

GREEDY TECHNIQUE

NOTE:- Most of the problems in greedy contain 'n' no. of inputs and our objective is finding a subset which will satisfy our conditions and optimizes our goal.

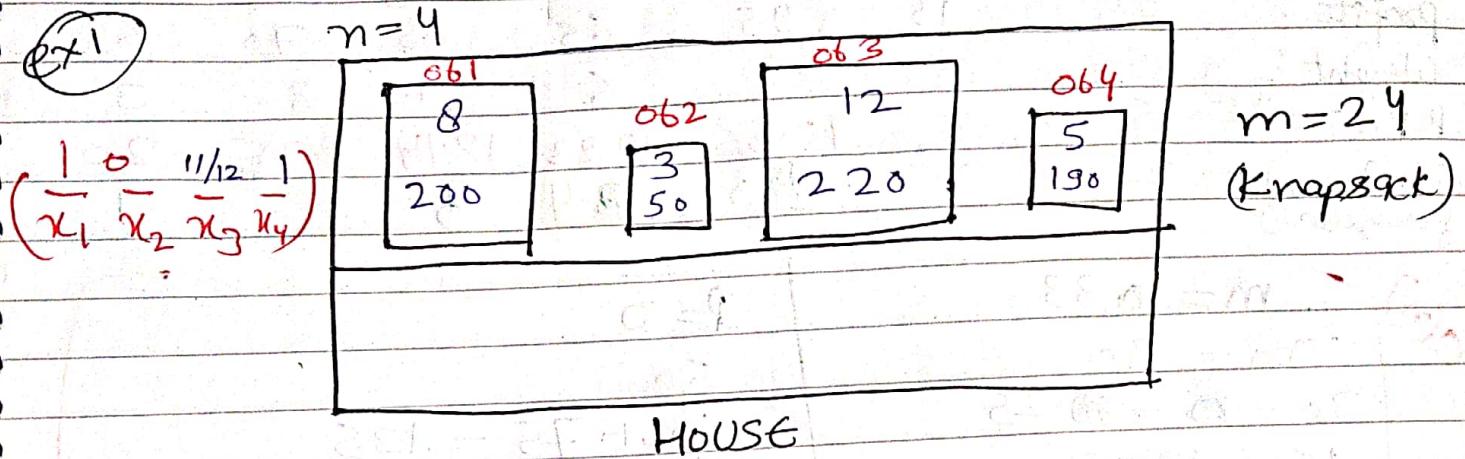
Basics of Greedy Technique :-

- 1.) Solution space:- set of all possible solutions over the given input is known as solution space.
- 2.) Feasible solution:- set of all those solutions, which will satisfy our conditions is known as feasible solution.
- 3.) Optimal solution:- Those feasible solution, which will optimise our goal is known as optimal solution.

APPLICATIONS OF GREEDY TECHNIQUE:-

1. Knapsack
2. Job Sequencing with deadlines
3. Huffman Coding
4. Optimal Merge Pattern
5. Minimum Cost Spanning Tree
 - (i) Prim's
 - (ii) Kauskal
6. Single Source Shortest Path
 - i.)
 - ii.)
 - iii.)

1. KNAPSACK PROBLEM



Greedy about both (Profit & Weight)

$$\begin{aligned} 061: \quad 8 &\rightarrow 200 \\ &1 \rightarrow 200 = 25 \\ &\hline 8 \end{aligned}$$

$$\begin{aligned} 063: \quad 12 &\rightarrow 220 \\ &1 \rightarrow \frac{220}{12} = 18.3 \end{aligned}$$

$$\begin{aligned} 062: \quad 3 &\rightarrow 50 \\ &1 \rightarrow 50 = 16.6 \\ &\hline 3 \end{aligned}$$

$$\begin{aligned} 064: \quad 5 &\rightarrow 190 \\ &1 \rightarrow \frac{190}{5} = 38 \end{aligned}$$

$$M=24$$

$$24 - 1 \times 5 = 19$$

$$19 - 1 \times 8 = 11$$

$$11 - \frac{11}{12} \times 12 = 0$$

$$P=0$$

$$0 + 1 \times 190 \Rightarrow 190$$

$$190 + 1 \times 200 \Rightarrow 390$$

$$390 + \frac{11}{12} \times 220 = \underline{\underline{591.66}}$$

$x_1 \ x_2 \ x_3 \ x_4 \ x_5 \ x_6 \ x_7$

8
8

(ex2) $n = 7$, $m = 33$

objects:	o_{b_1}	o_{b_2}	o_{b_3}	o_{b_4}	o_{b_5}	o_{b_6}	o_{b_7}
profits:	75	15	95	65	85	70	60
weight:	5	3	9	6	7	8	3
P.P.U:	$\frac{15}{5} = 3$	$\frac{15}{3} = 5$	$\frac{95}{9} = 10.56$	$\frac{65}{6} = 10.83$	$\frac{85}{7} = 12.14$	$\frac{70}{8} = 8.75$	$\frac{60}{3} = 20$
$O(n \log n)$ <small>{averaged}</small>	2	2	5	4	3	6	1

$m = 33$

$m = 30$

$30 - 5 = 25$

~~15 - 7 =~~

$25 - 7 = 18$

$18 - 6 = 12$

$12 - 9 = 3$

~~3 - 0~~

$3 - \frac{3 \times 8}{8} = 0$

$p = 0$

60

$60 + 75 = 135$

$135 + 85 = 220$

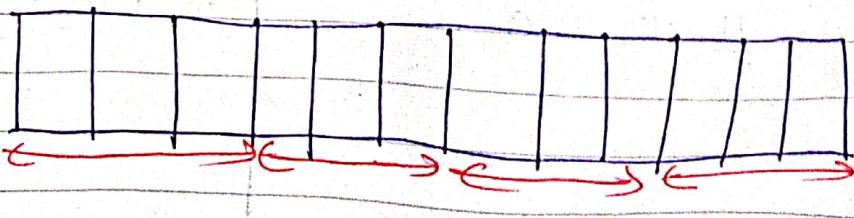
$220 + 65 = 285$

$285 + 95 = 380$

$380 + \frac{3 \times 70}{8} = 406.25$

$$\left(\begin{array}{ccccccc} 1 & 0 & 1 & 1 & 1 & \frac{3}{8} & 1 \\ x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 \end{array} \right)$$

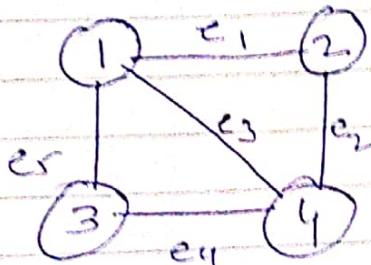
1. $O(n)$
2. $O(n \log n)$
3. $O(n)$



$O(n \log n)$ [Ec]

SINGLE SOURCE SHORTEST PATH :-

$G(V, E)$ \rightarrow set of edges
Vertices



$$V = \{1, 2, 3, 4\}$$

$$E = \{e_1, e_2, e_3, e_4, e_5\}$$

Types of graph

Simple

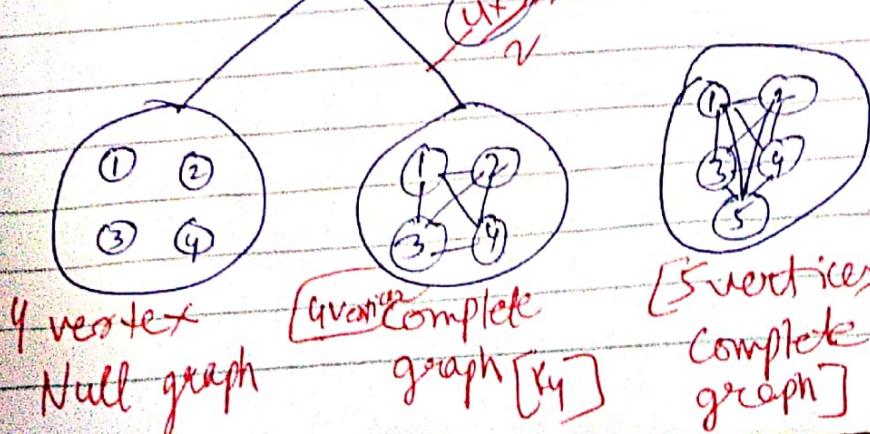
- 1.) No self loop
- 2.) No parallel edge

Multi graph

- 1.) Self loop
- 2.) Parallel Edge.

- A simple graph with N vertices, Maximum degree of any vertices $(n-1)$ & minimum degree '0'.
- A multigraph with N vertices can have Maximum degree of any vertices ∞ , minimum degree '0'.

TYPES OF SIMPLE GRAPH:



4 vertex
Null graph

4 vertex
Complete
graph [K4]

5 vertices
Complete
graph

$$E(K_n) = \frac{n(n-1)}{2}$$

Let $G(V, E)$ be a Simple Graph.

$$|E| \leq \frac{V(V-1)}{2}.$$

$$\leq C \cdot V^2$$

$$E = O(V^2)$$

$$\log E = O(\log V)$$

$$V \log E$$

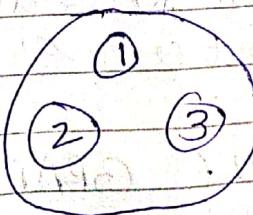
$$\Downarrow$$

$$O(V \log V)$$

No. of simple graphs possible with 'n' vertices.

$$n=3 \\ \text{Max edge} = \frac{3 \times 2}{2} = 3$$

e_1	e_2	e_3
0	0	0
0	0	1
0	1	0
1	0	0
1	0	1
1	1	0
0	1	1
1	1	1



Space complexity = $O(v^2)$ - Adjacency Matrix
 $O(v+e)$ - Adjacency List

161

$n=5$

$$\text{Max edges} = \frac{5 \times 4}{2} = 10 \text{ edges}$$

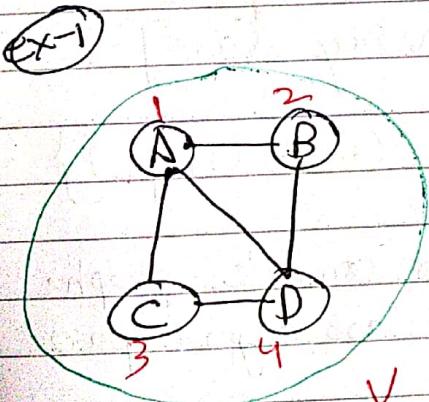
2^{10} - simple graphs.

No. of simple graphs using \Rightarrow
 n vertices

$$\boxed{\frac{n(n-1)}{2}}$$

GRAPH REPRESENTATIONS:-

1.) Adjacency Matrix



2.) Adjacency List

1	2	3	4
1	0	1	1
2	1	0	0
3	1	0	0
4	1	1	0

$\rightarrow O(v^2)$
same
& every case

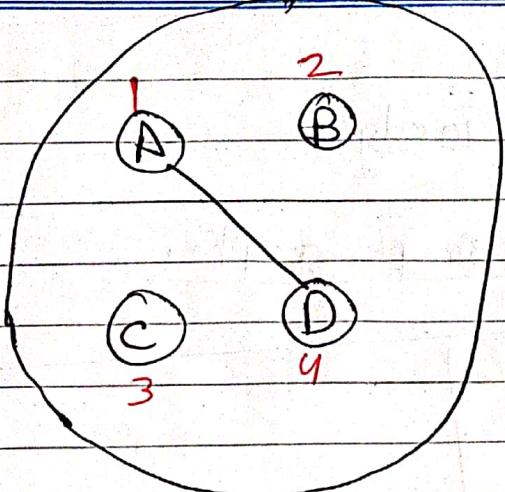
$$= v + 2e \\ = O(v + e)$$



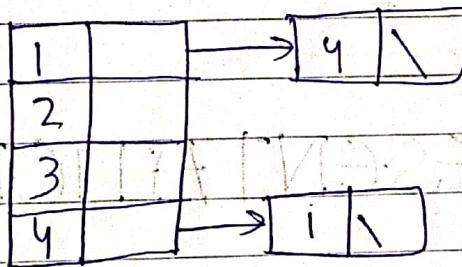
By default Adjacency List.

Less edges - sparse graph.
More edge - (complete graph) - Dense graph.

Ex-2



	1	2	3	4
1	0	0	0	1
2	0	0	0	0
3	0	0	0	0
4	1	0	0	0



Adjacency Matrix Representation $\rightarrow O(V^2)$ space
[Every Case]

{ It is best for more edges - (Dense Graph) }

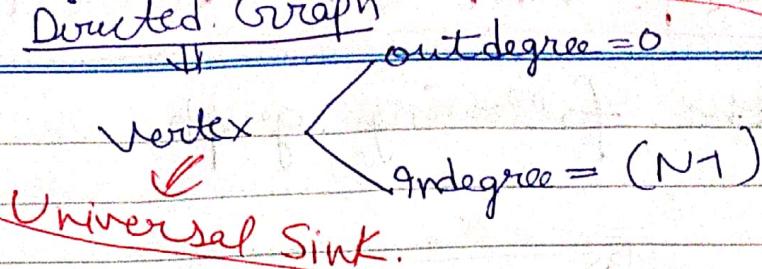
NOTE :- Adjacency Matrix is better for complete graph.
→ Adjacency List is better for less edges. (Sparse graph).

{ → To check adjacent list or not, $O(1)$ - best, $O(V)$ - worst }

{ → To check adjacent Matrix, $O(1)$ - in every case }

{ Degree of adjacency Matrix is - $O(V)$ E.C.
List is - .
 $B.C = O(1)$
 $W.C = O(V)$ }

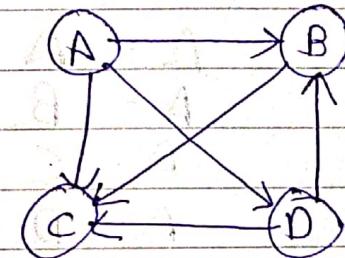
Directed Graph



In the given directed graph, when outdegree of vertex = 0 & indegree of vertex = (N-1), the graph is said to be Universal Sink.

To find universal sink, time complexity = ?
 { Graph represented using Matrix }.

		indegree			
		A	B	C	D
outdegree	A	0	1	1	1
	B	0	0	1	0
C	0	0	0	0	1
D	0	1	1	0	0

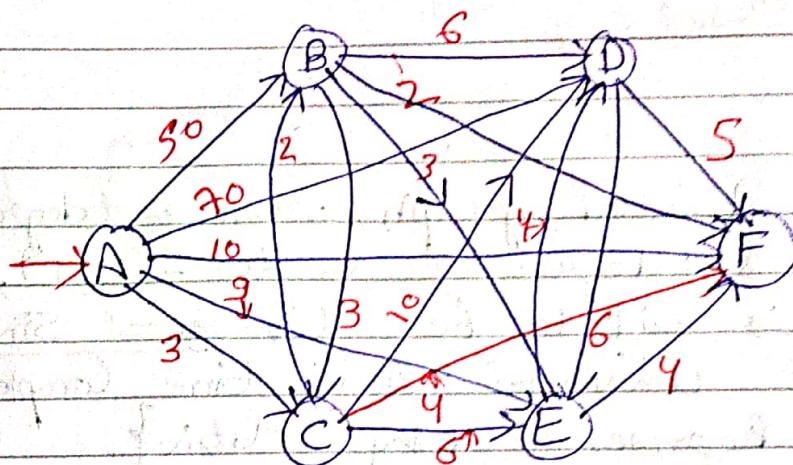


$$= O(v).$$

by

DIJKSTRA'S ALGORITHM

Q.) Consider the following graph



$$A - A = 0$$

$$A - B = 5$$

$$A - C = 3$$

$$A - D = 11$$

$$A - E = 3$$

$$A - F = 7$$

Dijkstra's Present in heap

	A	B	C	D	E	F
A	0	∞	∞	∞	∞	∞
C	$5 \log V$	N	N	N	N	N
B	$10 \log V$	5	13	9	9	$\Rightarrow 4 + 4 \log V$
F	$11 \log V$	$11 \log V$	$11 \log V$	$7 \log V$	$7 \log V$	$\Rightarrow 4 + 0 \log V$
E	$10 \log V$	$\Rightarrow 2 + 10 \log V$				
D	$\cancel{10 \log V}$	$\Rightarrow 2 + 0 \log V$				
						$T = E^{\log V} + \epsilon \log V$

NOTE:- In min heap or maxheap increase key or decrease key opn. will take
 $O(\log n)$ - average & worst case
 $O(1)$ - best case

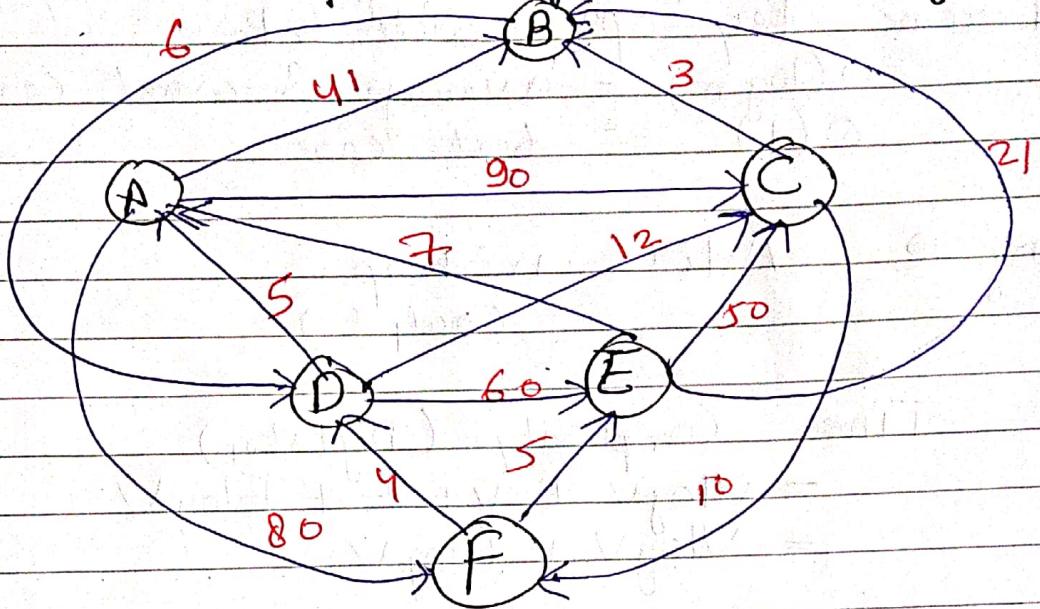
① Using \Rightarrow A. list, minheap
 (output)

$$\begin{aligned} \text{Time Complexity (Dijkstra)} \\ = V \log V + V + E + E \log V \\ = V \log V + E \log V \\ = O[(V+E) \log V] \end{aligned}$$

② Using A. Matrix, minheap

$$\begin{aligned} TC(\text{Dijkstra}) &= V \log V + V + V^2 + E \log V \\ &= V^2 + E \log V \quad \left[\begin{array}{l} \because E=V^2 \\ \therefore E=0 \end{array} \right] \\ &= O[V^2 + E \log V] \end{aligned}$$

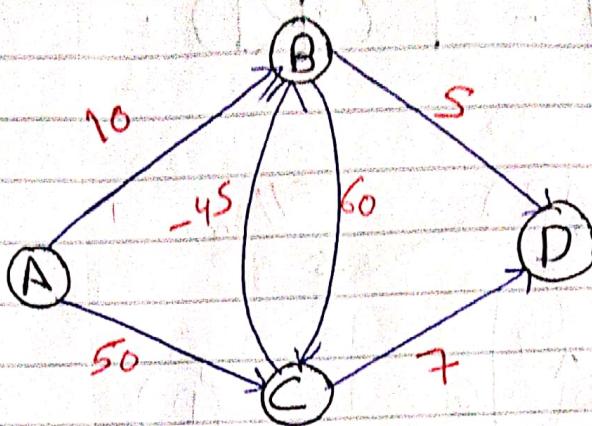
Consider the following directed graph



- (i.) Point out the sequence of vertices identified by Dijkstra's algorithm when algo starts from vertex A.
- (ii.) what will be the cost of shortest path from A to E.
- (iii.) what is the shortest path from A to E.

	A	B	C	D	E	F
A	N	N	N	N	N	N
D	41	12			7	80
F	41	12			A	F
E	28	"			A	

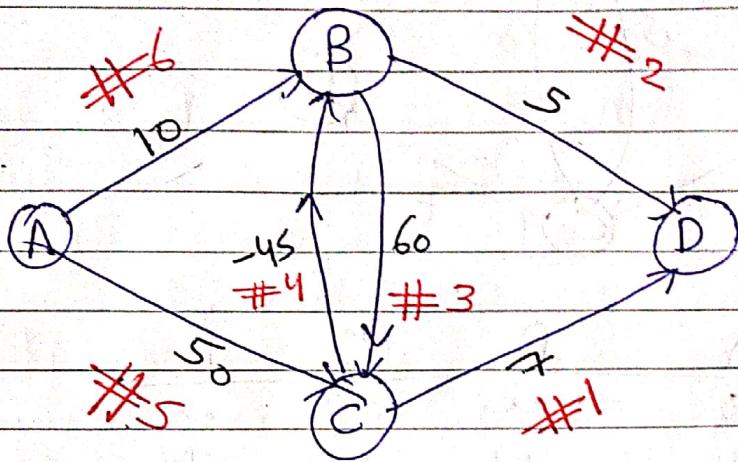
Ex-3.)



	A	B	C	D
A	O N	∞	∞	∞
B	10 A	∞	50 A	∞
D		50 A	15 B	∞
C			∞	∞

(B & D values are wrong using Dijkstra's Algorithm).
In dijkstra's every vertex will be relaxed one.

BELLMAN - FORD :-



Present in array

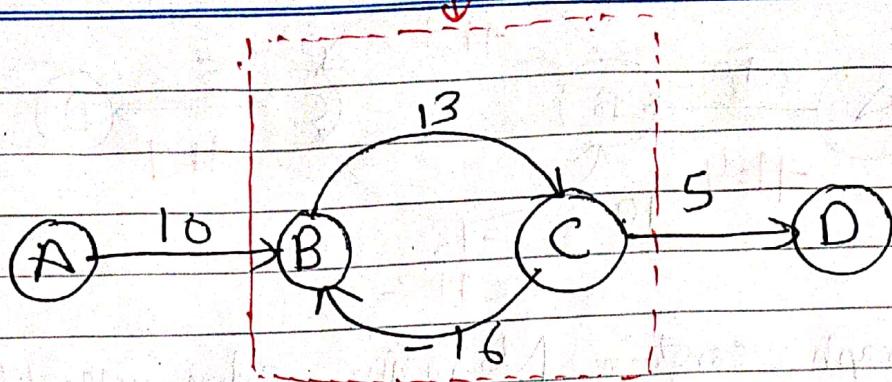
	A	B	C	D	
	0	∞	∞	∞	$\Rightarrow E \times O(1) \Rightarrow O(E)$
	N	N	N	N	
$i=1$	0	10	50	∞	$\Rightarrow E \times O(1) \Rightarrow O(E)$
	N	A	A	N	
	0	5	50	15	$\Rightarrow E \times O(1) \Rightarrow O(E)$
	N	C	A	B	
$i=2$	0	5	50	10	$\Rightarrow E \times O(1) \Rightarrow O(E)$
	N	C	A	B	
	0	5	50	10	$\Rightarrow E \times O(1) \Rightarrow O(E)$
$i=3$	N	C	A	B	
	0	5	50	10	$\Rightarrow E \times O(1) \Rightarrow O(E)$
$i=4$	N	C	A	B	$O(VE)$

↑ For verification

→ Graph contains all positive edge weight then
 Dijkstra's Algorithm always gives correct answer,
 otherwise Dijkstra's Algorithm may fail.

negative edge weight cycle.

(ex-4)



Dijkstra :-

	A	B	C	D
O	∞	∞	∞	∞
N	N	N	N	N
i = 1	10	∞	∞	∞
A	A	N	N	N
i = 2	10	23	∞	∞
B	B	N		
i = 3	10	23	20	28
C	C	B	C	

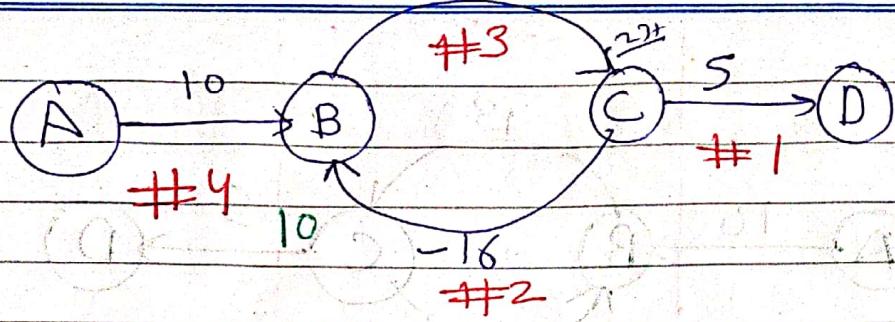
B is wrong, that's why C & D are also wrong (Depend on B). Bellman Ford :-

	A	B	C	D
O	∞	∞	∞	∞
N	N	N	N	N
i = 1	10	∞	∞	∞
A	A	N	N	N
i = 2	10	23	∞	∞
B	B	N		
i = 3	10	23	20	28
C	C	B	C	
i = 4	10	17	25	∞
C	C	B	C	

modified cost

Note → If in verification case, if any value is changing, there is a negative edge weight cycle. $\Rightarrow O(VE)$

13



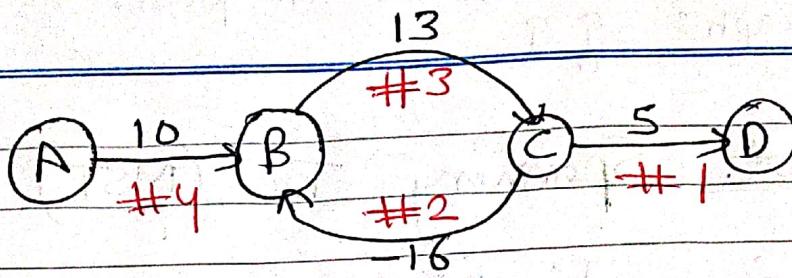
NOTE:- If graph contain Negative edge weight cycle, Dijkstra's Algorithm will always fail.

→ Graph contain only negative edge weight, but not -ve edge weight cycle, Dijkstra's Algorithm may fail.

→ Graph contain +ve edge weighted, -ve edge weight or both, but not -ve weight cycle, Bellman Ford will always give correct answer.

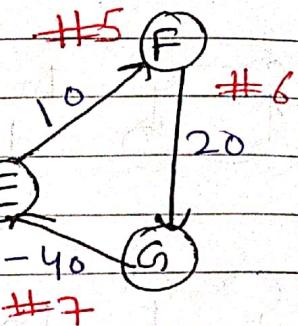
→ If graph contain -ve edge weight cycle, Bellman Ford Algo will give correct answers for those vertices which dont have ~~negative~~ any relation with -ve edge wt cycle, for other vertices answers is undefined.

NOTE:- Bellman Ford Algo will find out all the -ve edge weighted cycle present in the given graph if they are reachable from source.



	A	B	C	D	E	F	G
A	0	10	8	8	8	8	8
B	2	0	2	2	2	2	2
C	2	2	0	2	2	2	2
D	2	2	2	0	2	2	2
E	2	2	2	2	0	2	2
F	2	2	2	2	2	0	2
G	2	2	2	2	2	2	0

i=1 i=2 i=3 i=4 i=5 i=6



Bellman Ford algo can't detect the negative weight cycle, because it is not reachable from source.

→ Bellman Ford Algo will find out all -ve edge weight cycle present in the given if they are reachable from source.

→ Connected graph: In the given undirected graph b/w two vertices there must be an path.

→ Disconnected graph:

Minimum Cost Spanning Tree (MST)

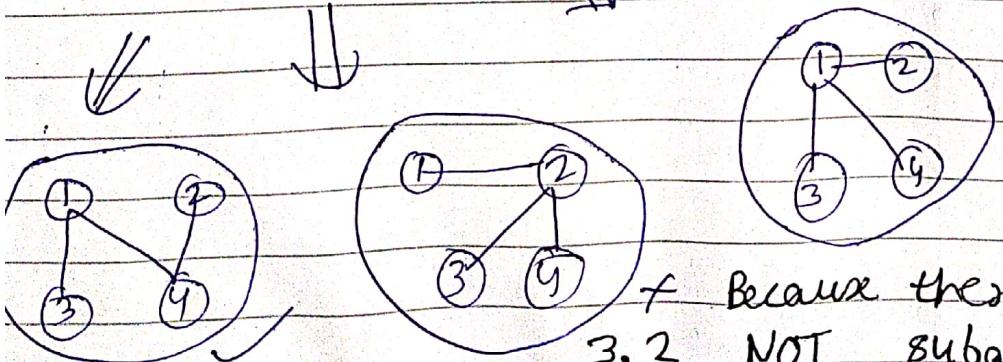
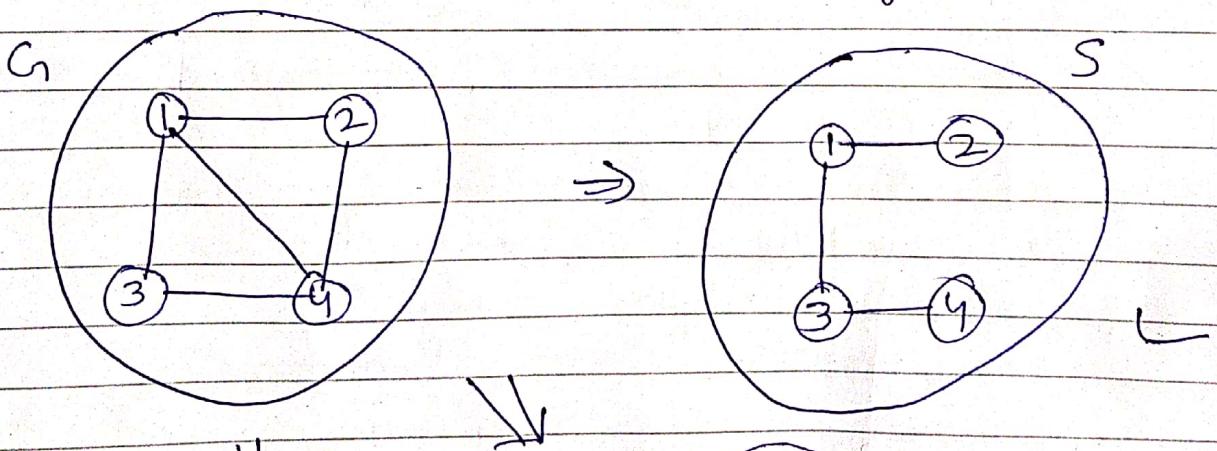
In the given directed graph without removing direction if graph is connected it is strongly connected graph.

In the given directed graph after removing direction if graph is connected then it is weakly connected graph.

Spanning Tree

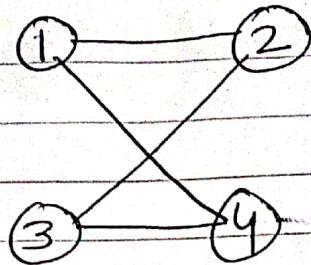
A subgraph ' s ' of the given graph ' G ' is said to be spanning tree if and only if.

- (1) Subgraph ' s ' should contain all vertices of ' G '
- (2) Subgraph ' s ' should contain $(v-1)$ edges without cycle, where ' v ' is no. of vertices.

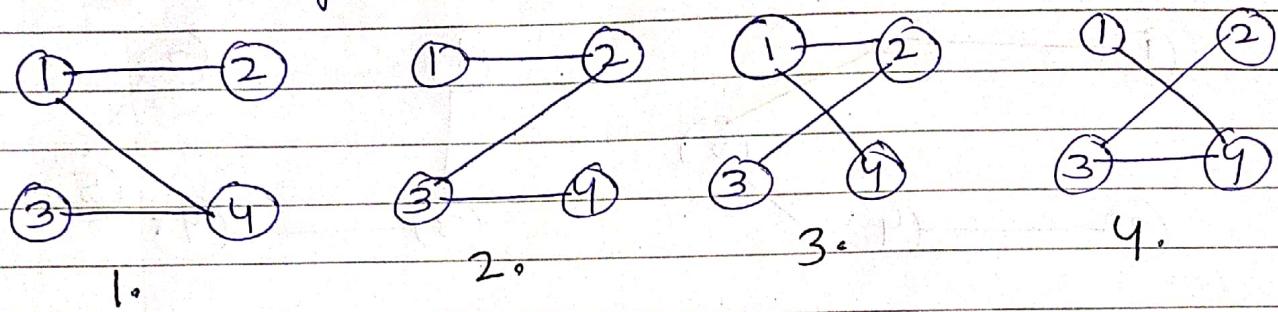
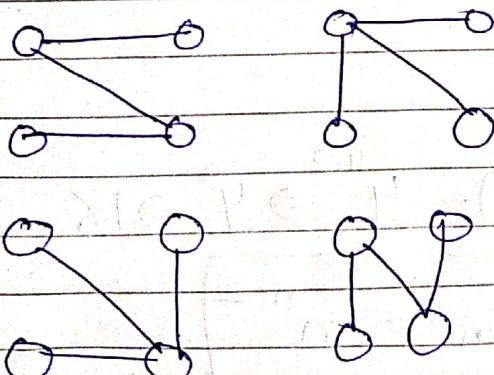
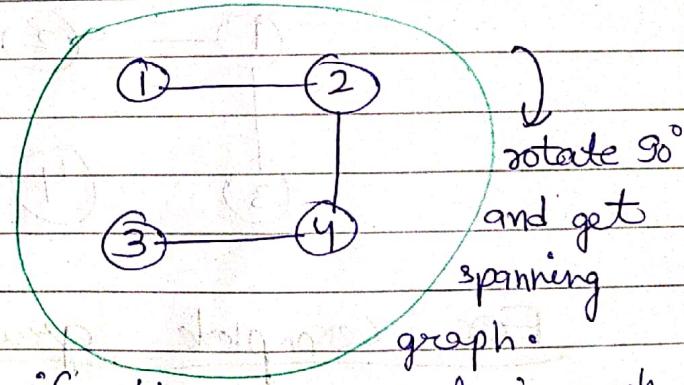
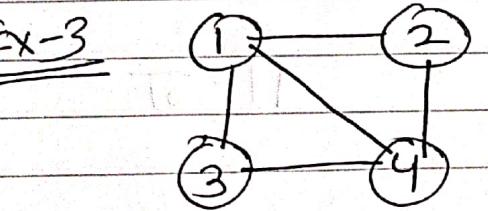


+ Because there is no edge b/w 3, 2 NOT subgraph in graph

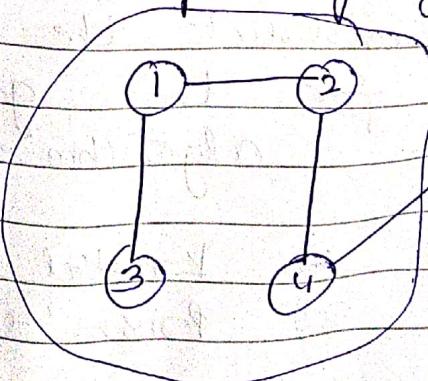
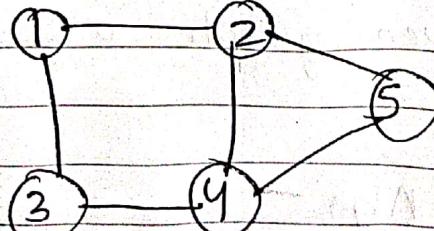
' G ', so you can't take in subgraph ' S '.

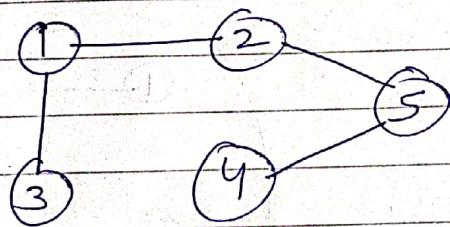
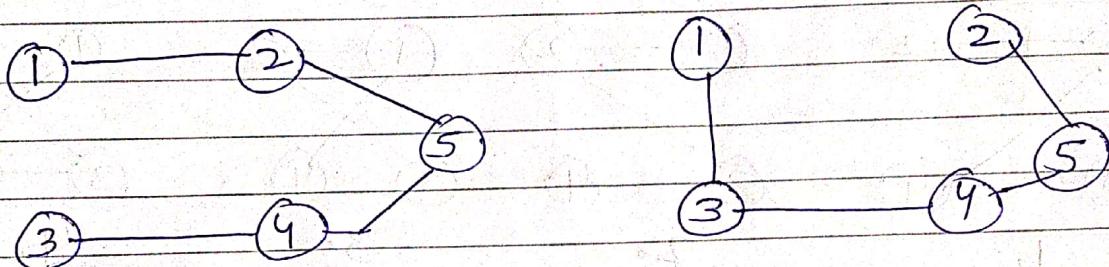
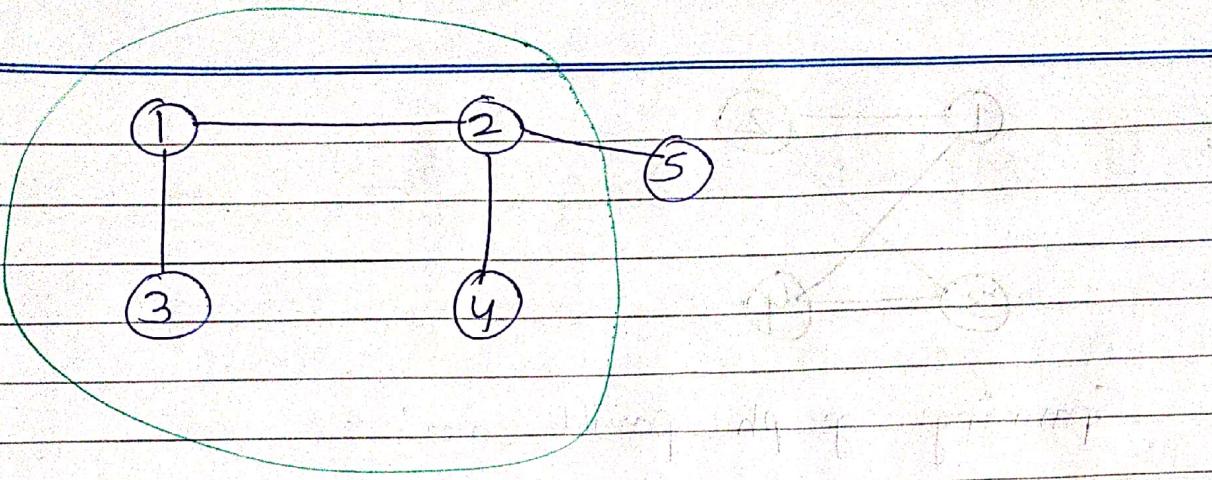
Ex-2

Spanning graph possible are:-

Ex-3

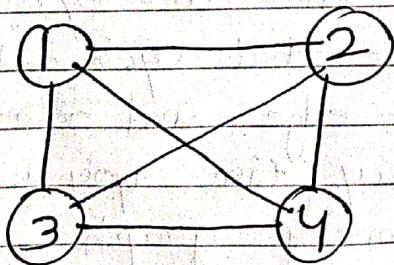
if there are cycle in graph
take that cycle and remove
one edge and rotate 90°
every time you get one
spanning graph.

 $4+4=8$ Spanning graph.Gx-4



11-ST

For complete graph:-



$$ST(K_4) = 4^{4-2} \Rightarrow 4^2 \Rightarrow 16$$

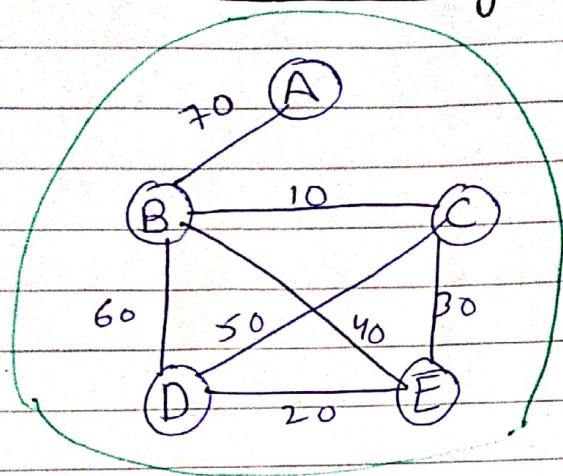
$$ST(K_n) = n^{n-2}$$

n-vertices.

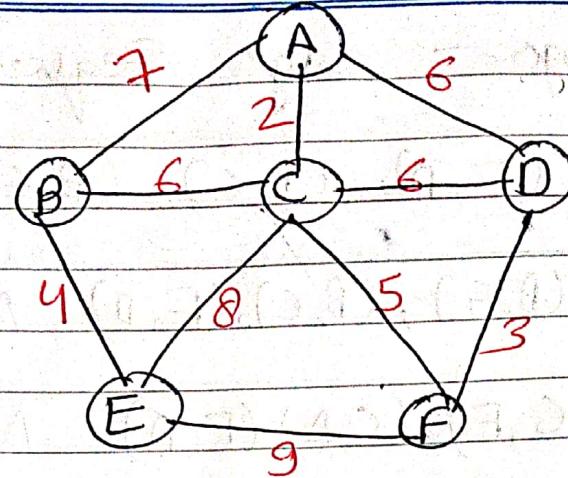
→ To construct the minimum cost spanning Tree, for the given graph we have two algorithm.

- 1.) Kruskal Algo
- 2.) Prism's Algo

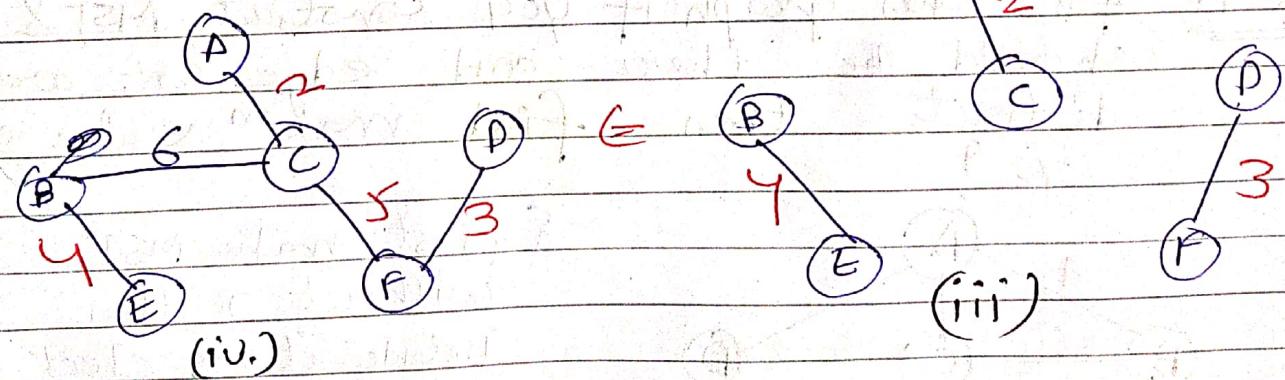
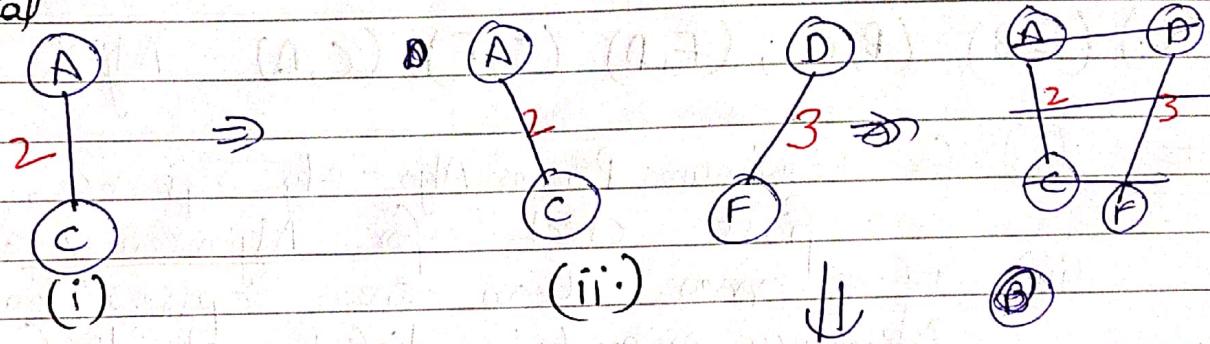
Kruskal Algorithm:-



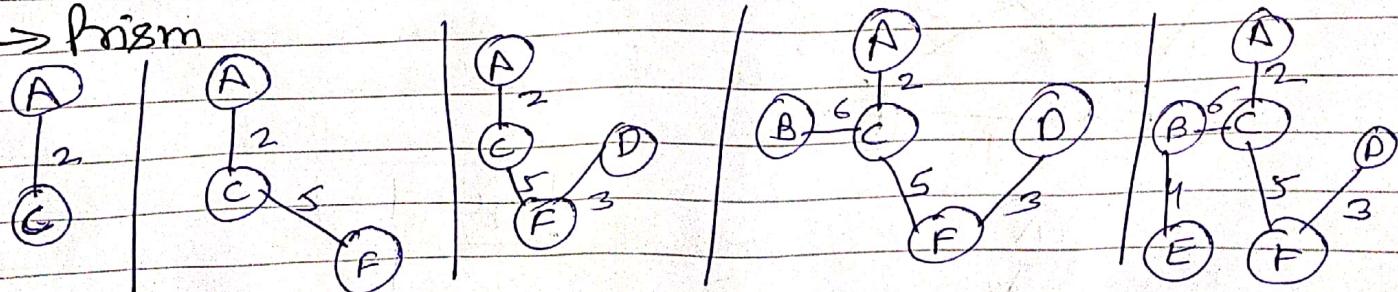
6×2



→ Kruskal



→ Prism



Use this method

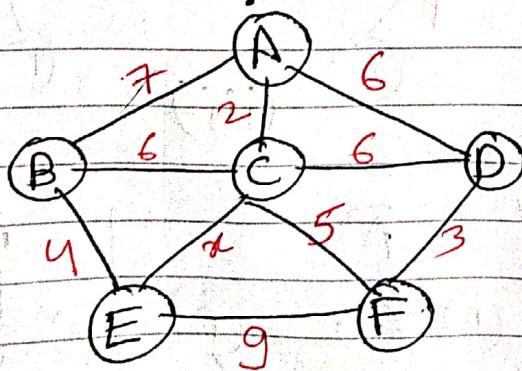
Ex-3) Prism - Algo - edge Sequence?

- i.) $(A, C), (C, F), (F, D), (B, C), (B, E)$ Adj \leftarrow min
- ii.) $(A, C), (C, F), (B, E), (B, C), (F, D)$ Adj \times
- iii.) $(E, B), (B, C), (C, F), (C, A), (F, D)$ Adj \times
- iv.) $(E, B), (B, C), (F, D), (C, F), (C, A)$ Adj \times

Note:- (i.) for minimum Prism - Algo - edge Sequence, first check for Adjacency.

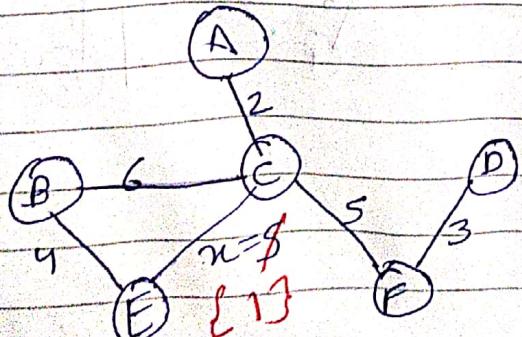
(ii.) If more than one option have Adjacency property satisfied, check for min. (minimum distance).

Ex-4) for this graph if you construct MST & x should be there and edge wt are distinct then find \max^m value of x ?



{first make MST without x , then include it & check}

$$x = 1$$



internal node) in directed $\{ \text{outdegree} = 0 \rightarrow \text{leaf node} \}$
degree = 2 in undirected $\{ \text{degree} = 1 \rightarrow \text{leaf node} \}$

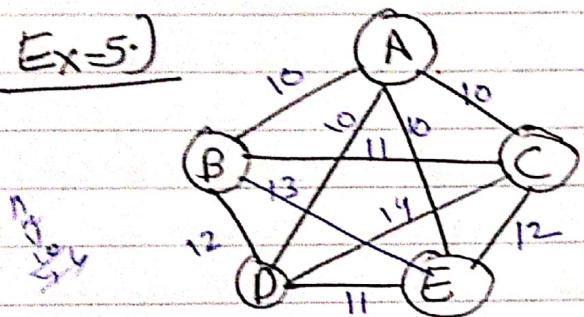
183

($\therefore x$ should be there, in order to remove cycle delete the maximum, i.e. 6 and assign x^{\max})

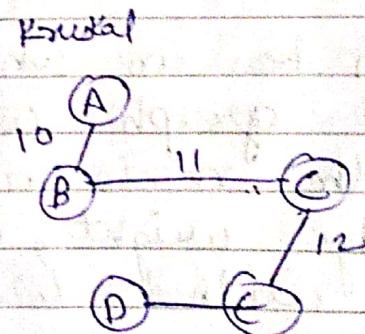
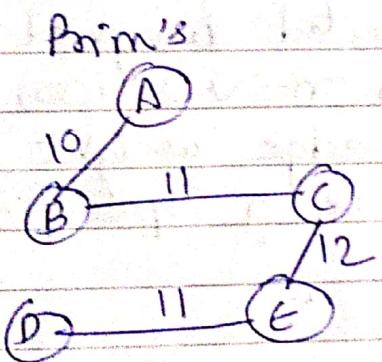
{ Not distinct question $\xrightarrow{\text{max}} x = 6 \text{ or } 5$ }
Distinct given $\xrightarrow{\text{max}} x = 5 / 4 / 3 / 2 / 1 \rightarrow$ They are already in graph.

(if $\min \text{ cost} = 1, 2, 3, 4, 5, < 6 \rightarrow$ Not distinct
 $\xrightarrow{\text{max}} 1 \rightarrow$ distinct)

Ex-5.)



What is the cost of the MST for this graph in such way that MST 'A' will be the leaf node.



Ex-6.) Let G be an undirected weighted connected graph with distinct edge weights, e_{\max} be the \max^m -edge weight & e_{\min} be its minimum edge weight.

T/F:-

- a.) G - contain - UMST ✓
- + b.) MST of G may $\xrightarrow{\text{must}}$ contain e_{\min}
- c.) " " " " " " " " e_{\max}
- d.) if e_{\max} is in MST then its removal from G must disconnect G .

ex7) Let G be an undirected weighted connected graph with n -vertices, w be the minimum edge weight among all edge weight, e be a specific edge with w .

T/F.

- + a.) G contain unique MST
- + b.) Every MST of G must contain e
- ✓ c.) Every MST of G must contain atleast one edge with weight w .
- ✓ d.) If e is not in MST then in that cycle all edges contain same weight w .
(which is not there)

(a. Ans.)

(b. Ans.)

(c. Ans.)

(d. Ans.)

[Q.1]

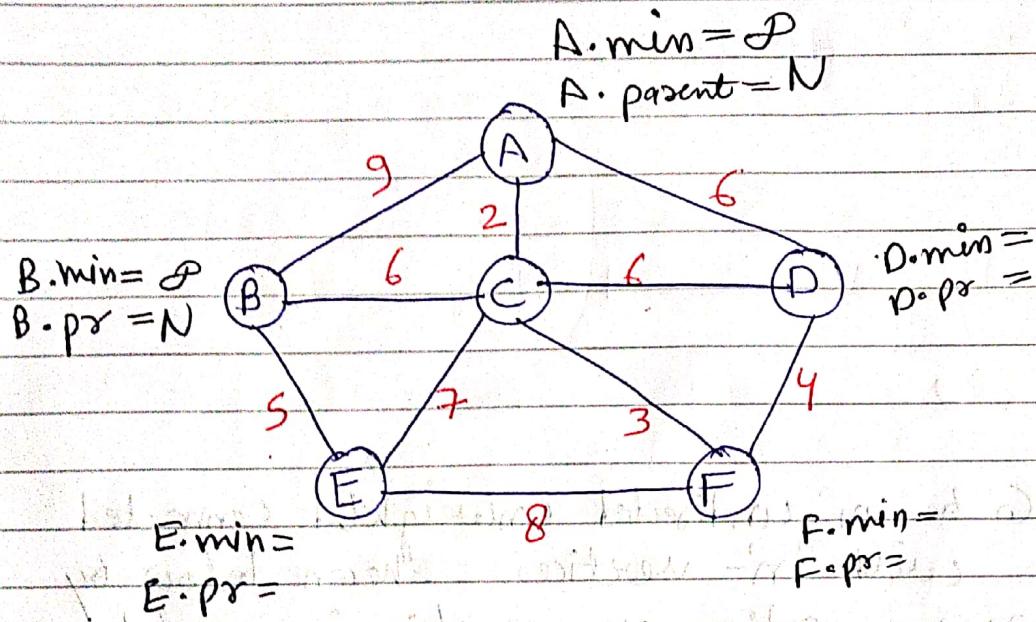
(ex-8.) Let G be an undirected unweighted connected graph with ' n ' vertices shown below by an $n \times n$ adjacency matrix in which

- (i) All diagonal element 0's
- (ii) All Non-diagonal element 1's

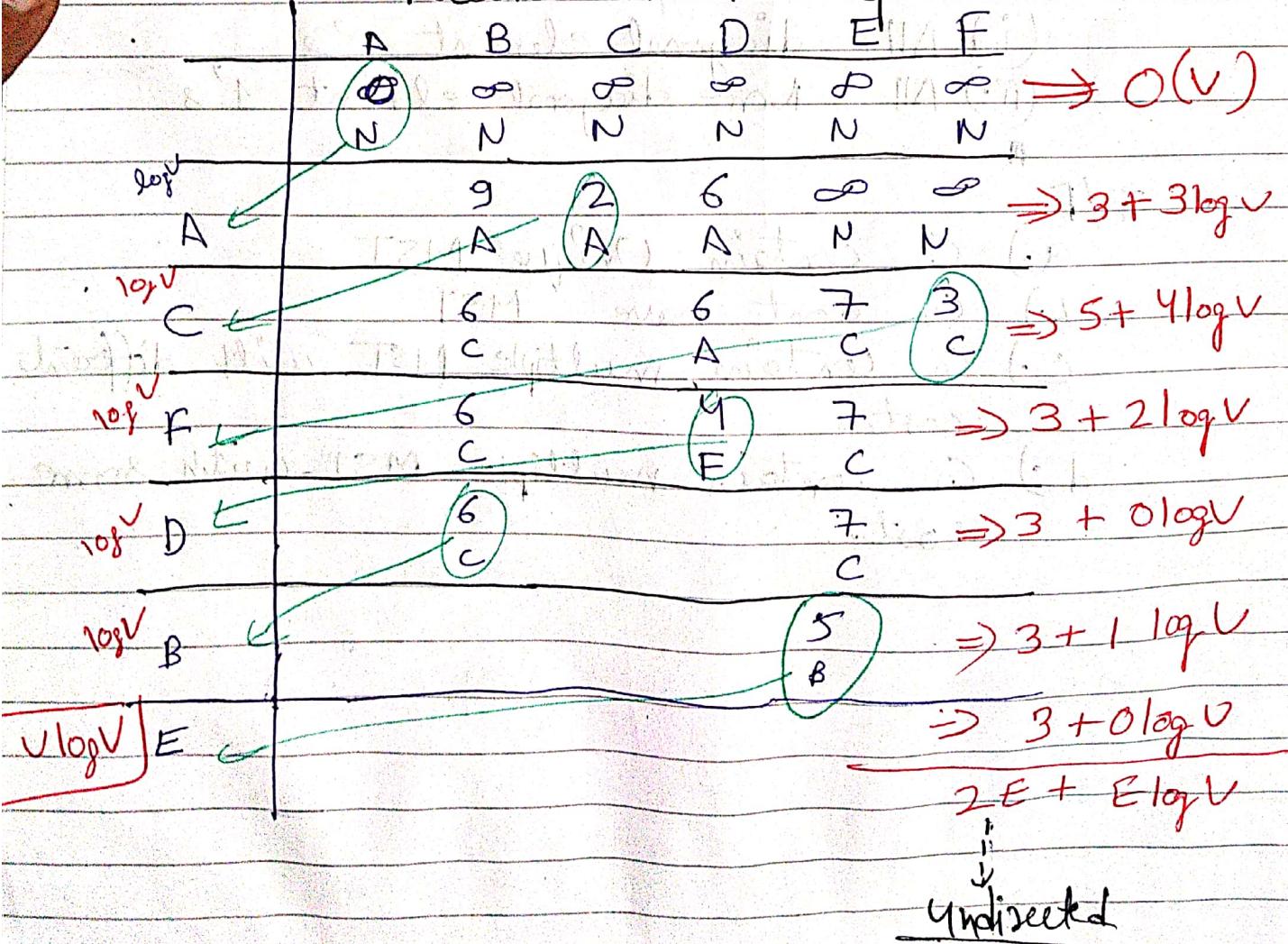
T/F

- a.) G contains Unique MST
- b.) G don't have MST
- c.) G contain multiple MST with different cost.
- d.) G contain multiple MST with same cost.

TIME COMPLEXITY OF PRISM'S ALGORITHM



Present in Min heap



$$\begin{aligned}
 \text{Time Complexity} &= V \log V + 2E + E \log V + V \\
 &= (V+E) \log V \\
 &= O[(V+E) \log V]
 \end{aligned}$$

using Adjacency Matrix, minheap.

$$\begin{aligned}
 TC &= V \log V + V^2 + E \log V + V \\
 &= O(V^2 + E \log V)
 \end{aligned}$$

using Adjacency list, sorted array

$$\begin{aligned}
 T.C &= V + 2E + EV + V \\
 &= O(EV).
 \end{aligned}$$

Using adj matrix, sorted Double Linked list

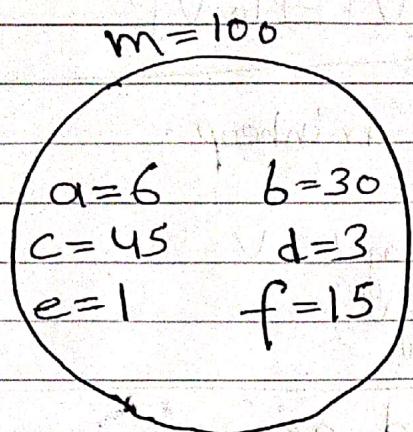
$$\begin{aligned}
 T.C &= V + V^2 + EV + V \\
 &= O(V^2 + EV)
 \end{aligned}$$

Using adj list and normal unsorted array

$$\begin{aligned}
 T.C &= V^2 + 2E + E + V \\
 &= O(V^2).
 \end{aligned}$$

HUFFMAN CODING :-

(ex-1)



{ Data compression tech }
{ Non uniform coding technique }

ASCII



$$a \rightarrow 6 \times 8$$

$$b \rightarrow 30 \times 8$$

$$c \rightarrow 45 \times 8$$

$$f \rightarrow 15 \times 8$$

$$\underline{100 \text{ char} = 800 \text{ bit}}$$

$$\text{char} = 8 \text{ bit/char.}$$

By default
2 way

Merge those who are minimum

OPTIMAL MERGE PATTERN

Ex-1) $n=3$ (3 files)



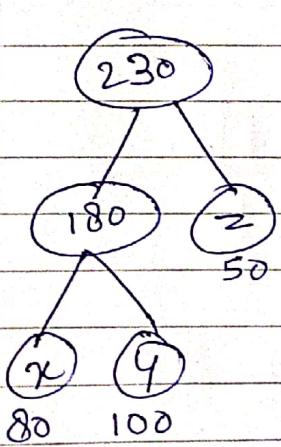
$x = 80$ (80-records)

$y = 100$

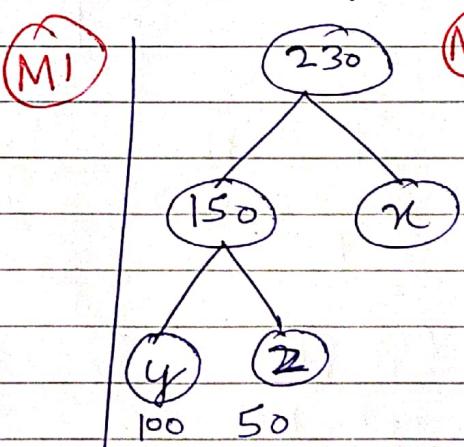
$z = 50$

Minimum moves = ?

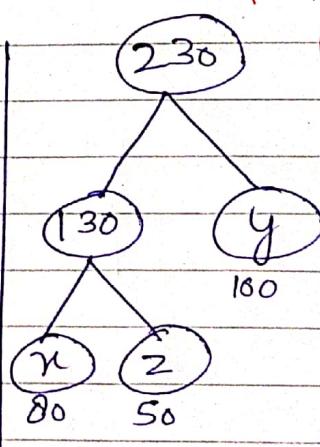
↑ Merge 2 minimum
for min moves.



$$\begin{array}{r} 180 \\ + 230 \\ \hline 410 \end{array}$$



$$\begin{array}{r} 230 \\ + 150 \\ \hline 380 \end{array}$$



$$\begin{array}{r} 230 \\ + 130 \\ \hline 360 \end{array}$$



ex-2

$n=7$



$$A=20 \quad B=15$$

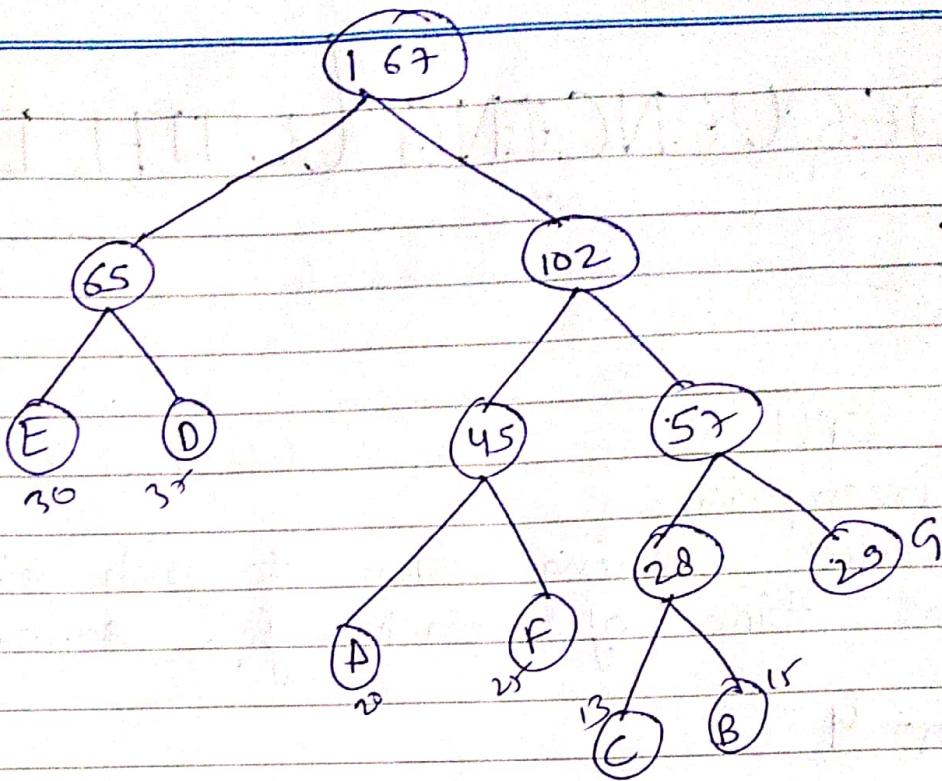
$$C=13$$

$$D=35$$

$$E=30$$

$$F=25$$

$$G=29$$



Minimum record moves = Sum of internal nodes.

$$\begin{aligned}
 &= 28 + 57 + 102 + 45 + 65 + 167 \\
 &= \underline{464}.
 \end{aligned}$$

JOB SEQUENCING WITH DEADLINES

LINES:-

1. Single CPU
2. No Preemption.
3. One unit running time to each job.
4. Arrival time of each job same.

ex1.) $n=4$

Jobs	J_1	J_2	J_3	J_4
profit	175	125	150	200
Deadline	2	1	2	1

$$(J_2, J_1) \Rightarrow 300$$

$$(J_4, J_3) = 350$$

$$(J_2, J_3) = 275$$

$$(J_1, J_3) = 325$$

$$(J_3, J_1) = 325$$

optimal soln $\rightarrow (J_4, J_1) = 375 \leftarrow \text{Max profit}$

feasible

J_1

J_2

J_3

J_4

1

$$6+4+1 = 11 \text{ feasible}$$

Shortcut:

- ① Take the Max Deadline at 2 place and then take Deadline at 1 place
Max profit \rightarrow

J ₄	J ₁
1	2

* Order of optimal solⁿ J₄, J₁

* optim solⁿ profit = 375

* which Job you left J₂, J₃

* what profit you left = 275

(ex-2) n=7

Jobs	J ₁	J ₂	J ₃	J ₄	J ₅	J ₆	J ₇
profit	300	250	150	200	175	275	270
Deadline	5	3	5	3	2	1	5
	o(n)	o(n)	o(n)	o(n)	o(n)		

J ₆	J ₄	J ₂	J ₇	J ₁
1	2	3	4	5

optimal solⁿ = 1295

seqn = J₆, J₄, J₂, J₇, J₁

penalty = 325

[These where didn't done called penalty]

ex-3

Jobs	J ₁	J ₂	J ₃	J ₄	J ₅	J ₆	J ₇	J ₈	J ₉
75	50	25	70	55	30	60	45	68	
5	3	7	3	5	4	2	5	3	

J ₇	J ₁	J ₄	J ₅	J ₃	J ₆	J ₈	J ₂	J ₉	J ₃
1	2	3	4	5	6	7			

shortcut

- ① Sort in Decreasing order of profit \Rightarrow (n log n) merge sort

Jobs: J₁ J₄ J₉ J₇ J₅ J₂ J₈ J₆ J₃

Profit: 75 70 68 60 55 50 45 30 25

Deadline: 5 3 3 2 5 3 5 4 7

$O(n)$ $O(n)$ $O(n)$ $O(n)$ $O(n)$ $O(n)$ $O(n)$

A diagram illustrating a linked list structure. It consists of seven rectangular boxes arranged horizontally, each containing a letter label: J_7 , J_9 , J_4 , J_5 , J_1 , J_6 , and J_3 . Below each box is a numerical index: 1, 2, 3, 4, 5, 6, and 7. A curved arrow originates from the bottom of the J_6 box and points to the bottom of the J_7 box, indicating a connection between the two nodes.

$$n^2 + n \log n \geq O(n^2) \quad [\text{Worst Case}]$$

$$O(n \log n) \quad [\text{Best Case}]$$

* if adjacent job have same Job dead line then you can swap them.

J₁ J₂

Read line \rightarrow 3 3 swap possible