chapter no : 5

-: NP - Hard and NP - complete problems.

I. P :

- P is the polynomial problem.
- Polynomial problem is solved in the polynomial time by the deterministic algorithm.
- Polynomial problem can be solved in minimum time
- Polynomial problem is in the deterministic format.
- Following are the polynomial problem example
  1. Searching Technique.

II. NP :

- NP is the non - polynomial problem.
- NP problem can be solved in the non - polynomi -al time by using non - deterministics algorithm.
- NP problem can be solved in maximum time as compare to the polynomial problem.
- NP problem can be superset of the polynomial problem.

- Non polynomial is in the non-deterministic format.
- Following are the example of Non-polynomial problem
  1. Knapsack problem
- The relationship between the polynomial and non-polynomial problem.

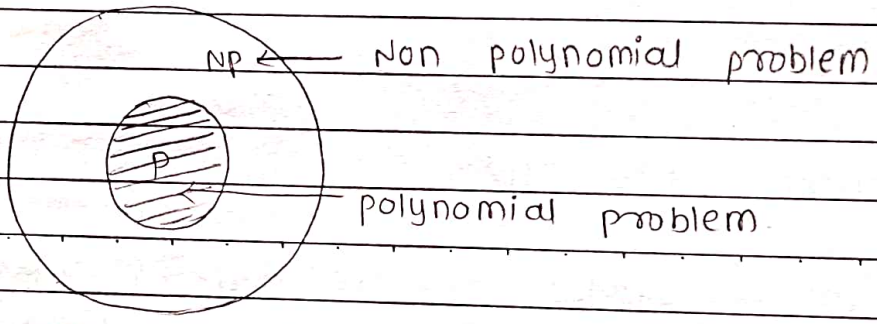
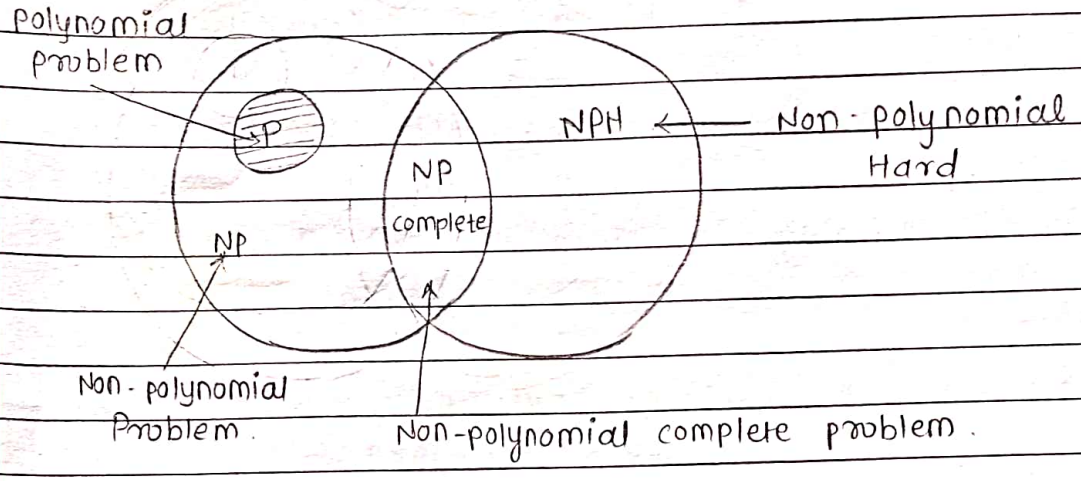


fig.-: Relationship between polynomial and Non-polynomial problem :-

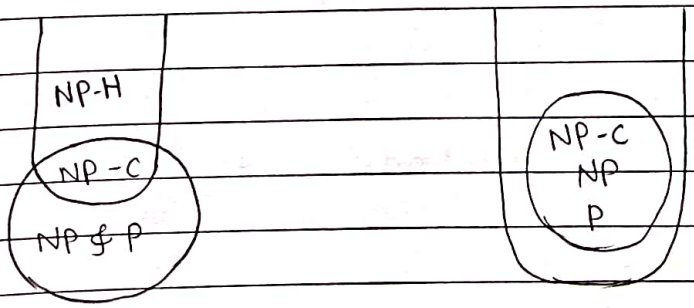
III. NP complete :

- NP complete is the Non-polynomial complete problem.
- Np complete problem is a superset of Non-polynomial problem or polynomial problem.
- Every Non-polynomial problem contain a polynomial problem.



IV. NP Hard :

- NP Hard is the Non-polynomial Hard problem.
- The NP complete problem is the subset of the NP Hard problem.
- When the $P \neq NP$ Hard but the $P = NP$
- In the following diagram it can be represented.



$P \neq NP$ complete. $P = NP$

fig: Non-polynomial Hard.

NP Hard scheduling problem :-

1. Scheduling Identical processors :-

Let Pi be example of printers in line or computer.

Now $i < i < m$ be the machines or processors for execution then.

a. Mean finish time

$$MFT(S) = \frac{1}{n} \sum_{i \leq i < m} fi$$

b. Weighted mean finish time

$$WMFT = \frac{1}{n} \sum_{i \leq i < m} wifi$$

c. Finish time (FT)

$$FT = \max_{i < im} \{Ti\}$$

11. Flow shop scheduling :

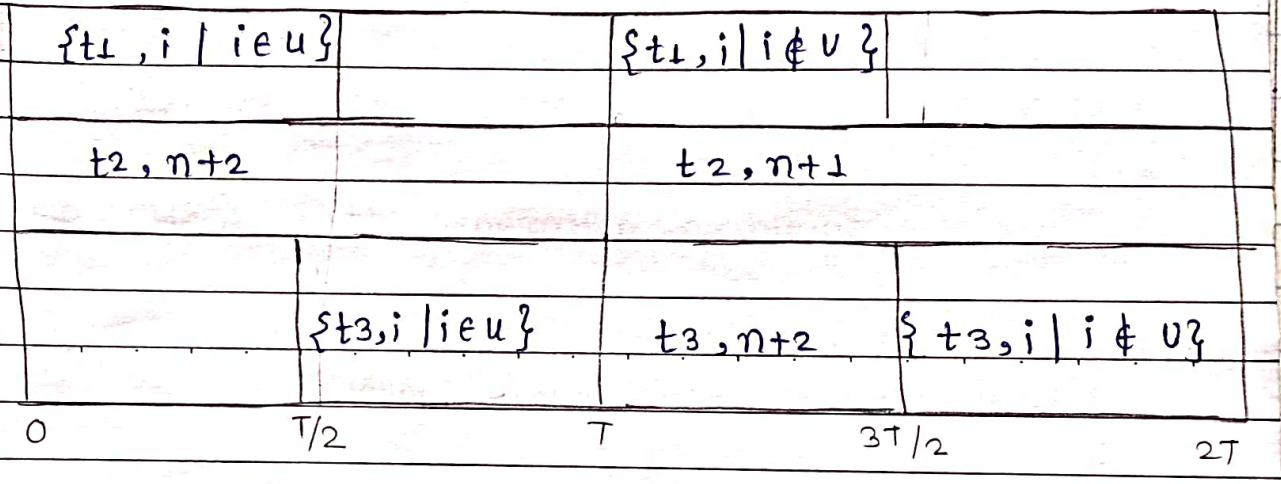
If n jobs are scheduling then time, $o(n \log n)$

The schedule length $0 < d \leq 2T$
Not more than $2T$.

$t_{1,i} = q_i$ $\quad t_{2,i} = 0$; $\quad t_{3,i} = a_1$, $1 \leq i \leq n$

$t_{1,n+1} = T/2$ $\quad t_{2,n+1} = T$; $\quad t_{3,n+1} = 0$

$t_{1,n+2} = 0$ $\quad t_{2,n+2} = T$; $\quad t_{3,n+2} = T/2$



A possible schedule.

III. Job shop scheduling :

M different processors.
N jobs to be scheduled.

The time of jt task for job ji is tkij.

The task for job ji to be carried out after
1, 2, 3, ----, j-1
∀. (j > 1)
It will not earlier than j-1

It can't be started until (i-1) is completed.

{t,i,1 / i∈u}	t1, n+1, 2	{t1, i, 1 / i∉u}	t1, n+1, 4
t2, n+1, 1	{t2, i, 2 / i∈u}	t2, n+1, 3	{t2, i, 2 / i∉u}

0 T/2 T 3T/2 2T

Another schedule.

NP - Hard code generation problems.

- In compiler code generation we are converting into assembly level language or machine level language

we are using one register as accumulator.
eg : (a+b) / (c+d)

{ addition, substraction, division, multiplication
eg :

```
LOAD  A          LOAD  C
ADD   b          ADD   d
STORE T1         STORE T1
LOAD  c          LOAD  a
ADD   d          ADD   b
STORE T2         DIV   T1
LOAD  T1
DIV   T2
```

25 % more efficient.

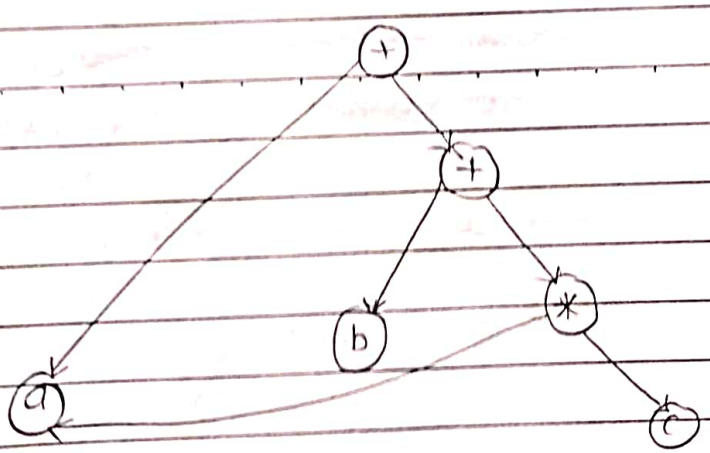
1 Code Generation with common subexpression :

Arithmatic expressions with common subexpressions represented by directed acyclic graph (DAG)

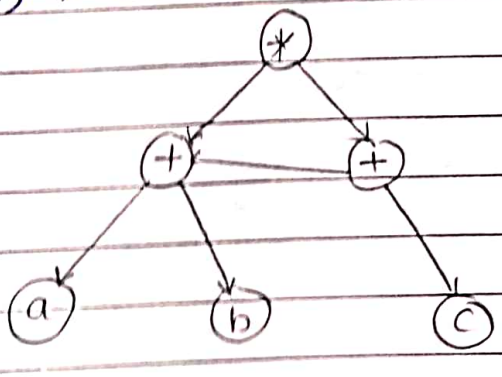
Every internal node is operator & external node is operand.

leaf :- out degree 0
shared node :- more than one parent

eg :- i. a + (b + a * c)



ii. (a+b) * (a+b+c)



11. Implementing parallel Assignment Instructions:

Parallel assignment instructions has the format

$(V_1, V_2, \ldots\ldots, V_n) = (e_1, e_2, \ldots\ldots, e_n)$

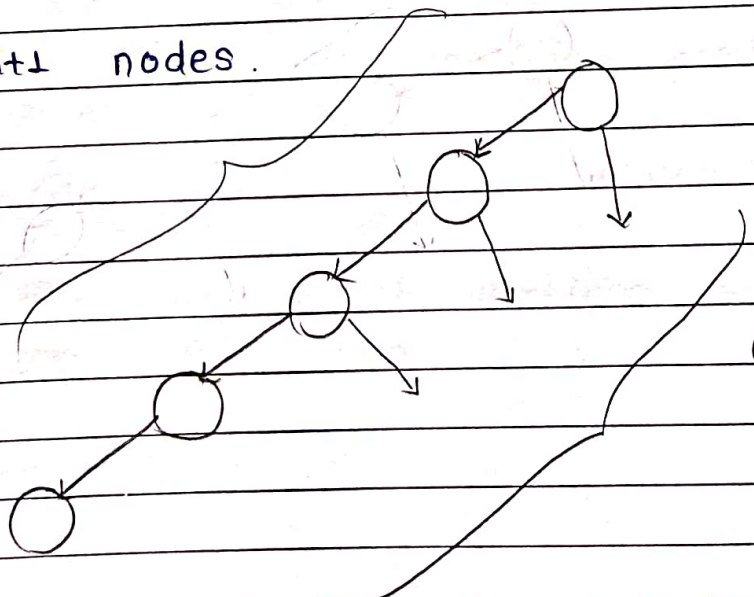
V_1 = variable names.
e_i = Expressions.

eg. $(A, B, C) = (D, A+B, A-B)$.

It gives 3! different relizations the costs:

R	C(R)	
1, 2, 3	2	
1, 3, 2	2	Not to store C(R) = 0.
2, 1, 3	2	
2, 3, 1	1	
3, 1, 2	1	
3, 2, 1	0	

n+1 nodes.



connections to tail Nodes.

NP - Hard Graph problems :

1. Clique Decision problem :

CDP $\in$ NP so
CDP $\in$ NP complete.

Now,

$$F = (x_1 \vee x_2 \vee x_3) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_3})$$

Now by setting,

$$x_1 = x_2 = x_3 = 1 \quad \&$$
$$\overline{x_1} = \overline{x_2} = \overline{x_3} = 0$$

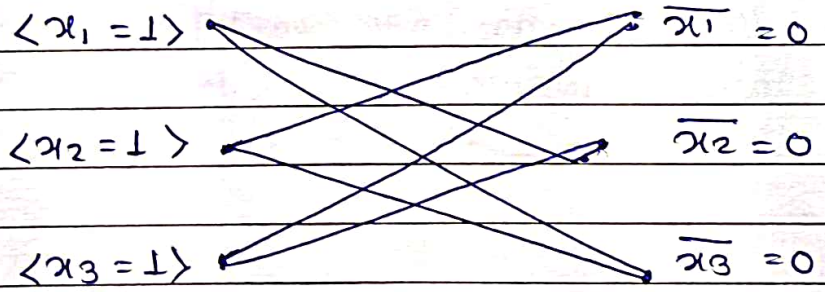
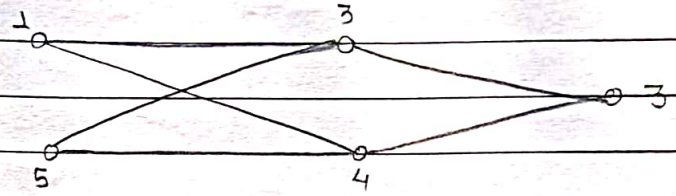


fig. clique Decision problem.

II. Node cover Decision problem : (NCDP).

The number of vertices covering in any set called NCDP

eg. = if $G = \langle V, E \rangle$ is graph then we check for.



eg.

if $S_1 = \{1, 2\}$ Node cover decision problem = 2

$S_2 = \{1, 3, 4\} \Rightarrow$ NCDP = 3

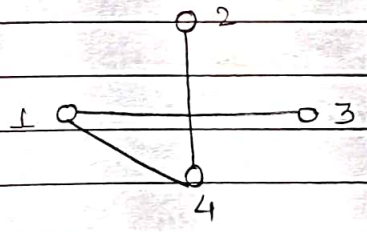
III. Complement of Graph :

If $G<V, E>$ is any graph,

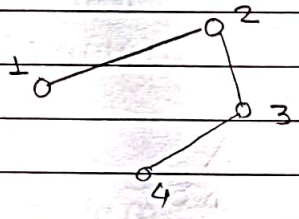
V = vertices or nodes

E = Edges, the complement of graph denotes all opposite edges of same graph

i.e. $\bar{G} = <V, E'>$



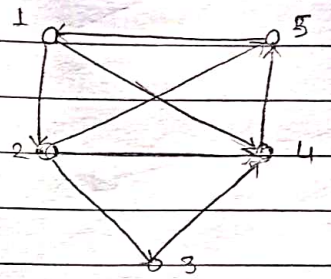
graph $<G' = V, E>$



graph $\bar{G} = <V, E'>$

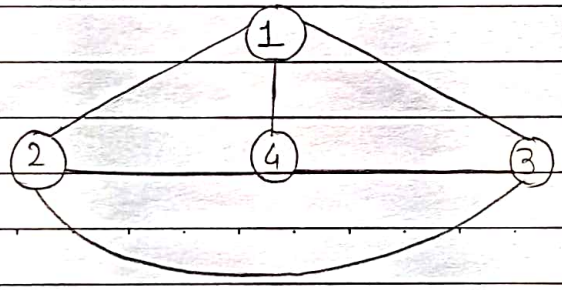
IV. Directed Hamiltonian Acyclic (DHC).

If is graph $G<V, E>$ where starting & ending Nodes are same.



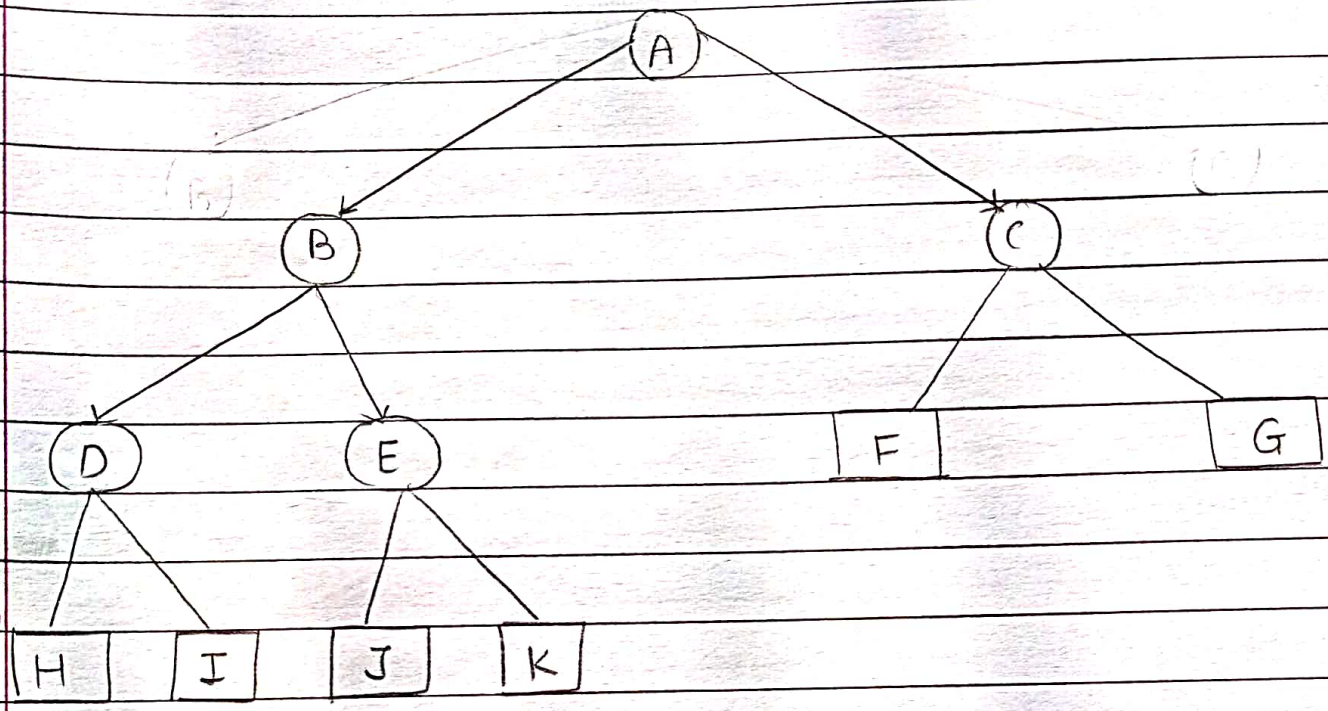
Eg. 1-2-3-4-5-1.

v. Travelling Salesperson decision problem :-



It is the problem of dynamic programming in which we have to find shortest path that visit every node exactly 1 & return to the starting node.

vi. AND / OR Graph Decision problem.


A
B
C
D
E
F
G
H
I
J
K

1) P:

- P is the polynomial problem.
- Polynomial problem is solved in the polynomial time by the deterministic algorithm.
- Polynomial problem can be solved in minimum time.
- Polynomial problem is in the deterministic format.
- following are the polynomial problem example.
  1) Searching Technique

2) NP:

- NP is the Non-Polynomial problem.
- NP problem can be solved in the Non-Polynomial time by using non-deterministics algorithm.
- NP problem can be solved in maximum time as compare to the polynomial problem.
- NP problem can be supperset of the polynomial problem.
- Non Polynomial is in the non-deterministic format.
- following are the example of Non-Polynomial problem.
  1) knapsack problem.

• The relationship between the polynomial and nonpolynomial problem.

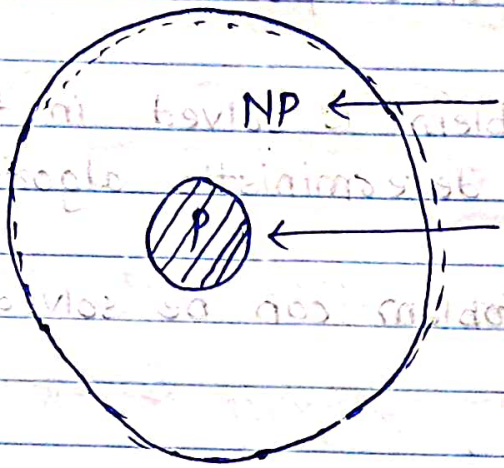


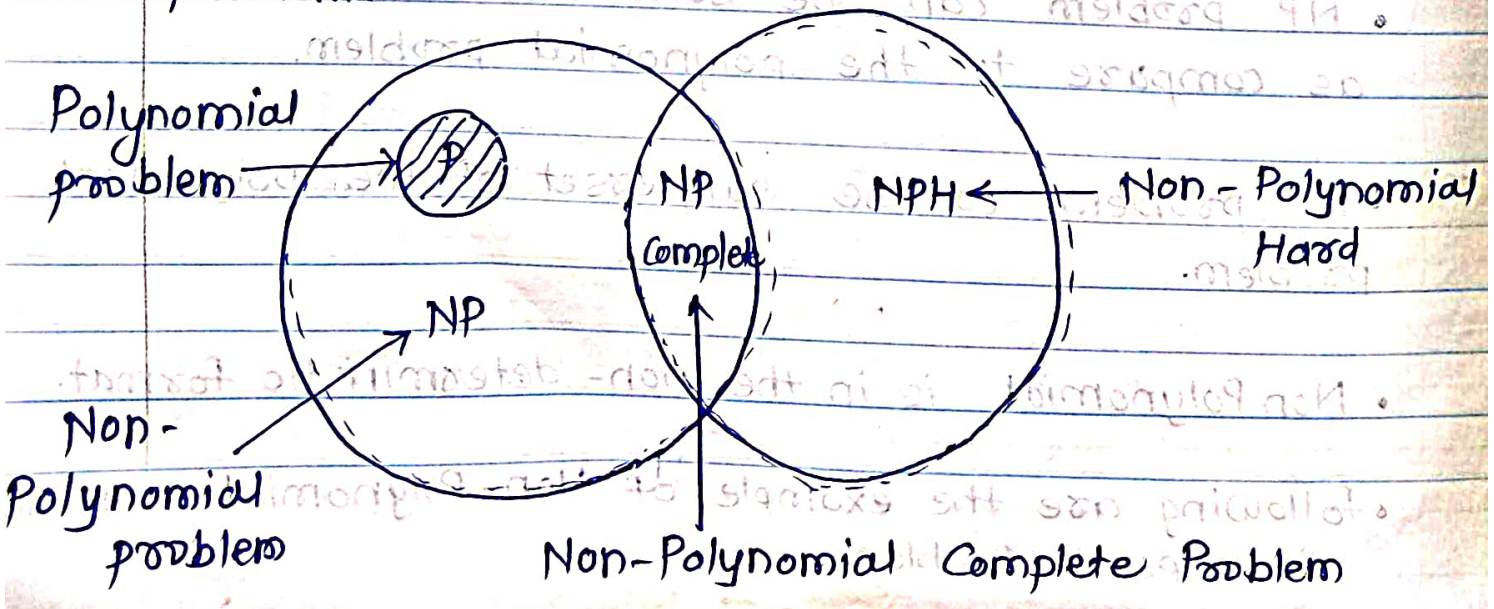
fig: Relationship Betn polynomial and Non-Polynomial problem.

3) NP Complete :

• NP complete is the Non-Polynomial Complete problem.

• NP complete problem is a superset of Non-Polynomial problem or Polynomial problem.

• Every nonpolynomial problem contain a polynomi problem.



(4) NP Hard :

- NP Hard is the NonPolynomial Hard problem.
- The NP complete problem is the subset of the NP hard problem.
- When the $P \neq NP$ Hard but the $P = NP$.
- In the following diagram it can be represented.

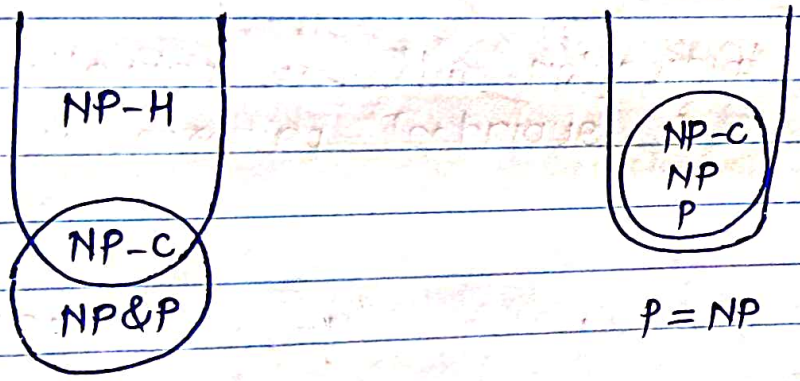


fig: Non-Polynomial Hard

NP Hard Scheduling Problem :

1) Scheduling Identical Processors :

Let Pi be example of printers in line or computer
Now i<i<m be the machines or processors for execution then,

ⓐ Mean finish time

$$MFT(S) = \frac{1}{n} \sum_{i \le i < m} fi$$

ⓑ Weighted Mean finish time

$$WMFT = \frac{1}{n} \sum_{i \le i < m} wifi$$

ⓒ finish time (FT)

$$FT = \max_{i<im} \{ Ti \}$$

2) Flow shop Scheduling

If n jobs are scheduling then time,

$O(n \log n)$

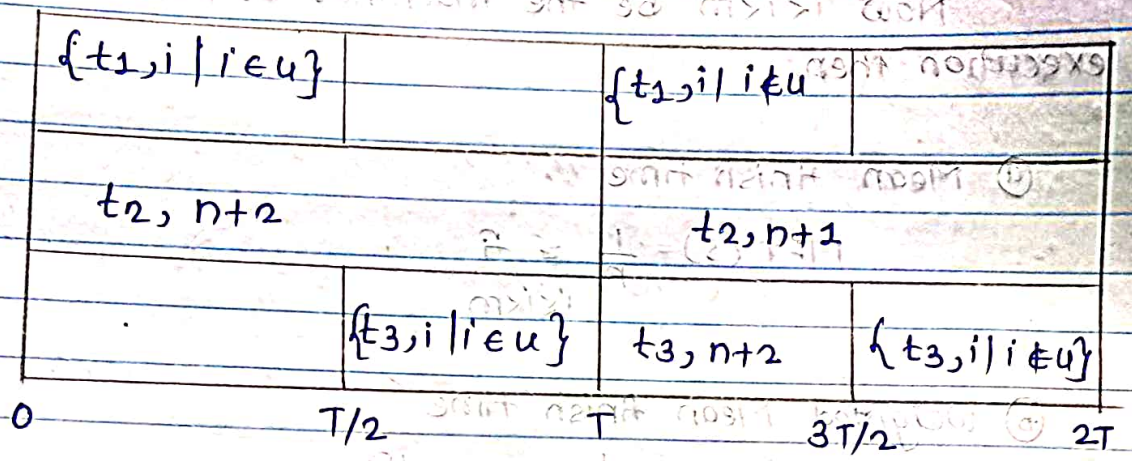
The schedule length $0 < \lambda \le 2T$

Not more than 2T.

$t_{1,i} = q_i$ $\quad t_{2,i} = 0$ $\quad t_{3,i} = q_i$, $1 \le i \le n$

$t_{1,n+1} = T/2$ $\quad t_{2,n+1} = T$ $\quad t_{3,n+1} = 0$

$t_{1,n+2} = 0$ $\quad t_{2,n+2} = T$ $\quad t_{3,n+2} = T/2$



A Possible Schedule

3) Job shop scheduling

M different processors

N jobs to be scheduled

The time of jt task for job ji is t_{kij}

The task for job ji to be carried out after 1, 2, 3, ---- j-1

$\forall$, (j'>1)

It will not earlier than j'-1

It can't be started until (j'-1) is completed

$\{t, i, 1 \mid i \in u\}$	$t_1, n+1, 2$	$\{t_1, i, 1 \mid i \in u$	$t_1, n+1, 4$
$t_2, n+1, 1$	$\{t_2, i, 2 \mid i \in u\}$	$t_2, n+1, 3$	$\{t_2, i, 2 \mid i \in u\}$

0 T/2 T 3T/2 2T

Another schedule

NP-Hard Code Generation Problems

In compiler code generation we are converting into assembly level language or m/c level language.

We are using one register as accumulator.

eg: (a+b) / (c+d)

addition, substraction, Division, Multiplication

eg:

```
LOAD  A          LOAD  C
ADD   b          ADD   d
STORE T1         STORE T1
LOAD  C          LOAD  a
ADD   d          ADD   b
STORE T2         DIV   T1
LOAD  T1
DIV   T2
```

25% more efficient

1) Code Generation with common subexpression:

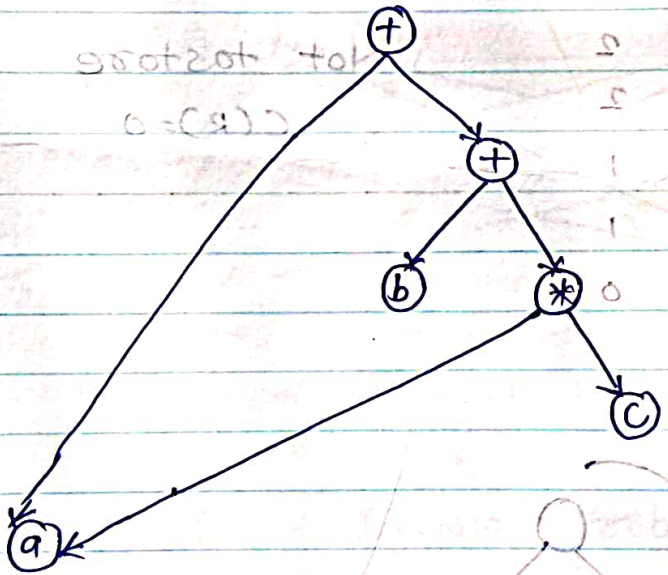
Arithmetic expressions with common subexpressions represented by directed acyclic graph (DAG).

Every internal node is operator & external nodes operand.

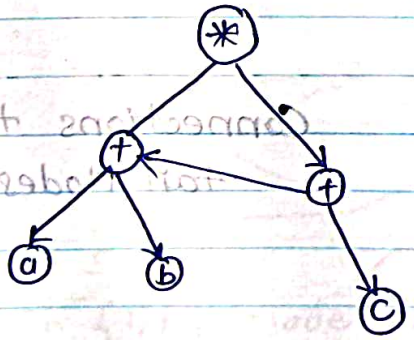
leaf :- out degree 0

Shared Node :- More than one parent

eg :- 1) a+(b+a*c)



2) (a+b)*(a+b+c)



2) Implementing Parallel Assignment Instructions:

Parallel assignment instructions has the format

$(V_1, V_2, \ldots, V_n) = (e_1, e_2, \ldots, e_n)$

V_1 = variable names

e_i = Expressions

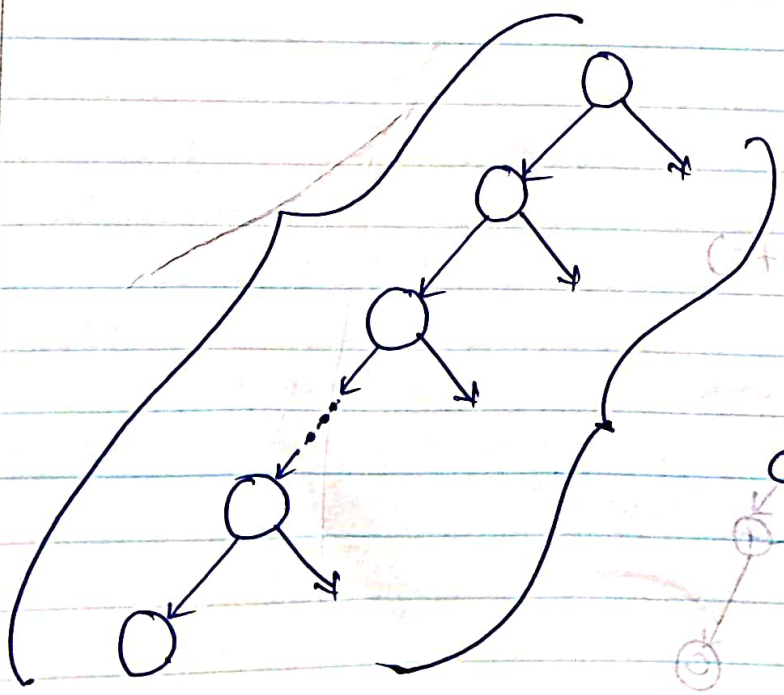
eg $(A, B, C) = (D, A+B, A-B)$

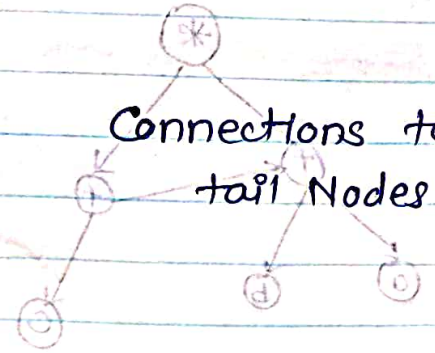
it gives 3! different relizations the costs.

R	C(R)
1, 2, 3	2
1, 3, 2	2
2, 1, 3	2
2, 3, 1	1
3, 1, 2	1
3, 2, 1	0

Not to store C(R)=0

n+1 nodes





* NP - Hard Graph Problems

1) Clique Decision Problem:

CDP ∈ NP so

CDP ∈ NP complete

Now,

$$f = (x_1 \vee x_2 \vee x_3) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_3})$$

Now by setting,

$x_1 = x_2 = x_3 = 1$ &

$\overline{x_1} = \overline{x_2} = \overline{x_3} = 0$

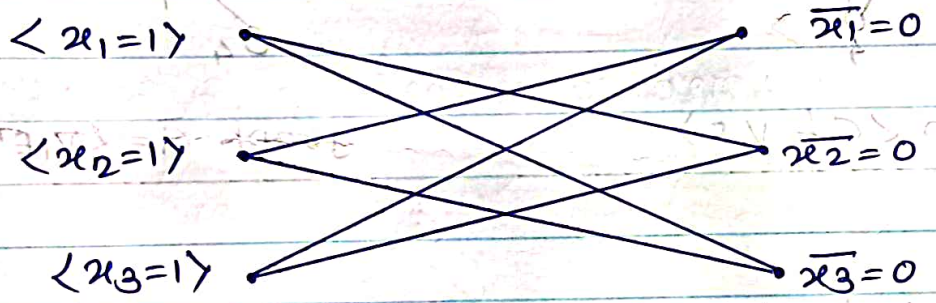
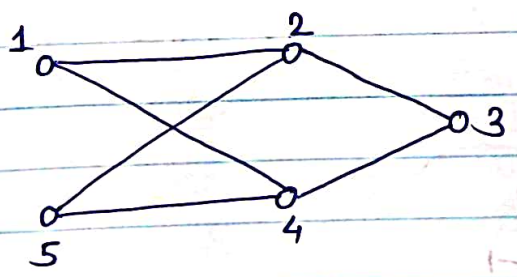


fig: Clique Decision problem

2) Node Cover Decision Problem: (NCDP)

The no. of vertices covering in any set called NCDP.

eg= if $G = \langle V, E \rangle$ is graph then we check for.



eg. if $S_1 = \{1,2\}$ Node cover decision problem=2

$S_2 = \{1,3,4\} \Rightarrow$ NCDP = 3

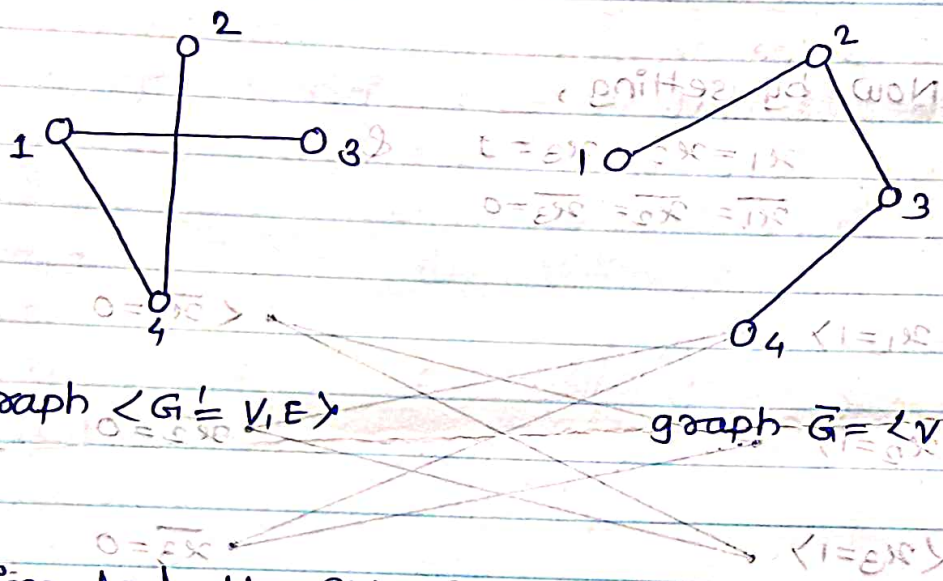
3) Complement of Graph:

If $G\langle V,E\rangle$ is any graph,

V = Vertices or Nodes

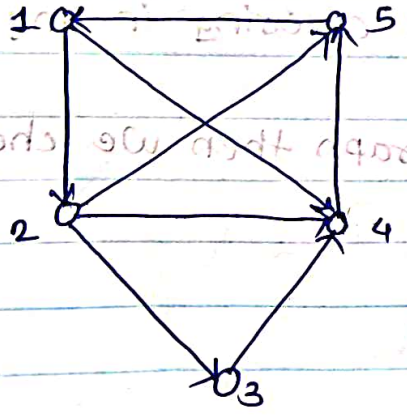
E = Edges, the complement of graph denotes all opposite edges of same graph.

(i.e $\bar{G} = \langle V, E' \rangle$)



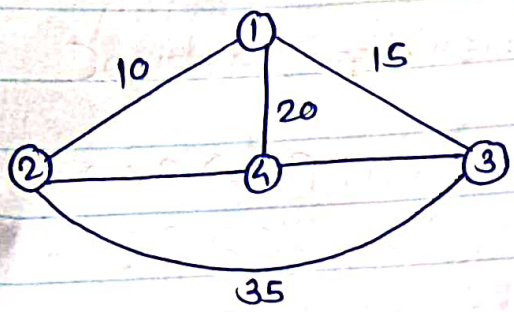
4) Directed Hamiltonian Acyclic (DHC)

If is graph $G\langle V,E\rangle$ where starting & ending Nodes are same,



Eg: 1-2-3-4-5-1

5) Travelling Salesperson Decision Problem :



It is the problem of dianamic programming in which we have to find shortest path that visit every node exactly 1 & return to the starting node.

6) AND / OR Graph Decision problem :