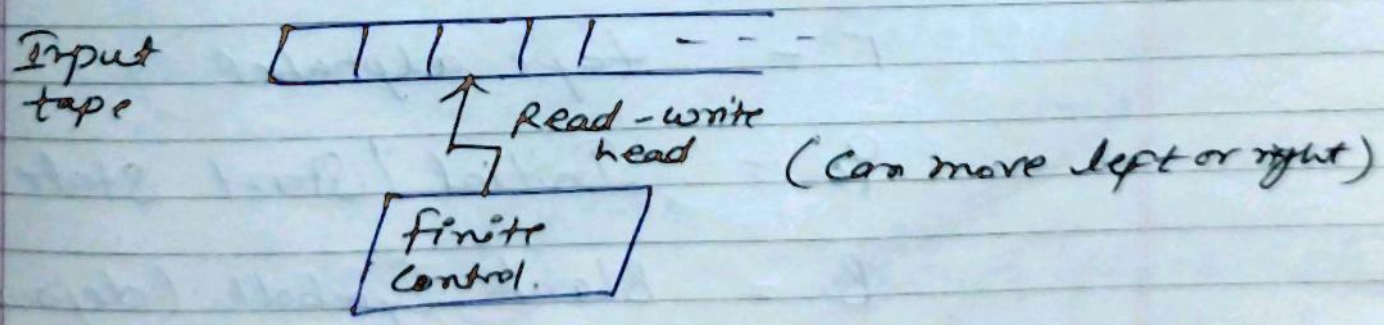
Unit V

Turing Machine

[TM > PDA > FA]

- It is most powerful mathematical model.
- It is proposed by Alan Turing in 1936.
- It is similar to finite automata but with an unlimited & unstructured memory (tape).
- It is a much more accurate model of a general purpose computer.
- It can do everything that a computer can do.
- It can both write on the tape & read from the tape.
- The read - write head of turing machine can move to both left & right direction.
- It has input tape of infinite length.



Scanned by Scanner Go

* Language recognize:-

- If turing machine enter final state & halts. String 'w' accepted.

- If turing machine halts & not find state. String 'w' rejected.

Note:- Turing machine can solve problem which are not solved by pushdown automata.

- Turing machine can also be used for calculating values like addition, substitution multiplication etc.

~~Numerical on TM~~

Formal Defination

- formally a turing machine is defined by 7 tupes.

$$M = \{Q, \Sigma, \Gamma, \delta, q_0, B, F\}$$

gamma

where

Q = finite set of state

Σ = finite set of input symbols

Γ = tape alphabet

q_0 = Initial / Start state

B = Blank Symboll (default symbol on input tape)

Scanned by Scanner Go

$F \subseteq Q$ → subset of = set of accepting or final state

δ = Transition function

$$\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times D$$

(State, tape alphabet) → (State, Symbol, direction)

Direction $D \Rightarrow$ Left or Right

* Design a turing machine that accept the following language

$$L = \{a^n b^n c^n / n > 0\}$$

Sol: $L = \{abc, aabbcc, aaabbbccc \ldots\}$

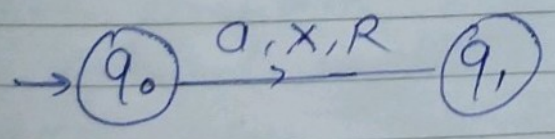
Input tape

B	B	a	a	a	...	b	b	b	...	c	c	c	B	B	...

↑ R/W head.

Let $w = aaabbbccc$

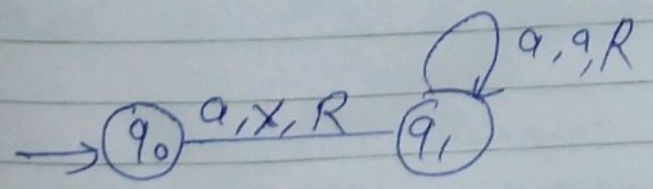
(i)



B	X	a	a	b	b	b	c	c	c

↑ R/W head.

(ii)

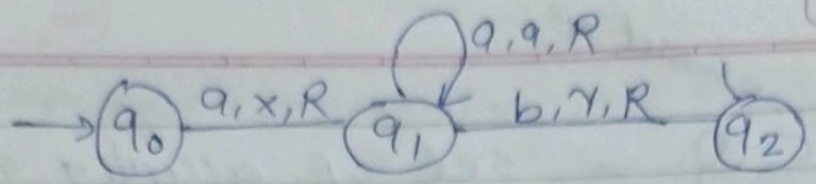


B	X	a	a	b	b	b	c	c	c

↑ R/W head.

Scanned by Scanner Go

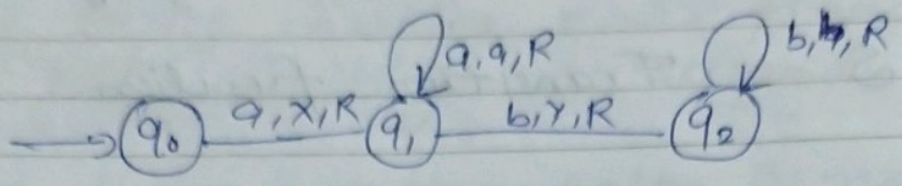
(iii)



B	x	a	a	Y	b	b	c	c

↑ R/W

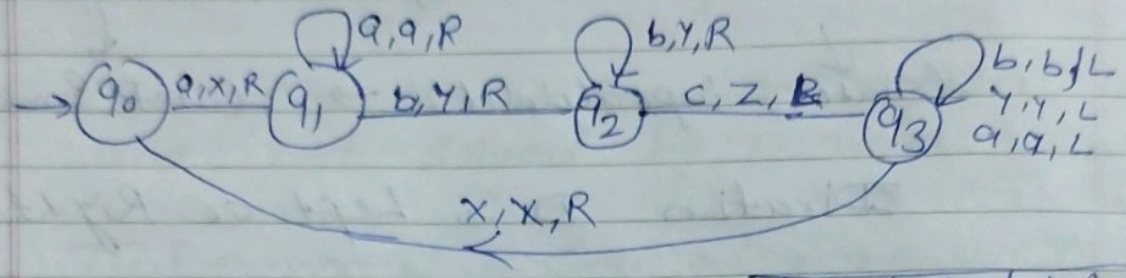
(iv)



B	x	a	a	Y	b	b	c	c	c

↑ R/W

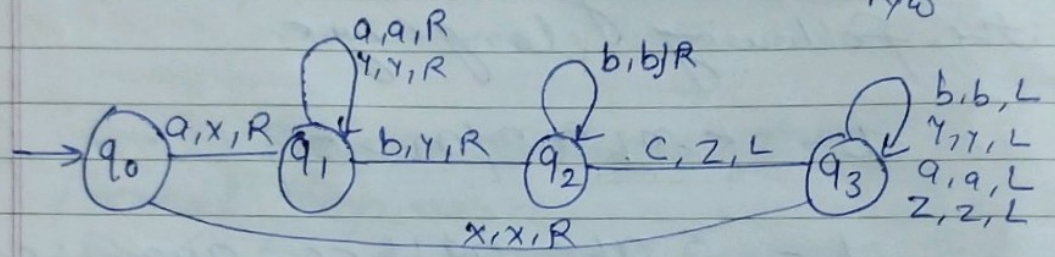
(v)



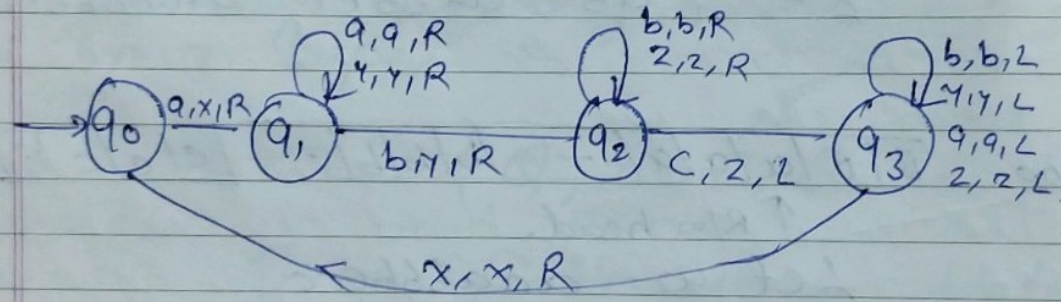
B	x	a	a	Y	b	b	Z	c	c

↑ R/W

(vi)



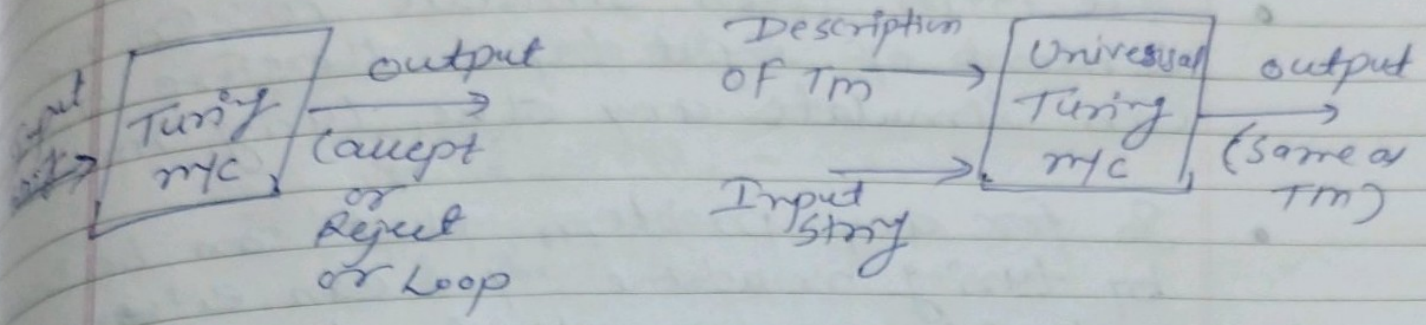
(vii)



B	x	x	x	Y	Y	Y	Z	Z	Z	B

Scanned by Scanner Go

Universal Turing machine:-



1) Hardwired m/c	1) Programmable TM
2) It does one task (execute one program)	2) It can do different type of task

- A universal turing machine is a ~~specified~~ special turing machine that can simulate the behavior of any turing machine.

- To design a universal turing machine, we have to increase the number of read/write heads, dimension of input tape & adding a special purpose memory in basic model of turing machine.

- The problem with turing machine is that a different machine must be constructed for every new computation to be performed, that is for every input output relation.

- To solve this problem of turing machine, univeral turing machine introduce, which

Scanned by Scanner Go

which takes description of a machine 'm' & input on the tape.

- The UTM, then simulate 'm' on the contents of input tape, therefore UTM can simulate any other turing machine

- So for any problem, that can be solved by turing machine you can either use a turing machine that directly solve the problem. 'or' you could use a UTM & give the description of TM that directly solve it.

- As a turing machine can solve any computing problem, a universal turing machine can be programmed to do the same

- UTM is a general purpose machine that can be used to compute any computable problem.

Scanned by Scanner Go

Multitape turing machine:-

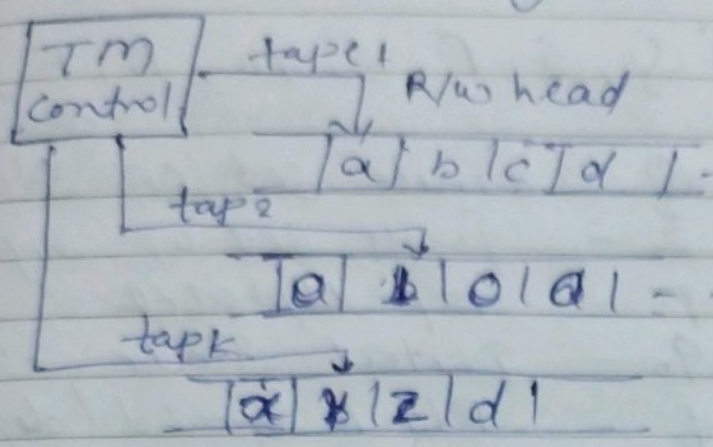
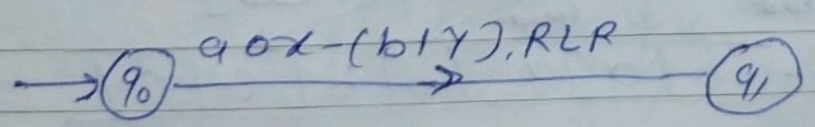


Fig:- Turing machine with k-no of tapes

- A Regular turing machine with several tape is called a multitape TM.
- Each tape has its own Read/write head. Each head can move independent of the others. It has no more additional power than single tape TM.
- It can perform same task as simple tape TM.
- We can build single tape TM corresponding to multiple TM.

or

Every language accepted by a k-tape TM is also accepted by a single tape TM.



$\delta(q_0, a, 0, x) = (q_1, b, 1, Y, R, L, R)$

Scanned by Scanner Go

- It is also defined by 7 tuples.

$$M = \{Q, \Sigma, \Gamma, \delta, q_0, B, F\}$$

where
Q - finite set of states
Σ - finite set of input symbol
Γ - tape alphabet
q_0 - Initial state
B - Blank symbol / default symb
F - set of final / acceptry state
δ = state transition function

$$\delta : Q \times \Gamma^k \longrightarrow Q \times \Gamma^k \times D$$

$\rightarrow (L, R)^k$
$\rightarrow$ 3 direction

$\boxed{\text{3 tapes than k value - 3}}$

☆ Multihead Turing machine

- A turing machine with multiple numbers of Read/write head is called as multihead TM.

- Each Read/write head works independently.

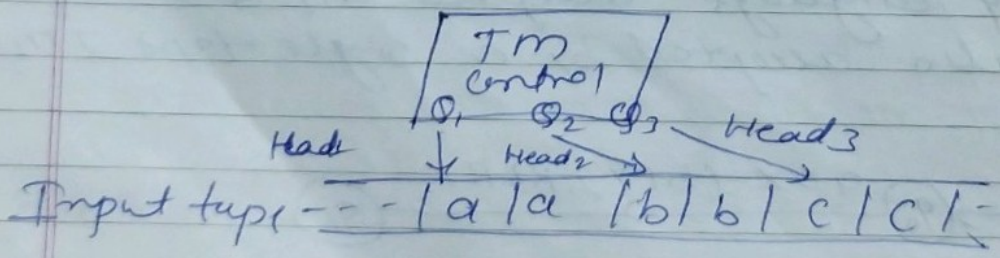


fig:- TM with 3 Read/write head.

Scanned by Scanner Go

Ex:- Head 1 - Read, Head 2 - write, Head 3 - Stationary.

- Here there is single tape with multiple R/W heads.

- In every state only one head may be used so, for k-head we have a partition of state, $q_1, q_2, \ldots q_k$.

  q_i: State we the ith head.

- A multihead turing machine is equivalent to normal T.M.

- Multihead TM defined by 7 tupes.

$$M = \{Q, \Sigma, \Gamma, \delta, q_0, B, F\}$$

where

Q - finite set of states

Σ = set of input symbol

Γ - tap alphabet

q_0 = Initial state

B = Blank symbol

F = set of final/accepting state

$\delta : Q \times \Gamma = Q \times \Gamma \times D$

3 read/write head → 3 direction

Scanned by Scanner Go

☆ Multidimensional Turing Machine:

- A multidimensisional turing machine has a tape that can extend infinitely in more than one ~~direction~~ dimension.

Eg: Two dimensional turing machine.

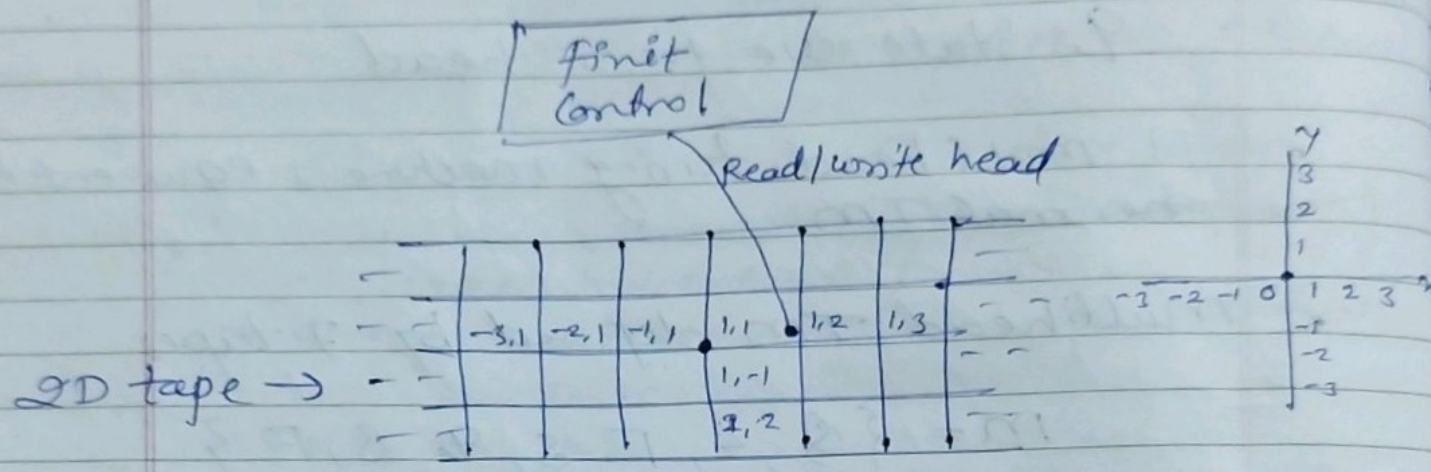


fig: Two dimensional TM

- The transition function must specify movement along both dimensions.

- It is also defined by 7 tuples as normal TM

$$M = \{Q, \Sigma, \Gamma, q_0, B, F, \delta\}$$

but difference in transition function which is defined as.

$$\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R, U, D\}$$

L = Left, R = Right

- Multidimensional turing machine have U = UP. D = Down the same power with standard TM.

Scanned by Scanner Go

NP Completeness :-

Problem

Polynomial Exponential

1) Polynomial :- A problem is said to be polynomial if there exists an algorithm that solves the problem in time

$$T(n) = \overset{\text{Big O}}{O}(n^c)$$

where c is constant. $[= n^2$ Polyn.

Examples:
- Sorting - $O(n \log n) = O(n^2)$
- Searching - $O(n)$
- All pairs shortest path $O(n^3)$
- Minimum cost spanning tree $O(E \log E) = O(E^2)$

(P class problem)

2) Exponential :- A problem is said to be exponential if no polynomial time algorithm can be developed for it. or algorithm solve the problem in time

$$T(n) = O(2^n) = O(k^n)$$

Not efficient Algo

Examples:- K - constant number
- 0/1 knapsack problem
- Travelling sale man problem
- Hamiltonian problem.

(NP class) (non deterministic)

Scanned by Scanner Go

GoodLuck Page No
Date

* What are P, NP, NP Hard & NP-Complete problem.

- **P Class (Polynomial) :-** P is set of problem that can be solved in polynomial time using deterministic algorithm

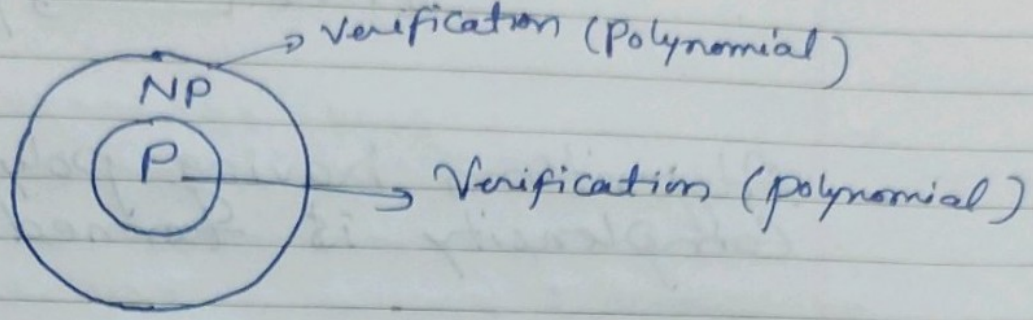
  - If a problem can be solved in polynomial time on a regular computer, that problem is known as class 'P' problem.
  - As 'P' class problem verified in polynomial time also, hence it also a subset of NP class problem.
  - Solved & verify in polynomial time, easy to solve easy to verify, It takes polynomial time to solved & verify.
  - $P \subseteq NP$ (as it is also verified in polynomial time).

- **NP Class (Non-deterministic polynomial) :-** NP is a set of problems that can be solved in polynomial time using non-deterministic algorithm.

  - NP problem are problem which can be solved in polynomial time on non-deterministic turing machine.
  - It can be verified in polynomial time.

Scanned by Scanner Go

- NP class problem, it is verified in polynomial time, Hard to solve but easy to verify, it take exponential time to solve.



- P is subset of NP class :-

Any problem that can be solved by deterministic machine in polynomial time, can also be solved by non-deterministic machine in polynomial time.

- NP problem can be solved by polynomial time using "non-deterministic algo", a magical algo that always make a right guess among the given set of choices.

Ex:- problem - Graph Coloring :-

Color a graph in such a way that each adjacent node has different color.

→ This is a NP, class problem it is solved in exponential time, but solution verification is easy, because once a solution is provided its much easier to check for correctness.

Scanned by Scanner Go

problem → Solution time & Verification time

- Solution time → polynomial time → P class problem
- Solution time → Exponential time
- Verification time → polynomial time → NP class problem

- Algorithm having polynomial time complexity is termed as efficient.

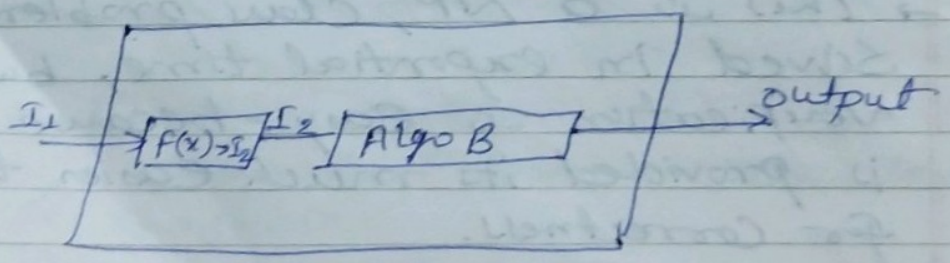
☆ NP Hard & NP Complete :-

Reduction :- Let we have 2 decision problem P_1 & P_2.

P_1 problem → Input I_1 → Algorithm A(?)

P_2 problem → Input I_2 → Algorithm B (Written)

Suppose algorithm B can be used to solve problem A.



Algorithm A

Scanned by Scanner Go

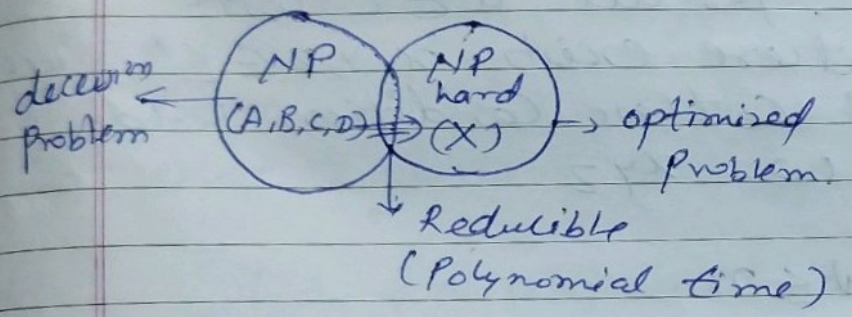
- Problem P_1 is reducible to problem P_2, if there is function which convert input of A into input of B & solution of that instance provide solution to problem

$$P_1 \propto P_2$$

converted

Here we assume conversion cost is also polynomial time.

→ **NP Hard** :- A problem is NP Hard if every problem in NP can be polynomial reduced to it.

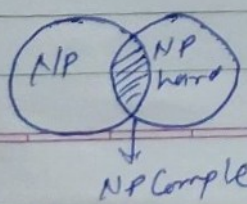


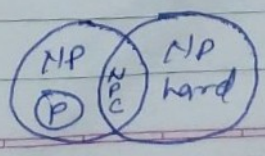
$A \propto X$
$B \propto X$ X is NP hard.
$C \propto X$
$D \propto X$

⇒ **NP Complete** :- A decision problem L is NP Complete if

yes/no

1) L is in NP.
2) L is NP-hard (every problem in NP is reducible to L in polynomial time)





Relation b/w P, NP, NP Complete, NP hard

GoodLuck Page No
Date

☆ Pumping Lemma :-

It is used to prove that a language is not regular.

- A regular language can be designed using finite automata.
- A non regular language can not be designed using finite automata.
- Pumping lemma is not used to prove that a language is regular.

→ **Pumping lemma :-** for any regular language L, there exists an integer 'n' (Pumping length) such that, for all $w \in L$ with $|w| \geq n$, there exists $x, y, z \in \Sigma^*$, such that all these condition holds.

$$w = xyz$$

i) $|xy| \leq n$
ii) $|y| \geq 1$
iii) $xy^iz \in L \quad \forall\ i \geq 0$

- pumping length 'n' depends on language.
- If L is regular language, select any string w. of length $\geq n$ then divide it into 3 parts x, y, z & then check the above 3 condition if all satisfied then pumping lemma is satisfied.

Scanned by Scanner Go

- In pumping lemma, string 'y' is pumped (inserted) any number of times (i) but still we get the string which belong to L. If L is regular language.

Note:- If a language is regular, it always satisfies pumping lemma.

- If after applying pumping lemma, there exists atleast one string made from pumping (xy^iz, $i \geq 0$) which is not in L, then we say that given language is definetly non regular.

☆ Numerical on pumping lemma.

Q. Let $L = \{a^n b^n / n \geq 0\}$ prove that L is not regular (using pumping lemma).

Sol:- $L = \{\varepsilon, ab, aabb, aaabbb \text{------}\}$

Step I:- Assume L is a regular language & L have a pumping length (P)

Step II:- Take a string $w = a^P b^P$

Step III:- Divide w into 3 parts x, y & z

Let $P = 5$

Scanned by Scanner Go

Case I

y is in the a's part

$w = \underbrace{a}_{x}\underbrace{aaaa}_{y}\underbrace{bbbbb}_{z}$

Case II

Y is in the b's part

$\underbrace{aaaaa}_{x}\underbrace{bbbb}_{y}\underbrace{b}_{z}$

Case III

y has both a's & b's

$\underbrace{aaaa}_{x}\underbrace{abb}_{y}\underbrace{bbb}_{z}$

$$\boxed{\begin{array}{l} |w| \geq P \\ |w| = 10 > 5 \end{array}}$$

(i) $|xy| \leq P$

$|\underbrace{aa}_{x}\underbrace{aaa}_{y}| = 5 \leq P(5)$

(a) $|xy| \leq P$

$|aaaaabbbb| \leq 5$

$9 \leq 5$

✗

(b) $|xy| \leq P$

$|aaaaabb| \leq 5$

$7 \leq 5$

✗

(ii) $|y| \geq 1$

$|aaaa| \geq 1$

(iii) $xy^iz \in L \quad \forall\, i \geq 0$

$a(aaaa)^i bbbbb$

$i = 0$ abbbbb ✗ not in L.

$i = 1$ aaaaabbbbb ✓

$i = 2$ aaaaaaaaabbbbb ✗ in not in L.

- If L is regular than it must satisfy pumping lemma but $xy^iz \notin L$ for $i = 0$.

So the given language is not regular.

Scanned by Scanner Go

Enumerable Language

★ Recursive ~~Grammer~~ Language or Type 0 ~~Grammer~~.

- It is generated by type 0 grammer.
- A recursive enumerable language can be accepted / recognized by turing machine.

- The language $L = \Sigma^*$ is recursive enumerable if there exist a turing machine 'm' that satisfy the following conditions.-

  i) if $w \in L$ then 'm' halt in the final state.
  ii) if $w \notin L$ then 'm' halts in non final state or loops forever.

- Recursive enumerable language are also called as turing recognizable language because a turing machine can recognize a string in the language (accept it).

- But turing machine might not be able to decide if a string is not in the language since the machine might loop for that input.

- A half solvable language is commonly refered to as recursively enumerable.

★ Recursive language / Decidable language

- A language is called recursive if there is a turing machine that accepts on any input within the language & rejects on

Scanned by Scanner Go

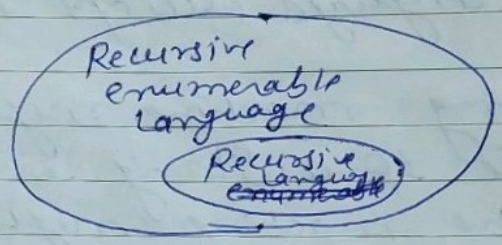
any other input that are not part of language.

- It means, turing machine for recursive language halt on every string (no loop)

- Example
  $L = \{a^n b^n c^n \mid n \geq 1\}$ is recursive language

  As we can construct a turing machine which will move to final state if the string is of the form $a^n b^n c^n$ else it move to non-final state so turing machine ~~halt~~ will always halt.

- Recursive language is a subset of recursive enumerable language.



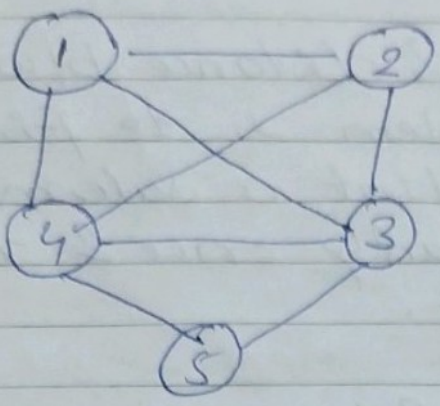
- If we have a deterministic turing machine for a language than language is called recursive language.

- The problem ~~which~~ whose language is recursive is called decidable otherwise the problem is undecidable.

Scanned by Scanner Go

(Backtracking algo)

Hamiltonian Cycle:-

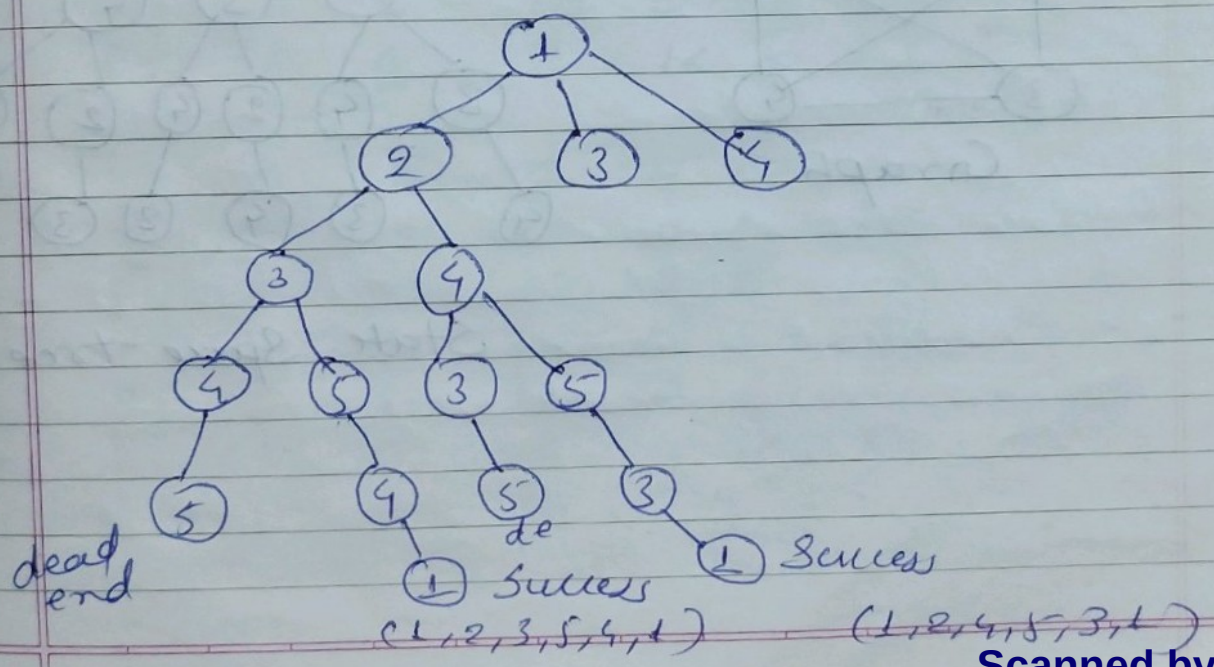


1, 2, 3, 5, 4, 1
1, 2, 4, 5, 3, 1
1, 3, 5, 4, 2, 1
1, 4, 5, 3, 2, 1

- Hamiltonian cycle is a path in a given graph that visit each vertex exactly once & there is edge/path between first & last vertex.

- There may be any number of hamiltonian cycle exist in a graph.

- Here we have to determine weather a given graph contains hamiltonian cycle or not. If it contain, then print the path.



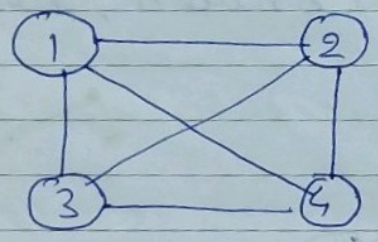
Scanned by Scanner Go

Imp

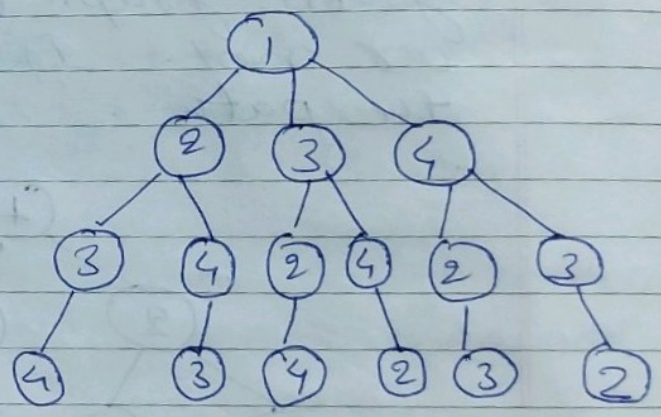
☆ Travelling Salesman problem:

Given a set of cities & distance b/w every pair of cities, the problem is to find the shortest possible route that a salesman has to visits every city exactly once & returns to the starting part.

- Assuming a salesman has to travel 'n' cit
- He starts from a particular city, visit eac city once & then returns to his starting point.
- The objective is to select the sequence in which the cities are visited (with shortest possible route) in such a way that his total travelling cost is minimized



Graph

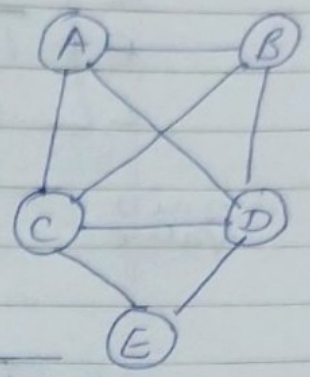


State Space tree

Scanned by Scanner Go

Q. A Saleman has to visit five cities A, B, C D, E, the distance b/w the five cities is as follows

$$\begin{array}{c|ccccc} & A & B & C & D & E \\ \hline A & \infty & 20 & 30 & 10 & 11 \\ B & 15 & \infty & 16 & 4 & 2 \\ C & 3 & 5 & \infty & 2 & 4 \\ D & 19 & 6 & 18 & \infty & 3 \\ E & 16 & 4 & 7 & 16 & \infty \end{array}$$



If the Saleman Starts from City A & has to come back to his starting point, will travell route should be select, so that total distance travelled is minimum.

Cost adjacency matrix

Sol:- Row Minimization

$$\begin{array}{c|ccccc|c} & A & B & C & D & E & \\ \hline A & \infty & 20 & 30 & \underline{10} & 11 & 10 \\ B & 15 & \infty & 16 & 4 & \underline{2} & 2 \\ C & 3 & 5 & \infty & \underline{2} & 4 & 2 \\ D & 19 & 6 & 18 & \infty & \underline{3} & 3 \\ E & 16 & \underline{4} & 7 & 16 & \infty & 4 \\ \hline & & & & & & \text{Total} - 21 \end{array}$$

$$\begin{array}{c|ccccc|l} & A & B & C & D & E & \\ \hline A & \infty & 10 & 20 & 0 & 1 & R_1 = R_1 - 10 \\ B & 13 & \infty & 14 & 2 & 0 & R_2 = R_2 - 2 \\ C & 1 & 3 & \infty & 0 & 2 & R_3 = R_3 - 2 \\ D & 16 & 3 & 15 & \infty & 0 & R_4 = R_4 - 3 \\ E & 12 & 0 & 3 & 12 & \infty & R_5 = R_5 - 4 \end{array}$$

Scanned by Scanner Go

Column minimization

	A	B	C	D	E
A	∞	10	20	0	1
B	13	∞	14	2	0
C	1	3	∞	0	2
D	16	3	15	∞	0
E	12	0	3	12	∞
min Value	1	0	3	0	0

$= 1 + 3 = 4$

	1 A	2 B	3 C	4 D	5 E
1 A	∞	10	17	0	1
2 B	12	∞	11	2	0
3 C	0	3	∞	0	2
4 D	15	3	12	∞	0
5 E	11	0	0	12	∞

$= C$

Fully Reduced matrix

Cost of reduced matrix

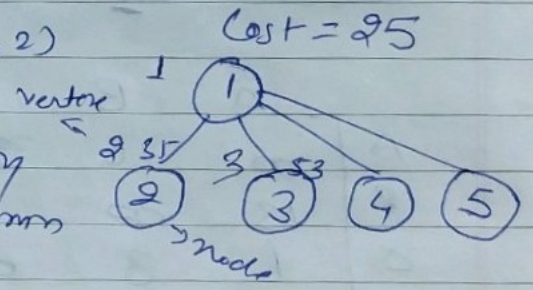
Row minimization + Column

$21 + 4$

$= 25$

- Cost of node 2 (1, 2)

Cost = 25



Replace each ~~entry~~ entry of 1st row & 2nd Column of Reduced matrix to ∞. Also made (2, 1) to ∞

$$\begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & \infty & \infty & \infty & \infty & \infty \\ 2 & \infty & \infty & 11 & 2 & 0 \\ 3 & 0 & \infty & \infty & 0 & 2 \\ 4 & 15 & \infty & 12 & \infty & 0 \\ 5 & 11 & \infty & 0 & 12 & \infty \end{array}$$

Here all row & Column are reduced each contain zero Value

$Cost(2) = C(1,2) + Cost(1) + \text{Reduced Cost}$

$= 10 + 25 + 0$

$Cost(2) = 35$

- Cost of node 3 :- (1,3) Path

Replace each entry of 1st row & 3rd Column of reduced matrix to ∞. also made (3,1) to ∞.

$$\begin{array}{c|ccccc|c} & 1 & 2 & 3 & 4 & 5 & \\ \hline 1 & \infty & \infty & \infty & \infty & \infty & 0 \\ 2 & 12 & \infty & \infty & 2 & 0 & 0 \\ 3 & \infty & 3 & \infty & 0 & 2 & 0 \\ 4 & 15 & 3 & \infty & \infty & 0 & 0 \\ 5 & 11 & 0 & \infty & 12 & \infty & 0 \\ \hline \text{min Value} & 11 & 0 & 0 & 0 & 0 & \end{array}$$

$$\begin{array}{c|ccccc} 1 & \infty & \infty & \infty & \infty & \infty \\ 2 & 1 & \infty & \infty & 2 & 0 \\ 3 & \infty & 3 & \infty & 0 & 2 \\ 4 & 4 & 3 & \infty & \infty & 0 \\ 5 & 0 & 0 & \infty & 12 & \infty \end{array}$$

$G = G - 11$

$Cost(3) = C(1,3) + Cost(1) + \text{Reduced Cost}$

$= 17 + 25 + 11$

$= 53$

Scanned by Scanner Go

- Cost of node 4 :- make 1st row & 4th column of reduced matrix to ∞ and also c(4,1) to ∞

$$
\begin{array}{c|ccccc|c}
 & 1 & 2 & 3 & 4 & 5 & \\
\hline
1 & \infty & \infty & \infty & \infty & \infty & 0 \\
2 & 12 & \infty & 11 & \infty & 0 & 0 \\
3 & 0 & 3 & \infty & \infty & 2 & 0 \\
4 & \infty & 3 & 12 & \infty & 0 & 0 \\
5 & 11 & 0 & 0 & \infty & \infty & 0 \\
\hline
 & 0 & 0 & 0 & 0 & 0 &
\end{array}
$$

$$
\begin{aligned}
Cost(4) &= c(1,4) + cost(1) + \text{Reduced cost} \\
&= 0 + 25 + 0 \\
&= 25
\end{aligned}
$$

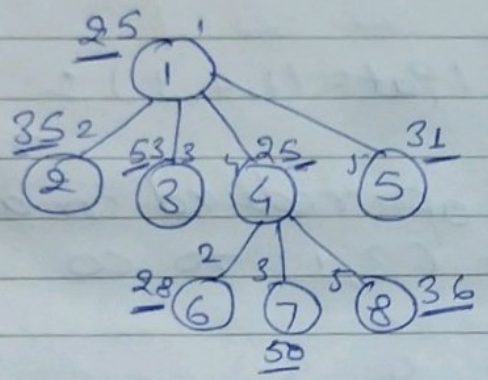
- Cost of node 5 :- make 1st row & 5th Column of reduced matrix to ∞ & also c(5,1) to ∞.

$$
\begin{array}{c|ccccc|c}
 & 1 & 2 & 3 & 4 & 5 & \\
\hline
1 & \infty & \infty & \infty & \infty & \infty & \\
2 & 12 & \infty & 11 & 2 & \infty & -2 \\
3 & 0 & 3 & \infty & 0 & \infty & -0 \\
4 & 15 & 3 & 12 & \infty & \infty & -3 \\
5 & \infty & 0 & 0 & 12 & \infty & -0 \\
\hline
 & 0 & 0 & 0 & 0 & 0 &
\end{array}
$$

Scanned by Scanner Go

$$
\begin{array}{c|ccccc}
 & 1 & 2 & 3 & 4 & 5 \\
\hline
1 & \infty & \infty & \infty & \infty & \infty \\
2 & 10 & \infty & 9 & 0 & \infty \\
3 & 0 & 3 & \infty & 0 & \infty \\
4 & 12 & 0 & 9 & \infty & \infty \\
5 & \infty & 0 & 0 & 12 & \infty
\end{array}
\qquad
\begin{array}{l}
\\ \\ R_1 = R_1 - 2 \\ \\ R_3 = R_1 - 3
\end{array}
$$

$$
\begin{aligned}
Cost(5) &= C(1,5) + Cost(1) + \text{Reduced Cost} \\
&= 1 + 25 + 5 \\
&= 31
\end{aligned}
$$



(least cost node explore)

Now for further calculation we use reduced matrix of node 4.

$$
C = \begin{array}{c|ccccc}
 & 1 & 2 & 3 & 4 & 5 \\
\hline
1 & \infty & \infty & \infty & \infty & \infty \\
2 & 12 & \infty & 11 & \infty & 0 \\
3 & 0 & 3 & \infty & \infty & 2 \\
4 & \infty & 3 & 12 & \infty & 0 \\
5 & 11 & 0 & 0 & \infty & \infty
\end{array}
$$

- Cost of node 6 :- (1, 4, 2) :- make 4th row & 2nd column of reduced matrix to ∞ & also make $c(2,1)$ to ∞

Scanned by Scanner Go

$$
\begin{array}{c|ccccc}
 & 1 & 2 & 3 & 4 & 5 \\
\hline
1 & \infty & \infty & \infty & \infty & \infty \\
2 & \infty & \infty & 11 & \infty & 0 \\
3 & 0 & \infty & & \infty & 2 \\
4 & \infty & \infty & \infty & \infty & \infty \\
5 & 11 & \infty & 0 & \infty & \infty
\end{array}
$$

$Cost(6) = c(4,2) + Cost(4) + \text{Reduced Cost}$

$= 3 + 25$

$Cost(6) = 28$

- Cost of node 7 :- (Path (1,4,3) :-

make 4th row & 3rd column of reduced matrix to ∞ & also make (3,1) to ∞.

$$
\begin{array}{c|ccccc}
 & 1 & 2 & 3 & 4 & 5 \\
\hline
1 & \infty & \infty & \infty & \infty & \infty \\
2 & 12 & \infty & \infty & \infty & 0 \\
3 & \infty & 3 & \infty & \infty & 2 \\
4 & \infty & \infty & \infty & \infty & \infty \\
5 & 11 & 0 & \infty & \infty & \infty \\
\hline
 & 11 & & & & 2
\end{array}
$$

$$
\begin{array}{c|ccccc}
 & 1 & 2 & 3 & 4 & 5 \\
\hline
1 & \infty & \infty & \infty & \infty & \infty \\
2 & 1 & & \infty & \infty & \\
3 & \infty & \underline{1} & \infty & \infty & 0 \\
4 & \infty & & \infty & \infty & \\
5 & 0 & 0 & \infty & \infty & \infty
\end{array}
\quad R_3 = R_3 - 2
$$

$C_4 = C_4 - 11$

Scanned by Scanner Go

$Cost(7) = c(4,3) + Cost(4) + \text{Reduced Cost}$
$= 12 + 25 + 11 + 2$
$= 50$

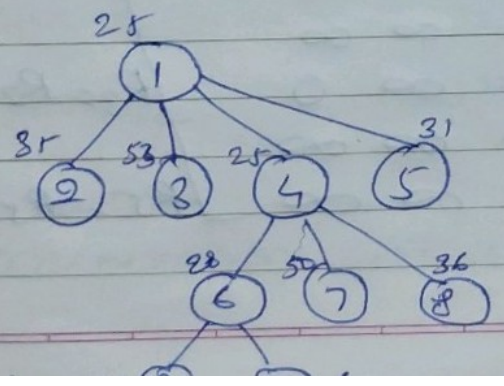
- Cost of node 8 :- (1, 4, 5) :-

make 4th row & 5th column of reduced matrix to ∞ & also make (5, 1) to ∞.

$$\begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & \infty & \infty & \infty & \infty & \infty \\ 2 & 12 & \infty & 11 & \infty & \infty \\ 3 & 0 & 3 & \infty & \infty & \infty \\ 4 & \infty & \infty & \infty & \infty & \infty \\ 5 & \infty & 0 & 0 & \infty & \infty \end{array} \quad -11$$

$$\begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & \infty & \infty & \infty & \infty & \infty \\ 2 & 1 & \infty & 0 & \infty & \infty \\ 3 & 0 & 3 & \infty & \infty & \infty \\ 4 & \infty & \infty & \infty & \infty & \infty \\ 5 & \infty & 0 & 0 & \infty & \infty \end{array} \quad R_2 = R_2 - 11$$

$Cost(8) = c(4,5) + Cost(4) + \text{Reduced Cost}$
$= 0 + 25 + 11$
$= 36$



Scanned by Scanner Go

- ~~Cost of node (9) :- path (1, 4, 2, 3)~~

for further calculation we will use reduced cost matrix of node 6.

$$C = \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & \infty & \infty & \infty & \infty & \infty \\ 2 & \infty & \infty & 11 & \infty & 0 \\ 3 & 0 & \infty & \infty & \infty & 2 \\ 4 & \infty & \infty & \infty & \infty & \infty \\ 5 & 11 & \infty & 0 & \infty & \infty \\ & & & 11 & & \end{array}$$

- Cost of node 9 :- path (1, 4, 2, 3) :- make 2nd row & 3rd column to ∞ of reduced matrix ∞ & also make (3, 1) to ∞

$$\begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & \infty & \infty & \infty & \infty & \infty \\ 2 & \infty & \infty & \infty & \infty & \infty \\ 3 & \infty & \infty & \infty & \infty & 2 \\ 4 & \infty & \infty & \infty & \infty & \infty \\ 5 & 11 & \infty & \infty & \infty & \infty \end{array} \begin{array}{l} \\ \\ 2 \\ \\ 11 \end{array}$$

$$\begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & \infty & \infty & \infty & \infty & \infty \\ 2 & \infty & \infty & \infty & \infty & \infty \\ 3 & \infty & \infty & \infty & \infty & 0 \\ 4 & \infty & \infty & \infty & \infty & \infty \\ 5 & 0 & \infty & \infty & \infty & \infty \end{array} \begin{array}{l} \\ \\ R_3 = R_3 - 2 \\ \\ R_5 = R_5 - 11 \end{array}$$

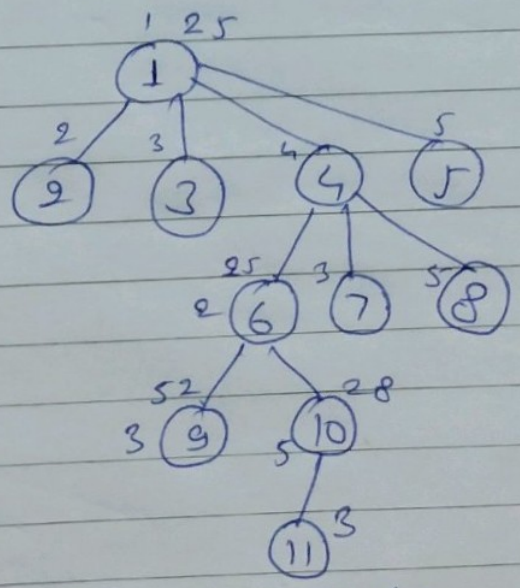
Scanned by Scanner Go

Cost (9) = C(2,3) + cost(6) + Reduced cost
= 11 + 28 + 2 + 11
Cost (9) = 52

- Cost of node 10 :- path (1, 4, 2, 5) :- make 2nd row 5th column of Reduced matrix ∞ & also make (5,1) to ∞.

$$\begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & \infty & \infty & \infty & \infty & \infty \\ 2 & \infty & \infty & \infty & \infty & \infty \\ 3 & 0 & \infty & \infty & \infty & \infty \\ 4 & \infty & \infty & \infty & \infty & \infty \\ 5 & \infty & \infty & 0 & \infty & \infty \end{array}$$

Cost (10) = C(2,5) + cost(6) + Reduced cost
= 0 + 28 + 0
Cost (10) = 28



minimum cost of path = (1, 4, 2, 5, 3, 1)
= 28

Scanned by Scanner Go