UNIT - III

PART-1 — Greedy Method.

→ Mainly used to solve optimization problem.

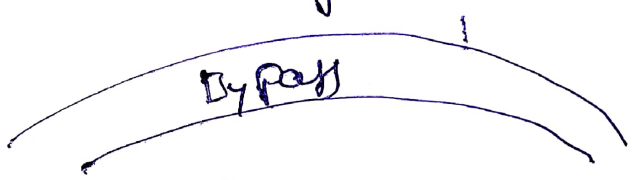
→ Optimization problem — which can have Multiple Alternate.

→ Procedure (or) process which gives a solution — is called as feasible solution.

→ Set of feasible solution based on some constraints

→ Best solution among set of feasible solution is known as optimal solutions. —

Eg:- In Real world you are travel



feasible Sol^{n} = { Bypass, streets }

If you use Bypass — less trafic — <1hr Bypass — 15 KMs

streets — trafic & signals — ≥1hr streets — 10 KMs

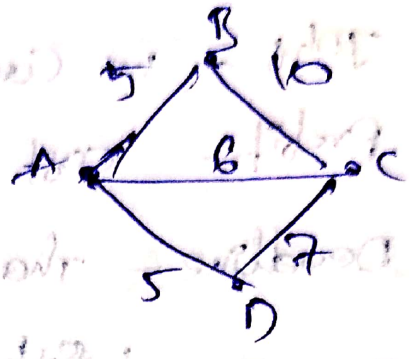
Constraints :-

walk in b/w City.

→ these types problems are Sloved in Greedy method.

Application of Greedy method.

* Selection sort
* finding shortest path.
* fractional knapsack
* Job sequencing
* finding Minimal Spanning Tree



A → C — via B — 16

A → C — 6

A → C — via D — 12

Example:-

25/- → Need change with Minimum number of coins.

So, coins available — 5/-, 2/-, 1/-.

The Sol^{n}-1 ⇒ 5 * 5/- → 5 coins ✓ selected

Sol^{n}-2 ⇒ 4 * 5/- + 2 * 2/- + 1 * 1/- → 7 coins

Sol^{n}-3 ⇒ 25 * 1/- → 25 coins.

Sol^{n}-4 ⇒ 2 * 5/- + 5 * 2/- + 5 * 1/- ⇒ 12 coins

Sol^{n}-1 is selected because of the constraint Minimum Number of coins.

Job sequencing with deadline

① Jobs - we can consider as no. of process.

② Profit - cost

③ Deadline - that particular job should complete with in deadline, and get max. profit.

→ we need to remember

* only one processor.
* No preemption.
* Every job takes 1 unit for completion

Example:-

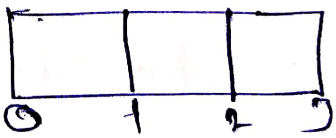
Let us consider 5 jobs

Jobs →	J_1	J_2	J_3	J_4	J_5
profit →	15	10	20	25	5
Deadline →	2	1	1	3	2

So, we have to find out the MAXIMUM deadline.

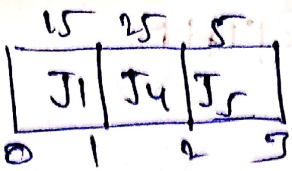
MAX Deadline = 3

Now, we need to draw Gantt chart.



next, sort all the jobs in descending with respect to profit.

Sorted profit	J_4	J_3	J_1	J_2	J_5
	25	20	15	10	
	3	1	2	1	5
					2

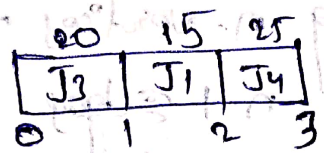


⇓

Profit → 15+25+5 = 45.

→ this is one feasible soln, there may be a lot of soln's in that we select optimal soln. → now in order to find out that max. profit.

So,



⇓

profit → 20+15+25 ⇒ (60). Highest profit

Alg:-

Step-1:- find maximum dead line

Step-2:- Draw Gantt chart with maximum dead line

Step-3:- Sort all Jobs with respect to profit.

Step-4:- Search Empty slot to fix Job. having in deadline near to its max deadline.

Time Complexity:-

$O(n \log n) + O(n^2)$ ← n-Jobs, n-slots. i.e (n^2)

Time Complexity, $T_c = O(n^2)$

Knapsack problem

"n" objects are given. Each object is having Profit (P_i) & weight (w_i). we have a knapsack or bag. whose weight is denoted by 'm'.

⇒ The Main objective is to place all the objects in knapsack with Maximum Profit.

⇒ Optimal Soln: is represented by a knapsack vector x_i.

x_i value denotes $\boxed{0 \le x_i \le 1}$

if your object is placed into knapsack then $x_i = 1$

if your object is not placed into " " $x_i = 0$

$x_i = 1$, $x_i = 0$, $x_i = \frac{\text{Remaing Size of knapsack}}{\text{Actual weight of the object.}}$

Algo: ① calculate $\frac{P_i}{w_i}$ for every object

② Arrange all the objects in decreasing order based on P_i/w_i

③ Place objects in knapsack.

Example:-

5 objects → $(P_1, P_2, P_3, P_4, P_5)$

$m = 100$ → size of knapsack.

$= (20, 30, 66, 40, 60)$ corresponding ~~cost~~ profits.

$= (10, 20, 30, 40, 50) \Leftarrow (w_1, w_2, w_3, w_4, w_5)$

Sol:-

$\frac{P_1}{w_1} = \frac{20}{10} = 2$,

$\frac{P_2}{w_2} = \frac{30}{20} = 1.5$,

$\frac{P_3}{w_3} = \frac{66}{30} = 2.2$,

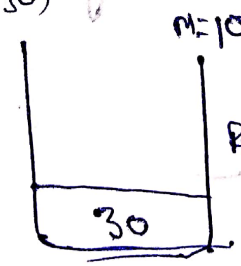
$P_4 = \frac{40}{40} = 1$

$P_5 = \frac{60}{50} = 1.2$

Step ② arrange ↓ing order.

$\frac{P_3}{w_3} > \frac{P_1}{w_1} > \frac{P_2}{w_2} > \frac{P_5}{w_5} > \frac{P_4}{w_4}$ Objects are placed into knapsack in this order.

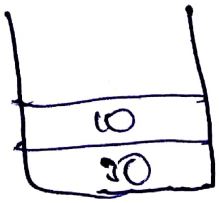
(P_3, w_3)
$(66, 30)$



Remaining Size of bag = 70

$x_3 = 1$

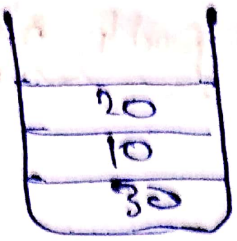
(P_1, w_1)
$20, 10$



R.S. of bag = 60

$x_1 = 1$

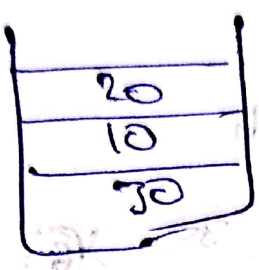
(P_2, w_2)
$(30, 20)$



R.S.k — 40

$x_2 = 1$

(P_3, w_3)
$(60, 30)$



Full

$x_3 = \frac{40}{50} = \frac{4}{5}$ fraction of object is placed i.e 40kg.

So, knapsack fulled there is no more space.

$x_4 = 0$ we can't placed in bag.

Next calculate profit. $= \sum P_i x_i$

$= 30 \times 1 + 30 \times 1 + 66 \times 1 + 40 \times 0 + 60 \times \frac{4}{5}$

$= 20 + 30 + 66 + 48$

Max. profit. $= 164$

PRIM'S ALGORITHM

Spanning Tree

↓

SubGraph $\subseteq$ G(V,E)

with no cycles.

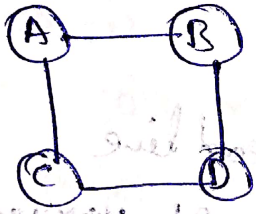
→ Spanning tree should connect all vertices of Graph.

→ If Graph is having n-vertices then Spanning Tree should have n-1 edges

→ should not contain cycles.

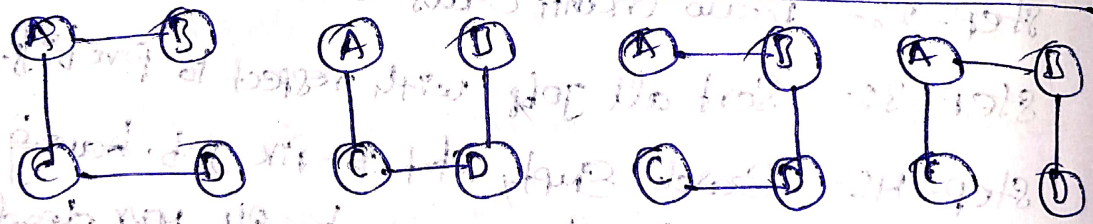
Let us take an Example: with 4 nodes

Eg①



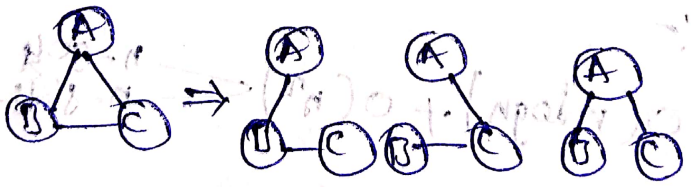
⟹ now construct SpanningTree

no. of vertices must be Equal.



All these are Spanning Trees.

Eg②



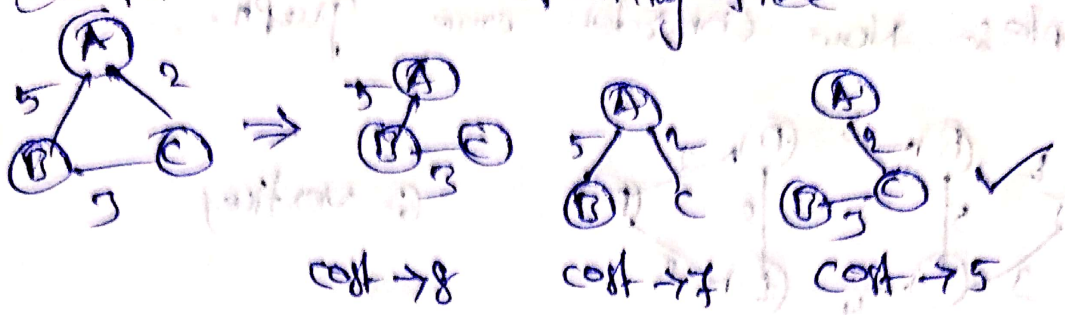
Minimum Spanning Tree (or)

Minimum cost Spanning Tree :-

Each Edge having Some weights.

Now Consider an Example.

Graph → Spanning tree



There are two ways to find the minimum cost spanning trees. i.e

① PRIM's Algorithm &

② Kruskal's Algorithm

PRIM's Algorithm

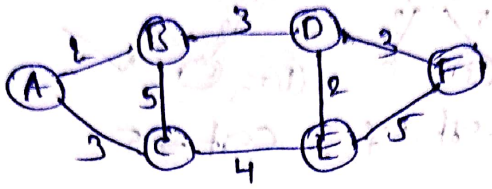
→ Graph should be weighted, connected & undirected.

→ Step-1: consider any vertex

Step-2: find all edges from selected vertex to all new vertices select least weighted edge and include it in Tree. If least weighted edge forms a cycle then discard it and consider next least weighted edge and include it in tree

Step-3: Repeat step-2 until all the vertices are included in Tree.

Example:- Now Consider one graph.



6-vertices

Consider any vertex ie

(A)

→ A↔B-2✓

A↔C-3

now tree is

→

~~A↔C-3~~

B↔C-5

B↔D-3✓

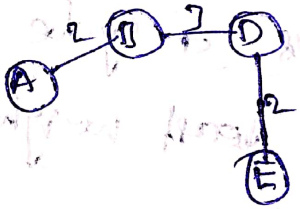
→

(A)↔C-3

B↔C-5

D↔E-2✓

D↔F-3



→

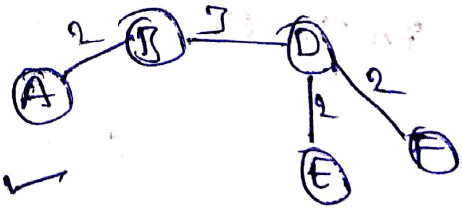
A↔C-3

B↔C-5

D↔F-2✓

E↔C-4

E↔F-5



→

A↔C-3✓

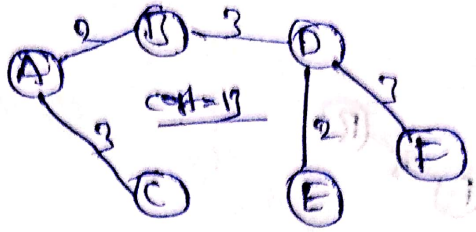
B↔C-5

E↔C-4

E↔F-5

F↔D-3✗

→ So,



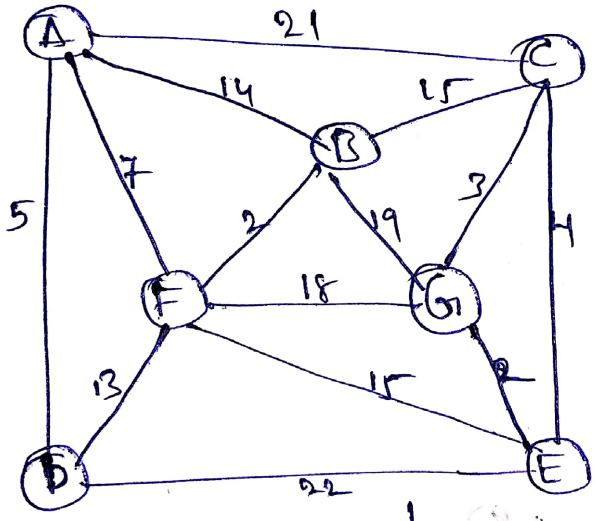
All vertices are covered.
We obtain here.

B → C — 5
E → C — 4
E → F — 5

} discard because cycle formed.

∴ Cost = 3+2+3+3+2 = 13

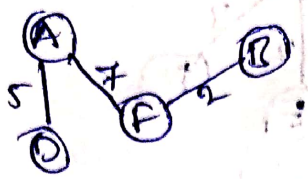
Example — 2:-



1 → A
A → D — 5 ✓
A → F — 7
A → B — 14
A → C — 21

2 → A, D
5
A → F — 7 ✓
A → B — 14
A → C — 21
D → F — 13
D → E — 22

3 → A, D, F
5, 7
A → B — 14
A → C — 21
D → F — 13
D → E — 22
F → E — 15
F → G — 18
F → B — 2 ✓

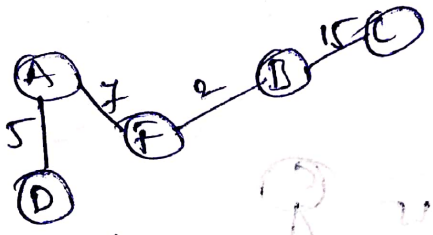
→ 

consider rerais edges i.e

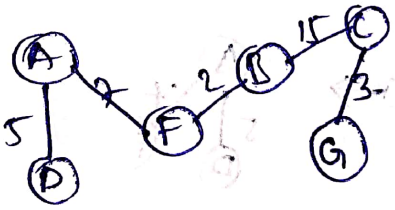
A → B — 14 ✓
A → C — 21
D → F — 18 ✗
D → E — 22
F → E — 15 ✓
F → G — 18
B → G — 19
B → C — 15 ✓

we form cycle so, discard. then we consider f→E(on ✓ D→C -15

now the Tree will be

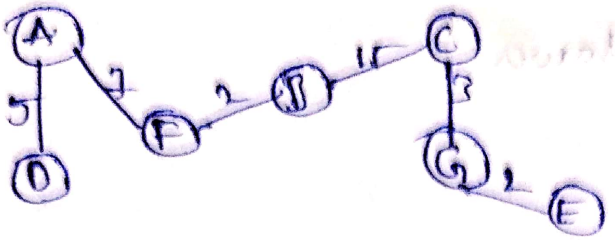
→ 

A → B — 14
A → C — 21
D → E — 22
F → E — 15
F → G — 18
B → G — 19
C → G — 3 ✓
C → E — 4

→ 

✗ A → B — 14
✗ A → C — 21
✗ D → E — 22
✗ F → E — 15
✗ F → G — 18
✗ B → G — 19
✗ C → E — 4
✗ G → F — 18
G → E — 2 ✓

→



Here all the vertices has been covered.
this is minimum cost spanning tree.

vertices — 7
edges — 6.

cost is — 5+7+2+15+3+2 ⇒ 34

Min. cost = 34

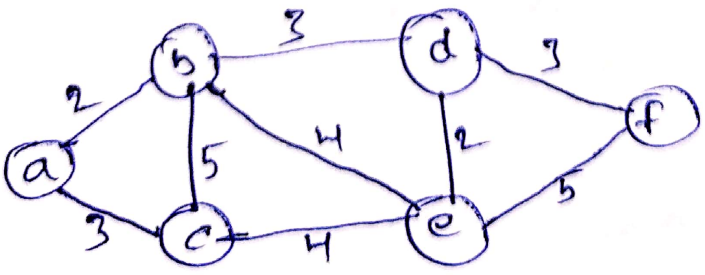
(2) KRUSKAL'S Algorithm

step-1 :- Arrange all the Edges in ASCENDING order based on cost.

step-2 :- Consider least cost weighted edge and include it in tree. if the Edge forms a cycle just ignore and Consider next least cost weighted edge.

step-3 :- Repeat step-2 until all the vertices are included in Tree.

Eg :-



(1)

a ↔ b – 2 ✓ lower

d ↔ e – 2 ✓

a ↔ c – 3 ✓

b ↔ d – 3 ✓

d ↔ f – 3 ✓

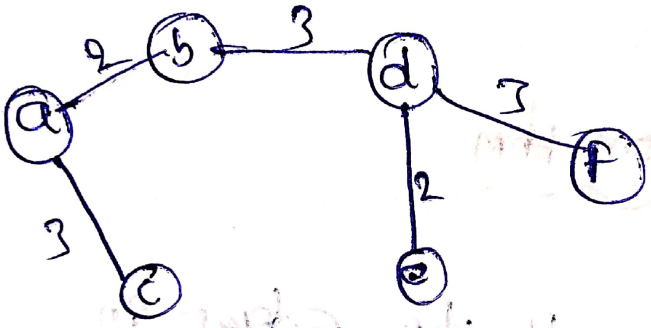
b ↔ e – 4 ✗

c ↔ e – 4 ✗

b ↔ c – 5 ✗

e ↔ f – 5 ✗ Heplet

(2)



MST

Graph having 6 vertices

Spaing Tree having 6 vertices

" " " 5 edges (vertices – 1) ✓

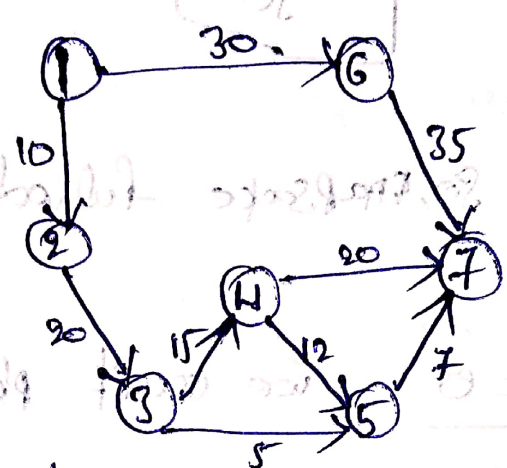
Minimum Cost Spaing Tree = 3 + 2 + 3 + 3 + 2 = 13 ✓

Single Source Shortest Path Algorithm

[Dijkastra's Alg]

Let $G_1 = (V, E)$

Directed weighted Graph



Let us consider two columns

Selected vertex	visited vertex Set	d[2]	d[3]	d[4]	d[5]	d[6]	d[7]
1	{1}	(10)	∞	∞	∞	30	∞
2	{1,2}	(10)	(30)	∞	∞	30	∞
3	{1,2,3}	(10)	(30)	45	35	(30)	∞
6	{1,2,3,6}	(10)	(30)	45	(35)	(30)	65
5	{1,2,3,6,5}	(10)	(30)	45	(35)	(30)	(42)
7	{1,2,3,6,5,7}	(10)	(30)	(45)	(35)	(30)	(42)
4	{1,2,3,6,5,7,4}	(10)	(30)	(45)	(35)	(30)	(42)

1	col
2	10
3	30
4	45
5	35
6	30
7	42