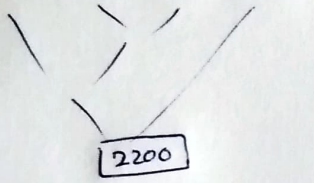
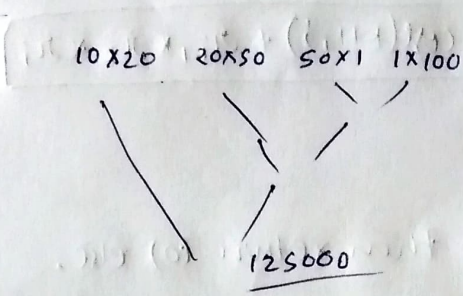
Matrix Product Problem:

$M = M_1 M_2 \cdots M_n$. minimum.

$10\times20 \quad 20\times50 \quad 50\times1 \quad 1\times100$





Let $X(n)$ be the number of ways of parenthesizing the n median

It is to say that $X(0) = X(1) = 1$ for $n > 2$

$$X(n) = \sum_{K=1}^{n-1} X(k) \cdot X(n-k)$$

$\underbrace{(M_1 \cdot M_2 \cdots M_K)}_{X(k)\text{ way}} \cdot \underbrace{(M_{K+1} \cdots M_n)}_{X(n-k)\text{ ways}}$

$M_1 M_2 M_3$
$((M_1 M_2) M_3)$
$(M_1 (M_2 M_3))$

$$X(3) = \sum_{K=1}^{2} X(k) \cdot X(n-k)$$

$$= X(1) \cdot X(2) + X(2) \cdot X(1)$$

$$= 1 + 1 = 2$$

$$X(4) = \sum_{K=1}^{3} X(K) \cdot X(n-K)$$

$$= X(1) * X(3) + X(2) \cdot X(2) + X(3) \cdot X(1)$$

$$= 2 + 1 + 2 = 5$$

$X(n) \geq 2$

Let $cost(i,j)$ be the minimum cost of computing.

$$M_i \cdot M_{i+1} \cdots M_j$$

what is the cost of building the product at M_k?

$$(M_i \cdot M_{i+1} \cdots M_k)(M_{k+1} \cdots M_j)$$

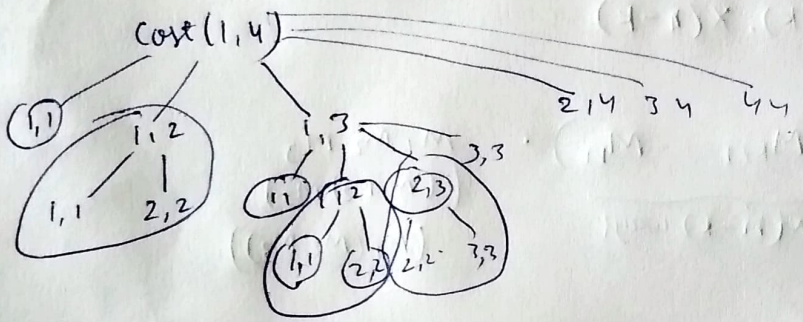
$(M_i . M_{i+1} \cdots M_k)(M_{k+1} \cdots M_j)$

$cost(i,k) + cost(k+1,j) + r_{i-1} * r_k \cdot r_k * r_j$

$$\boxed{cost(i,k) + cost(k+1,j) + r_{i-1} * r_k * r_j}$$

```
cost(i,j)
{
  If i=j then return (0) else,
  return (Min_{i≤k<j} (cost(i,k) + cost(k+1,j) + r_{i-1} * r_k * r_j))
}
```

~~cost(i,j)~~ $T(n) = 3T(n-1) + d$ $O(3^n)$



Store $cost(i,j)$ in $m[i,j]$. $m[i,j] = 0$ if $i=j$ and

$$\min_{i \le k < j} \left(m[i,k] + m[k+1,j] + r_{i-1} r_k r_j\right)$$

0	10K	1-2K	2-2K	
	0	1K	3K	/
		0	5K	/
			0	/
				0

$r_0 = 10$. $r_1 = 20$ $r_2 = 50$ $r_3 = 1$ $r_4 = 100$

$$m[1,2] = \min_{1 \le k < 2} \left(m[1,k] + m[k+1,2] + r_0 r_k r_2\right)$$

$= r_0 r_1 r_2 = 10000$

$$m[1,3] = \min_{1 \le k \le 3} \left(m[1,k] + m[k+1,3] + r_0 r_k r_3\right)$$

$= \min(m[1,1] + m[2,3] + r_0 r_1 r_3,$
$\quad m[1,2] + m[3,3] + r_0 r_2 r_3)$

$= \min(0 + 1000 + 200,\ 10000 + 0 + 500)$

$= 1200$

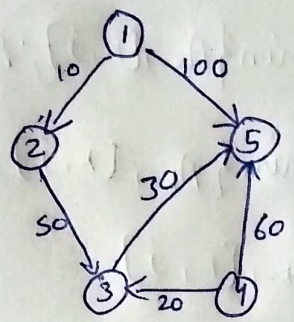
```
matrix(n)
for i=1 to n do m[i,i]=0.
for d=1 to n-1 do
  for i=1 to n-d do
    j = i+d
    m[i,j] = min_{i≤k≤n} (m[i,k] + m[k+1,j] + r_{i-1} r_k r_j)
return(m[1,n])
```

$O(n^3)$

Dynamic Programming. 30/11/2021

All pairs Shortest Path Problem.

graphs $G = (V, E)$



Given labelled directed graph $G = (V, E)$, find for each pair of vertices $v, w \in V$ the cost of the shortest path from v to w

Define A_k to be an $n \times n$ matrix with $A_k[i,j]$ the cost of the shortest path from i to j with internal vertices numberd $\leq k$.

$A[i,j]$:

- If $i \neq j$ and $(i,j) \in E$ then the cost of the edge from i to j
- If $i \neq j$ and $(i,j) \notin E$ then ∞
- If $i \neq j$ then $\emptyset$

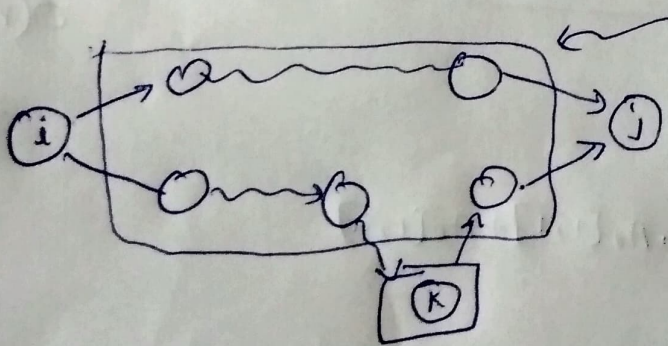
Computing A_k.

Consider the shortest path from i to j with internal vertices $1 \dots k$

either - It does not go through k, in which place it cost $A_{k-1}[i,j]$

- It goes through k, in which each it goes through k only once

So it costs. $A_{k-1}[v,k] + A_{k-1}[k,j]$



$A_k[i,j] = \min\{A_{k-1}[i,j] ; A_{k-1}[i,k] + A_{k-1}[k,j]\}$ 30/11/2021

Row k. $A_k[k,j] = \min\{A_{k-1}[k,j], A_{k-1}[k,k] + A_{k-1}[k,j]\}$

$= \min\{A_{k-1}[k,j], A_{k-1}[k,j]\}$

$= A_{k-1}[k,j]$

Column k. $A_k[i,k] = \min\{A_{k-1}[i,k], A_{k-1}[i,k] + \cancel{A_{k-1}[k,k]}^{0}\}$

$= A_{k-1}[i,k]$

Floyds Algorithm:

```
for i=1 to n do
  for j=1 to n do
    if (i,j) ∈ E.
      then A[i,j] = cost of (i,j)
      else A[i,j] = ∞.
    A[i,j] = ∅

for k=1 to n do
  for i=1 to n do
    for j=1 to n do
      if A[i,k] + A[k,j] < A[i,j] then
        A[i,j] = A[i,k] + A[k,j].
```

$O(n^3)$

A_0

0	10	∞	30	100
∞	0	50	∞	∞
∞	∞	0	∞	10
∞	∞	20	0	60
∞	∞	100	∞	0

$A_1[1,2]$

$A_1[i,j] \cdot A_0[1,2], A_0[1,1] + A_0[1,2]$

$A_0 \longrightarrow A_1 \longrightarrow A_2 \longrightarrow A_3 \longrightarrow A_4 \longrightarrow A_5$

Warshall's Algorithm.

↳ Given a directed graph $G \in (V, E)$ find for each pairs of vertices $v, w \in V$ whether this is a path from v to w

$A[i,j] \neq \infty$

```
for i=1 to n do
  for j=1 to n do
     A[i,j] = (i,j) ∈ E.
       A[i,j] = true.
for k=1 to n do
  for i=1 to n do
    for j=1 to n do
        A[i,j] = A[i,j] or (A[i,k] and A[k,j])
```

CLR text book.
↳ and solve the Problem H.W

Complexity Theory.

Reduction.

Problem P belongs to complexity class $f(n)$ iff best algorithm solves P in time $f(n)$

$\Rightarrow$ no algorithm with time $< f(n)$ is possible.

P: Class of problems which have an algorithm taking time $O(n^k)$ where k is any fix integer.
n is size of input.

EXP time: Problem for which algorithm take time $O(k^n)$
↳ Polynomial.

[Bubble sort. $O(n^2)$
then it is P class Problem

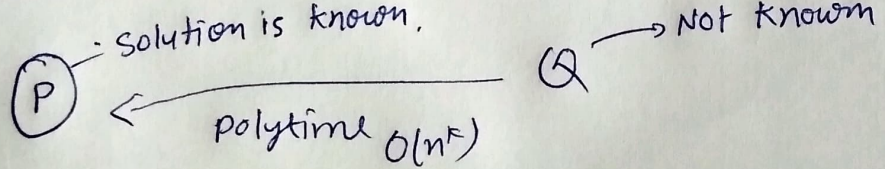
[Array are sorted.
↳ so it is decision Problem

P: class of Problem for which "fast" algorithm exist are "easy" Problem.

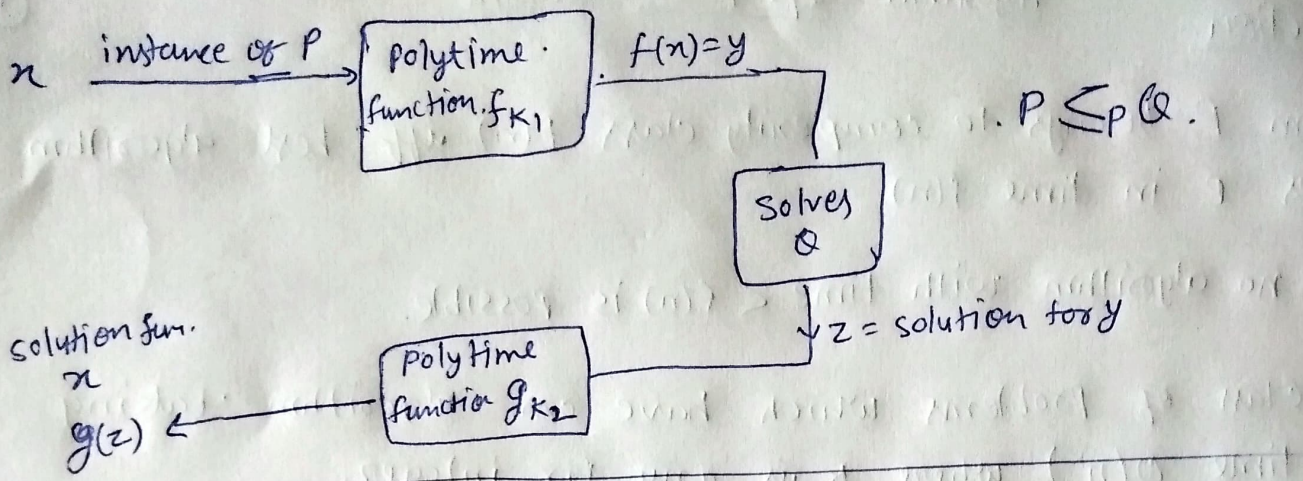
Problem not in P are "Hard" $\Longrightarrow$ fast algorithm do not exist.

Matrix Mult. $O(n^3)$
Strassion Mult. $O(n^{2.81})$

[if sorting easy then it is easy Prob
[if sorting Hard then it is Hard
[if it belong to P then it is easy Prob



Polynomial time reduction.: Problem P can be polytime reduced to Problem Q iff there exists polytime compatable function f & g such that given an instruction. x of P, $f(x)$ is an instance of Q, given a solution z to an instance y of Q, $g(z)$ is the solution to x of $y = f(x)$


x
instance of P
Polytime function f_{K_1}
f(x)=y
Solves Q
z = solution to y
Polytime function g_{K_2}
solution fun. x
g(z)

$P \leq_p Q$.

Complexity Theory :-

Theorem : Suppose P can be polytime reduced to Q, then if there exists a ~~Polytime~~ polynomial time algorithm for Q, then there exists a polytime algorithm for P.

$P \leq_P Q$

Q has polytime algo / P has polytime algo.

Proof :- $|s|$ = length of s in bits.

Time to compute f : $|x|^{k_1}$

$|y| \leq |x|^{k_1}$ [x is instance of P]

Given that A solves Q in time n^{k_2}

Time for solving $|y| = |y|^{k_2} \leq |x|^{k_1 k_2}$

Q is no easier than P.

- Suppose we can prove that no polytime algorithm exists for P $\Rightarrow$ No polytime algorithm for Q.
- Suppose no polytime algorithm exists for Q $\not\Rightarrow$ no polytime algorithm exists for P.

↳ does not mean to say that.

Decision Problem : Answer = Yes or No.

Example: can this graph be colored in 3 colors?

Related non-decision problem,

for the given graph return the minimum no of colours required. for coloring. ___ $k = 1, 2, 3 \cdots$

P is a decision Problem if every instance has a unique solution which is either 0 or 1.

Theorem : Clique reduces to PCS.

Proof :-

Insert class

can G_T be scheduled on P Problem such that completed in d steps?

0, 1 — Integer linear Programming:

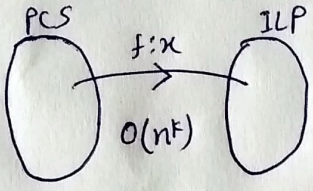
Input : matrix A. $m \times n$

vector b. $m \times 1$

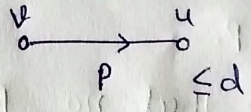
Decision Problem: Does there exists a vector x $(n \times 1)$ such that,

$A.x \leq b$ and $x \in \{0, 1\}$

(1) PCS reduces to ILP.



PCS - Directed acyclic graph.
- no. of process
+ deadline.



t_u - time at which vertex u is schedule.

$x_{ut} = 1$; if u is schedule at time t.

$= 0$; otherwise

Constraint: for every time step t, only P node can be scheduled

$$\sum_x x_{ut} \leq P \text{ for all } t.$$

O(n) — $x_{11} + x_{21} + x_{31} + \cdots + x_{n1} \leq P$

Every node must be scheduled repatly at one instance,

$$\sum_t x_{ut} = 1.$$

O(t) — $\sum_t x_{ut} \leq 1$ & $-\sum_t x_{ut} \leq -1.$

Time at which u is scheduled $= 1 \cdot x_{u1} + 2 \cdot x_{u2} + 3 x_{u3} + \cdots + d x_{ud}$.

$1 \cdot x_{u1} + 2 x_{u2} + 3 x_{u3} + \cdots < 1 \cdot x_{v1} + 2 x_{v2} + \cdots$.

$1 \cdot x_{u1} + 2 x_{u2} + 3 x_{u3} + \cdots \leq 1 \cdot x_{v1} + 2 \cdot x_{v2} + \cdots - 1$.

O(d)

polytime in nature. $O(n) + O(t) + O(d)$

Hence. $PCS \leq_P ILP$.

Algorithm. 07/12/2021

Reductions.

Theorem: Clique reduces to Procedance constrained scheduling

PCS.: Task graph T.

#Processors = P deadline = d.

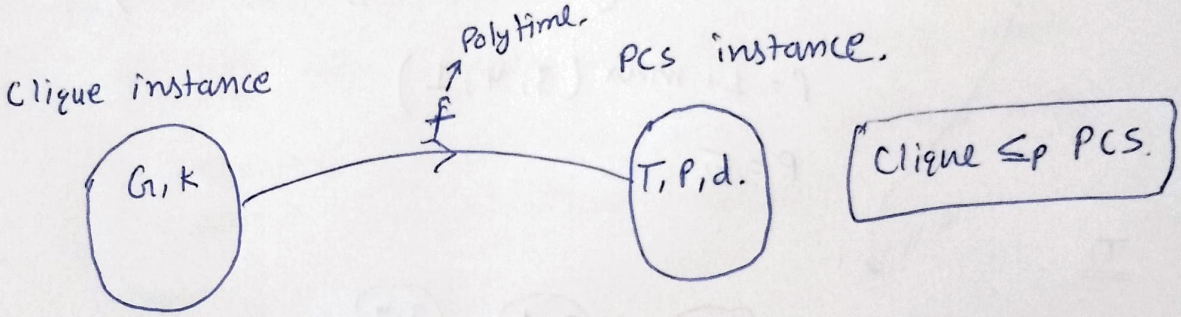
Decision: can T be scheduled using P Processors in time d?

Clique.: Graph G, integer k.

Desision: Does G contain as a subgraph a clique of size k?

Clique of size k = complete graph of k vertices.

Proof:



for every G, k, d = 3

$$P = 1 + \max\left(k,\ n-k+\frac{K(k-1)}{2},\ m-\frac{K(k-1)}{2}\right)$$

m = # edgs in G.

n = # vertices in G.

$T = T_1, T_2$

T_1 = verten set $\doteq$ { one verten f(u) for every verten u of G & one verten f(u,v) for every edge (u,v) of G.

: edge set = { Directed edge from vertices f(u) to f(u,v) }

T_2 = 3 level graph.

↓

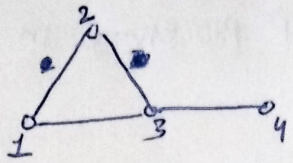
T_2 = 3 level graph.

IIIrd level · $P - \left(m - \frac{K(K-1)}{2}\right)$

IInd level $P - \left(n - K + \frac{K(K-1)}{2}\right)$

Ist level · $P - K$ tasks

Example:



$K = 3$
$n = 4$ (vertices)
$m = 4$ (edge)

$x = G_1$.

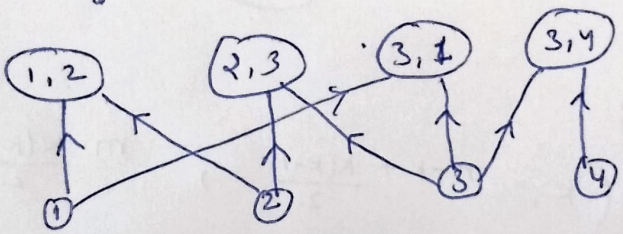
$f(x) = (T, P, d)$

$d = 3$ $P = 1 + \max\left(3, 4 - 3 + \frac{3*2}{2}, 4 - \frac{3*2}{2}\right)$

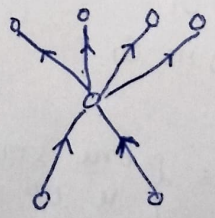
$P = 1 + \max(3, 4, 1)$

$P = 5$.

T_1



T_2



$P - K = 2$.

Number of tasks in T_1 & T_2

$|T_1| + |T_2| = m + n + 3P - (m+n)$

$= \underline{3P}$.

schedule exists iff every processor is busy at every step.

Claim.: Suppose a schedule exists for T, P, d then G_1 must contain a clique of size K.

Proof:

07/12/2021

Proof: Number of nodes of T_2 scheduled in top level.

$$\underbrace{P}_{\text{Total}} - \underbrace{\left(P - m + \frac{k(k-1)}{2}\right)}_{\text{nodes of } T_2}$$

$$m - \frac{k(k-1)}{2}$$

must all be edge nodes.

$\frac{k(k-1)}{2}$ edge nodes are in level 2

All vertices associated with these edge task must be level-1. But level-1 has space for k tasks which implies.

$\Rightarrow \frac{k(k-1)}{2}$ edge tasks in level-2 are such that associated edges are incident only on k vertices. which implies.

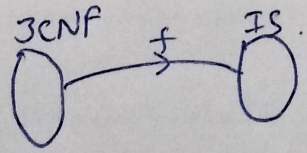
$\Rightarrow$ K clique exists.

Clique $\leq_p$ PCS $\leq_p$ ILP

Formula satisfiability.: Input :· formula.

o/p : True if there exist an assignment of value to variable such that the formula evaluation true else false

Theorem: formula set (3CNF) $\leq_p$ Independent set.



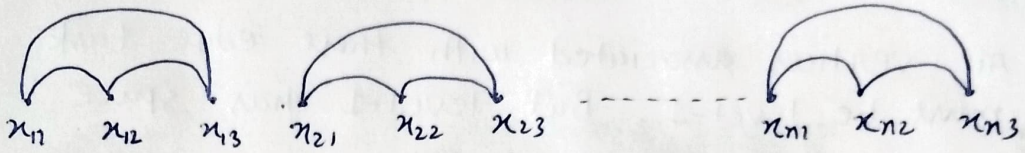
Theorem : Formula satisfiability $\leq_p$ Independent set

Proof : Definition of f.

Let input formula be $x = \underbrace{(x_{11}+x_{12}+x_{13})}_{A}\underbrace{(x_{21}+x_{22}+x_{23})}_{B}\cdots(x_{n1}+x_{n2}+x_{n3})$

where x_{ij} in the literal

$f(x)$: G will have $3n$ vertices. 3CNF

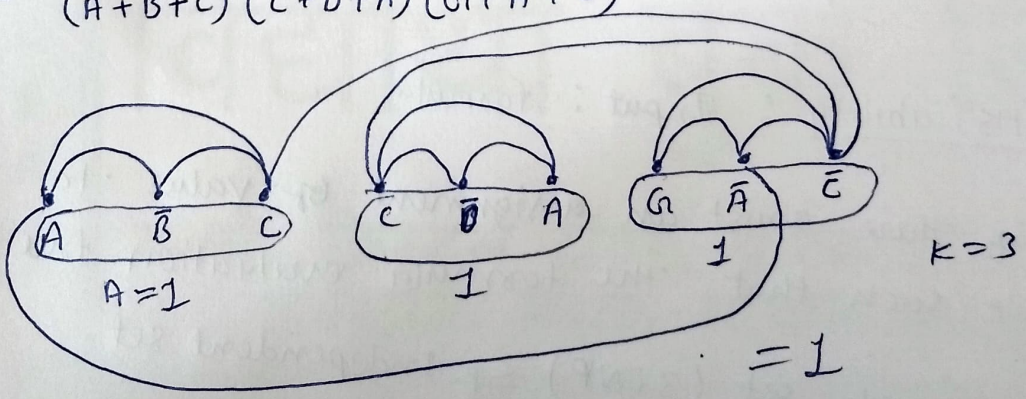


Edges : Two kinds of edges.

(i) Triangle counting x_{i1} x_{i2} x_{i3} $\forall i$

(iii) For every x_{ij} & x_{kl} such that x_{ij} is negation of x_{kl}, Put an edge.

Example : $(A+\bar{B}+C)(C+\bar{D}+A)(G+\bar{A}+\bar{C})$



If $P(x)$ is true then $Q(f(x))$ is true.

Proof : $P(x)$ = true $\Rightarrow$ In each sum there exits a literal x_{ij} that is true.

Claim : The set of vertices corresponding to x_{ij} is independent.

Proof: Suppose there exist an edge (x_{ij_i}, x_{kj_k})

$\Rightarrow x_{ij_i} = \overline{x}_{kj_k}$

This is not possible. Hence proof.

If $Q(f(x)) = true \Rightarrow \exists$ independent set of size n.

$\Rightarrow$ Due to type 1 edges exactly one vertices x_{ij_i} from each triple must be in the independent set.

Consider the assignment $x_{ij_i} = 1 \ \forall i$

Set other variables arbitrarily.

Since x_{ij_i} & x_{kj_k} are never compliment the 3CNF is true.

Circuit Satisfiability:

Given: combinational circuit made up of 2-input NAND gates with a single output.

Question: Does there exist an input assignment such that output = True.

Theorem: Circuit sat $\leq_p$ CNF satisfiability.

Proof: x : circuit.

$f(x) = 3CNF$

$f(x)$ will that the input value and the output values for each gates are consistent with its type & output.

Express using 3CNf : x is NAND(C,D)

$$CD \longrightarrow \overline{x}$$

$$\equiv \overline{CD} + \overline{x}$$

$$\equiv \overline{C} + \overline{D} + \overline{x}$$

~~f(x) = w Π CNF expr~~

$f(x) = w \; \Pi$ CNF . that input - o/p . to the i^{th} gate are consistant with its type.

G_i

$f(x) = 1 \Rightarrow w = 1$, G_i = True.

$\leq_p$ Clique $\leq_p$ PCS $\leq_p$ ILP $\leq_p$ circuit Sta. $\leq_p$ Formula Sta.

Ind. set $\leq_p$

~~f(x) = w = 1 CNF expr~~

$f(x) = w = 1$ CNF that input - o/p ; to the i^{th} gate are consistant with its type.

G_i

$f(x) = 1 \Rightarrow w = 1$, G_i = True.

$\leq_p$ Clique $\leq_p$ PCS $\leq_p$ ILP $\leq_p$ circuit Sta. $\leq_p$ Formula Sta.

Ind. set $\leq_p$

09/12/2021 Algorithm

Definition: Let P be a decision problem. we will say that V is a verifier for (true instance of) P iff :

1. V takes two assignments x & y and return true / false.
2. ∀ instance x

P(x) = true ⇒ ∃y s.t. V(x,y) = true

&

P(x) = false ⇒ ∀y V(x,y) = false.

→ Every true statement must have some Proof.

→ If a statement is false, it must not have a Proof.

Proof. e , says whether or not y is a proof for x

↓

V is a polytime verifier if time taken by V is polynomial in length of the first input.

Definition: We will say that a decision problem belongs to the class NP of P has a polynomial time verifie

---

Cooks Theorem: All problem in class NP can be reduced to circuit satisfibility (Hw. CLR Textbook)

Theorem: Graph Coloring Decision Problem is in NP.

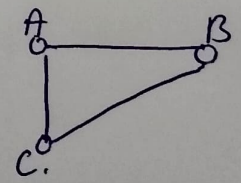
(Graph ; color)
(G, K)

if or can color can be k color. or ans. will be yes or not so this is a decision problem.

Proof. x : instance.
y : coloring. ; sequence of n-numbers.
$c_1\ c_2 \cdots c_n$; c_i is color of node i

verifing ← V(x,y): for every edge (i,j) in G do.

if $c_i = c_j$
return false
else.
for every i do
if $c_i > k$ or $c_i < 1$
return false.
else.
return true:



$O(E) + O(E+i) = O(E)$

Hence: Graph Coloring Problem is in NP

09/12/2021

Theorem. PCS is in NP.

Proof: $x : (T, P, d)$

$y : t_1 t_2 \cdots t_n$ · n = no. of nodes.

t_i = time at which node i is ~~calculated~~. Executed

$V(x, y)$: (i) for every edge (i, j)

if $t_i \geq t_j$ ··

return false

(ii) $P_T = \{i \mid t_i = T\}$

set of nodes executed at step T.

if $P_T > P$ return false

(iii) Check that · $1 \leq t_i \leq d$ · for all i

$O(E+V)$ Polytime.

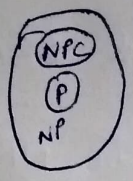
Pcs is in NP.

~~traveling~~ Hamiltonian integer ~~fa~~

NP - complete Problems.

All Problem in NP can be reduced circuit-set

Co-NP.



$P \neq NP$?
$P = NP$?

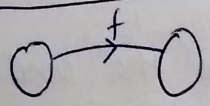
$V(x,y)$ T/F

Polytime $P \in NP$

NP-complete.

Problem Q is NP-complete if class $NP = \{\ldots\} \leq_p$ Ckt-set

1. $Q \in NP$
2. $Q' \leq_p Q$ for all $Q' \in NPC$



why this logic is sufficient?

Since $Q' = NPC$, all Problem in NP is reducible to Q'

~~Since Q'~~.

Show Q' is reducible to Q.

Then all Problem in NP is reducible to Q.

$\therefore$ Q is NPC.

Example:

PCS is NPC.

Proof:

① PCS $\in$ NP.

$x = (T, R, d)$

$y = t_1 \text{ ---- } t_n$.

$V(x,y)$

$t_i \geq t_j$ — false.

$P_T = \{i \mid t_i = T\}$

if $P_T \triangleright P$ — false

$1 \leq t_i \leq d$ return true

Polytime.

(1) PCS $\in$ NP

② Circuit-satisfiability $\leq_p$ PCS. — ①

Circuit-sat $\in$ NPC.

$\forall Q' \in NP$, $Q' \leq_p$ Circuit-sat — ②

from ① & ②

$\forall Q' \in NP$ $Q \leq_p$ PCS Hence PCS is NPC.

General strategy for Proving Q is NPC.

(i) Find a polytime verifier for Q.
(ii) Find some problem R such that R is NPC & $R \leq_p Q$.

The directed Hemiltonian cycle is known to be NPc use this ~~fact~~ fact to Prove that undirected HC is NPC.

Proof: (i) Undirected HC $\in$ NP. ——— (A)

(ii) Directed HC $\leq_p$ Undirected HC. —— (B)

↳ Define the transformation T.

Let $G = (V, E)$ be a directed graph Define G to the undirected graph $G' = (V', E')$ by the following transformation.

$= u \in V \longrightarrow v^1, v^2, v^3 \in V'$ and $(v^1, v^2), (v^2, v^3) \in E'$

$= (u, v) \in E \longrightarrow (u^3, v^1) \in E'$

$v_1 v_2, v_3 \cdots v_n, v$ has a HC.

then,

$v_1^1, v_2^2, v_3^3 \cdots - v_n^1, v_n^2, v_n^3, v_1^1$ is an undirected HC for G.

$v_i^1, v_{i_1}^2, v_{i_2}^3, v_{i_2}^1, v_{i_2}^2, v_{i_2}^3, \cdots\cdots v_{in}^1 \cdot v_{in}^2 \; v_{in}^3 : v_{i_1}^1$

$3(V+E)$

From (A) & (B) we can say that, undirected HC is NPC

Example:

Show that Subset Sum is NPC. (23)

Stratergy for sharing Problem $Q \in NPC$.

① Construct a polytime verifire for Q.

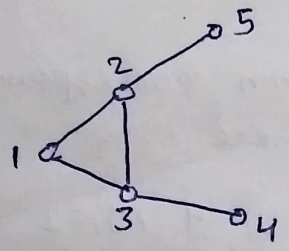
② Find a problem $R \in NPC$ s.t $R \leq_p Q$

Verten cover.

I/P : Undirected graph G, Integer k.

O/P : Yes iff G has a verten cover of size k.

Verten cover.: $G(V,E)$ $V' \subseteq V$ a verten cover of every edge in E, has atleast on endpoint in V'



{3,2} verten cover.

{1,3,5} verten cover.

Subset Sum Problem. ⟶ It belong to the class NP.

I/P : $S = \{S_1, S_2 \cdots S_n\}$

t : integer target.

O/P: True if $\exists S' \subseteq S$ such that sum of number in s' is enactly t

eg. $S = \{5, 17, 3, 2\}$ $t = 11$. — No

$S = \{5, 17, 3, 2\}$ $t = 10$. Yes

Polytime verifire for subset sum.

Theorem. : Verten cover $VC \leq_p SS$ (subsetsum)

$G(V,E)$ $\xrightarrow{f}$ $S = \{x_1, x_2 \cdots x_m, y_1, y_2 \cdots y_n\}$

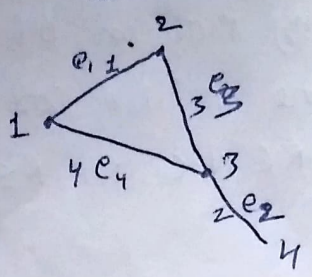
m = # edge

n = # verten

Edge incidence matrix.

Edge incidance matrix.

	e_1	e_2	e_3	e_4
v_1	1	0	0	1
v_2	1	0	1	0
v_3	0	1	1	1
v_4	0	1	0	0



Interpret each verten as a radix 4 number N_i

$$x_i = N_i + 4^m$$

$$x_1 = 4^0 \cdot 1 + 0*4 + 0*4^2 + 1*4^3 + 256$$

$$= 321$$

y_i = Place '1' in column corresponding to edge e_i & 0 elsewhere.

$y_4 = 64$, $y_3 = 16$, $y_2 = 4$, $y_1 = 1$

$$t = k4^m + 2.4^{m-1} + 2.4^{m-2} + \cdots + 2.4^0$$

Claim: If $vc(x) =$ true then $SS(x) =$ true.

Proof: Suppose vc in verten $V_1, V_2 \cdots V_k$.

Subset: $S' = \{x_{v_1}, x_{v_2} \cdots x_{v_k} \cdots\}$

$k.4^m +$ for every edge position we have either 1 or 2

Include y_i in S' if vc cover edge i exactly once.

Claim: If $SS(x) =$ true then $vc(x) =$ true.

Proof: suppose we know S', can we know suitable v'?

$$S' = \{x_{i_1}, x_{i_2} \cdots x_{i_n}, y_{i_1}, y_{i_2} \cdots\}$$

atleast k x_{i_s}

$V' = (i_1, i_2, i_3)$

suppose V' does not cover a certain edge.

$e = (u, v)$

$= x_u$ & x_v are not in S'

Consider column corresponding to d.

Sum of any subset of $S - \{x_u, x_v\}$ will not produce 2 in this position.

Set Cover:

Input: ① Sequence of sets $S_1, S_2 \cdots S_n$.

$S_i \subseteq X$ ⟶ Universal set.

② Integer k.

Output: yes, iff $\exists k$ sets $S_{i_1}, S_{i_2} \cdots S_{i_k}$ such that.

$S_{i_1} \cup S_{i_2} \cup \cdots \cup S_{i_k} = X$

eg: $X = \{1, 2, 3, 4, 5, 6\}$.

$S_1 = \{1, 2, 3\}$. $S_2 = \{2, 3, 4\}$ $S_3 = \{4, 5\}$

$S_4 = \{4, 5, 6\}$ $S_5 = \{6\}$

$X = \{S_1 \cup S_2 \cup S_3\}$

$X = \{S_1 \cup S_4\}$ $X = \{S_1 \cup S_2 \cup S_3 \cup S_4\}$

Minimize the size of the set cover.

weighted set cover - minimize total weight of the set cover.

Theorem: Set cover is NPC } H.W

Proof : set cover $\in$ NP. }

$V(x, y)$