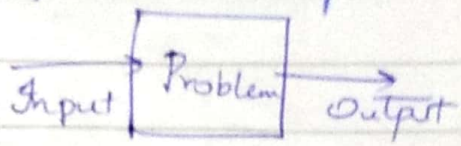
q_{start}:

To show impossibility, we want use Turings language to give the loop.

classmate
Date
Page

Sathish
Sanjeev

What are Algorithmic Problems

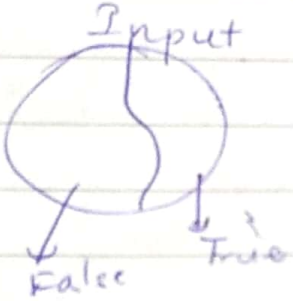


Output is one bit

Bcoz if output is 2 bits ⇒ Equivalent to 2 problems. i.e

What is the first bit?

What is the second bit.



Problem ⊆ Input Space

Problem is the True Submit state.

Σ → Input space

Problem $\subseteq \Sigma^*$

Subset of Σ^* is a language

Problem ≡ L

WAP to input a graph & check if it is connected

$L = \{\langle G \rangle \mid \text{Graph is connected}\}$

WAP to check if input is even

$L = \{x \mid x \text{ is even}\}$

M. Sipser: Introduction to Theory of Computation

21/10/17

Lec-20

Set of all subsets of Σ^* = $P(\Sigma^*)$

No. of problems: $2^{n(\Sigma^*)}$

Problems for which amount of ~~course~~ memory required can be found: Automata Theory $O(1)$

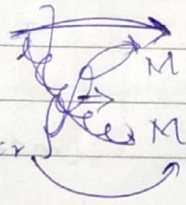
Computability Theory (∞ resources)

Complexity Theory (variable resources)

Turic Machine M is said to recognise/solve ~~solve~~ a language L if $\forall w \in L$, M accepts w.

$\forall w \notin L$, M doesnot accept w. (rejects)

Decider vs Recognizer



M doesnot accept w ~~rejects~~

M reject w (May take ∞ time)

WAP to find integer soln to

$$x^3 + y^3 = z^3$$

→ ∞ possibilities

No Int.

=

Computation problems for which no solution exists

Halting problem:

Can

A set X is same as set Y if

$\exists$ a bijection from X to Y.

$f: X \to Y$

If no one-one: Codomain

If no onto: Domain is larger

Number of real numbers in $(0,1)$ is uncountable

Proof: Suppose it is countable

A bijection

$f: N \longrightarrow (0,1)$

$f(1) : 0.\ d_{11}\ d_{12}\ d_{13}\ d_{14}\ d_{15} \ldots$ $\quad d_{ij} \in \{0 - 9\}$

$f(2) : 0.\ d_{21}\ d_{22}\ d_{23}\ d_{24} \ldots$

$f(3) : 0.\ d_{31}\ d_{32}\ d_{33}\ d_{34} \ldots$

$x \in (0,1)$

Chose $d_1 \neq d_{11}$

$d_2 \neq d_{22}$

$d_3 \neq d_{33}$

$x = 0.\ d_1 d_2 d_3 \ldots$ where $d_i \neq d_{ii}$

$x \neq f(1), f(2), \ldots f(n)$

An x exists, which is not in the range

25/10/17 Lec-21

$T = \{M / M$ is a Turing Machine$\}$

T is countable ($\aleph_0$)

Σ^* is countable

↳ Set of strings in countable

Every C program is a Binary string

$f: N \longrightarrow T$

$f(1) = \epsilon$

$f(2) = 0$

$f(3) = 1$

$f(4) = 00$

Δ is finite

↓

Symbols used in a program

$S = \{L / L$ is a subset of $\{0,1\}^*\}$

Theorem:

S is not countable.

Proof:

$\{1\ 2\ 3\ 4\ 5\} = 11100$

$\{1\ 2\ 3\} = 11100$

$\{\epsilon\} = 01000$

$\{0,1\}^* = \bullet\ \epsilon, 0, 1, 00, 01, 10, 11, \ldots$

$L \subseteq \{\epsilon, 0, 1, 00, 01, \ldots\}$

So we can say whether ϵ is the language / not

Dec $b_1 b_2 b_3 \ldots$ (Bits)

$b_i = \begin{cases} 0 & \text{if } f_i \in L \\ 1 & \text{if } f(i) \notin L \end{cases}$

$S = \{\text{"all infinite length bit strings"}\}$

Consider

$0.b_1 b_2 \ldots$ [Base 2]

$0 \leq \text{Number} \leq 1$

Total number of Langs = No of Real nos b/w 0 and 1

= Uncountable

Directly, we can prove by Diagonalization

$L_1 = f(1) = b_{11} b_{12} b_{13} b_{14} \ldots$

$L_2 = f(2) = b_{21} b_{22} b_{23} b_{24} \ldots$

$L :- \bar{b}_{11} \bar{b}_{22} \bar{b}_{33} \ldots$

So $L \neq L_i$

$\rightarrow$ L exists that is not in the range

$\rightarrow$ Bijective

Or intuitively

Power set of countably ∞ set is uncountable.

$\exists L \notin RE$

2) Again use diagonalization to show $RE \not\supset R$

Consider

$$A_{TM} = \{ \langle M, w \rangle \mid M \text{ accepts } w \}$$

Theorem: A_{TM} is undecidable

A_{TM} is recognizable: simulate M for input w and give its output as output of A_{TM}

Suppose TM H that decides A_{TM}

$$H(\langle M, w \rangle) = \begin{cases} \text{Accepts} & \text{if } M \text{ accepts} \\ \text{Rejects} & \text{if } M \text{ doesn't accept} \end{cases}$$

D: On input M (a TM)

runs $H(M, \langle M \rangle)$

If H accepts, Reject

Else if H rejects, Accept

If H exists $\Rightarrow$ D exist

So D doesnot exist $\Rightarrow$ H doesnot exist

What is $D(\langle D \rangle)$?

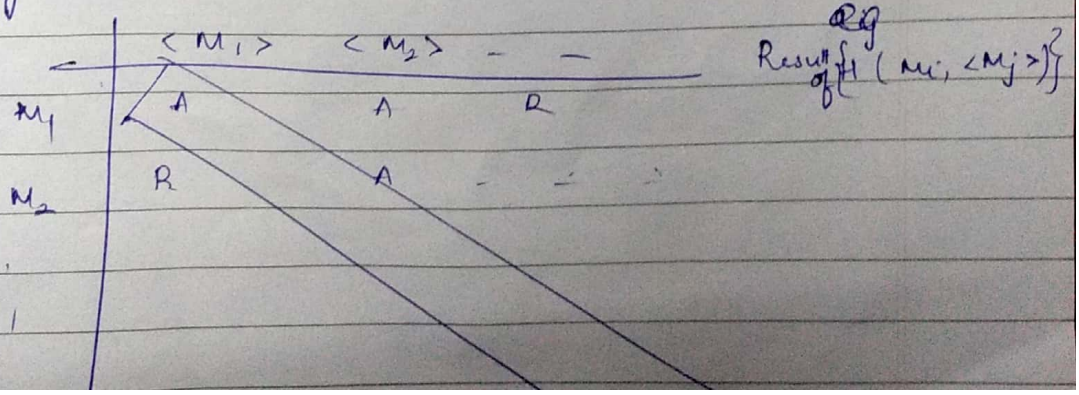
$$D(\langle D \rangle) = H(D, \langle D \rangle)$$

D accepts $\rightarrow$ H rejects

D rejects $\rightarrow$ H accepts

So no decider

Diagonalisation:



$H(D, \langle D \rangle)$

L : { $O(f(n))$
not $o(f(n))$

Space complexity: Total number of spaces taken up by a turing machine

TM : B that decides L for $O(f(n))$
• B is different from any algorithm taking space less than B.

1> Compute $f(|w|) = k$
2> Cut the tape @ k cells, If ever you need more reject f
Let B take M as input
3> B runs M on $\langle M \rangle$, if M accepts - B rejects and vice-versa

If M uses smaller than $f(n)$ space, B can also solve it.
But not same as

If M doesnot halt in $2^{f(n)}$ space, reject.

Lec - 22

Randomized Algos:

Primality Testing

Normally: $O(\sqrt{N})$ → Superpoly in Log

$N > 2$, odd

Consider $A_1, A_2, \ldots A_3, \ldots$

where A_i tests on $\frac{2}{3}^{rds}$ of the set

⇒ So each input gets $\frac{2}{3}^{rd}$ % correct output

Suppose 32 bit number then

Probability of failure is $O\left(\frac{1}{2^{32}}\right)$

Using IC

Fermat's little Theorem:-

N is prime → $\forall\ a \in [1, \ldots N-1]$

$a^{N-1} \bmod N = 1$

(NOT Vice-Versa)

$a \quad 2a \quad 3a \quad 4a \quad \ldots \quad (N-1)a$

$i\,a \,\%\, N = j\,a \,\%\, N$

$N \mid (i\,a - j\,a) = 0$

$N \mid (i-j)\,a = 0$

$N \mid |i-j| a = 0$

N is prime ⇒ $(i-j)$ or a

N can't divide a but $i - j < N$

$\Rightarrow N \mid i-j \Rightarrow i = j$

$\prod a \times 2a \times 3a \times \ldots (N-1)a = \prod_{i=1}^{N-1} i \pmod N$

$N \mid (N-1)!$ when N is prime

$(N-1)!\, a^{N-1}$

If N is prime, then

$x^2 \equiv 1 \mod N$

has exactly 2 roots in $[0, N-1]$

$x = 1$

$x = N-1$

$N \mid x^2 - 1$

$N \mid (x-1)(x+1)$

$x+1 = 0$ or $x-1 = 0$

$x = -1, \quad x = 1$

1) N is odd

$(N-1) = \text{even} = 2^r s$ s is odd, $r \geq 1$

2) Create an Array

$A_{k \times (r+1)}$

choose at random $a_0, a_1, \ldots a_{k-1} \xleftarrow{R} [2, N-1]$

$A_{ij} \leftarrow [a_i^{2^j \cdot s} \mod N]$

3) Read each row of A from Right → Left

if ∃ a row where the first number that is not 1 is not N-1

Print N is composite & halt

4) If still not halted, put N is a prime, halt

If N is not prime

$P[\text{Algo prints "Prime"}] \leq \frac{1}{2^k}$

If N is prime, algo

2/11/17

Lec-23

What is a computational problem:
→ A language

What is an Algorithm:
→ A turing machine / A C-Prog / A-Python Prog /

Are there algorithmically unresolvable
Yes, Diagonalisation

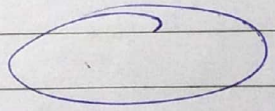
Are there problems that are not solvable in finite time?

How does randomization help algo-design?

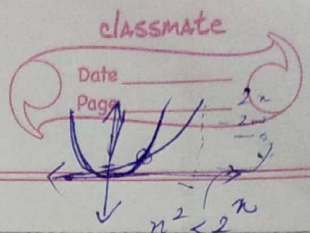
» What is an efficient algorithm

Efficiency + ∈
Given a problem which grows at the rate of $f(n)$
if soln is also $f(n)$, then no use bcoz, it is another language of the same problem

Suppose a class of algos



$f \in$ Class
$g \in$ Class
$f+g \in$ Class . $fg \in$ Class
f is $O(n^{O(1)})$



Turing machine can simulate this within another class.

This is called the P class

$\text{Time}(n^k) = \{L \mid \exists \text{ TM } M \text{ that decide } L \text{ in } O(n^k) \text{ steps}\}$

$P = \bigcup_{k=0}^{\infty} \text{Time}(n^k)$

↳ Poly time Probs

NP Class:→

↳ No solution can be found

↳ But verifiers exist

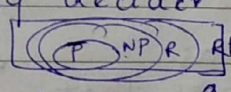
A TM V is a said to be a verifier for language L if the language

$L = \{w \mid V(w, c) = \text{accepts for some string } c\}$

$NP = \{L \mid L \text{ has a Poly-time Verifier}\}$

If a prob has Poly decider ⇒ Poly Verifier

$P \subseteq NP$



NP-Hard: { ∃ a polytime reduction

$\forall L' \in NP \Rightarrow L' \leq_p L$ }

NP-Complete:

↳ Implication: If one prob solved, every other NP can be solved efficiently.

and no-one has solved it yet (:'(

Notes

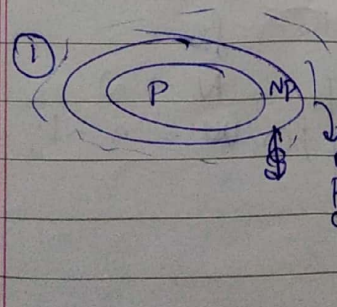
① Poly-time ≤ reduction

$A \leq_P B$ (Polynomial time)

⇒ A lang L is NP complete if

a) $L \in NP$

b) $\forall L' \in NP, L' \leq_p L$

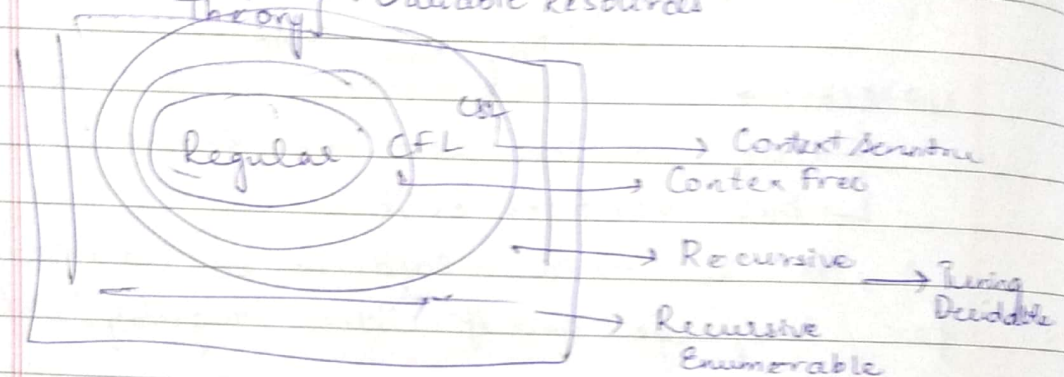


Sec: Notes

20. Automata Theory: Easy Probs → Finite Resources → Decide before knowing input

Computability Theory: Can solve / Can't be solved with ∞ resources

Complexity Theory: Variable Resources



CSL: Linearly Bound Automata
Prob ∝ Inp Size

CFL: ∞ stack memory

RE has probs that don't have a decider
≥ RE can have probs that can't have recogniser too

To prove some probs have no solns ⇒ n(Algos) < n(Probs)

—

Proving equality
⇒ Prove Bijection
⇒ Map it to another fn

eg E → N | N → N
f(n) = n/2 | f(n) = n − 1;
f(even n) = n/2

$f : n \to Z$

$$f(n) = \begin{cases} 0 & \text{if } n = 1 \\ -n/2 & n \text{ even} \\ \frac{n-1}{2} & n \text{ is odd} \end{cases}$$

① RTP: T is countable.

T ↓ Turing Mac = Set of symbols

$T \subseteq |\Delta^*|$ (all combination of symbols)

Show bijection $N \to T$

$f(1) = \epsilon$, $f(2) = 0$, $f(3) = 1$, ...

Each lang is a sequence of 0 & 1

So occurs only once in RHS

and LHS is countable & RHS is unique.

// One-One + Onto

So T is countable

② $S = \{ L / L$ is a subset of $\{0,1\}^* \}$

$\Rightarrow S$ is uncountable

ϵ, 0, 1, 00, 01 ...

$L \subseteq \{0,$

Reductions

If A can be solved & once A is solved ⇒ we can use B to solve A then B is reduced to A

eg: Solving { set of Linear eqns } reduced to { Solving det }

$A \leq B$

↓

SSLE ≤ Det

A is ~~solved~~ reduced to B

If A is solved, I know how to solve B

Mapping Reduction:-

$A \leq_m B$ if ∃ a computable M

$f : \Sigma^* \longrightarrow \Sigma^*$ such that

$\forall w,\ w \in A \iff f(w) \in B$

Computable fn. takes some input X on the tape & returns $f(x)$ onto the tape.

theorem:

i) If $A \leq_m B$ & B is decidable ⇒ A is decidable

ii) $A \leq_m B$ and if A is undecidable then B is undecidable

iii) If $A \leq_m B$ and $B \leq_m C \Rightarrow A \leq_m C$

Theorem:- Halting Prob. is undecidable

$HALT = \{ \langle M, w \rangle \mid M \text{ halts on } w \}$

Meaning: If M halts on w ⇒ M ∈ HALT

Aim: $A_{TM} \leq_M HALT$

Let H be a decider for HALT

To solve/decide $A_{TM} = \{ \langle M, w \rangle : M \text{ accepts } w \}$

On input M, w

Run H(M, w)

If H rejects, reject

If H accepts, run M on w and if M accepts accept, else reject.

→ We can see that A_{TM} is decidable if H is decidable.

Since we know that A_{TM} is undecidable from Theorem (ii)nd point H is undecidable

II $E_{TM} = \{ \langle M \rangle \mid L(M) = \phi \}$

↓

Rejects all inputs

Theorem: E_{TM} is undecidable

Proof: $A_{TM} \leq_M \overline{E_{TM}}$

Let TM, E decide E_{TM}

On input $\langle M, w \rangle$

1> Write a new TM M'

M' on input x

if x ≠ w reject

else x == w, run M on w,

if M accepts reject

else accept.

2> Run E(M')

If E accepts, ACCEPT

If E rejects, REJECT

A_{TM} is solved with E_{TM} ∴ $A_{TM} \leq_m E_{TM}$

∵ A_{TM} is undecidable, E_{TM} is undecidable

f converts A_{TM} to E_{TM} in step (1)

Theorem: $EQ_{TM} = \{\langle M_1, M_2 \rangle \mid L(M_1) = L(M_2)\}$.

Proof: $E_{TM} \leq_M EQ_{TM}$

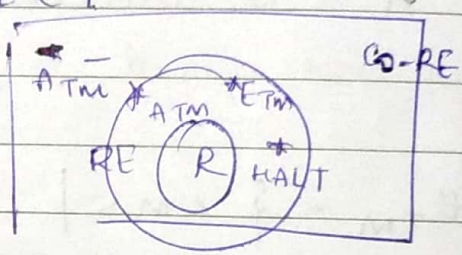
Let Q decide EQ_{TM}

E: On input M → Run $Q(M, M')$

If Q accepts, Accept

else If Q rejects, REJECT

M' on input x, REJECT



Thm: If $L \in RE$ and $\bar{L} \in RE$ then L is decidable.

$L \in RE$: ∃ a TM M so that

∀ $w \in L$, M accepts w

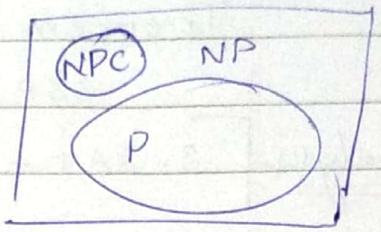
∀ $w \notin L$, M doesnot accept w

$\bar{L} \in RE$: ∃ a TM, T so that

∀ $w \in L$, T accepts w

∀ $w \in \bar{L}$ T doesnot w accept

From the theorem, we can see: If $L \notin R$ and $L \in RE \Rightarrow \bar{L} \notin RE$



- Prob that is verifiable but not decidable
  → Password - Hash
  ↳ can verify a Pw given Hash
  ~~and~~ but u can't find the Pw

Hope Security Doesn't exist

P = NP : No secure Password
it would mean
Bcoz anything efficiently verifiable ⇒
It will be decidable.

A class NP - Complete exists such that
NPC ⊆ NP
If that problem can be solved then any prob in NP can be solved.
L is NP - Complete if: a) L ∈ NP
b) ∀ A ∈ NP, $A \leq_p L$

Cook-Lemma Theorem: SAT is NP complete

SAT = { ϕ / ϕ is satisfiable }
→ SAT's one facet is
Given a Boolean formula
Can atleast 1 row be found that satisfies that formula (i.e Not a contradiction)
→ Another facet is:
Max Clique Problem in
CLIQUE = { <G, k> / G has a clique of size k }

{ If Max Clique can be solved ⇔ Boolean SAT can be solved

Clique = A complete sub-graph

Theorem:

$3SAT \leq_p$ Clique

Analysis $3\text{-}SAT = \{\phi \mid \phi$ is satisfiable$\}$

↳ CNF (each clause has exactly 3 literals)

↳ (Boolean Normal form)

$\phi = (x_4 \vee x_2 \vee \bar{x_3}) \wedge (x_3 \vee x_4 \wedge x_4) \wedge (x_1 \wedge x_5)$

Any NP prob is convertibe to 3-Sat and 3sat is convertible to Clique

$\Rightarrow$ NP ~~is~~ Clique is alsofis ~~NP complete~~ ↳ part (1)

and if Clique $\in$ NP

$\Rightarrow$ Clique is NP - Complete

On input $\langle G, k, c \rangle$

- if G is k-sized clique in G, accept

else reject

Clique is NP beoz naive algo is search for sub graph of size - k $= {}^nC_k$

$= \frac{n!}{(n-k)!\,k!}$

$\approx O$ (exponential)

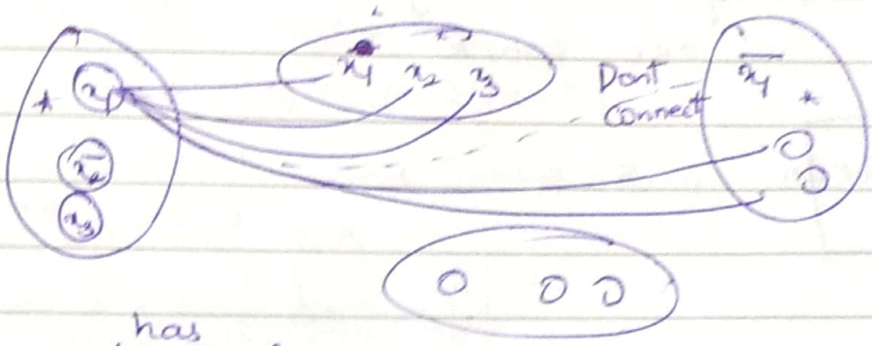
On Input $\langle G, K, C \rangle$

Assuming Clique is solvable, How can we solve 3-SAT ??.

n variables, l clauses

$$\phi = (x_1 \vee x_2 \vee x_3) \wedge (\bar{x}_3 \vee \bar{x}_2 \vee x_4) \wedge (\quad \vee \quad \vee \quad)$$

Each clause is a vertex.



Graph has $\Rightarrow 3l$ vertices.

and $x_i \neq \bar{x}_i$

If a clique of size 'l' exists, then prob is solvable

In each literal choose 1 variable which is true (the first one)

It can't be a clique, if 2 edges are not connected $\Leftrightarrow$ 'Either they belong to same literal

(or)

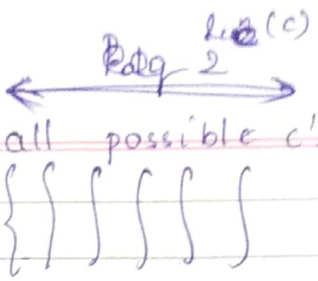
either they are of the form $x_j, \bar{x}_j$

But if $x_j = T \Rightarrow \bar{x}_j = F$

So a clique of size l.

And vice-versa:

Take 1 variable from each clause and assign true to each of that variable and then the Boolean Expression is Satisfiable.

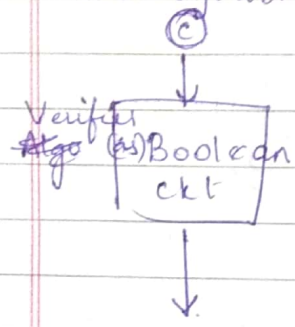


If Poly

Verifier given c ↔ Outputs in poly time

NP: Out of all c's one c works

Satisfiability: Out of all inputs, does one work

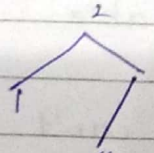


Approximate Algorithms:

Relax the Ending point.

Minimum Vertex Cover:

3-SAT $\leq_p$ VC



{1,3} = VC

2. Approximate efficient algo for Min VC

Size(Approx algo) $\leq$ 2. Size(Exact Algo)

VC ← ϕ

Repeat

↳ pick an Edge (u,v) s.t

u ∉ VC, v ∉ VC

- VC ← VC ∪ {u,v}

Until no edge can be

closer output vc

Verifier:

{G, K, C} → Non Poly

{G, K} → Not Poly, bcoz Count(C) ≈ $^{n}C_{k}$ ≈ Expo

|VC*| is min vertex cover

No. of edges be h

R.T.P |VC| ≤ 2|VC*|

1> |VC| = 2h → Bcoz 2 vertices are added everytime

2> VC* ≥ h

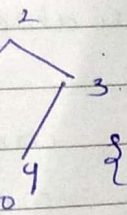
Ways to End-Well:

Proactivization:- Password change 6 month

Pseudo Solution { Solving a hard prob } → { To tell whether solved } → NP

⇓

eg: Pseudo random no generation



{1, 3} = VC

11/11/17

Lec-05

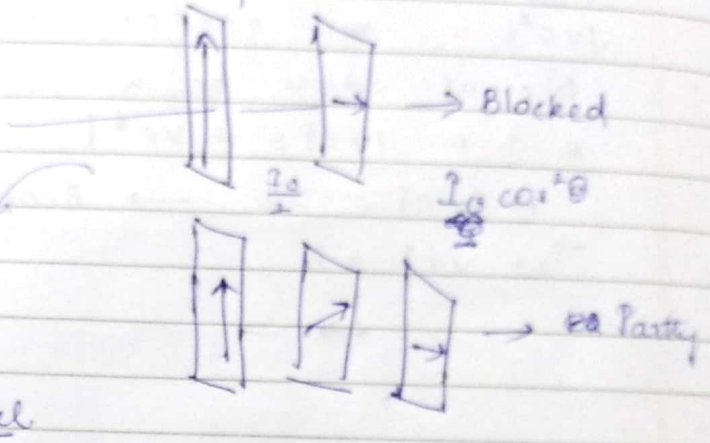
Quantum Algorithms

A Model akin to Schrodinger Eqn

Quantum World

↳ Superposition (Principle)

Photon Polarization Expt:



Qubit model

Works for Both info and Reality

↳ Simplest

Qubit is a vector in $\mathbb{C}^2$

$\psi = a|0\rangle + b|1\rangle$

Probability of 0 is $|a|^2$

Probability of 1 is $|b|^2$

And once you measure, ψ collapses.

Case 1 $\psi = a|\rightarrow\rangle + b|\uparrow\rangle$

when $\uparrow$ is used

$\psi = 0|\rightarrow\rangle + 1|\uparrow\rangle$

Now if $\rightarrow$ is used

Probability is $|0|^2 = 0$

case 2: $\psi = a|\rightarrow\rangle + b|\uparrow\rangle$

when ~~45°~~ $\uparrow$ is used

$\psi = 1|\uparrow\rangle$

when 45° is used

$\psi = \frac{1}{\sqrt{2}}|\uparrow\rangle + \frac{1}{\sqrt{2}}|\nearrow\rangle$

$P\left(\frac{1}{2}\right)$

3. ~~Shor's Algo~~

and when $\psi = \frac{1}{\sqrt{2}}|\nearrow\rangle + \frac{1}{\sqrt{2}}|\nwarrow\rangle$

$|\rightarrow\rangle$ is ~~one~~ used

If (a, b) is there, $[a|0\rangle + b|1\rangle]$ naturally evolves with a unitary matrix

(a', b')

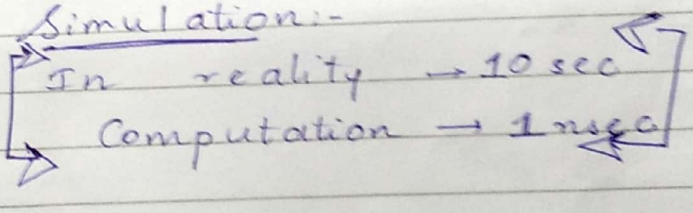
If $(a', b') = U(a, b)$

$U^{\dagger} = U$

Superposition holds for more than 2 comps too.

{2 bits, 3 bits} → Quantum Computers can hold

Simulation:-



In QMech systems: To find what happens in 5 mins, it takes a much longer time. So using a realistic model is better.

What is the use, when the exact value can't be ~~used~~ found

→ Search an unsorted array: $(\sqrt{N})$

→ Prime factorization in Poly time (by Shore) ↓ Algo

Create 2^N numbers and delete all no's except 1 with the given distro.

$N = pq$

If $x^2 \equiv 1 \mod N \rightarrow N \mid x^2 - 1$

and $x \neq 1 \; \gamma (N-1)$

Can be reduced to: Period finding problem.

$a^r \equiv 1 \mod N$ for some even r

$\Rightarrow (a^{r/2})^2 \equiv 1 \mod N$

$x^2 \equiv 1 \mod N$

Take random a and ask for peri