34: NP- Completeness

* Shortest vs. longest simple paths :→ We saw that that even with negative edge weights, we can find shortest path from a single source in a directed graph $G=(V,E)$ in $O(VE)$ time.

→ Finding a longest simple path b/w 2 vertices is difficult, However. Merely determining whether a graph contains a simple path with a least a given number of edges is NP- Complete.

* Euler tour vs. hamiltonian cycle :

→ An Euler tour of a connected, directed graph $G=(V,E)$ is a cycle that traverses each edges on G exactly once, through it is allowed to visit each vertex more than once.

→ We can determine whether a graph has Euler tour in only $O(E)$ time, & infact, we can find the edges of the Euler tour in $O(E)$ time.

→ A hamiltonian cycle of a directed graph G is a simple cycle that contains each vertex in V. Determining whether a (un)directed graph has a hamiltonian cycle is NP- Complete

* 2- CNF Satisfiability vs. 3- CNF Satisfiability :

→ A boolean formula contains variables where values are 0 or 1: boolean connection such as $\wedge$ (AND), $\vee$ (OR), $\sim$/$\neg$ (NOT); and parentheses. A boolean formula is satisfiable if there exists some assignment of the values 0 & 1 to its variables that causes it to evaluate to 1. ~~we shall define terms more~~

→ A boolean formula ~~is~~ in K-conjuctive normal form. or K-CNF, if it is the AND of clauses of ORs of exactly K variables or their negation.

Ex:- $(x_1 \vee \sim x_2) \wedge (\sim x_1 \vee x_3) \wedge (\sim x_2 \vee \sim x_3)$ is in 2CNF

It has satisfying assignment $x_1 = 1, x_2 = 0, x_3 = 1$.

→ Although we can determine in polynomial time whether a 2CNF formula is satisfiable.

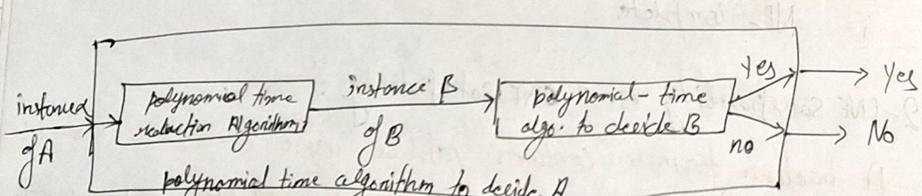
→ 3-CNF formula is satisfiable is NP-Complete.

Reductions :

Suppose that we have a procedure that transforms any instance α of A into some instance β of B with the following characteristics

→ The transformation takes polynomial time

→ The answers are the same. i.e. the answer for α is "yes" iff the answer for β is also "yes".



A concrete problem is polynomial-time solvable, therefore if there exits ~~time~~ An algorithm to solve it in time $O(n^k)$ for some constant k.

→ Hence, Complexity class P as the set of concrete problems that are polynomial-time solvable.

Formal Language

language = L, empty string = ε, empty language = ϕ

The language of all string over Σ by Σ^*.

Ex:- $\Sigma = \{0,1\}$ then $\Sigma^* = \{\varepsilon, 0, 1, 00, 01, 10, 11 \ldots\}$
⇓
all set of binary string.

$\therefore (\Sigma \subseteq \Sigma^*)$

Ex:- * Complement of L by $\bar{L} = \Sigma^* - L$

* The concatenation $L_1 L_2$ of two language L_1 & L_2

$L = \{x_1 x_2 : x_1 \in L_1 \ \& \ x_2 \in L_2\}$.

* Closure or Kleene star of Language L is

$L^* = \{\varepsilon\} \cup L \cup L^2 \cup L^3 \cup \ldots$

Where, L^k is language obtained by concatenating L to itself k times.

Ex:- PATH $= \{ \langle G, u, v, k \rangle : G = (V, E)$ is an undirected graph, $u, v \in V$, $k \geq 0$ is an integer & $\exists$ a path from u to v in G consisting of at most k edges$\}$.

Hence, Complexity class P:

$P = \{ L \subseteq \{0,1\}^* :$ there exists an algorithm A that decides L in polynomial time $\}$.

In fact, P is also the class of languages that can be accepted in polynomial time.

\# HAM-CYCLE $= \{ \langle G \rangle :$ G is hamiltonian graph$\}$.

\# Verification algorithms as being a 2-argument algorithm A, where one argument is an ordinary I/P string x & the other is a binary string y called certificate.

→ A two argument algorithm A verifies an I/p string x if there exists a certificate y such that $A(x,y) = 1$.

$$L = \{ x \in \{0,1\}^* : \exists\, y \in \{0,1\}^* \text{ such that } A(x,y) = 1 \}.$$

\# The complexity of NP is the class of languages that can be verified by a polynomial-time algorithm.

i.e. a language L belongs to NP iff $\exists$ a 2-I/p polynomial time algo A & a constant c such that

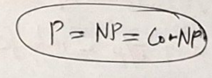
$$L = \{ x \in \{0,1\}^* : \exists \text{ a certificate } y \text{ with } |y| = O(|x|^c) \Rightarrow A(x,y) = 1 \}.$$

Ex:- If $L \in P$ then $L \in NP$, since if there is a 2-argument verification to decide L, the algo. can be easily converted to a 2-argument verification algorithm that simply ignores any ~~converted~~ certificate & accepts exactly those I/p strings it determines to be in L. Thus $\underline{P \subseteq NP}$

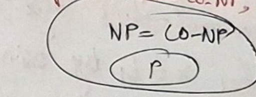
\# Complexity class Co-NP: the set of language L such that $\bar{L} \in NP$.

i.e. $\boxed{P \subseteq NP \cap Co\text{-}NP}$

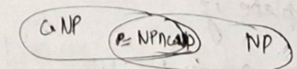
(a)



(b) If NP is closed under complement then NP = Co-NP, but Not P = NP

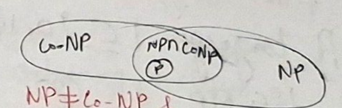


(c)



P = NP ∩ CoNP but NP is not closed under complement.

(d)

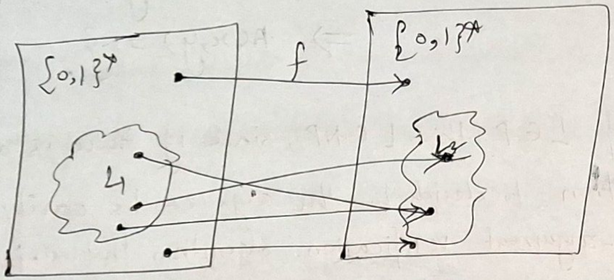


$NP \neq Co\text{-}NP$ &
$P \neq NP \cap Co\text{-}NP$

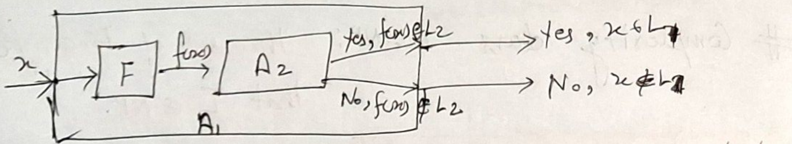
Reducibility: we say that a language L_1 is polynomial-time reducible to a language L_2, written $L_1 \leq_p L_2$, if there exists a polynomial-time computable function $f : \{0,1\}^* \rightarrow \{0,1\}^*$ such that $\forall\, x \in \{0,1\}^*$

f → Reduction funct^n.

i.e. $x \in L_1$ iff $f(x) \in L_2$



NP- Completeness



The algorithm F is a reduction algorithm that computes the reduction funct^n 'f' from L_1 to L_2 in polynomial-time, & A_2 is a polynomial-time algo. that decides L_2. Algorithm A_1 decides whether $x \in L_1$ by using F to transform any I/p x into $f(x)$ & then using A_2 to decide whether $f(x) \in L_2$.

A language $L \subseteq \{0,1\}^*$ is NP-complete if
① $L \in NP$, & ② $L' \leq_p L$ for every $L' \in NP$

The circuit-satisfiability problem is "Given a boolean conditional circuit composed of AND, OR, and NOT gates, is it satisfiable?"

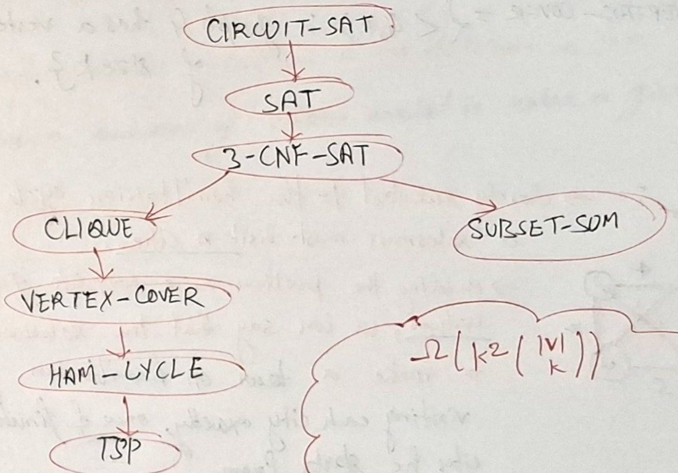
CIRCUIT-SAT = {<C> : C is satisfiable boolean conditional circuit}.

The clique problem

→ The clique in an undirected graph $G = (V, E)$ is a subset $V' \subseteq V$ of vertices, each pair of which is connected by an edge in E.

→ A clique is a complete subgraph of G. The size of a clique is the number of vertices it contains.

→ The clique problem is the optimization problem of finding a clique of maximum size in a graph.



$\Omega\left(k^2 \binom{|V|}{k}\right)$

→ CLIQUE = {<G, k> : G is a graph containing a clique of size k}.

Vertex-cover problem

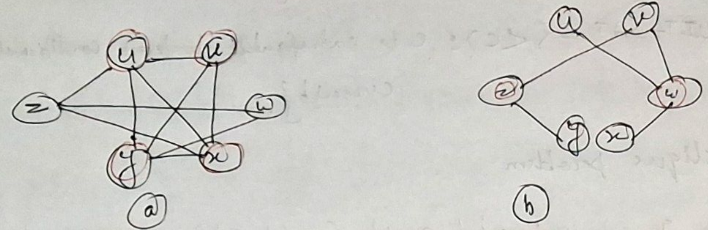


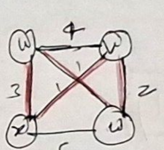
Figure 34.15: Reducing CLIQUE to VERTAX-COVER

(a) An undirected graph $G=(V,E)$ with clique $V' = \{u, v, x, y\}$

(b) The graph $\bar{G}$ produced by reduction algorithm that has vertex cover $V-V' = \{w, z\}$.

VERTAX-COVER $= \{\langle G,k\rangle :$ graph G has a vertex cover of size $k\}$.

TSP : → closely related to the hamiltonian cycle problem, a salesman must visit n cities.



→ Modeling the problem as a complete graph with n vertices, we can say that the salesman wishes to make a tour on hamiltonian cycle, visiting each city exactly once & finishing at the city he starts from.

$TSP = \{\langle G, c, k\rangle :$ $G=(V,E)$ is a complete graph, c is a funct^n from $V \times V \to N$, $k \in N$ & G has a TS tour with cost at most k $\}$.

$$c(i,j) = \begin{cases} 0 & \text{if } (i,j) \in E \\ 1 & \text{if } (i,j) \notin E. \end{cases}$$

SUBSET-SUM $= \{\langle S, t\rangle :$ ∃ a subset $S' \subseteq S$ such that $t = \sum_{s \in S'} s\}$.

Graph coloring: k-coloring is a function $c : V \to \{1, \dots, k\}$

$\Rightarrow c(u) \neq c(v) \; \forall$ edge $(u,v) \in E$.

→ The numbers $1, 2, \dots, k$ represent the k-colors & adjacent vertices must have different color.

→ The graph-coloring problem is to determine the minimum number of colors needed to color a given Graph