MODERN COMPLEXITY THEORY

- **Complexity Theory**: classify into easy and hard problem
- **Computability Theory**: classify into solvable and unsolvable

Intro to Algorithm: set of instruction that if followed, accomplishes a particular task

→ 0/more input must produce atleast one/more output

→ definiteness: divide by zero is not possible

→ all algorithm must terminate after finite amount of time

→ all algorithm must be feasible (able to solve in poly time)

→ infeasible (exponential time)

✓ IFP: integer factor problem

$n = p \times q$ (p, q: prime)

Space Complexity Amount of memory/space algorithm need to run to completion

fixed part → .a.out file has assembly instruction

variable part → depend on problem being solved

$S(P) = C + S_P$ (space requirement)

C → instance characteristics depend on input

C → constant depend on data structure used

- **Instance characteristics**: special attribute of problem instance being solved
- **Component variable**: vary more based on input data
- **reference variable**: store ref/pointer to data structure
- **Recursion stack space**: depth of recursion / memory required in each recursion call.

Eg: example if we need to store array. numsize: n

$S_{sum}(n) \geq n+3$

curvature stack space also depend on n

Recursion Tree

$R(a, n-1)$

TIME COMPLEXITY

T(P): compile time + runtime (execution)

algorithm

$\vdash$ no. of add n/subs

$tp(n) = c_a \cdot Add(n) + c_s \cdot Sub(n)$ ——

Time needed

(focus on run time)

ASYMPTOTIC NOTATION

$f(n) = O(g(n))$ iff there exist some positive constant c such that $f(n) \le c \cdot g(n)$

upper bound of fn f(n)

$O(n^k) \rightarrow$ polynomial time comp

Theorem can always express in order of highest power

if $f(n) = a_m n^m + a_{m-1} \cdot n^{m-1}$ — —

$\boxed{f(n) = O(n^m)}$

OMEGA (Ω) (lower)

$f(n) = \Omega(g(n)) \Rightarrow$

$f(n) \ge c \cdot g(n) \; \forall \; n \ge n_0$

$\boxed{\text{lower bound of } f(n)}$

Theorem

if $f(n) = a_m n^m + a_{m-1} n^{m-1}$ — —

if $(a_m > 0)$

$f(n) = \Omega(n^m)$

combine $\boxed{O(n) \,\&\, \Omega(n) = \Theta(n)}$

Theta (Θ)

$\boxed{\text{both upper \& lower bound}}$

f(n)

$f(n) = \Theta(g(n))$ iff $\exists (c_1, c_2, n_0) > 0$

when

$c_1 \cdot g(n) \le f(n) \le c_2 \, g(n) \; \forall \; n \ge n_0$

little o(n) (o)

$f(n) = o(g(n))$ iff $\lim_{n \to \infty} \frac{f(n)}{g(n)} = 0$

Notation

Sub-polynomial = $2^{\text{polynomial}}$

Undecidable problem

- no algorithm can solve
- Even turing machine can't solve

Decidable problems

→ An algorithm can be written to solve it

→ corresponding algo may / may not be feasible

→ If can be solved in polynomial : tractable

→ If can be solved in exponential time. intractable

Church-Turing Thesis

- Alonzo Church : λ calculus formal way to define algo
- Alan turing used turing machine

↓

Lme Intuitive notation.

Automata

An automation is self propelled machine which follows predefined sequence of operation automatically

→ an automation with finite number of state is called finite automata (FA) or finite state machine (FSM).

DFA (Deterministic finite Automata)

At each state one can decide the state to which it will move

formal defination : A dfa is a 5 tuple $\langle Q, \Sigma, \delta, q_0, F \rangle$

Q → finite set called states

Σ → Alphabet

δ : $Q \times \Sigma \rightarrow Q$ is transition function

$Q \times \Sigma$ → Cartesian product

$\{(a, b) \mid a \in A \text{ and } b \in B\}$

If $|A| = m$ and $|B| = n$, $|A \times B| = mn$.

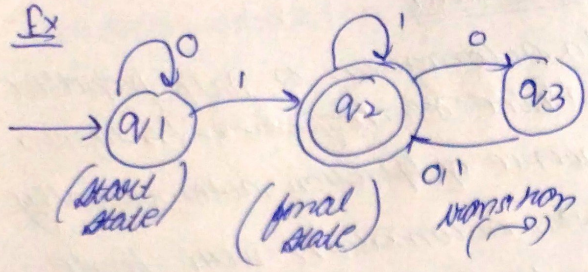
NFA (Non deterministic finite automata)

In NFA for a specific input symbol machine can move forward any combination of state within the machine.

4) $Q_0 \in Q$: start state

5) $F \subseteq Q$ (accepted / final state)

Ex



DFA M_1

$M_1 = (Q, \Sigma, \delta, q_1, F)$

① $Q = \{q_1, q_2, q_3\}$

② $\Sigma = \{0, 1\}$

③ δ:

	0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_2	q_2

④ q_1: start state

⑤ $F: \{q_2\}$ (final state)

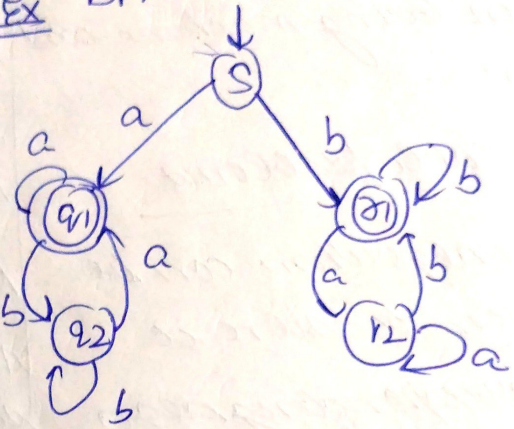
If Input string = 1101 on M_1

- read 1, follow transition from $q_1 \to q_2$
- Read 1, follow transition from $q_2 \to q_2$.
- read 0, follow transition from $q_2 \to q_3$
- read 1 follow transition from $q_3 \to q_2$. accept $q_2 \in F$

A: set of all strings the machine m accepts → A is language of machine m

that $L(m) = A$

Ex DFA M_2



$M_2 = \{Q, \Sigma, \delta, S, F\}$

1) $Q = \{S, q_1, q_2, r_1, r_2\}$

2) $\Sigma = \{a, b\}$

3) $Q \times \Sigma \to Q$

δ	a	b
S	q_1	q_2
q_1	q_1	q_2
q_2	q_1	q_2
r_1	r_2	r_1
r_2	r_2	r_1

$L(M_2) = A = \{a, aa, bb, b, \ldots$

; w | w string that s start and w ka some symbol,

Formal defination of computation

Let $M = (Q, \Sigma, \delta, q, F)$ be a DFA and $w = w_1 w_2 \ldots w_n$ be a string over alphabet Σ.

We call M accepts w if a seq of states $r_0, r_1 \ldots r_n$ exist in Q with following conditions (3).

- ✓ $r_0 = q_0$ (must be starting state of machine)
- ✓ $\delta(r_i, w_{i+1}) = r_{i+1}$
- ✓ $r_n \in F$ (final state)

Regular language: Language is regular if some finite automata recognizes it $L = \{w \mid M \text{ accept } w\}$

Regular Operation

1) Union: Let A and B be langs

$A \cup B = \{x \mid x \in A \text{ or } x \in B\}$

2) Concatenation:

$A \circ B = \{xy \mid x \in A \text{ and } y \in B\}$

3) Star (*): $A^* = \{x_1 x_2 \ldots x_n \mid k \geq 0 \text{ for each } x_i \in A\}$

(empty string also)

Theorem [Union]

Class of regular language is closed under union operation

If A_1, A_2 are regular langs, then $A_1 \cup A_2$ is also reg lang

Proof: Let M_1 recognize A_1, where DFA, $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$

and M_2 recognize A_2, where DFA, $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$

Goal is to construct new deterministic finite automata $M = (Q, \Sigma, \delta, q_0, F)$ such that M should accept w. If either M_1/M_2 accept w

- ✓ $Q = Q_1 \times Q_2 = \{(r_1, r_2) : r_1 \in Q_1, r_2 \in Q_2\}$
- ✓ Σ is same as that of of M_1, M_2
- ✓ δ for M is defined as
- ✓ $\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a))$
- ✓ $\delta : Q \times \Sigma \longrightarrow Q$

  $(Q_1 \times Q_2) \times \Sigma \longrightarrow (Q_1 \times Q_2)$
- ✓ $q_0 = (q_1, q_2)$
- ✓ $F = \{r_1, r_2 \mid r_1 \in F_1 \text{ or } r_2 \in F_2\}$

Determinism

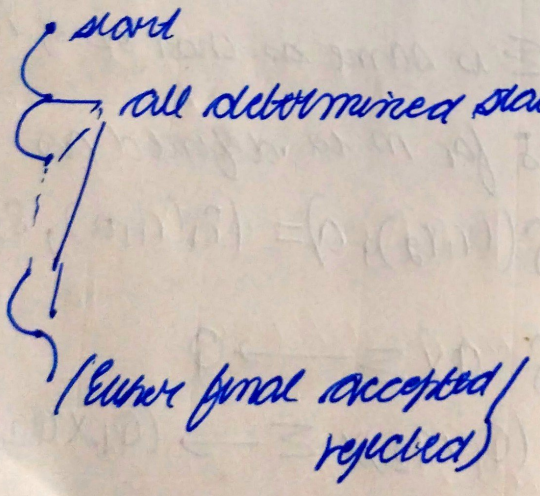
When a machine (DFA) is in a given state and read next symbol from input string we know what the next state will be ∴ determined

Non-determinism

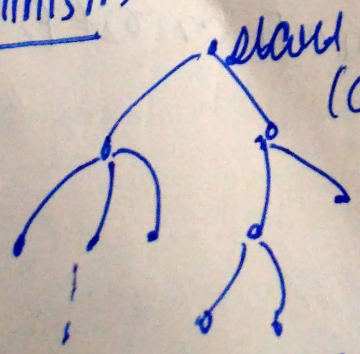
Several choices may exist for next state at any point

(non determinism can be viewed as a kind of parallel computation wherein several processes can be running concurrently

Determinism



determinism



(many choices any accept

If A is set, P(A): power set of A

set of all possible subset of A.

$A = \{a, b\}$

$P(A) = \{\phi, \{a\}, \{b\}, \{a, b\}\}$

$|A| = n \quad P|A| = 2^n$

Formal definition of NFA

A NFA is a 5 tuple $(Q, \Sigma, \delta, q_0, F)$

where

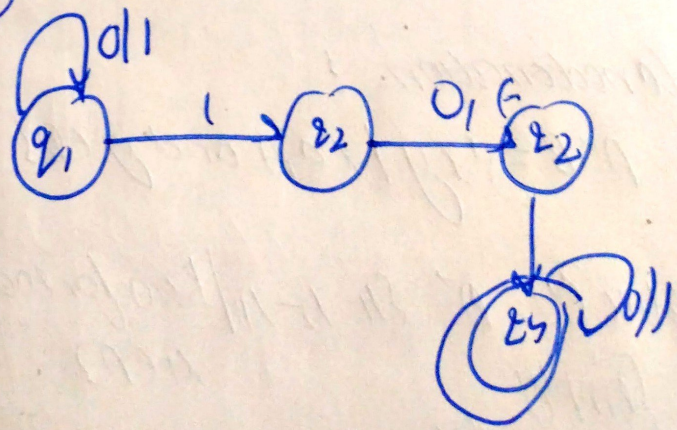
1) Q: finite set of states
2) Σ: finite alphabet
3) $\delta: Q \times \Sigma_\epsilon \to P(Q)$

$\Sigma_\epsilon = \Sigma \cup (\epsilon)$ → empty string

(*)

4) $q_0 \in Q$ start state
5) $F \subseteq Q$ is set of accept state

Problem Consider NFA M as follows



$M = (Q, \Sigma, \delta, q_1, F)$

(q_1, q_2, q_3, q_4)

$(0, 1)$

	0	1	ϵ
q_1	$\{q_1\}$	$\{q_1, q_2\}$	ϕ
q_2	$\{q_3\}$	ϕ	$\{q_3\}$
q_3	ϕ	$\{q_4\}$	ϕ
q_4	$\{q_4\}$	$\{q_4\}$	ϕ

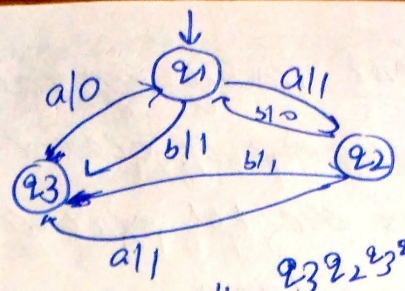
Formal defination of Computation

Let $N = (Q, \Sigma, \delta, q_0, F)$ be an NFA and w be a string over Σ. We say then N accept w if we can write $w = y_1 \ldots y_m$ where each $y_i \in \Sigma_\epsilon$ and sequence of state $r_1 \ldots r_m \in Q$ with

1) $r_0 = q_0$
2) $\delta(r_i, y_{i+1})$ for some $i \in 0, 1, \ldots N-1$

Theorem Every nondeterministic finite automata has equivalent DFA.
(that are branch is afc)

- Finite State Transducer (FST) is type of deterministic finite automata whose output is a string not just accept / reject



say input "abaa": $q_3 q_2 q_3 q_2 / 1110$

formal defination of FST

A FST is a ~~simple~~ 5 tuple $(Q, \Sigma, \Gamma, \delta, q_0)$ where

- Q → state
- Σ → input alphabet
- Γ → output alphabet
- q_0 → Initial state
- δ → $Q \times \Sigma \longrightarrow Q \times \Gamma$

We say that a FST $M = (Q, \Sigma, \Gamma, \delta, q_0)$ accept an input string $w = w_1 w_2 \ldots$ over Σ if it output string $v = v_1 v_2 \ldots v_n$ and sequence of state $r_0, r_1 \ldots r_n \in Q$ with following conditions

1) $r_0 = q_0$
2) $\delta(r_i, w_{i+1}) = (r_{i+1}, v_{i+1})$
$(i = 0, 1, 2 \ldots n-1)$

Theorem Class of regular languages is closed under the '.' operation

$A \circ B = \{xy \mid x \in A \text{ and } y \in B\}$

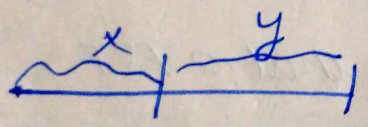
proof:

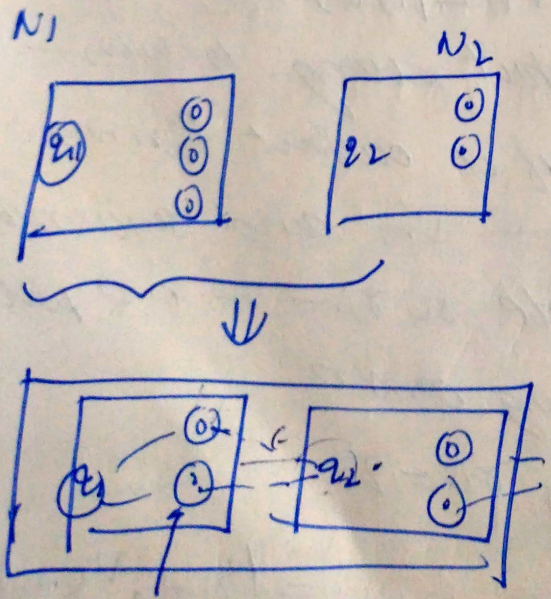
let $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognizes A_1 (NFA).

$M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognizes A_2

Goal is to construct NFA $M = (Q, \Sigma, \delta, q_0, F)$ to recognize $A_1 \circ A_2$.

$w = xy$





these then become dead not final state

1) $Q = Q_1 \cup Q_2$

2) State q_1 is same as start state of M_1 $\boxed{q_0 = q_1}$

3) accept state are same as accept state of M_2 $F = F_2$

4) transition function is defined as $\delta : Q \times \Sigma_\epsilon \to \mathcal{P}$ for any state $q \in Q$ and $a \in \Sigma_\epsilon$

~~$\delta(q,a)$~~:

$\delta_1(q,a)$ if $q \in Q_1$ and $q \notin F_1$

$\delta_1(q,a)$ if $q \in F_1$ and $a \neq \epsilon$

$\delta_1(q,a) \cup \{q_2\}$ if $q \in F_1$ and $a = \epsilon$

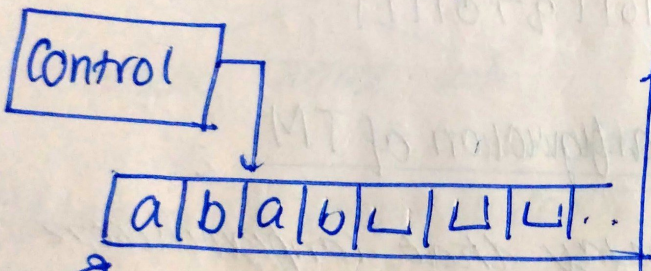
$\delta_2(q,a)$ if $q \in Q_2$ ~~and $a \in \Sigma$~~

Lecture 4 (12 Aug)

TURING MACHINE:

Turing machine is similar to finite automaton but with an unlimited and unobstructed memory which is accurate model

- Turing machine can do almost everything a real computer can do
- Infinite tape is used as an unlimited memory



Input string + blanks

✓ Information to stored is written onto tape

- output accept/reject are obtained by entering designated accepting/rejecting state

↓ if it doesn't enter accept/reject

It will go forever no halt

Difference between finite automata & Turing machine

1) A turing machine can both write and read from tape
2) Read-write head can move both to left and right
3) tape is infinite
4) Special state for rejecting & accepting take effect immediately.

Formal defination of deterministic Turing machine

A deterministic turing machine is 7 tuple $(\underbrace{Q, \Sigma, \Gamma}_{\text{finite sets}}, \delta, q_0, q_{accept}, q_{reject})$

- Q: set of states
- Σ: input alphabet not containing blank symbol (⊔)
- Γ: tape alphabet
- q_0: start state
- $q_{accept} \in Q$: accept state
- $q_{reject} \in Q$: reject state

$$\boxed{q_{accept} \neq q_{reject}}$$

S_1 (a) → S_2, (T)

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

one step to next

so $\delta(q_1 a) \longrightarrow (r, b, L)$

machine write symbol b replacing a, goes to state r. L/R are basically the movement of head

→ Computation by deterministic TM on input string $w = w_1 w_2 \ldots w_n \in \Sigma^*$

$M = \langle Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject} \rangle$

basically M receive input of $w_1 \ldots w_n$ on leftmost n square of tape and rest tape is blank

head start on left most square, since Σ does not take any blank symbol so first blank appear on tape marks end of input

→ once m start, computation proceeds according to transition function δ

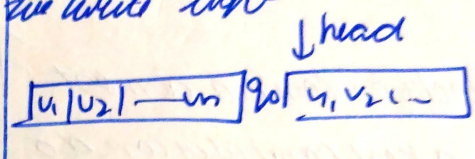
If M tries to move head to left on it stay on same state.

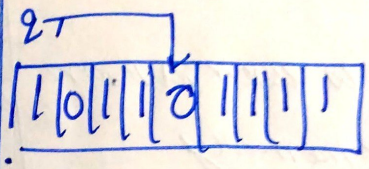
computation continue till either accept/reject.

→ There are three item defining configuration of TM

current state, current head, current tape content

→ for a state q two string u and v over tape alphabet Γ we write uqv





1011 q_7 01111

Configuration of TM

we say that configuration C_1 yields another configuration C_2 if turing machine can legally go from C_1 to C_2 in single step

say $a, b, c \in \Gamma$ and $u, v \in \Gamma^*$

$\delta(q_i, b) = (q_j, c, L)$

$uaq_ibv \rightarrow uq_jacv$

start configuration of M on input w : $q_0 w$

start state : q_0 with head at left most

A deterministic TM M accept input w if sequence of conf $C_1 — C_K$

start accept

A TM accept collection of string called language of M

$L(M) = \{w \mid M \text{ accept } w\}$

if recognized : Turing recognizable

accept reject lost

Language L is turing decidable if some turing machine decide it

Language $L = \{a^i b^j c^k \mid i \times j = k, i,j,k \geq 1\}$

design a TM that decide L

this is how it goes

- scan input from l → r to determine if its member of a^i, b^j, c^k and reject if not
- move head to left end of tape and scan to right till blank occurs.

Non Deterministic Turing machine

At any point machine may proceed acc to several possibilities

Transition function of NDTM

$$\delta : Q \times \Gamma = P(Q \times \Gamma \times \{L, R\})$$

Theorem Every non deterministic TM has an Equivalent deterministic TM.

Language is turing recognizable if and only if some non deterministic TM recognizes it

blem : (DTM): Let language A $A = \{0^k 1^k \mid k \geq 0\}$. Design TM decide it.

① Scan across tape reject if 0 is found to right of 1.

② Repeat if both 0's and 1's remain on tape
: Scan across tape crossing single 0, single 1

③ finally if 0's remain when all 1 are crossed or 1's remain after crossing all 0's → reject, none remain: accept

Searching Problem

✓ Deterministic Algorithm: Given set of Integer (n element) $A[1 \ldots n]$ and a search Element x.

Goal: Check if x is present in A or not.

Algorithm Input: $A[\,]$, x

Output: index j such $A[j] = x$

Steps for $j = 1$ to n
    if $A[j] = x$
        write (j)
        return success

·) Non deterministic algorithm

- choice(s) arbitrarily chooses one of element of given set S.
- failure(): Signal unsuccessful comp
- success(): Signal successful comp

·) Nondeterministic search

Input : A[1..n], x
output : Index j such that A[j]=x

Steps 1: Selecting index using f(n)

Choice

j=choice(1,..n) S=[1,n]

if A[j]= x, then
write(j)
succes().

write(0) failure } would come here only if unsuccessful

O(1)

[Kaise si bhai]

TIME COMPLEXITY

decision problem : any problem for which output is either true(1) or false(0) : decision problem

An algorithm for a decision problem: decision algorithm

Time Classes

Let $t: N \to N$ be a function where N is set of natural no

define

·) time(t(n)) := {L | L is a lang or problem decided by an O(t(n)) - time deterministic TM}

·) ntime(t(n)): {L | L is ... O(t(n)): time NTM}

The Class P

P:= {L | L is a language | problem decided by polytime DTM}

$\bigcup_{k \in N} TIME(n^k)$: is the class of (set of natural numbers)

all language that are quickly decided on DTM.

P is class of all practically decidable | solvable problem

P is class of language that are decidable in polynomial time by deterministic single tape TM.

$P = \bigcup_k TIME(n^k)$

Graphs

$G = (V, E)$

$\langle G \rangle$: encoding of a graph
- adjacency matrix
- linked list

Let G be a directed/undirected graph. Then adjacency matrix or linked list representation $\langle G \rangle$ is a reasonable encoding of graph G.

Let us define a formal problem

PATH := $\{\langle G, s, t \rangle \mid$ G is directed graph with path from vertex node s → vertex t $\}$

Theorem: PATH is in P, PATH ∈ P

(we need to design a DTM that decides it in polytime)

Proof: construction of poly time DTM

Input: $\langle G, s, t \rangle$

accept if there is path

reject. otherwise

Algo

1) Place a mark on node vertex s.

2) Repeat following steps until one fails to mark additional node

3) Scan all edges, if there a edge $(u, v) \in E(G)$ (edge set) from marked u to unmarked v then mark v

4) If node T marked (path v / path x) unmarked

Complexity TC: $O(m^2)$ — P

$O(m)$

Theorem P is closed under ~~UA~~, union, concatenation, complement, ~~*~~

P is are closed under union

for any two P languages d_1 and d_2 and let m_1, m_2 be corresponding DTM

Lec 5

Theorem P is closed under concatenation

Let we have two P languages L_1 and L_2 and let M_1 and M_2 be the dtm that decide L_1 and L_2. In polytime.

Construction to construct a dtm M' that decides concatenation of L_1 L_2 in polynomice time

M' = string of form $w = w_1 \ldots w_n$

✓ on each way split w into w = xy

✓ simulate M_1 on x and M_2 on y

if both accept "accept"

✓ if w is rejected after all possible split repeat

$T(n) = 1\cdot(n-1) + 2(n-2) + 3\cdot(n-3)$

$\sim O(n^2)$ - -

Theorem P is closed under (*).

Proof let $A \in P$ and M be dtm deciding A in polynomial time

Construct a dtm to decide A^*

using divide & conquer + dynamic prog

S.T. on input $w = w_1 w_2 \ldots w_n$

if $w = \epsilon$ (empty string), "accept"

- Initialize $T[i,j] = 0$ for $1 \le i \le j \le n$
- for $i = 1 \ldots n$ do
  - set $T[i,i] = 1$ if $w_i \in A$
- // test each substring of length l
- for $l = 2$ to n // here l is length of substring
- for $i = 1$ to $i = n-l+1$ // possible start of substring of length l
- now $j = i+l-1$ // this is end position
- if $[w_i w_{i+1} \ldots w_j] \in A$ set $T(i,j) = 1$
- → from $k = i$ to $k = j-1$ // try splitting
- → now if $T[i,k] = 1$ and $T[k+1, j] = 1$ ✓ set $T(i,j) = 1$

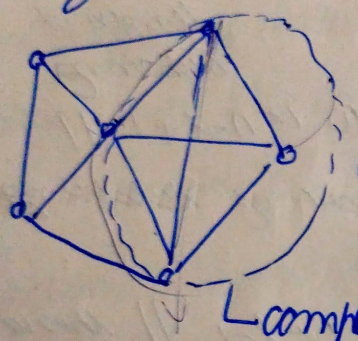
if $T[1][n] = 1$ accept otherwise reject !!

$TC(O(n^3))$

Clique: A k-clique in a (directed) undirected graph. $G=(V,E)$

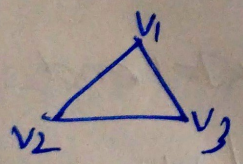
A subgraph G' of G which is isomorphic to the complete graph

($k>0$ is a positive integer = size of clique).



complete subgraph hence clique: G' size = 4 = k

Problem A triangle in an undirected graph $G=(V,E)$ is a 3-clique



define:

TRIANGLE: $\{\langle G\rangle \mid G$ contains a triangle$\}$

TRIANGLE $\in$ P

Goal: we need to construct a polytime DTM M that decide triangle.

M: Input $\langle G\rangle$ where G is undirected graph

1) for each triple of vertices (v_1, v_2, v_3) in G' do

2) If edges $(v_1,v_2), (v_2,v_3), (v_1,v_3)$ are all edges of original graph G then "accept".

3) no triplet found reject

Time Complexity

total number of ways to pick (v_1, v_2, v_3) in G' : $m C_3$

$\sim O(m^3)$

PROBLEM Let G be a connected undirected graph and define

Connected = $\{\langle G\rangle \mid G$ is connected undirected graph$\}$

Connected $\in$ P

Proof Undirected Graph G is connected if every node can

be reached from every other node
We construct polytime DTM which
decides this.

The class NP

(multiple branch of computations)

$NP = \{L \mid L$ is a language decided by a polytime NTM$\}$

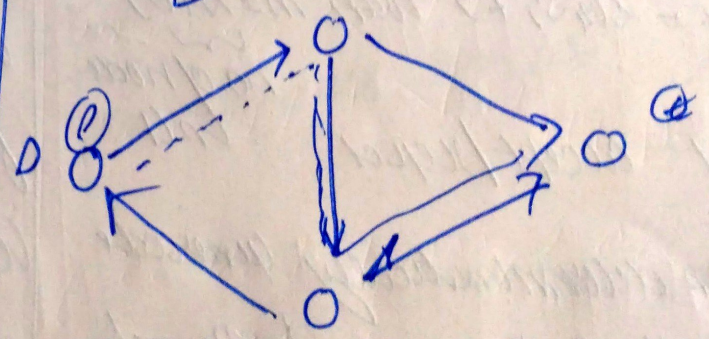
$\bigcup_{k \in \mathbb{N}} NTIME(n^k)$

is class of language that can be decided (verified) quickly on dtm.

PROBLEM

$HAMPATH = \{\langle G, s, t \rangle \mid$ there is directed hamiltonian path from vertex/node "s" to node "t" in G.

→ an s-t hamiltonian path is an s-t path that goes through each vertex of G exactly once



DTM, M: on input $\langle G, s, t \rangle$:

1) Construct all possible s to t paths in G

-) check if any s-t path is hamiltonian
↳ if : accept
↳ else : reject.

Time Complexity : Exponential
(may be many paths than verifying takes time)

HAMPATH ∈ EXP, Exp class is set of all languages L such that L is decided by DTM.

$\bigcup_{k \in N} Time(n^{n^k})$

Theorem $\boxed{\text{HAMPATH} \in NP}$

construction of NTM for hampath

Input = ⟨G, s, t⟩ with $m = n(G)$ (no of nodes in G)

output: accept/reject

→ non deterministically generate m nodes a sequence $p_1, p_2 \ldots p_m$

each p_i chosen from $\{1, 2, \ldots m\}$

$O(n)$

NP problem have poly time verifier

2) if $p_1 \neq s$ or $p_m \neq t$ reject
3) if there is repetition in $p_1 \ldots p_m$ reject.
4) if for some $i = 1 \ldots m$, edges (p_i, p_{i+1}) are not edges Reject
5) otherwise accept

$T(C) = O(m) + O(m) + \boxed{O(m^2)}$ ③
↳ $\neq$.

to solve problem in NP we consider NM
(note that we didn't generate permutation)

Verifier of language L

Let L be a language. A verifier for L is a DTM say V such that

$L = \{\alpha \mid \langle \alpha, \beta \rangle$ is accepted by V for some string $\beta\}$

for every $\alpha \in L$ there exist a certificate β such that V accepts $\langle \alpha, \beta \rangle$

If (V) runs in poly time w.r.t size of input x
poly time verifier of L

Theorem
has p
proof
given
that
de
un
acc
wo
be
Al
Sr
ou

Theorem Let $L \in NP$, such that L has polytime verifier

Proof [Necessary condition]

Given that $L \in NP$, let N be a NTM that decides L in polynomial time.

Let $x \in L$.

There is branch that leads to acceptance of x, certificate B would be prescription of that branch

Claim: A polytime verifier $V(x)$.

Input: $\langle x, B \rangle$

output: accept / reject

→ Simulate N on x with choice of transition, prescribed deterministically by B.

→ If N accept on branch of comput: accept / reject

Construct NTM N

Input = x

output: accept / reject

Steps

- non deterministically generate string of length $\leq n^k$.
- Simulate verifier B on input state x
- If any branch accept: accept else reject

B_1: run V on $\langle x, B_1 \rangle$. If accept "accept".

B_2: run V on $\langle x, B_2 \rangle$ If accept "accept"

x: determined by NTM

$\bigcup_{k \in \mathbb{N}} TIME(n^k)$ generating all branches } NP

Vice Versa:

Let L have polytime verifier V.

assume L runs $\leq n^k$ steps for some

Theorem CLIQUE ∈ NP

Proof polytime verifier

verifier V.

Input $\langle\langle G, k\rangle, T\rangle$

↓

where $T \subseteq V(G)$ is certificate. $\langle G, k\rangle$

output accept / reject

steps 1) if T doesnot contain k nodes "reject"

2) if G doesnot contain (u, v) where $(u,v) \in T$, reject else.

3) accept.

0) [polytime NTM N]

Input: $\langle G, k\rangle$

output: Accept / reject

steps

V nondeterministically generate a k subset T of V(G)

next

accept

conclude P, NP

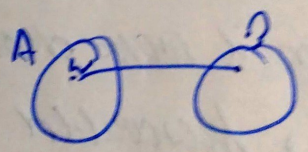
P ⊆ NP

- **polynomial Time Reduction ($\leq_p$)**

language A is polynomial time mapping reducible to language B denoted by

$\boxed{A \leq_p B}$, if a polynomial time computable fn. $f: \Sigma^* \to \Sigma^*$ exists where for every input $w \in A$ if and only if $f(w) \in B$.

function f is called polynomial reduction from $A \to B$.



If d_1, d_2, d_3 be problem

If $d_1 \leq_p d_2$, $d_2 \leq_p d_3 \Rightarrow d_1 \leq_p d_3$

problem if $A \leq_p B$, and $B \in P$ then $A \in P$

proof: Let M be the polytime DTM

deciding B. $f: \Sigma^* \to \Sigma^*$ is a polytime computable fn?

Cor: $A \leq_p B$, if $w \in A$ then $f(w) \in B$.

Construction of DTM N for A

: Non input string W.

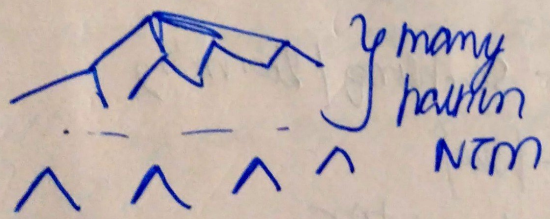
✓ compute f(w)

✓ then N on f(w) ∈ B and output whatever M output

✓ then M accept ~~whatever~~ f(w) when w ∈ A

A ∈ P

Equivalence of NTM & DTM



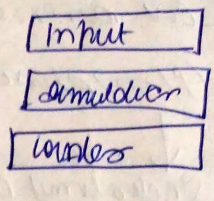
dfs: may go down to infinite path

bfs: ✓

b: max no of children / max successors comb

3 tape DTM will be used

→ initially tape 1 has input, tape 2,3 empty



✓ copy tape 1 to tape 2

✓ simulate NTM in tape 2. Counter guides you path/choices.

✓ replace tape 3 with next lexicographically shortest string

✓ all path in certain level reject
↳ reject, atleast one continue continue.

hence DTM → NTM

Diagonalization

(1) (0,1) is uncountable

(2) There are computational problem that machine can't solve

(3) Program Equality is unrecognized

(4) halting problem is undecidable

(5) for every ε > 0 and +ve integer k there exist problem that are decidable in $O(n^{1+\varepsilon})$ but can't be decidable in $O(n^k)$

(V) for (almost) all function $f(n)$ $\exists$ problem that can be decided in $O(f(n))$ space but cannot be decided in $o(f(n))$ space.

upper limit on amount of space

strictly limit algo will use less space

Given a function $f: N \to N$ we define the set of functions $O(f(n))$ as

$$O(f(n)) = \{ g(n) \mid \exists n_0 \in N, \exists c \text{ st } \forall n \geq n_0 \; g(n) \leq c f(n) \}$$

$O(f(n))$: upper limit of how much memory algo uses as n grows

$o f(n)$ (strictly limit)

or function $f: N \to N$, $o f(n)$

$$f(n) \{ g(n) \mid \lim_{n \to \infty} \frac{g(n)}{f(n)} \to 0 \}$$

(II) Relativization

(III) Reduction

(IV) Diagonalization

(V) ~~Rational~~ natural proof

Proof

① Godel ~~fn~~ recursive function theory (functional programming)

② church Grammer (parsing) Problem

③ Turing machine model

④ λ calculus

resource of computation

- Time
- Space
- Energy
- message / communication
- Sampling / learning
- Proof
- description
- random - design
- uniform - access

Problem: basically isomorphic graph are one that have same number of vertices, edge and same edge connectivity

Problem: we call two graphs G and H be isomorphic if node of G may be reorder so it is identical to H.

. ~~Claim~~ Let ISO = [<G,H> / G and H are isomorphic graphs]

Claim ISO ∈ NP

- G and H are isomorphic

⇒ ∃ a bijection $f: G \to H$ such that f respect adjacency property in two graph $|G| = |H|$

Let us construct non deterministic algorithm:

"On input <G,H>, where G and H are undirected graph,

1.) Let m be the no. of nodes (vertices) of G and H, if $|G| \neq |H| = m$ then reject

2.) Non deterministicly select permutation π of element {1 .. m}

3) for each pair of node (x,y) in G check that $(\pi(x), \pi(y))$ is in H. If all agree "accept", if differ reject!

$S = \{1, 2 \ldots m\}$ (x, y)

$\downarrow \downarrow$

2 5

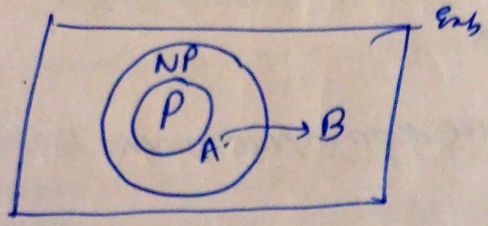
$\pi(S) = \{2, 5 \ldots\ldots 1\}$

Time complexity: $O(m!)$

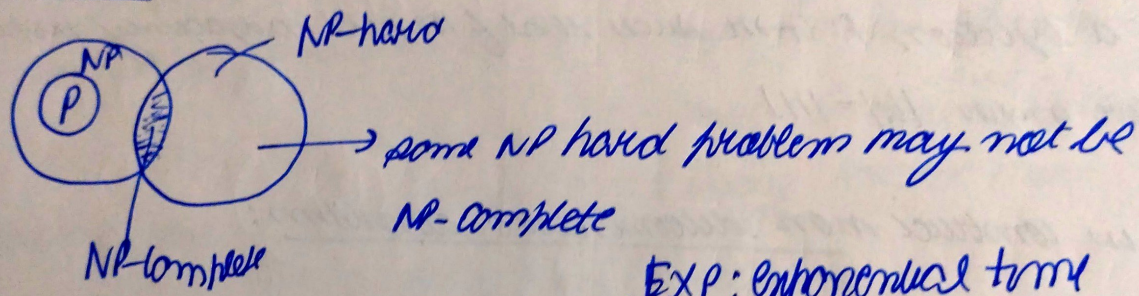
Explanation. basically if boarding kitkam delhi the G, meant; others to G2 meu hrta?

NP-hard

A problem B is called NP hard if every problem $A \in NP$ is poly time reducible to B, $A \leq_p B \ \forall A \in NP$ (atleast as hard a NP problem)



NP Complete



A problem B is called $\mathcal{C}$ complete if

- $B \in \mathcal{C}$
- B is $\mathcal{C}$ hard, $A \leq_p B \ \forall A \in \mathcal{C}$

$$NP = \bigcup_{K \in N} NTIME(n^K)$$

$$P = \bigcup_{K \in N} TIME(n^K)$$

Co-NP = Complement of NP-problems

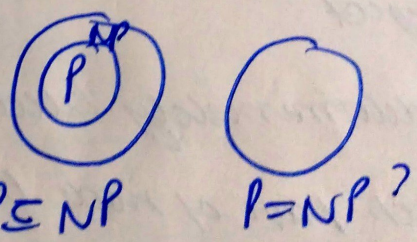
$\{\bar{\alpha} \mid \bar{\alpha} \in NP\}$

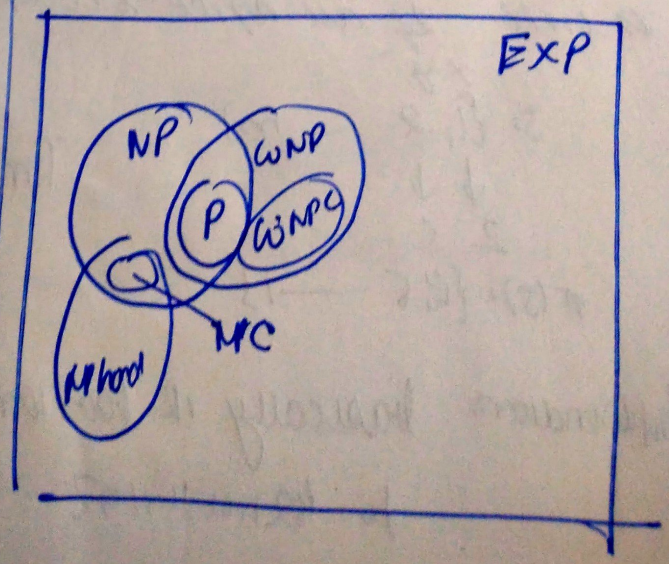
NPC: NP-complete

CONPC: CoNP-complete

EXP: exponential time

$$\boxed{\bigcup_{K \in N} Time(n^{2^K})}$$





Theorem If B is NP Complete and $B \leq_p C$ $\forall C \in NP$. Then C is NP-Complete

Proof: Given that $C \in NP$, we need to show that C is NP Complete. We must show C is NP hard i.e $A \leq_p C, \forall A \in NP$

Given B is NP-complete

$A \leq_p B \quad \forall A \in NP$ — (1)

$B \leq_p C$ — (2)

$\boxed{A \leq_p C}$ hence C is NP-hard

SAT: Boolean Satisfiability Problem

Let $x_1, x_2, \ldots$ — denote boolean variable (their value is either true (1) or false (0))

Let $\bar{x}$ denote negation of x_i.

A literal would either be boolean variable (x_i) or the negation ($\bar{x_i}$)

A boolean formula ~~is~~ the propositional calculus is the expression that can be constructed using literal and operations: and ($\wedge$) or ($\vee$).

for eg: $\varphi = \varphi(x_1, x_2, x_3) = (x_1 \vee \bar{x_2}) \wedge x_3$ is boolean formula

$\varphi = \varphi(x_1, x_2 \ldots x_n)$ no of comp : 2^n

Define $SAT = \{\langle\varphi\rangle \mid \varphi$ is satisfiable$\}$

Theorem $SAT \in NP$.

Construction of poly time verifier

Input: $\langle\langle\varphi\rangle, \beta\rangle$ — certificate of φ

output: accept, if φ is satisfiable otherwise reject

Let $\varphi = \varphi(x_1, x_2 \ldots x_n)$ be boolean formula

Algo

(i) If β does not contain n boolean variable assignment then reject

(ii) Evaluate φ on β

(iii) If value of $\varphi = 1$ accept, otherwise reject

CNF-SAT (and of clauses, or of literals)

A formula φ is in conjunctive normal form iff it can be expressed as

$$\varphi = \bigwedge_{i=1}^{k} C_i = C_1 \wedge C_2 \text{ — } C_k$$

C_i → clauses represented as

$C_i = \vee l_{ij}$ → literals.

Eg $\varphi = (x_3 \vee \bar{x}_4) \wedge (x_1 \vee \bar{x}_2)$ is in CNF

$(x_1=0, x_2=0, x_3=0, x_4=0)$ is satisfying assignment for φ

define

CNF-SAT = $\{\langle\varphi\rangle \mid \varphi$ is a satisfying CNF formula$\}$

Theorem CNF-SAT $\in$ NP

formula φ is in disjunctive normal form if it can be expressed,

$$= \bigvee_{i=1}^{k} C_i = C_1 \vee C_2 \vee \text{ — } C_k$$

for example

$\varphi = (x_1 \wedge x_2) \vee (x_3 \wedge \bar{x}_4)$ — is DNF

define (or of clauses, and of literals)

DNF-Sat = $\{\langle\varphi\rangle \mid \varphi$ is satisfying DNF formula$\}$

Theorem DNF-SAT in $\boxed{P}$

3-CNF

A 3-CNF formula is a CNF formula in which clause C_i has exactly three literals

Example

$\varphi = (x_1 \vee \bar{x}_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee \bar{x}_4 \vee \bar{x}_1)$ is 3 CNF

define

3-SAT = $\{\langle\varphi\rangle \mid \varphi$ is satisfiable 3-CNF problem$\}$

$\in$ NP

Theorem [Cook-Levin]: SAT is NP-Complete

To prove that SAT is NP-Complete we must demonstrate following two things

1) $SAT \in NP$
2) Every problem $A \in NP$ is polytime reducible to SAT ie $A \leq_p SAT \; \forall A \in NP$

[SAT is NP-hard]

Claim: SAT is NP-hard.

Let $A \in NP$, $\exists$ NTM say N which decide A is polynomial time.

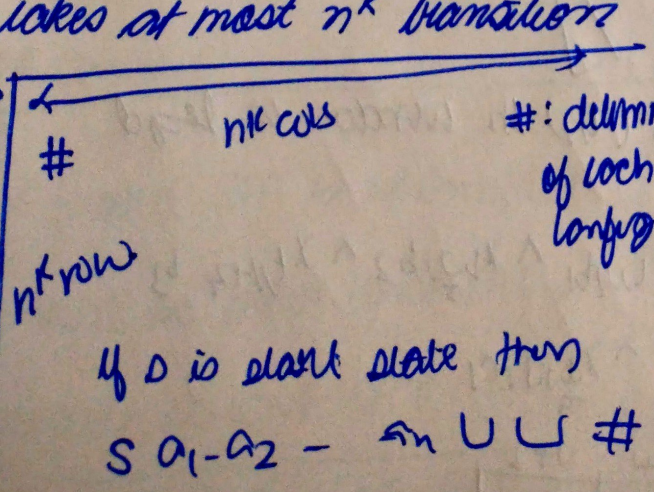
Let $\alpha \in A$ be input string for N,

Goal: Construct boolean formula ϕ from description of N and α such that

ϕ is satisfiable $\iff \alpha \in \alpha(N)$

Let $\alpha = a_1 a_2 \ldots a_n$ be input string of string $n = |\alpha|$

for every string α of size n, N takes at most n^k transition



If s is start state then

$s\, a_1 - a_2 - a_n \sqcup \sqcup \#$

$\square$ (y): define boolean variable of the cell

table is $n^k \times n^k$ table.

	1	2	3	n^k cols 4	5	6	7	8	n^k
1	#	s	a_1	a_2	a_3			⊔⊔	#
2	#	b	q	a_2	a_3	-			#
3	#								#
4	#								#
									#
n^k	#								

(since you have already assumed α is accepted by TM N hence you consider accepting n^k branches)

Define: Q is set of states of N

Γ: tape alphabet for N

F: set of final state of N

C: set of configuration.

$Q \cup \Gamma \cup \{\#\}$

for each cell location (i,j) in the tableau for each $c \in C$ define boolean variable $x_{i,j,c}$ such that

$$x_{i,j,c} = \begin{cases} 1 & \text{if cell } (i,j) \text{ contains } c \\ 0 & \text{otherwise} \end{cases}$$

$x_{1,1,\#} = 1$ (since cell has #)

CNF-SAT $\in NP$: C/F#3 ?

configuration

$1 \le i \le n^k$
$1 \le j \le n^k$

of variable is

$c = |n^k| |n^k| |\Gamma|$ ↳ constant

$= |n^{2k}| = O(n^{2k})$

We have boolean variable, we will now construct boolean formula.

define boolean formula ϕ as:

$\phi = \phi_{start} \wedge \phi_{cell} \wedge \phi_{accept} \wedge \phi_{move}$

ϕ_{start}: $x_{1,1,r\#} \wedge x_{1,2,s} \wedge x_{1,3,a_1} \wedge x_{1,4,a_2}$ — — — — $x_{1,n+1,a_{n+1}}$ — $x_{1,n+3,\sqcup} \wedge x_{1,n^k,\#}$.

↓ start config

ϕ_{cell}: true when atleast one variable is turned on and almost one variable is turned on: exactly / turned on

$$\bigwedge_{1 \le i,j \le n^k} \left[\left(\bigvee_{c \in C} x_{i,j,c} \right) \wedge \left(\bigwedge_{\substack{c,d \in C \\ c \ne d}} \left(\overline{x}_{i,j,c} \vee \overline{x}_{i,j,d} \right) \right) \right]$$

$$= \bigvee_{\delta \in \Gamma} x_{i,j,\delta}$$

$\forall i,j \quad 1 \le i,j \le n^k$

ϕ_{move}: is derived from all the legal 2x3 window in which we can window legal if it is part of two successive rows such that lower row is derived from upper row using one of M's transition.

legal:

a	q_1	b
q_2	b	c

$\delta(q_1, a) \to \{(q_2, b, L)\}$

Illegal:

b	q_1	b
q_2	b	q_1

&

ϕ_{move}: and of all legal window

$$\phi_{move} = \bigwedge_{\substack{1 \le i < n^k - 1 \\ 1 \le j \le n^k - 2}} \left(\text{Num}(i,j)^{th} \text{ window} \right)$$

↳ legal window

where (i,j) th window is legal

$$\bigvee \left(x_{i,j,b_1} \wedge x_{i,j+1,b_2} \wedge x_{i,j+2,b_3} \wedge x_{i+1,j,c_1} \cdots \right)$$

	j	j+1	j+2
i	b_1	b_2	b_3
i+1	c_1	c_2	c_3

$|\varphi| = |\varphi_{accept}| + |\varphi_{cell}| + |\varphi_{accept}| + |\varphi_{move}|$

$O(n^{2k}) + O(n^{2k}) + O(n^{2k}) + O(n^{2k})$

$= \underline{\underline{O(n^{2k})}}$ size of φ

φ is satisfiable if and only if $\in L(N)$

$\forall A \in NP$
$A \leq_p SAT$

Corollary! 3SAT is NP Complete

$\boxed{\text{CNF-SAT}}$

CNF-SAT $= \{\langle\varphi\rangle \mid \varphi$ is satisfiable CNF formula$\}$

If B is NP Complete and B is polytime reducible to C, C is NP: C is NP complete

We found SAT is NP Complete

We need to prove that SAT is reducible to CNF-SAT.

$\boxed{SAT \leq_p CNF\text{-}SAT}$

Convert boolean formula φ into CNF-formula

$\varphi = \underbrace{\varphi_{accept}}_{\text{CNF form formula}} \wedge \underbrace{\varphi_{cell}}_{\text{CNF}} \wedge \underbrace{\varphi_{accept}}_{\text{CNF form formula}} \wedge \varphi_{move}$

for φ_{move} we would use distributive formula

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

can convert (1,2) larger formula in CNF formula

φ_1: CNF

hence φ is a CNF!!!

this reduction to CNF takes an polynomial time only φ_{move} to reduce

now SAT $\rightarrow$ 3SAT in polynomial time

CNF-SAT $\leq_p$ 3SAT: Test new

Convert boolean formula φ into 3 CNF-SAT problem.

Use the rules

1) only one literal → three

$x = x \vee x \vee x$

2) $x \vee \bar{y} = x \vee x \vee \bar{y} = x \vee \bar{y} \vee \bar{y}$

We dont have any case with clause that has exactly three literal!

When number of literals > 4 then

$\underbrace{(x_1 \vee x_2 \vee x_3 \cdots \vee x_k)}$

Given clause containing k literal is satisfiable

$(x_1 \vee x_2 \vee z_1) \wedge (\bar{z}_1 \vee x_3 \vee z_2) \wedge (\bar{z}_2 \vee x_4 \vee z_3)$

where z_i are introduced new variable

set value of z accordingly!!

Hence its possible to convert having clause more than 3 variable to φ' of 3 literals.

hence

$\boxed{\text{CNF-Sat} \leq_p \text{3-SAT}}$

---

so NP-Complete

SAT is NP Complete

SAT is NP Complete

Boolean SAT problem.

$\langle \phi \rangle / \phi$ at least two satisfying assignments.

→ Boolean SAT is NP complete

↳ proof

(1) Boolean SAT ∈ NP

(2) Boolean SAT is NP hard

SAT ≤p Boolean SAT

Minimum

A function $f: \Sigma^* \to \Sigma^*$ is polynomial time computable function if there exists a TM M that when given input w halts with just f(w) on its tape when started with input w

to test whether we A the real reduction $f: A \to B$ to check if $\boxed{f(w) \in B}$

There are computational problem that computer cannot solve

Related Issue | Questions

- ✓ What is computational problem?
- ✓ What do you mean by 'solving' a problem?
- ✓ What are 'computes'
- ✓ How to prove such statement

no of unique problem > # of unique

$f: A \to B$ $\quad |A| = |B|$

bijection (one to one | onto)

13/9/23

CLIQUE = {<G,K> | G is an undirected Graph with a K clique where K>0 is integer}

Theorem Clique is NP-complete

Proof (i) Clique ∈ NP

(ii) Clique is NP-hard

i.e A ≤p Clique ∀ A∈NP

Take 3-sat is NP-complete

RTP (Required To Prove):

$\boxed{3SAT \leq_p CLIQUE}$

i.e Given a 3-CNF boolean formula convert it into an undirected Graph by having clique of some specified size".

al: G has a clique of some size if & only the 3CNF formula ϕ is satisfiable!

ction of G:

$\phi = (a_1 \vee b_1 \vee c_1) \wedge (a_2 \vee b_2 \vee c_2) \wedge \cdots (a_k \vee b_k \vee c_k)$

$= C_1 \wedge C_2 \wedge \cdots C_k$

↓

Ith clause

↳ for each clause $C_i = a_i \vee b_i \vee c_i$ create three vertices and label them!

Total vertices : 3k.
in G

↳ Add all possible edges among 3k nodes with following two exception

(i) No edges between nodes in triple is allowed

for $C_i = a_i \vee b_i \vee c_i$, the edge $(a_i, b_i), (b_i, c_i), (a_i, c_i)$ are not allowed

(ii) No edge in node with contradictory label are allowed

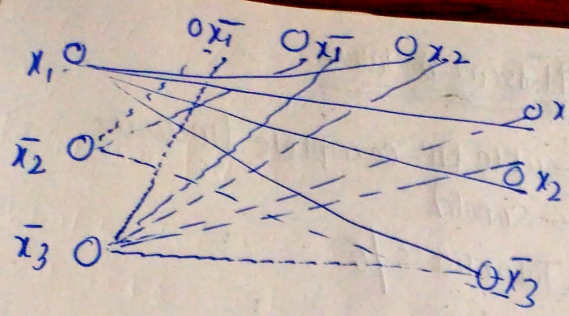
$(x_i, \bar{x}_i)$ is not allowed

Let $\phi = (x_1 \vee \bar{x}_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee x_2) \wedge (x_2 \vee x_2 \vee x_3)$

x_1 ∅ ∅ $\bar{x}_1$

$\bar{x}_2$ ∅ ∅ x_2

$\bar{x}_3$ ∅ ∅ x_3



(connect all non contr, shown some)

$|V| = 3 \times 3 = 9$.

When ϕ is satisfiable

$\Rightarrow$ ϕ has satisfying assignment

$\Rightarrow$ Every clause c_i in ϕ is true

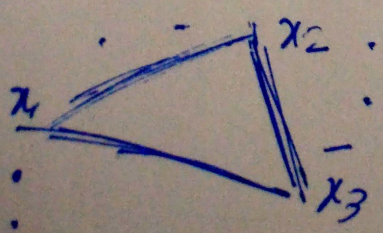
$\Rightarrow$ Each c_i has true literal

$\therefore$ pickup such true literal from each clause c_i !!

Claim: G has k-clique

- Choose one literal exactly from each c_i
- assign true value to each literal chosen.

In above ex — x_1: true, x_2: true, $\bar{x}_3$: true



HAMPATH: {<G, s, t> | G is directed graph having hamiltonian path from s to t.}

Theorem HAMPATH $\in$ NP-Comp.

(1) HAMPATH IN NP

(2) " " is NP-hard

Take a known NP-complete problem say 3-SAT show that $3SAT \leq_p$ HAMPATH

=

UHAMPATH = {<G, s, t> | there is hamiltonian path from vertex s to vertex t in undirected graph}

Theorem UHAMPATH is NP-Complete.

Proof (I) UHAMPATH $\in$ NP ✓ (same as hampath)

(II) UHAMPATH is NP-HARD

Take HAMPATH as known NP-complete problem

Claim: HAMPATH $\leq_p$ UHAMPATH

G is directed graph with s and t

<G, s, t>

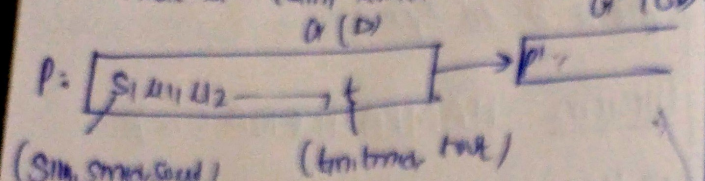
↓ convert

G' is undirected graph with s' and t'

<G', s', t'>

note that $u \longrightarrow$ (u_{in}) $\rightarrow$ (u_{mid}) $\rightarrow$ (u_{out})

node $u = (u_{in}, u_{mid}, u_{out})$



Construction of G' (undirected)

Let $G' = (V(G'), E(G'))$ and $G = (V, E)$

$V(G') = \underbrace{\{s^{out}}_{(s')}, \underbrace{t_{in}}_{(t')}\} \cup \{\bigcup_{u \in V(G')/\{s,t\}} (u_{in}, u_{mid}, u_{out})\}$

$E(G') =$

(i) Edges connect u_{mid} with u_{in} and u_{out} i.e edges (u_{in}, u_{mid}), (u_{mid}, u_{out}) are allowed

(ii) Edge connecting u_{out} with v_{in} if edge connect u to v in G!

if $|P| = m$ $|P'| = 3(m-2) + 2 = 3m+4$

P': $s^{out}, u_1^{in}, u_1^{mid}, u_1^{out}, u_2^{in}, u_2^{mid}, u_2^{out}$

$\cdots\cdots t_{in} = t'$

back to G.

Midsem Syllabus

- upto NP-complete (including Simplex)

Total 40's /6