

Chapter 3: Greedy Technique -

Most of the problems in greedy, contain n inputs and the objective is finding a subset which will satisfy our condition to give optimal solution.

feasible solⁿ $\rightarrow$ which will satisfy our conditions

Optimal solⁿ $\rightarrow$ one of the feasible solution which will optimize our goal.
(optimal solⁿ need not be unique)

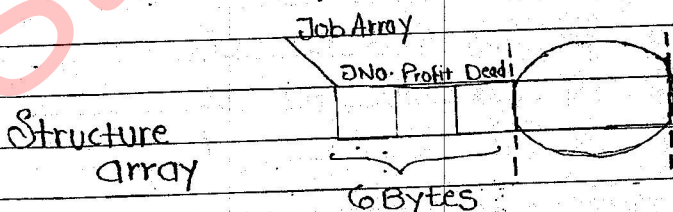
Applications of Greedy technique -

I. Job Sequencing with deadlines

1. Single CPU available.
2. Every job required one unit execution time (no SJF)
3. Arrival time of every job is same. (no FCFS)
4. Preemption not possible. (no RR)

Ex.1 $n=4$

Jobs	J ₁	J ₂	J ₃	J ₄
Profit	100	200	150	250
Deadlines	2	1	2	1



Representation in computer

```
struct JI
{
    int JNo;
    int profit;
    int deadlin;
};
```

Sequence Profit

$(J_2, J_1) \Rightarrow 300$

(J_1, J_2) Not feasible

(J_3, J_4) Not feasible

$(J_4, J_3) \Rightarrow 400$ (optimal)

$(J_1, J_3) \Rightarrow 250$

$(J_2, J_3) \Rightarrow 350$

$(J_3, J_1) \Rightarrow 250$

$() \Rightarrow 0$ feasible

$(J_1) \Rightarrow 100$

$(J_2) \Rightarrow 200$

$(J_3) \Rightarrow 150$

$(J_4) \Rightarrow 250$

1. Sort all jobs in decreasing order of the profits.

2. What left is the penalty.

Ex-2. $n=7$

Jobs	J_1	J_2	J_3	J_4	J_5	J_6	J_7
Profits	25	45	50	15	75	120	55
Dead-lines	5	3	4	2	1	3	2

Linear Search

Sequence J_5, J_7, J_6, J_3, J_1

1 2 3 4 5

Profit $75 + 55 + 120 + 50 + 25 = 325$

Ex-3 $n=9$

Jobs	J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8	J_9
Profits	5	15	20	25	18	13	30	70	12
Dead-lines	5	3	2	7	4	5	2	3	4

$J_3, J_7, J_8, J_5, J_6, J_4$

1 2 3 4 5 6 7

Profit = 176

2.

							J4
1	2	3	4	5	6	7	

Time complexity - $O((\log n)n)$ // to sort
 for 1-element $O(n^4)$ // worst case
 n-element $O(n^2)$
 for 1-element $O(1)$ // Best case
 n-element $O(n \log n)$

[No binary search possible since deadlines are not in sorted order]

II Knapsack Problem

feasible - not keeping more than capacity of knapsack.

$$1 \times 10 + 1 \times 2 + \frac{8 \times 15}{15} + 0 \times 8 \quad \text{Weight} \quad \begin{array}{|c|c|c|c|} \hline 10\text{kg} & 2\text{kg} & 15\text{kg} & 8\text{kg} \\ \hline 200 & 300 & 350 & 250 \\ \hline \end{array}$$

$$1 \times 200 + 1 \times 300 + \frac{8 \times 350}{15} + 0 \times 250 \quad \text{Profit}$$

Solution = $(1, 1, 8/15, 0)$ [feasible]

Ex-1 $n=3$

Object	obj1	obj2	obj3	
Weight	18	10	6	$m=20$
Profit	25	18	8	

Greedy about both profit & weight -

1. obj1 : $18 \rightarrow 25$
 $1 \rightarrow 25/18 = 1.38$

2. obj2 : $18/10 = 1.8$

3. obj3 : $8/6 = 1.33$

Take 2nd Object first

Arrange in the descending order of Profit/Weight

$$\left(\frac{1}{10/18} \right)$$

$$W = 20 - 10 = 10$$

$$P = 18$$

$$\left(\frac{1}{10/18} \right)$$

$$W = 10 - \frac{10 \times 18}{18} = 0$$

$$P = 18 + \frac{10 \times 25}{18}$$

$$\left(\frac{0}{10/18} \right)$$

$$W = 0$$

$$P = 31.8$$

Time complexity:-

$$O(n)$$

finding ratio

$$O(n \log n)$$

sorting n numbers

$$O(n)$$

calculations (add)

$$O(n \log n)$$

Ex 2. $n = 7$

Object	1	2	3	4	5	6	7	
Profit	25	5	50	40	10	90	60	$m = 45$
Weight	5	2	12	7	3	15	15	
Ratio	5	2.5	4.1	5.7	3.3	6	4	
Arrange	6	4	1	3	5	7	2	(Sorted)

$$\left(\frac{1}{1} \frac{0}{0} \frac{1}{1} \frac{1}{1} \frac{0}{0} \frac{1}{1} \right)$$

$$W = 45 - 15 = 30$$

$$P = 90$$

$$W = 30 - 7 = 23$$

$$P = 40$$

$$W = 23 - 5 = 18$$

$$P = 25$$

$$W = 18 - 12 = 6$$

$$P = 50$$

$$W = 6 - \frac{15 \times 6}{15} = 0$$

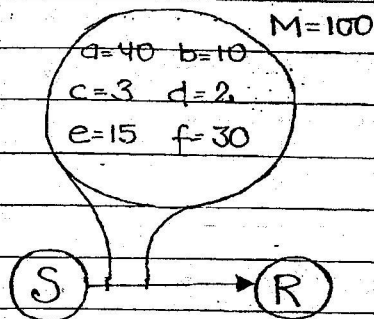
$$P = \frac{6 \times 60}{15} = 24$$

$$\text{Total Profit} = 90 + 40 + 25 + 50 + 24 = 229$$

If the i/p is already sorted, then $T(n) = O(n)$.

Greedy technique fails in case of 0/1 Knapsack problem

III Huffman Coding



ASCII

a - 8x40

b - 8x10

c - 8x3

d - 8x2

e - 8x15

f - 8x30

Encoding

Data compression

$$320 + 240 + 120 = 680$$

Total

$$100 \text{ chr} = 800 \text{ bits}$$

$$1 \text{ chr} = \frac{800}{100}$$

$$\text{Avg} = 8 \text{ bit/chr}$$

Can you compress

Huffman Coding
More freq - less bits
Less freq - more bits

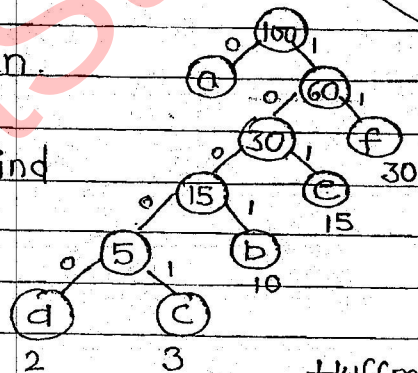
1. Take two minimum.

2. Add them & return.

Min-heap is used to find two minimum.

Deletion - $2 \log_2 n$ Insertion - $\log_2 n$ Total $3 \log_2 n$

If selection sort

find min - $2n$ insert - c Total $2n$ 

Huffman Encoding tree

a - 0 $40 \times 1 = 40$ b - 1001 $10 \times 4 = 40$ c - 10001 $3 \times 5 = 15$ d - 10000 $2 \times 5 = 10$ e - 101 $15 \times 3 = 45$ f - 11 $30 \times 2 = 60$

210

$$\text{avg} = \frac{210}{100} = 2.1 \text{ bit/char}$$

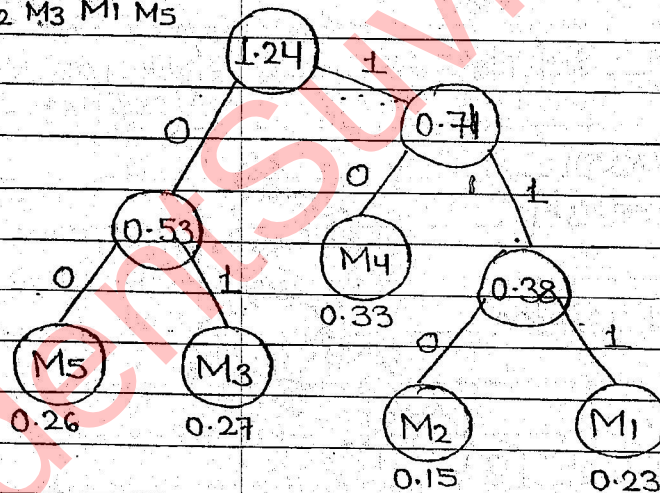
Time complexity -

$O(n)$ Create min-heap
 $(n-1) 3 \log n$ Insertion / Deletion $[(n-1) \text{ passes}]$
 $n + 3 \log n$ Total
 $= O(n \log n)$

$\times 2 \quad M = M_1 \cdot M_2 \cdot M_3 \cdot M_4 \cdot M_5$
 $0.23 \quad 0.15 \quad 0.27 \quad 0.33 \quad 0.26$

1. Huffman code
2. Average no. of bits.
3. Encoded message

110010010110011100
 $M_2 \quad M_3 \quad M_5 \quad M_4 \quad M_2 \quad M_3 \quad M_1 \quad M_5$



$M_1 = 111 \quad 3 \times 0.23 = 0.69$

$M_2 = 110 \quad 3 \times 0.15 = 0.45$

$M_3 = 01 \quad 2 \times 0.27 = 0.54$

$M_4 = 10 \quad 2 \times 0.33 = 0.66$

$M_5 = 00 \quad 2 \times 0.26 = 0.52$

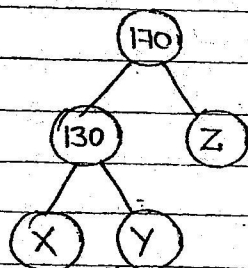
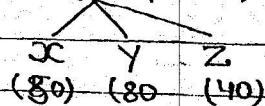
2.86

Average = $\frac{2.86}{1.24}$

= 2.30 bits/m

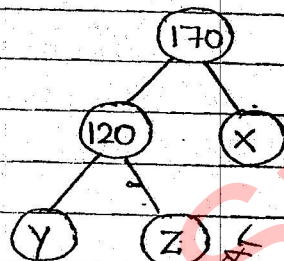
IV Optimal Merge-pattern

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

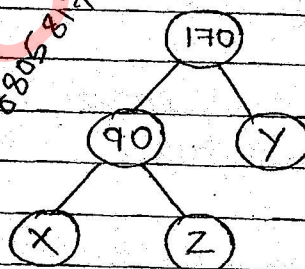


No. of record movements = $130 + 170$
= 300.

(Two maximum taken)



No. of record movements
= $170 + 120 = 290$



Record movements
 $170 + 90 = 260$

(Two minimum taken)

find

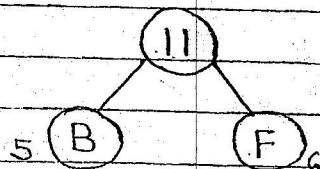
Optimal !

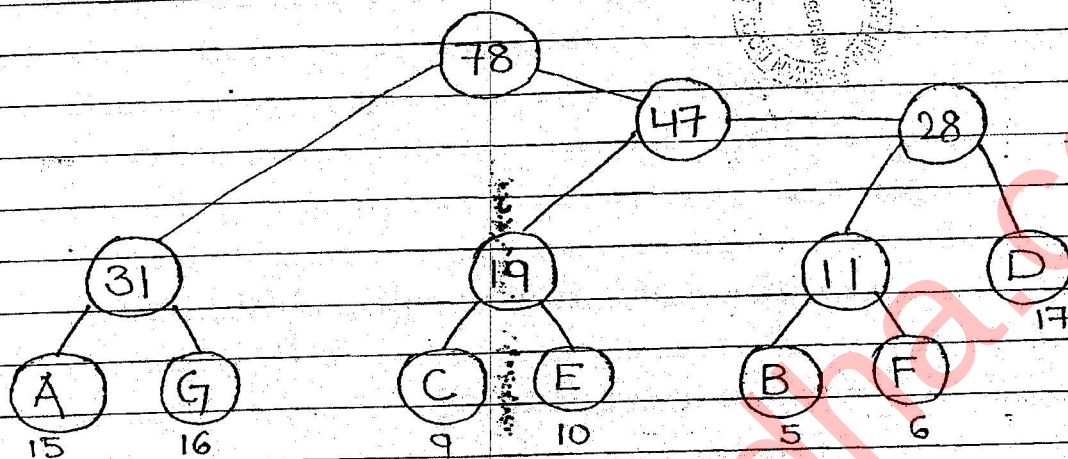
Solution is

take every time two minimum

Ex-2 $n=7$ (7 files).

A	B	C	D	E	F	G
15	5	9	17	10	6	16




$$47 + 78 = 214$$

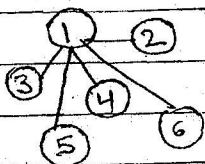
V Minimum Spanning Tree

Graph $G(V, E) \rightarrow$ Set of edges
 $\hookrightarrow$ set of vertices

Types of graphs-

- | | |
|-----------------|--|
| I. Simple-Graph | (No parallel edges) |
| II. Multi-Graph | (Parallel edges are allowed)
(self loops are allowed) |

Degree of a vertex - No. of vertices adjacent to a particular node.



Maximum degree 5 $(n-1)$ n nodes

Minimum degree 0

[Simple graph]

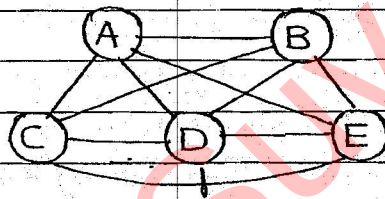
• for multi-graph, maximum degree is infinite and minimum degree is zero. for simple graph, maximum degree is $(n-1)$ & minimum degree is zero.

• $\equiv$ Null graph - A simple graph in which every node has minimum degree i.e. zero. (No. of edges are zero).

• $\equiv$ Complete graph - A simple graph in which every node has maximum degree.

A 4 node complete graph represented by K_4

K_5



5-node complete graph.

No. of edges = Sum of degrees / 2.

$$= 4 \times 5 / 2 = 10$$

$$E(K_n) = n(n-1)/2$$

Consider $G(V, E)$ be a simple graph, then

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

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

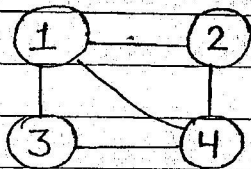
$$|E| = O(V^2)$$

Applying log both sides -

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

Representation of graphs -

Ex-1.



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

Adjacency matrix (2-D Array)

Time complexity depends on how graph is stored.

Degree of (3) = No of 1's in 3rd row.

Complexity $O(V)$ (Scan whole row)

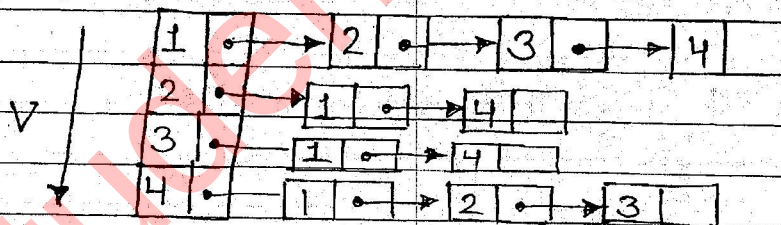
In an unweighted graph, edge weight is one.

If 20 vertices, 500 edges, the space required is 20×20 .

If n vertices, space required $O(V^2)$.

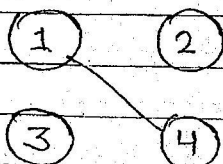
Better representation for complete graphs. i.e. having maximum no. of edges. (dense graphs)

Adjacency list (linked list).

 $2E$

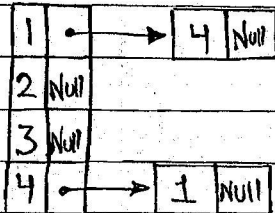
Space needed = $V + 2E \Rightarrow O(V + E)$.

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

 $O(V^2)$



$$O(V + 2E) \Rightarrow O(V)$$

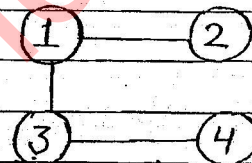
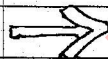
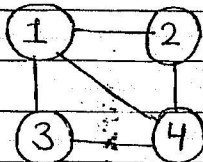
V dominating

Adjacency list is preferred when no. of edges are less. (sparse graphs)

Finding degree of a vertex in adjacency list

$O(1)$ Best case $O(V)$ worst case.

Spanning tree - A sub-graph S of graph G is ST iff all vertices, $n-1$ edges, no cycle.



Graph

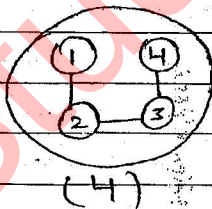
Connected

(To cover n nodes,
($n-1$) edges are
required (minimum))

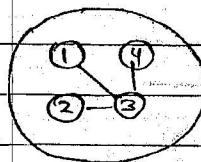
Spanning Tree

No. of vertices are
same & no cycle present
Covering all nodes with
minimum no. of edges.

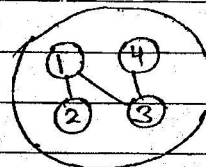
No. of spanning trees.



(4)

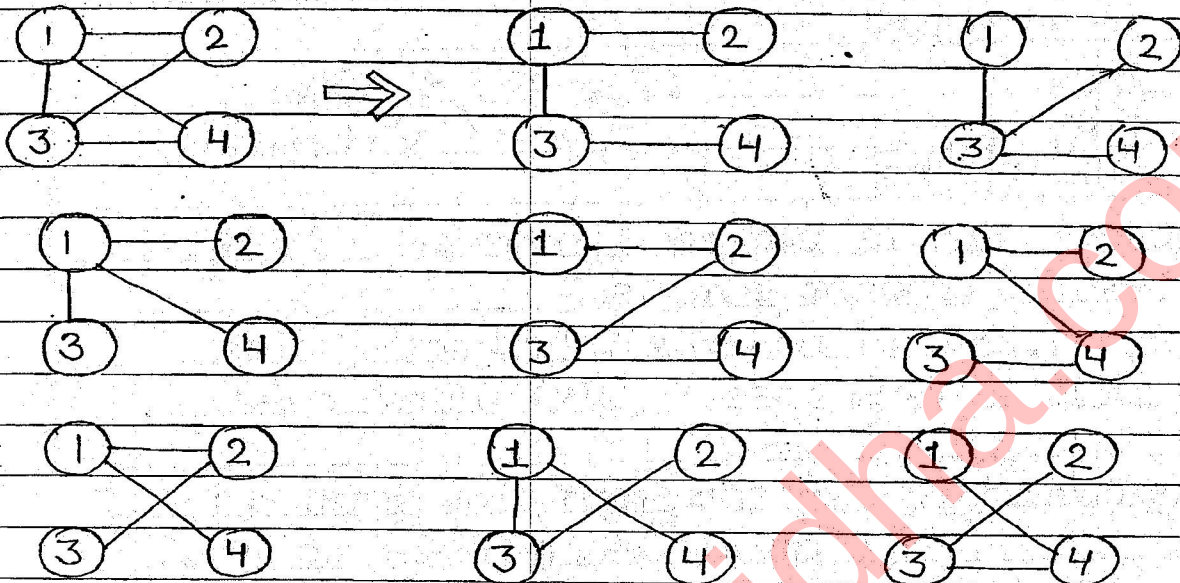


(2)



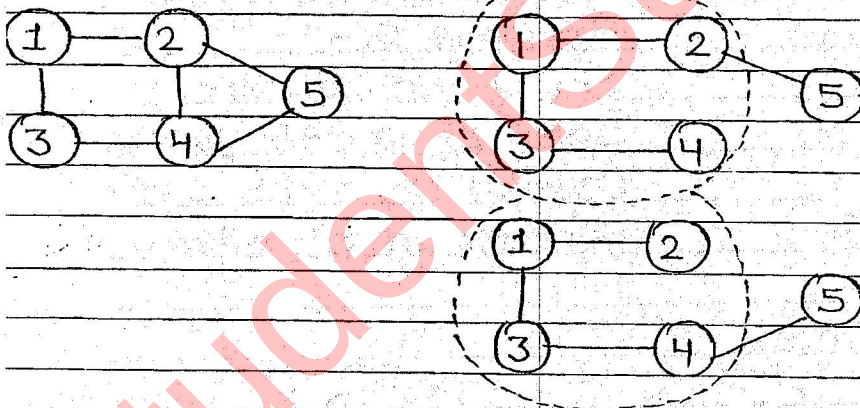
(2)

$$\text{Total} = 4 + 2 + 2 = 8$$

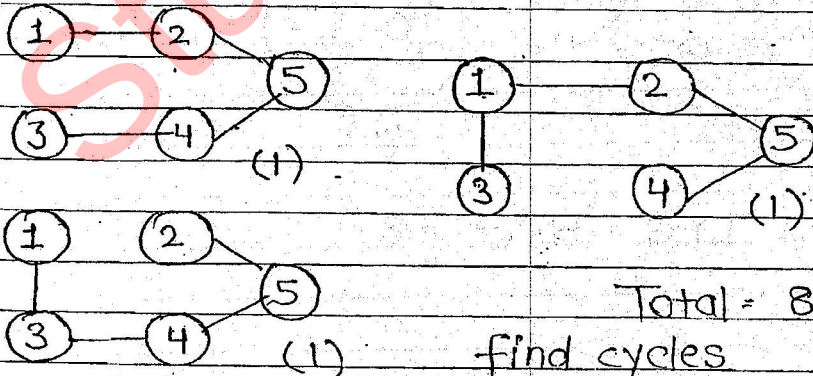


Total = 8

Count the no. of spanning trees -



[Check out the cycles]



Total = 8 + 3 = 11

find cycles

for complete graphs -

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

$$ST(K_3) = 3^{3-2} = 3$$

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

[Spanning tree
concept only
valid for undirected
graphs]

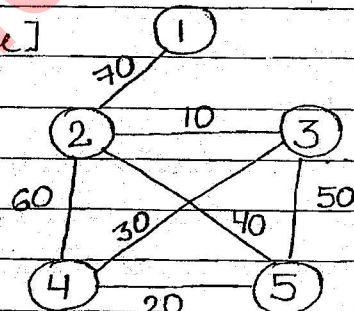
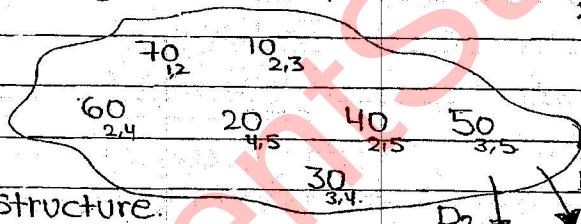
≡ If spanning tree not possible for a graph, then it means the graph is not connected one.

Minimum cost spanning tree :- total cost of all the edges minimum.

≡ Kruskal's Algorithm

[Edge by Edge]

(creating min-heap $O(E)$)



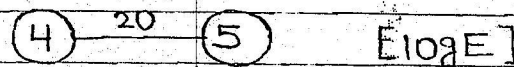
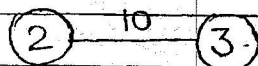
Array

Structure

1. Take min-edge weight

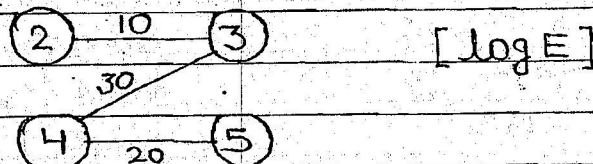


2. Take next-minimum to add to MST

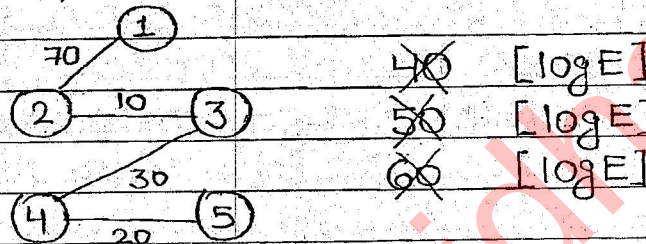


[Not necessary that at every step, connected graph.]

3. Take next min and add to MST if no cycle



1. Repeat the previous step.



Time complexity -

Best case :-

$O(E)$ // create min-heap // $O(E)$

$V-1 \log E$ // deletions // $E \log E$

$E + V \log E$ // Total // $E + E \log V$

$O(E + V \log E)$

$O(E \log V)$

Average case $\uparrow$

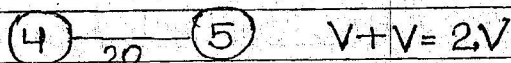
If edges already sorted, then worst case $O(E)$

= Prim's Algorithm -

[Vertex by Vertex]

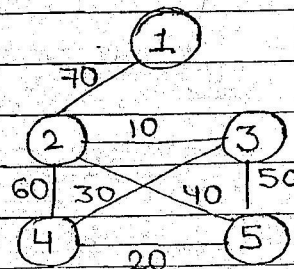
• Take any node. (Say 5)

• find adjacent of that node and take the minimum one.

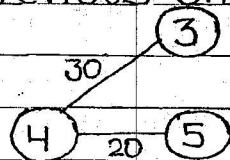


$V + V = 2V$

$= O(V)$

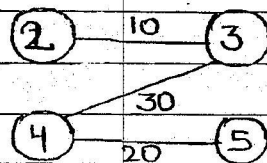


3. find adjacent of new vertex & take minimum of this & previous one.



[Always connected]

4. Repeat above to check for cycle.

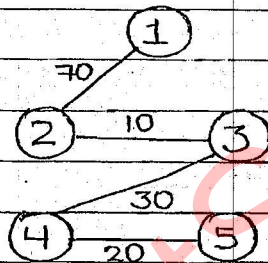


finding minimum

$$V + V = 2V$$

finding adjacent

5. Repeat until all nodes are covered.



Problem 1 Let G be an undirected weighted graph with n -vertices with different edge weights.

Let $e_{\max}$ be its edge with maximum edge wt. & $e_{\min}$ be edge with minimum edge weight. State T/F -

(a) G - UMST (Unique)

(b) Every edge MST of G must contain $e_{\min}$.

(c) Every MST of G never contain $e_{\max}$.

(d) Every MST of G must contain $e_{\max}$.

(e) If $e_{\max}$ is in MST, then its removal from G must disconnect G (S)

(f) $e_{\max}$ may be there in MST

(g) $e_{\min}$ may be there in MST

T
T
F
F

T
T
F

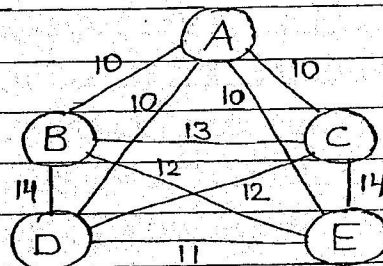
All edge weights are distinct, only one MST possible.
 One MST, not sure weights are distinct because of the cycle property.
 If not distinct, may or may not have one MST.

Problem 2.- Let G be an UWC graph with n -vertices and let w be its minimum edge-weight among all edge weights and e be specific edge with weight w . State T/F.

1. G is unique MST [F]
2. e should be there in every MST [F]
3. e may be there in every MST [T]
4. If e is not in MST of G then in that cycle all contain weight w . [T]
5. Every MST of G must contain atleast 1-edge with weight w . [T]

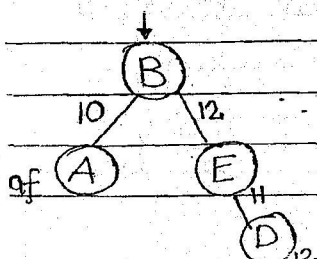
Problem 3 Consider the following graph

Cost of the MST? in such a way that A will be the leaf node.

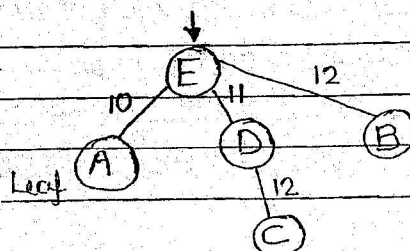


Prim's: Take B as starting node.

$$10 + 12 + 11 + 12 = 45$$

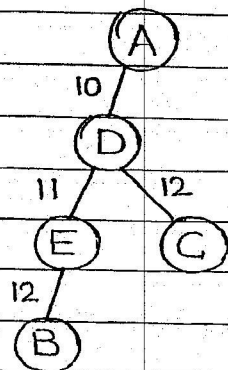


Take E as starting node



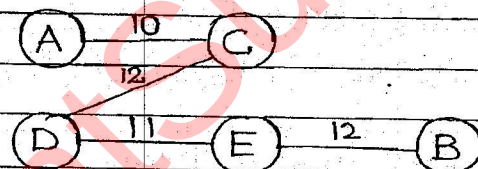
In any un-directed graph, a node is called a leaf node, if the degree is one. For directed graph, outdegree should be one.

Starting with 'A'



A degree have to be maintained.

Kruskal's: Take first minimum.



Problem 4: Let G be an undirected unweighted connected graph with n -vertices whose adjacency matrix is an $n \times n$ as given

(i) all diagonal elements are 0's

(ii) all non-diagonal elements are 1's

State True/False

(a) G contain unique MST with cost $n-1$

[F]

(b) G contain multiple MST with different cost

[F]

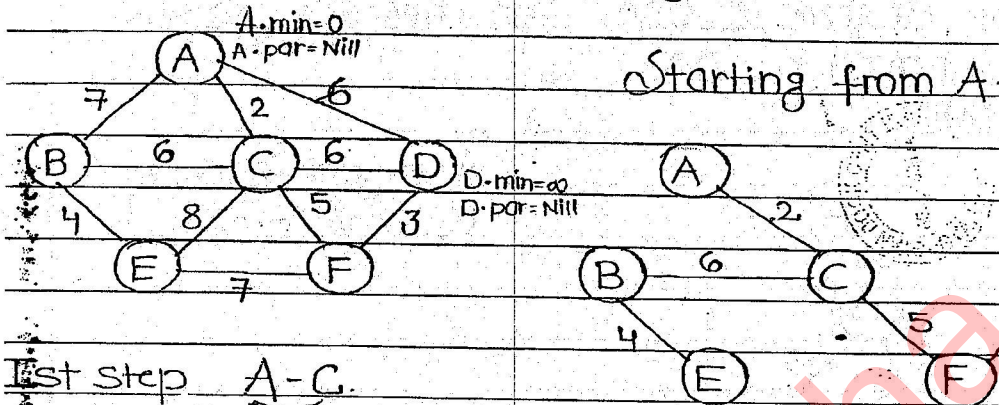
(c) G contain multiple MST with same cost $n-1$

[T]

(d) G don't have MST.

[F]

Time complexity of Prim's Algorithm -



Wrong order - (A,C) (C,F) (B,E) (F,D) (C,B)
(adjacency fails).

(A,C) (F,D) (C,F), (C,B) (B,E) fails

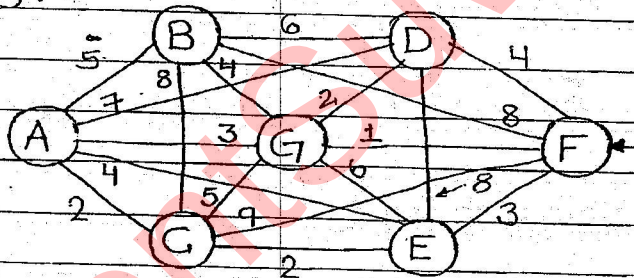
Insert in min-heap

	A	B	C	D	E	F	
	∞	∞	∞	∞	∞	∞	$\Rightarrow O(V)$
	N	N	N	N	N	N	
A	7	2	6	∞	∞		
	A	A	A	N	N		Deletion [3 log v]
C	6		6	8	5		
	C		A	C	C		Change-key [4 log v]
F	6		3	7			
	C		F	F			[2 log v]
D							[-]
B							[log v]
					2		[-]
					F		

If array instead of min-heap, then deletions will take $O(V)$ for one element. for changing keys, constant time, then

Time complexity = $O(V^2) + O(E)$
 $= O(V^2)$

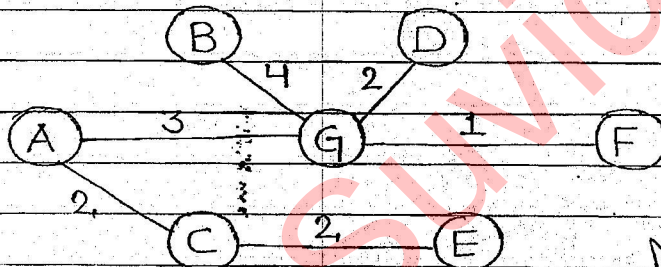
Problem 5:

[illegible]

A	4	(2)	3	$\Rightarrow 3 \log V + 5$
	G	A	F	
C	4		(2)	$\Rightarrow 2 \log V + 5$
	G		G	
E	(4)			$\Rightarrow 0 \log V + 5$
	G			
B				$\Rightarrow 0 \log V + 5$

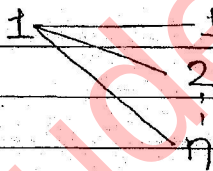
Adjacency list

$$E \log V + 2E$$

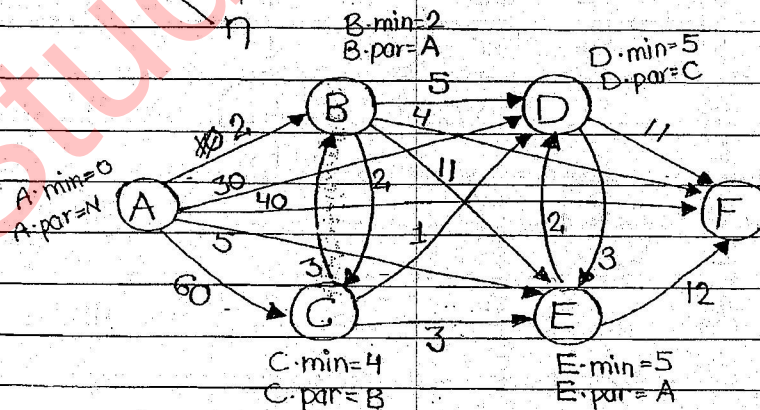


Minimum Spanning Tree

VI Single Source Shortest Path:-



EX-1.



$$\begin{aligned}
 A-A &= 0 \\
 -B &= 2 \\
 -C &= 4 \\
 -D &= 5 \\
 -E &= 5 \\
 -F &= 6
 \end{aligned}$$