UNIT 1 - PROBLEM SOLVING AND ANALYSIS

Q1) What is an Efficient Algorithm?
→ that takes least amount of execution time and memory usage possible while yielding a correct answer
2) Good Algorithm? → Correct
3) Great Algorithm? → Correct & Efficient.

* Every algorithm must satisfy the following criteria :-
- Input - zero or more quantities that are externally supplied
- Output - at least one quality is produced.
- Definiteness -
- Finiteness - [ppt 5]
- Effectiveness -

* Efficiency of Algorithm :-
→ Defined as number of computational resources used by the algorithm
→ Algorithm must be analyzed to determine its resource usage.
→ Can be measured based on the usage of different resources.
→ Maximum efficiency of algorithm ↑ ; ↓ minimize resource usage.
→ Imp resources such as time and space complexity cannot be compared directly, so time and space complexity could be considered for an algorithmic efficiency.

* Algorithm Efficiency => A measure of ~~age~~ average execution time necessary for an algorithm to complete work on a set of data.

→ Efficiency of algorithm is mainly defined by two factors: time and Space complexity.

→ A good algorithm less time and space.

→ Space Efficiency :- Measure of amount of memory needed for an algorithm to execute
or complexity

→ Time Efficiency :- Measure of amount of time required for an algo to execute.
or complexity.

* Space Complexity :- [two components]
① Fixed part - total space required to store certain data and variable for an algorithm.
② Variable part - total space required by variables, size depends on problem & its iteration

* Time Complexity :-
→ A measure of the amount of time for an algorithm to execute.
→ Time is measured by counting the number of key operations like comparisons in the sorting algorithm.
→ Depends on :-
- Computer Speed
- Compiler
- Amount of Data
- Actual Data.

→ 3 Cases :- Worst Case time Complexity
→ Average Case time Complexity
→ Best case time Complexity.

→ Two main measures for the efficiency of an algo :-
- Time Complexity
- Space "

* Best Case :- Situation which requires the least number of operations.

* Worst Case :- The worst case is when targetNumber is the very last item in the list, since requiring the repeating the 3 loop operations for every item in the list.

* Worst Worst Case :- The situation where target Number is not in the list at all.

* Average Case :- When target number is the middle of the list

* Complexity of Linear Search Algorithm :- is O(n)
  - → Best Case → O(1)
  - → Average Case → O(n/2)
  - → Worst Case → O(n)
  - → Space Complexity → O(1)

* Time Complexity :-
  - → Constant Time Complexity → O(1)
  - → Linear Time Complexity → O(n)
  - → Logarithmic time Complexity → O(log n)
  - → Quadratic Time Complexity [O(n²)]
  - Exponential Time Complexity : O(2^n)

O(N!)	Factorial
O(2^N)	Exponential
O(N³)	Cubic
O(N²)	Quadratic
O(N log N)	N x log N
O(N)	Linear
O(log N)	Logarithmic
O(1)	Constant

Fastest / Highest Efficiency

* Asymptotic Notations / order of magnitude / Asymptotic Categorization.

→ Languages that uses meaningful statements about time & space complexity.

→

① Big O

→ describes worst case

→ describe upper bound (worst case) of a asymptotic function.

$f(n) = O(g(n))$

↳ f of n is big Oh of g of n

iff (if and only if) positive constants c and n_0 exist such that

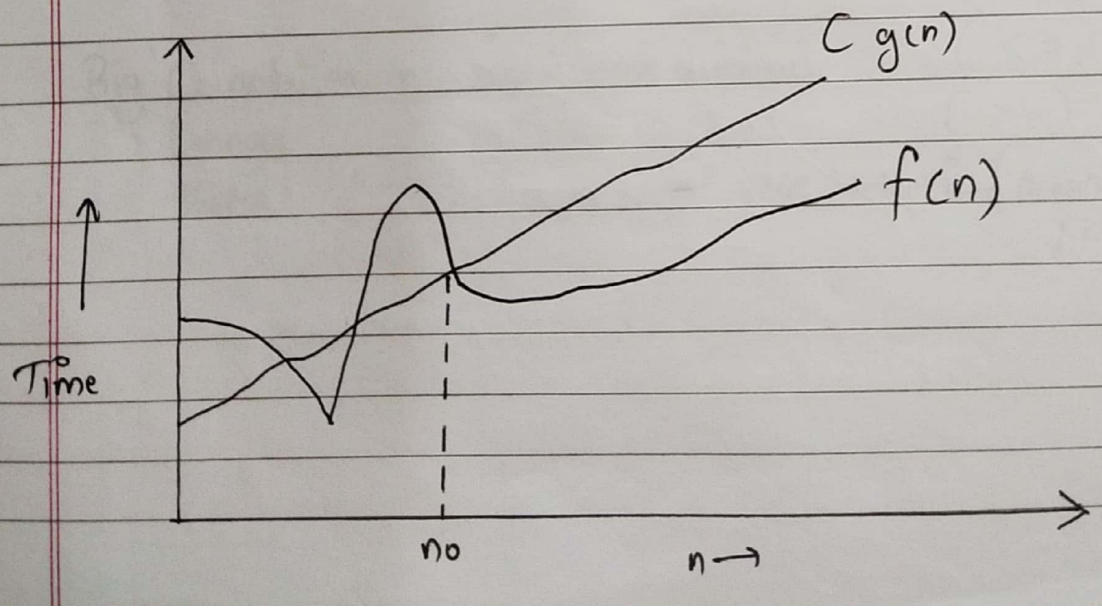
$f(n) <= cg(n)$ for all n, $n >= n_0$.

[n = used to give upper bound on a function]

$$|6x^4 - 2x^3 + 5| \leq 6x^4 + |2x^3| + 5$$
$$\leq 6x^4 + 2x^4 + 5x^4$$
$$= 13x^4$$

So,

$$|6x^4 - 2x^3 + 5| = 13x^4$$



(2) Big Omega Notation (Ω):-

→ Big Omega is the reverse Big O.

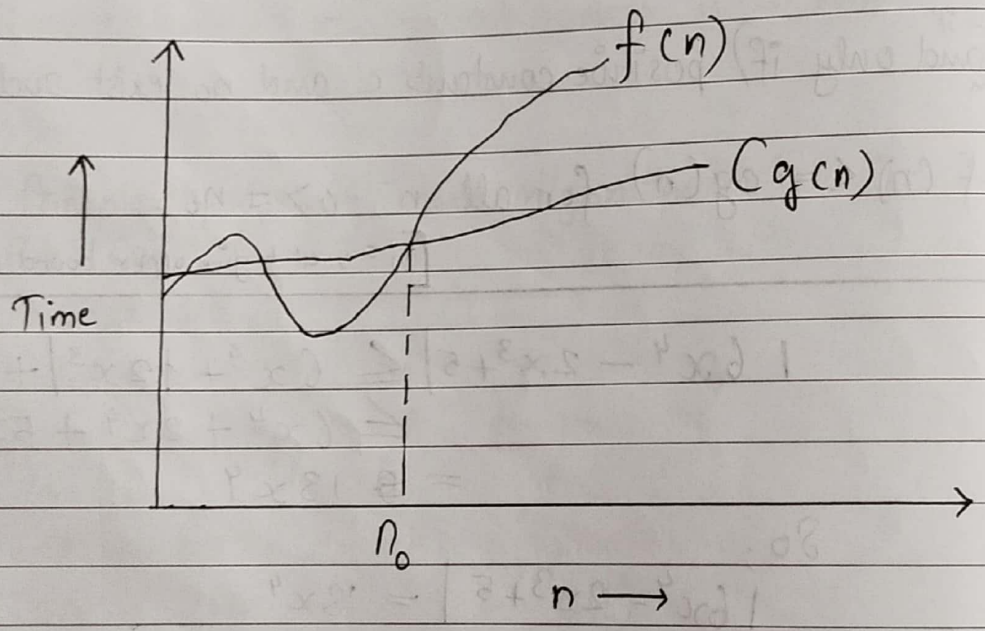
→ lower bound (best-case)

$f(n) = \text{omega}(g(n))$, ~~where~~

↳ f of n is omega of g of n,

iff (if and only if) positive constants 'c' & n_0 exist such that

$$f(n) >= cg(n) \text{ for all } n;\ n >= n_0$$



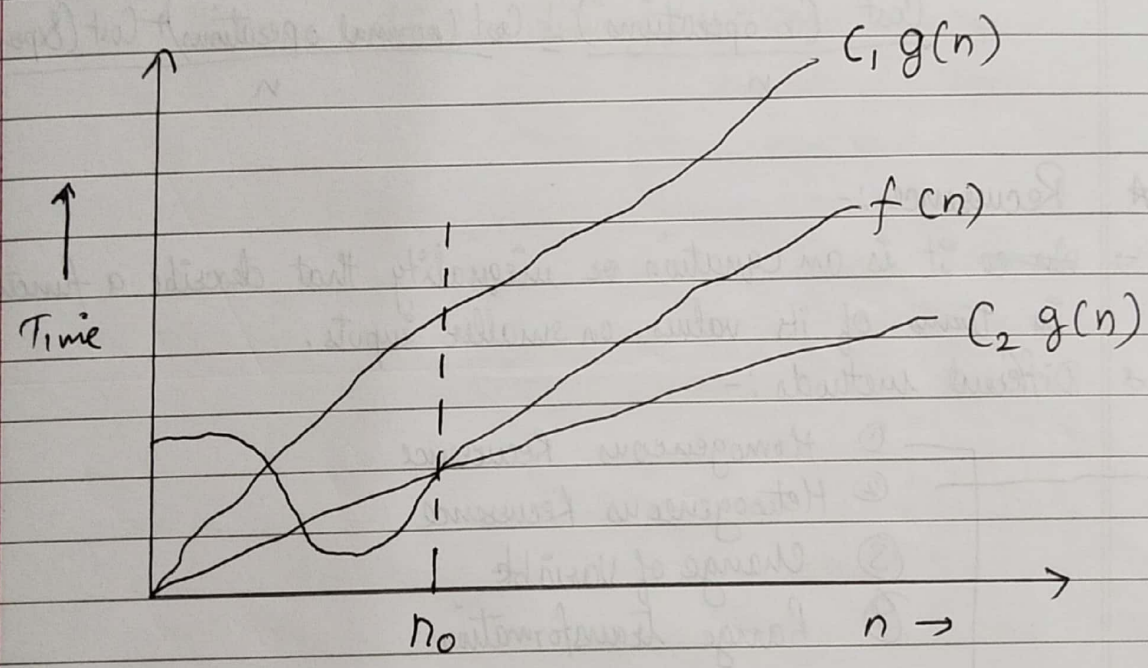
(3) Big Theta Notation Θ

→ lower bound = upper bound → tightest bound

$f(n) = \text{Theta}(g(n))$,
↳ f of n is theta of g of n

iff (if and only if) positive constants c_1, c_2 & n_0 exist such that

$c_1 g(n) <= f(n) <= c_2 g(n)$ for all n, $n >= n_0$.



Big O notation – worst case analysis (<=) Upper
Big Omega – best case analysis (>=) lower
Big theta – best and worst case analysis (average case) (==)
[tightest boond]

★ Amortized Analysis :-

→ Considers costly & less costly operations together over the whole sequence of operations.

→ Data structures : we need AA in Hash tables, Disjoint sets, Splay ~~Rees~~ Trees.

★ Performing Amortized Analysis :-

① Aggregate method

$$\frac{\text{Cost (n operations)}}{n} = \frac{\text{Cost (normal operations)}}{n} + \text{Cost (Expensive O)}$$

★ Recurrence :-

→ ~~An eq~~ it is an equation or inequality that describe a function in terms of its values on smaller inputs.

& Different methods :-

① Homogeneous Recurrence
② Heterogeneous Recurrence
③ Change of Variable
④ Range transformation

→ The recurrence relation is called homogeneous where

$$f(n) = 0$$

$$a_0 t_n + a_1 t_{n-1} + a_2 t_{n-2} + a_3 t_{n-3} = 0$$

where $a_0, a_1, a_2 \ldots$ are constant

$$a_0 t_n + a_1 t_{n-1} + \ldots a_n t_{n-k} = b^{p(n)}$$

where $0 \le i \le k$

b is constant

p(n) is a polynomial in n of degree d.

★ Change of Variable

→ possible to solve more complicated recurrences relation by making a change of variable.

$$T(n) = 3T(n/2) + n$$

★ Range Transformation :-

→ fundamental Recurrences

↳ $S(n) = S(n-1) + d(n)$

where $S(n) = 0$

$d(n)$ = driving functions

GREEDY ALGORITHMS AND DYNAMIC PROGRAMMING :-

→ Greedy Progeamming :-
- Minimal Spanning Tree
- Shortest Path
- Job Scheduling
- Traveling Salespeeson Problem

→ Dynamic Progeamming :-
- Chained Matrix Multiplication
- Optimal Search Tree
- 0/1 Knapsack

★ Greedy Approach
- Always makes the best choice that seems to be the best at that moment.
- It never goes back & reverses the decision.
- Never changes the judgement even if the option is wrong.
- Decision is made on the basis of current information only, regardless of future impact.
- Makes greedy choices.
- Maximum or minimum result.

★ Components of Greedy Algorithm :-
① Candidate Set - Solution created from set.
② Selection function - used to choose the candidate or subset.
③ Feasibility function - used to determine whether the candidate or subset can be used to contribute to the solution.
④ Objective function - used to assign the value to the solution or the partial solution

(5) Solution function - used to estimate whether the complete function has been reached or not.

★ Applications of Greedy Algorithm:-
→ used to find shortest path,
→ used to find minimum spanning tree using Prim's algo or Kruskal's Algo.
→ used in job sequencing
→ " to solve fractional knapsack problem.

★ Shortest Path Algorithm:-
→ To find a route between any pair of vertices along the edges, so the sum of weights of edge is minimum.

★ Common shortest Path Algorithms:-
- Dijkstra's Algo
- Bellman ford's Algo
- Floyd Warshall's Algo
- Johnson's Algo

★ Dijkstra's Algorithm:-
→ find the shortest path between a given node & all other nodes in a graph.
→ This algo uses the weights of the edges to find the path that minimizes the total distance (weight) between source node & other nodes.

Time Complexity - $O(E \log V)$ E- no. of edges
V- no. of vertices.

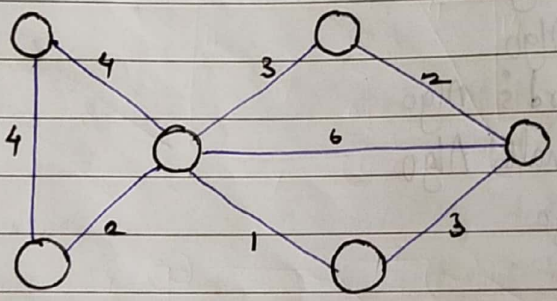
Space Complexity - $O(V)$

* Dijkstra's Algorithm Applications :-
- To find the shortest path
- In social networking applications
- In a telephone network
- To find the locations in the map

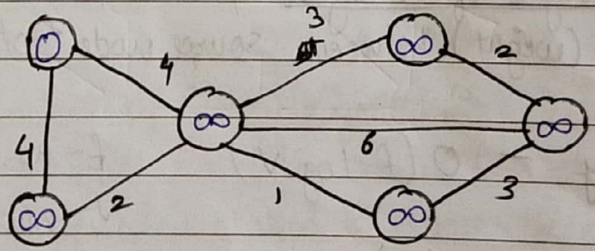
* which algorithm is better than Dijkstra's Algorithm?
- We have negative weights, we have to go with the Bellman-Ford algorithm.

* Example of Dijkstra's Algorithm :-

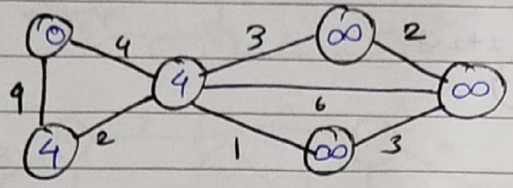
Step 1 - Start with a weighted graph



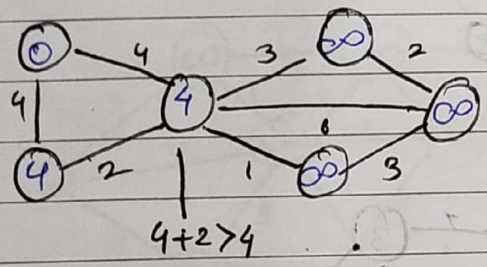
Step 2 - Choose a starting vertex and assign infinity path values to all other devices.



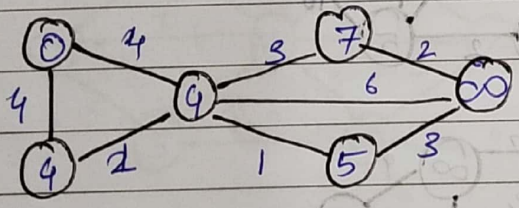
Step 3 - Go to each vertex and update its path length.



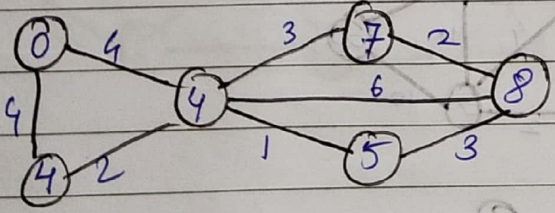
Step 4 - If the path length of the adjacent vertex is lesser than new path length, don't update it.



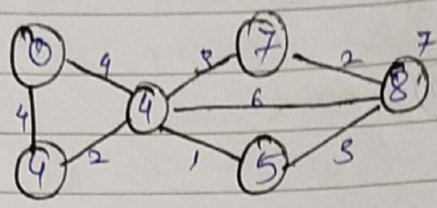
Step 5 - Avoid updating path lengths of already visited vertices :-



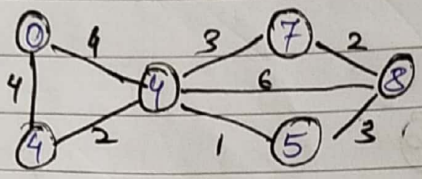
Step 6 :- After each iteration, we pick the unvisited vertex with the least path length. So we choose 5 before 7.



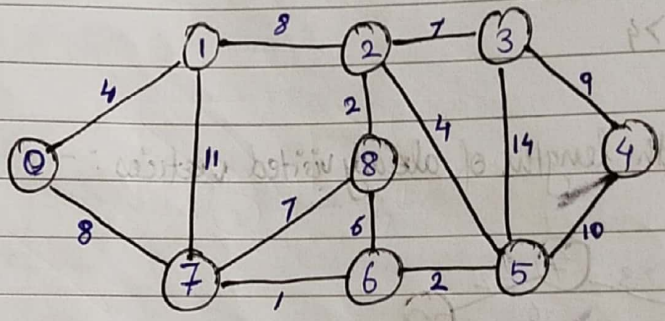
Step 7:- Notice how the rightmost vertex has its path length updated



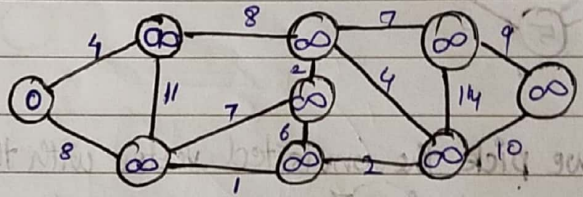
Step 8 - Repeat until all the vertices have been visited.



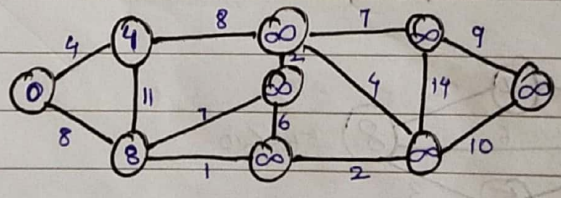
Eg (2)



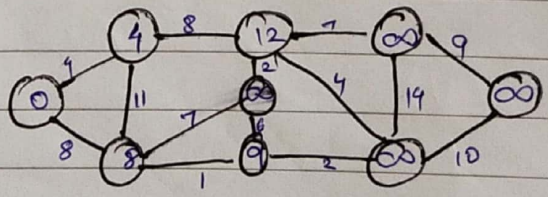
(1)



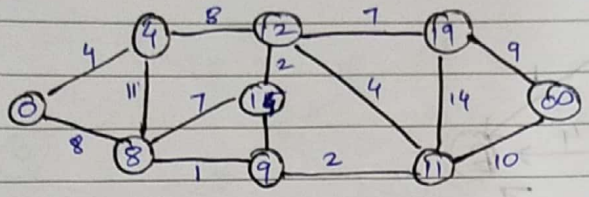
(2)



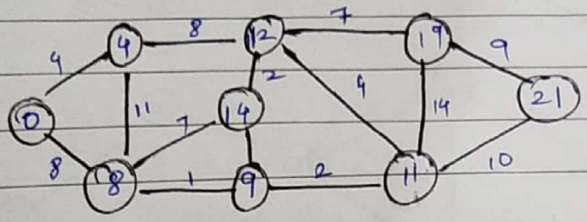
(3)



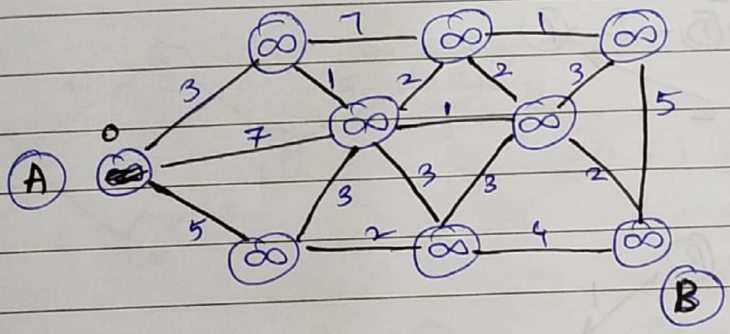
(4)

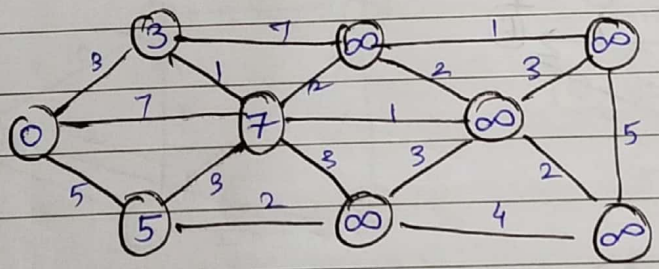


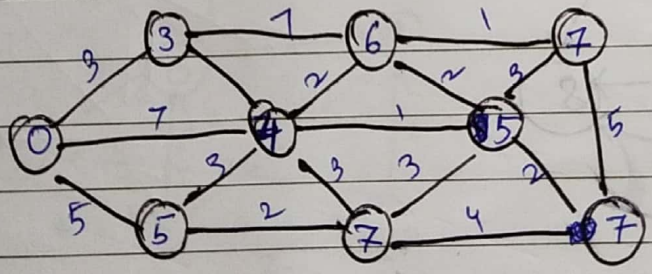
(5)



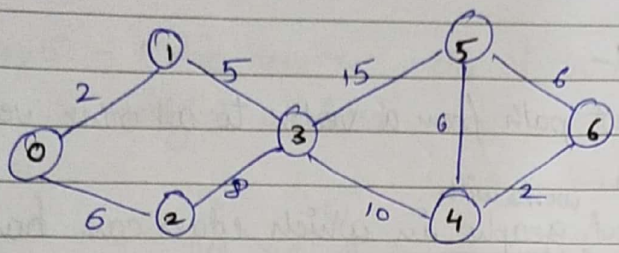
Eg (3)





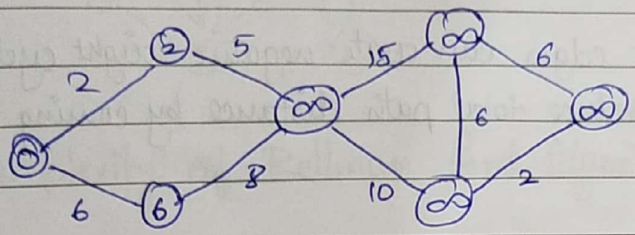


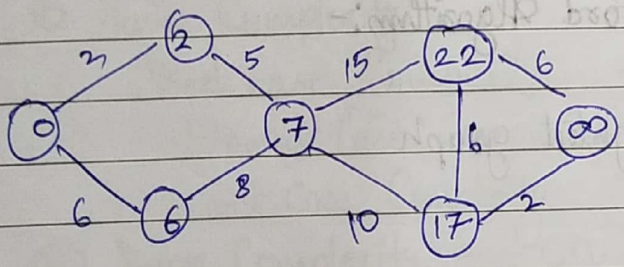
Eg (4)



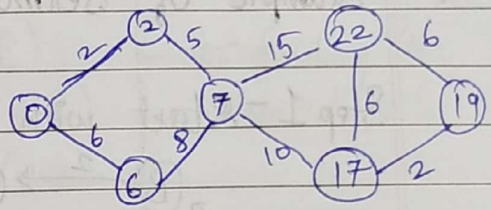
(a)

=)

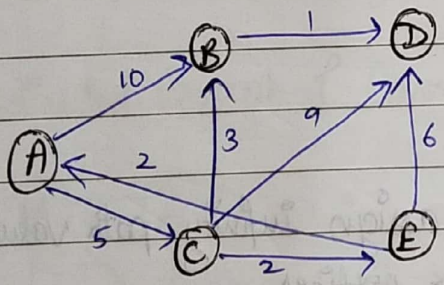




=)

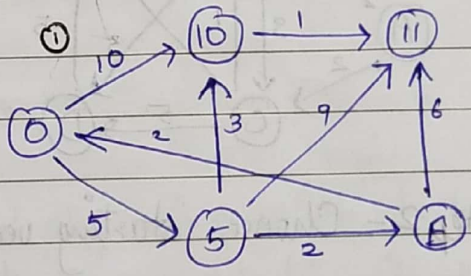


Eg (5)

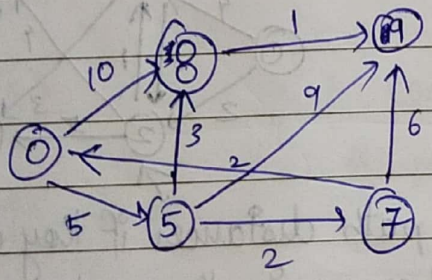


=

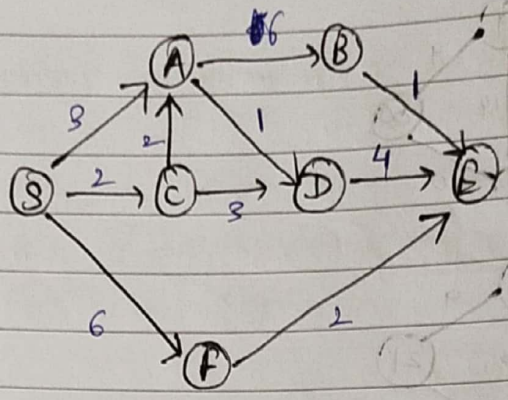
(1)



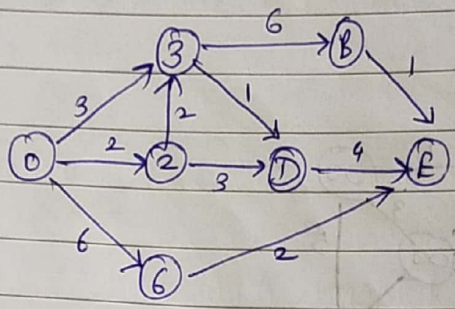
(2)



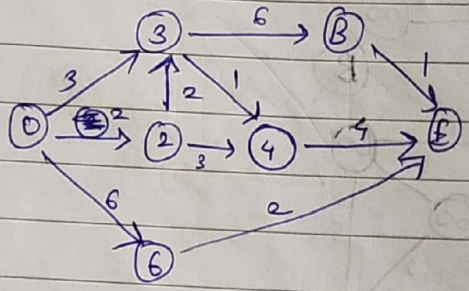
Eg (4)



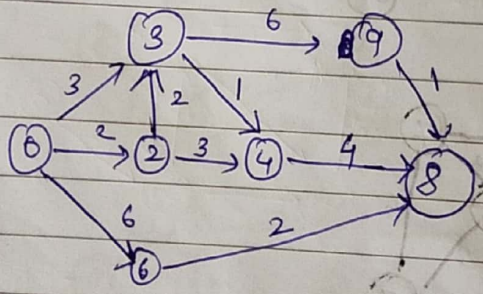
=)



=)



=)



☆ Bellman ford Algorithm:-

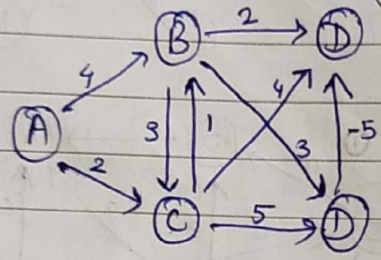
→ Helps to find the shortest path from a vertex to all other vertices of a weighted graph.

→ Similar to dijkstra's but works with graphs in which edges can have negative weight.

→ IMP :- Negative weight edges can create negative weight cycles ie a cycle that will reduce total path distance by coming back to the same point.

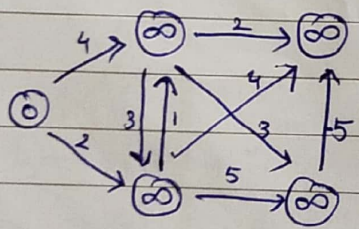
☆ Example of Bellman ford Algorithm:-

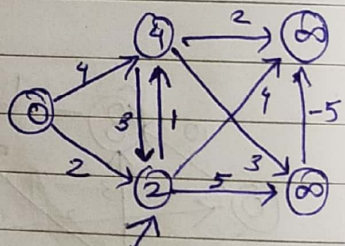
Step 1 - Start with weighted graph



Doubt

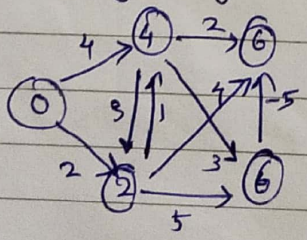
Step 2 - Choose starting vertex and assign infinity path values to all other vertices.



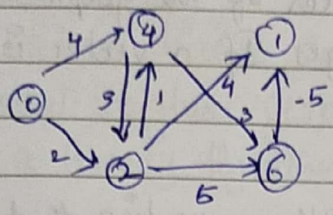


Step 3 - Visite each edge & relax the path distances if they are inaccurate

Step 4 - We need to do this V times because in the worst case, a vertex's path length might need to be readjusted V times.



Step 5 :- Notice how the vertex at the top right corner had its path length adjusted.



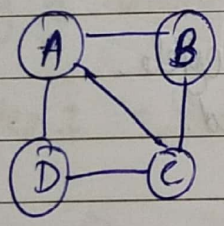
★ Complexity of Bellman Ford Algorithm :-

① Time Complexity
- Best Case Complexity - O(E)
- Average Case Complexity - O(VE)
- Worst Case Complexity - O(VE)

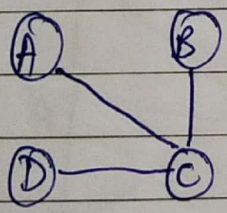
② Space Complexity - O(V)

Applications :-
- for calculating shortest paths in routing algo.
- for finding the shortest path.

★ Undirected Graph & Connected Graph :-



→ An undirected graph is a graph in which the edges do not point in any direction (ie the edges are bidirectional).



→ A connected graph in which there is always a path from a vertex to any other vertex.

★ Spanning Tree:-

→ A spanning tree is a sub-graph of an undirected connected graph, which includes all the vertices of the graph with a minimum possible number of edges. If a vertex is missed, then it is not a spanning tree.

→ The total number of spanning tree with n vertices that can be created from a complete graph is equal to $n^{(n-2)}$

★ Minimal Spanning Trees (MST)

→ No. of vertices in the spanning tree would be the same as the no. of vertices in the graph.

→ Two conditions exists :-

- $V' = V$
- The no. of edges in the spanning tree would be equal to the number of edges minus 1.
  $E = |V| - 1$
- The spanning tree should not contain any cycle.
- The spanning tree should not be disconnected.

★ Methods of Minimum Spanning Tree:-

① Kruskal's Algorithm → starts from edge
② Prim's Algorithm → starts from vertex.

— edges O vertices

★ Kruskal's Algorithm :-

→ It is an algo to construct a Minimum Spanning Tree for a connected weighted graph.

→ It is used to discover the shortest path between two points in a connected weighted graph.

→ It is a Greedy algorithm.

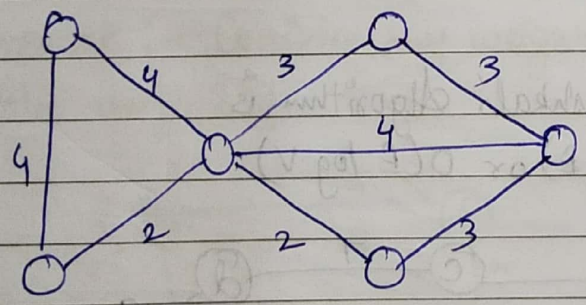
★ How it works ?

- Sort all the edges from low weight to high
- Take the edge with the lowest weight & add it to the spanning tree. If adding the edge created a cycle, then reject this edge
- Keep adding edges until we reach all vertices.

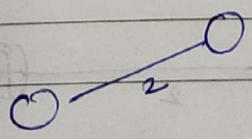
- The total time is $O(E \log E) = O(E \log V)$

★ Eg of Kruskal's Algorithm :-

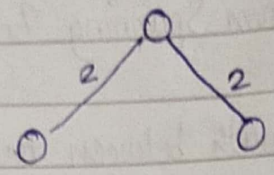
Step 1 :- Start with a weighted graph.



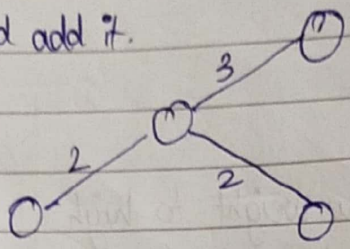
Step 2 - Choose the edge with the least weight, if there are more than 1 choose any 1.



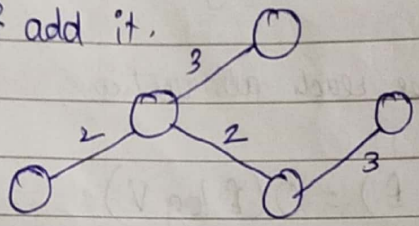
Step 3: Choose the next shortest edge & add it.



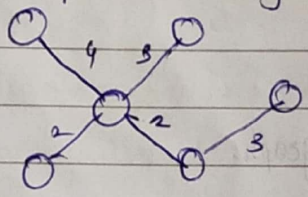
Step 4: Choose the next shortest edge that doesn't create a cycle and add it.



Step 5: Choose the next shortest edge that doesn't create a cycle & add it.

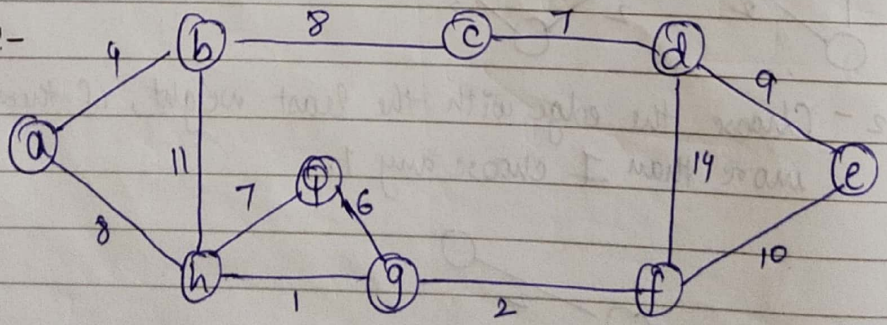


Step 6:- Repeat until you have a spanning tree.



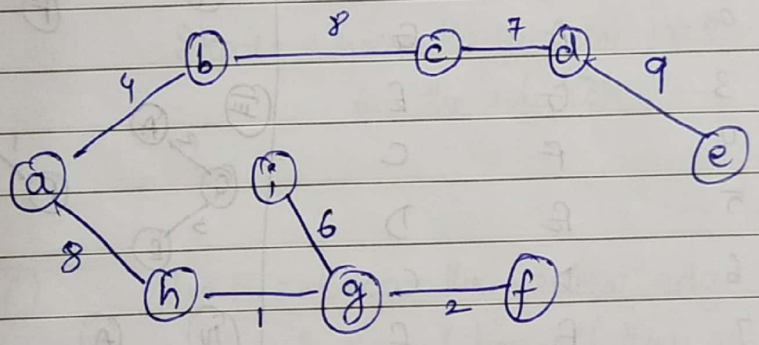
★ Time complexity of Kruskal's Algorithm is
$O(E \log E)$ or $O(E \log V)$

Eg:-



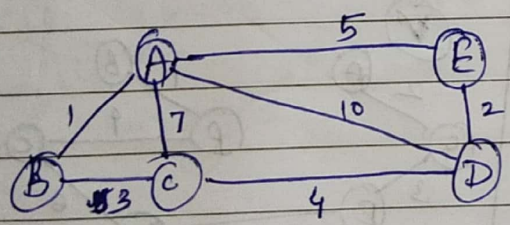
Weight	Source	Destination
1	h	g
2	g	f
4	a	b
6	i	g
7	c	d
7	h	i
8	a	h
8	b	c
9	d	e
10	f	e
11	b	h
14	d	f

→ Make it step by step.



* Homework :- Consider any weighted graph from an MST on it. Solve any 2 eg.

I



Weight	Source	Destination
1	A	B
2	E	D
3	B	C
4	C	D
5	A	E
7	A	C
10	A	D

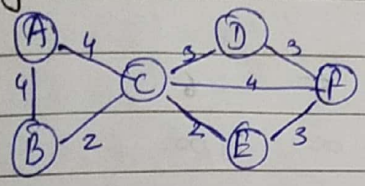
★ Prim's Algorithm :-

→ It is a greedy algorithm that finds the local optimum in the hopes of finding a global optimum.

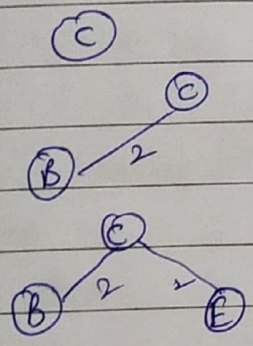
→ Starts with empty spanning tree.

→ At every step, it considers all the edges & picks the minimum weight edge. After picking the edge, it moves the other endpoint of edge to set containing MST.

Eg



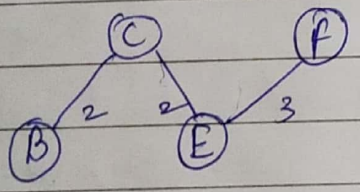
Step 1 - Start with a weighted graph.



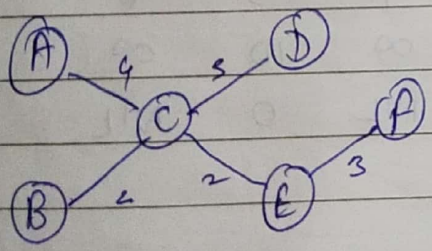
Step 2 - Choose a vertex

Step 3 - Choose the shortest edge from this vertex & add it.

Step 4 - Choose the nearest vertex not yet in the solution

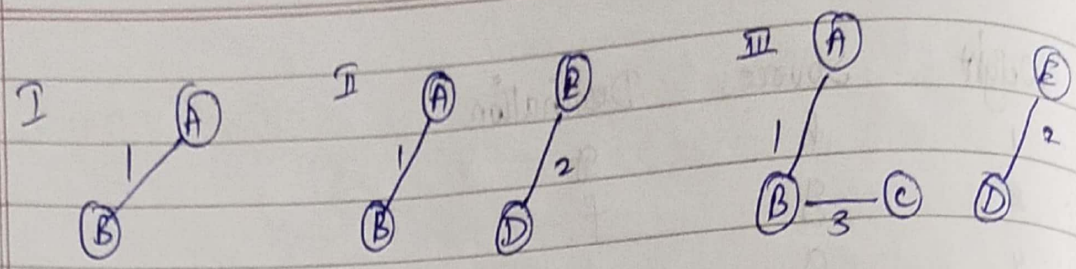


Step 5 - Choose the nearest edge not yet in the solution, if they are multiple choices, choose ~~any~~ one at random

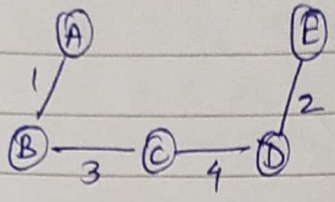


Step 6 - Repeat until you have a spanning tree.

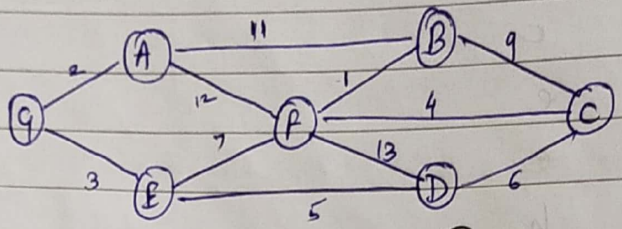
Time Complexity of Prism's Algorithm $O(E \log V)$.



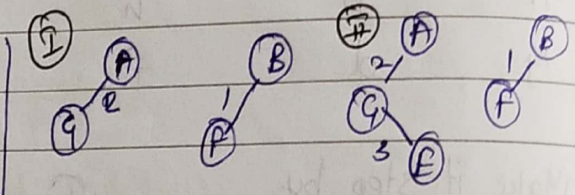
IV

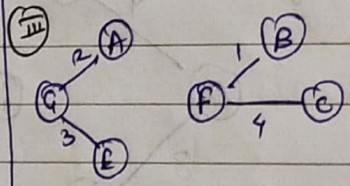


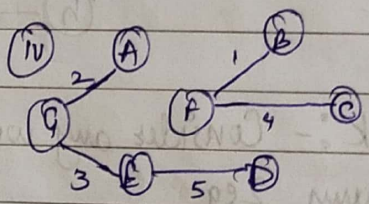
(2)

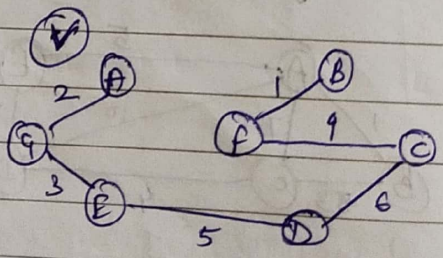


Weight	Source	Destination
1	F	B
2	A	G
3	G	E
4	F	C
5	E	D
6	D	C
7	E	f
9	B	C
11	A	B
12	A	f









∴ data key value & parent of 4 :-

Vertex	0	1	2	3	4	5	6
Key Value	0	28	∞	∞	25	10	∞
Parent	NIL	0	NIL	NIL	5	0	NIL

Step III :-

Adj[4] = {6, 3}

Key[6] = ∞ ; Key[3] = ∞

w(4, 6) = 24 ; w(4, 3) = 22

w(u, v) < key[v] ; w(u, v) < key[v]

w(4, 6) < key[6] ; w[4, 3) < key[3]

∴ Update key value of 6 as => 24 and 3 => 22

& π(3 & 6) = 4 ; π(3) = 4 ; π(6) = 4

[Now by EXTRACT_MIN(Q) Removes 3, because key[3] = 2]

Vertex	0	1	2	3	4	5	6
Key value	0	28	∞	22	25	10	24
Parent	NIL	0	NIL	4	5	0	4

Step IV :-

Adj[3] = {4, 6, 2}

key[4] = 25 ; key[6] = 24 ; key[2] = ∞

4 is already in heap & key[6] = 18 ;

∴ key[2] = ∞

w(3, 2) = key[2]

12 < ∞

∴ π[2] = 3

∴ key[2] = 12

key[6] = 18

w(3, 6) = key(6)

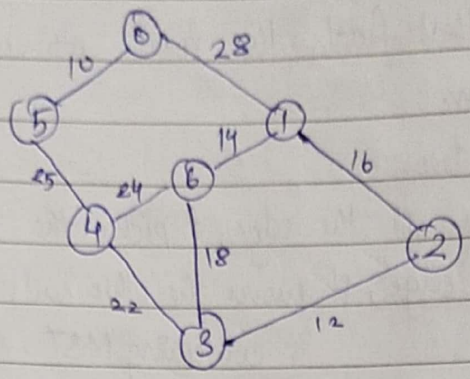
18 < ∞

∴ π[6] = 3

key[6] = 18

[Now by EXTRACT_MIN(Q) Removes 2, because key[2] = 12 is minimum]

Example :-



~~Step 1 :-~~

Vertex :	0	1	2	3	4	5	6
Key value :	0	∞	∞	∞	∞	∞	∞
Parent :	NIL	NIL	NIL	NIL	NIL	NIL	NIL

Consider here vertex 0 as root.

Step I :- Root = 0 ; Adj [0] = 5, 1

∴ Parent; $\pi(5) = 0$ & $\pi(1) = 0$

∴ key [5] = 0 ∞ & key [1] = ∞

∴ $w(0,5) = 10$; $w(0,1) = 28$

$w(u,v) < key[5]$; $w(u,v) < key[1]$

key [5] = 10 ; key [1] = 28

So we update key of 5 & 1 :-

Vertex	0	1	2	3	4	5	6
Key value	0	28	∞	∞	∞	10	∞
Parent	NIL	0	NIL	NIL	NIL	0	NIL

Step II :-

Adj [5] = 0, 4 ∴ Not considering 0; since its already visited.

Taking 4, key [4] = ∞ $\pi[4] = 5$

(u, v) < key [v] then key [4] = 25

$w = (5,4) = 25$; $w(5,4) < key[4]$

∴ 25 < ∞

Step V :-

~~Adj[2] = 1, 3 Adj[1]~~

Adj[2] = 1, 3

key[3] is a heap.

key[1] = 28 but

~~key[1]~~ $w(0,1) \not> w(1,2)$

$\therefore$ 28 > 16

$\therefore$ key[1] = 16 & $\pi(1) = 2$

Vertex	:	0	1	2	3	4	5	6
Key Values	:	0	16	12	22	25	10	18
Parent	:	NIL	2	3	4	5	0	3

Step VI :-

Adj[1] = {0, 6, 2}

0 & 2 are already in heap.

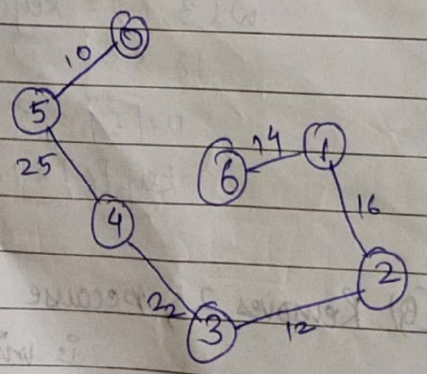
Taking 6, key[6] = 18

w[1,6] = 14

$\therefore$ w(1, 6) < key[6]

$\therefore$ key value of 6 is 14 & $\pi(6) = 1$

Vertex	0	1	2	3	4	5	6
Key Values	0	16	12	22	25	10	14
Parent	NIL	2	3	4	5	0	1



$\therefore$ Total Cost - 10+25+22+12+16+14

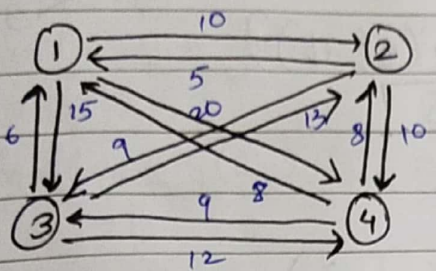
=> 99

★ Travelling Salesperson Problem (TSP) :-

→ To find the shortest possible route that visits every city exactly once & returns to the starting point.

→ The goal is to find a tour of minimum cost.

* Eg :-



Sol →

	1	2	3	4
1	0	10	15	20
2	5	0	9	10
3	6	13	0	12
4	8	8	9	0

① Starting Vertex => 1
Ending Vertex => 1

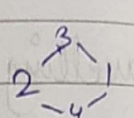
② Second last vertex can be either 2, 3, 4
Compute $g(i, \phi) = C_{ij}$
ie $g(2, \phi) = g(2,1) = C_{21} = 5$
$g(3, \phi) = g(3,1) = C_{31} = 6$
$g(4, \phi) = g(4,1) = C_{41} = 8$

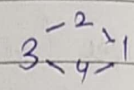
③ Three last vertex can have following paths :-
→

ⓐ $2 \to 3 \to 1$

$g(2, \{3\}) = \min(C_{23} + g(3, \phi))$

$= 9 + 6 = \boxed{15}$

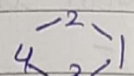




ⓑ $2 \to 4 \to 1$

$g(2, \{4\}) = \min(C_{24} + g(4, \phi))$

$= 10 + 8 = \boxed{18}$



ⓒ $3 \to 2 \to 1$

$g(3, \{2\}) = \min(C_{32} + g(2, \phi))$

$= 13 + 5 = \boxed{18}$

ⓓ $3 \to 4 \to 1$

$g(3, \{4\}) = \min(C_{34} + g(4, \phi))$

$= 12 + 8 = \boxed{20}$

ⓔ $4 \to 2 \to 1$

$g(4, \{2\}) = \min(C_{42} + g(2, \phi))$

$= 8 + 5 = \boxed{13}$

ⓕ $4 \to 3 \to 1$

$g(4, \{3\}) = \min(C_{43} + g(3, \phi))$

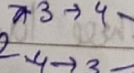
$= 9 + 6 = \boxed{15}$

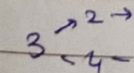
ⓖ Now last 4 vertex can have paths.

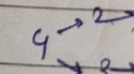
ⓐ $2 \to (3, 4) \to 1$

$\Rightarrow \min[C_{23} + g(3, 4, 1)]$

$\Rightarrow 9 + 20 \Rightarrow \boxed{29}$







(b) $2 \to (4,3) \to 1$

$\Rightarrow \min [C_{24} + g(4,3,1)] = 10 + 15 \Rightarrow \boxed{25}$

(c) $3 \to (2,4) \to 1$

$\Rightarrow \min [C_{32} + g(2,4,1)] = 13 + 18 = 31$

(d) $3 \to (4,2) \to 1$

$\Rightarrow \min [C_{34} + g(4,2,1)] = 12 + 13 \Rightarrow \boxed{25}$

(e) $4 \to (2,3) \to 1$

$\Rightarrow \min [C_{42} + g(2,3,1)] = 8 + 15 = \boxed{23}$

(f) $4 \to (3,2) \to 1$

$\Rightarrow \min [C_{43} + g(3,2,1)] = 9 + 18 = 27$

(5) Thus final path can be found as :-

$1 \to \{2,3,4\} \to 1$

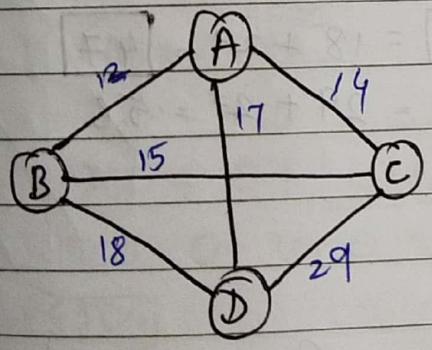
$g(1\{2,3,4\}) \Rightarrow \min (C_{12} + g(2\{3,4,1\}),$
$C_{13} + g(3\{2,4,1\}),$
$C_{14} + g[4\{2,3,1\}])$

$\Rightarrow \min (10 + 25,\ 15 + 25,\ 20 + 23)$ $\Rightarrow \min (35, 40, 43)$

$\Rightarrow 35$

$\therefore$ Path route $\Rightarrow 1 \to 2 \to 4 \to 3 \to 1$

Eg (2)



	A	B	C	D
A	0	12	14	17
B	12	0	15	18
C	14	15	0	29
D	17	18	29	0

(1) Starting Vertex = A

Ending Vertex = A

(2) Second last vertex can either be B, C, D :-

Compute $g(i, \phi) = C_{ij}$

$g(B, \phi) = g(B, A) = 12$

$g(C, \phi) = g(C, A) = 14$

$g(D, \phi) = g(D, A) = 17$

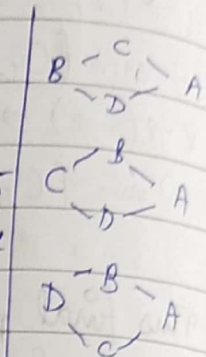
(3) There last vertex can have the following paths :-

(a) $B \to C \to A = g(B, \{C\}) = \min(C_{BC} + g(C, \phi)) = 15 + 14 = 29$

(b) $B \to D \to A = g(B, \{D\}) = \min(C_{BD} + g(D, \phi)) = 18 + 17 = 35$

(c) $C \to B \to A = g(C, \{B\}) = \min(C_{CB} + g(B, \phi) = 15 + 12 = 27$

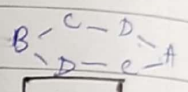
(d) $C \to D \to A = g(C, \{D\}) = \min(C_{CD} + g(D + \phi)$
$= 29 + 17 = 46$



(e) $D \to B \to A = g(D, \{B\}) = \min(C_{DB} + g(B + \phi) = 18 + 12 = 30$

(f) $D \to C \to A = g(D, \{C\}) = \min(C_{DC} + g(C + \phi) = 29 + 14 = 43$

(4) Now last four vertex can have paths :-



(a) $B \to (C, D) \to A = \min[C_{BC} + g(C, D, A)] = 15 + 46 = 61$

(b) $B \to (D, C) \to A = \min[C_{BD} + g(D, C, A)] = 18 + 43 = 61$

(c) $C \to (B, D) \to A = \min[C_{CB} + g(B, D, A)] = 15 + 35 = 50$

(d) $C \to (D, B) \to A = \min[C_{CD} + g(D, B, A)] = 29 + 30 = 59$

(e) $D \to (B, C) \to A = \min[C_{DB} + g(B, C, A)] = 18 + 29 = 47$

(f) $D \to (C, B) \to A = \min[C_{DC} + g(C, B, A)] = 29 + 27 = 56$

(5) The final path can be found as :-

$A \to (B, C, D) \to A =$

$g(A\{B, C, D\}) = \min(C_{AB} + g(B\{C, D, A\}),$
$C_{AC} + g(C\{B, D, A\}),$
$C_{AD} + g(D\{B, C, A\}))$

$\min(12 + 61,$
$14 + 50,$
$17 + 47) \Rightarrow \min(73, 64, 64)$

∴ The minimum Path route => $A \xrightarrow[14]{} C \xrightarrow[15]{} B \xrightarrow[18]{} D \xrightarrow[17]{} A = 64$

AND → $A \xrightarrow[17]{} D \xrightarrow[18]{} B \xrightarrow[15]{} C \xrightarrow[14]{} A = 64$

* Chained Matrix Multiplication:-

→ Method under dynamic Programming.

→ If a chain of matrices is given, we have to find the minimum number of the correct sequence of matrices to multiply.

* Example =) Input :- N = 4

Arr =) $\{10, 30, 5, 60\}$ (P_0 P_1 P_2 P_3)

$A_1 = [10, 30]$; $A_2 = [30, 5]$; $A_3 = [5, 60]$

	1	2	3	
1	0	1500	4500	1
2		0	9000	2
3			0	3

∴ $m(1,2) = A_1 \times A_2$

=) $[10 \times 30 \times 5]$

=) $[1500]$

$m(2,3) = A_2 \times A_3 = [30 \times 5 \times 60] = [9000]$

$m(1,3) = A_1 \times A_2 \times A_3$ =

$(A_1 * A_2) A_3$ ↓ ↓ $A_1 (A_2 * A_3)$

=) $1500 + 0 + [P_0 * P_2 * P_3]$ ← $A_1 \times A_2 \times A_3$

=) $1500 + 0 + 3000$

=) 4500

$[P_0 \times P_1 \times P_3] + 0 + 9000$

=) $(10 \times 30 \times 60) + 0 + 9000$

$18000 + 9000$

=) 27000

$(10 \times 30 \times 5)$

$(10 \times 5) \times (5 \times 60)$ ← A_3

$(10 \times 5 \times 60)$ =) $10 = P_0$

3000 $5 = P_2$

$60 = P_3$

∴ $A_1 \times A_3 = 4,500$; Parenthesis =)

$(A_1 * A_2) A_3$

$(a_1, a_2, a_3 \ldots a_n)$ (first, last)

$(a_1, a_2, \ldots a_7) a_8$ (first; last, last)

Eg[2] Input = N=4
are = [10, 100, 5, 50]
P_0 P_1 P_2 P_3
A_1 A_2 A_3

$A_1 = [10, 100]$
$A_2 = [100, 5]$
$A_3 = [5, 50]$

1	2	3	
0	5000	7500	1
	0	25000	2
		0	3

$A_1 \times A_2 = [10 \times 100 \times 5] = 5000$ ~~$A_2 \times A_3$~~
$A_2 \times A_3 = [100 \times 5] \times [5 \times 50] = [100 \times 5 \times 50] = 25000$

$A_1 \times A_3 \Rightarrow (A_1 * A_2) A_3$ | And $A_1 (A_2 * A_3)$

$m(1\times2) + m(3\times3) + P_0 * P_2 * P_3$ | $m(1\times1) + m(2\times3) + P_0 * P_1 * P_3$

$5000 + 0 + (10 \times 5 \times 50)$ | $0 + 25000 + (10 * 100 * 50)$

$5000 + 2500$ | $25000 + 50000$

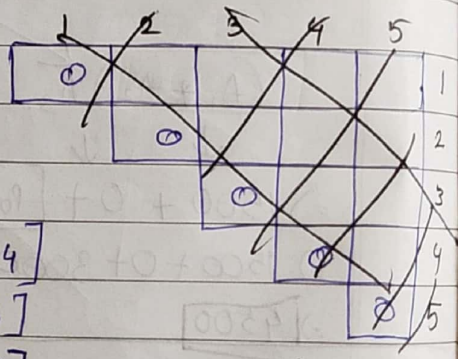
7500 | 75,000

$\therefore A_1 \times A_3 = m(1*3) = 7,500$

Parenthesis $\Rightarrow (A_1 * A_2) A_3$

Eg[3] N=5; are = {5, 4, 6, 2, 7}
P_0 P_1 P_2 P_3 P_4

1	2	3	4	
0	120	88	158	1
	0	48	104	2
		0	84	3
			0	4



$A_1 = [5, 4]$
$A_2 = [4, 6]$
$A_3 = [6, 2]$
$A_4 = [2, 7]$

$A_1 \times A_2 = [5 \times 4 \times 6] = 120$

$A_2 \times A_3 = [4 \times 6 \times 2] = 48$

$A_3 \times A_4 = [6 \times 2 \times 7] = 84$

$A_1 \times A_3 \Rightarrow (A_1 * A_2) A_3$ OR $A_1 (A_2 * A_3)$

$120 + 0 + [P_0 * P_2 * P_3]$ $\quad$ $[P_0 * P_1 * P_3] + 0 + 48$

$120 + 0 + (5 \times 6 \times 2)$ $\quad$ $(5 \times 4 \times 2) + 0 + 48$

$120 + 60 = 180$ $\quad$ $40 + 48 = \boxed{88}$

$\therefore A_1 \times A_3 \Rightarrow 88$; Parenthesis $\Rightarrow A_1 (A_2 * A_3)$

$A_2 \times A_4 \Rightarrow (A_2 * A_3) A_4$ OR $A_2 (A_3 * A_4)$

$48 + 0 + [P_1 * P_3 * P_4]$ $\quad$ $[P_1 * P_2 * P_4] + 0 + 84$

$48 + 0 + (4 \times 2 \times 7)$ $\quad$ $(4 \times 6 \times 7) + 84 = \boxed{168}$

$48 + 56 \Rightarrow \boxed{104}$

$A_1 \times A_4 \Rightarrow$

$M_1 (M_2 \times M_3 \times M_4)$ $\quad$ $(M_1 \times M_2 \times M_3) M_4$ $\quad$ $(M_1 \times M_2)(M_3 \times M_4)$

$0 + [5 \times 4 \times 7] + 104$ $\quad$ $[5 \times 2 \times 7] + 88 + 0$ $\quad$ $120 + 84 + \boxed{210}$

$140 + 104 = 244$ $\quad$ $70 + 88 = \boxed{158}$ $\quad$ $(5 \times 6 \times 7) = 210$

$120 + 84 + 210$

$330 + 84 = 414$

$A_1 \times A_4 = 158$; Parenthesis $\Rightarrow (M_1 \times M_2 \times M_3) M_4$

★ $N = 5$; arr $= [40, 20, 30, 10, 30]$

1 2 3 4 5

	1	2	3	4	
1	0	24000	14000	26000	1
2		0	6000	12000	2
3			0	9000	3
4				0	4

$A_1 = [40, 20]$

$A_2 = [20, 30]$

$A_3 = [30, 10]$

$A_4 = [10, 30]$

*

$A_1 = [40, 20]$
$A_2 = [20, 30]$
$A_3 = [30, 10]$
$A_4 = [10, 30]$

$A_1 \times A_2 = [40 \times 20 \times 30] = 24000$
$A_2 \times A_3 = [20 \times 30 \times 10] = 6000$
$A_3 \times A_4 = [30 \times 10 \times 30] = 9000$

$A_1 \times A_3 = (A_1 \times A_2) A_3$ AND $A_1 (A_2 \times A_3)$

$24000 + 0 + (40 \times 30 \times 10)$
$24000 + 12000$
36000

$0 + 6000 + (40 \times 20 \times 10)$
$6000 + 8000$
14000

$\therefore A_1 \times A_3 = 14000$; Parenthesis =) $A_1 (A_2 \times A_3)$

$\therefore A_2 \times A_4 = (A_2 \times A_3) A_4$ AND $A_2 (A_3 \times A_4)$

$6000 + 0 + (20 \times 10 \times 30)$
$6000 + 6000$
12000

$0 + 9000 + (20 \times 30 \times 30)$
$9000 + 18000$
27000

$\therefore A_2 \times A_4 = 12000$; Parenthesis :- $(A_2 \times A_3) A_4$

$A_1 \times A_4 \Rightarrow A_1 (A_2 \times A_3 \times A_4)$
$\Rightarrow 0 + 12000 + (40 * 20 * 30)$
$\Rightarrow 12000 + 24000$
$\Rightarrow 36000$

$(A_1 \times A_2 \times A_3) A_4$
$\Rightarrow 14000 + 0 + (40 * 10 * 30)$
$\Rightarrow 14000 + 12000$
$\Rightarrow$ **26000**

$(A_1 * A_2)(A_3 * A_4)$
$24000 + 9000 + (40 \times 30 \times 30 \times 30)$
$24000 + 9000 + (40 \times 30 \times 30)$
$24000 + 9000 + 36000$
69000

$\therefore A_1 \times A_4 = 26,000$; Parenthesis is $(A_1 \times A_2 \times A_3) A_4$

Q $N=6$; arr $[4, 10, 3, 12, 20, 7]$ ($P_0\ P_1\ P_2\ P_3\ P_4\ P_5$)

$A_1 = [4, 10]$; $A_2 = [10, 3]$; $A_3 = [3 \times 12]$; $A_4 = [12 \times 20]$

$A_5 = [20 \times 7]$

$A_1 \times A_2 = [4 \times 10 \times 3] = 120$

$A_2 \times A_3 = [10 \times 3 \times 12] = 360$

$A_3 \times A_4 = [3 \times 12 \times 20] = 720$

$A_4 \times A_5 = [12 \times 20 \times 7] = 1680$

	1	2	3 (264)	4	5	
1	0	120	144	1074	1344	1
2		0	360	~~3480~~ 1320	1350	2
3			0	720	1140	3
4				0	1680	4
5					0	5

$A_1 \times A_3 = (A_1 \times A_2) A_3$

$= 120 + 0 + (4 \times 3 \times 12)$ - wrong calculation

$= 120 + 24 =$ **144**

$\Rightarrow A_1 (A_2 \times A_3) = 0 + 360 + (4 \times 10 \times 12)$

$= 360 + 480 = 840$

$\therefore A_1 \times A_3 \Rightarrow (144)$; Parenthesis $= (A_1 \times A_2) A_3$

$A_2 \times A_4 \Rightarrow (A_2 \times A_3) A_4$

$\Rightarrow 360 + 0 + (10 \times 12 \times 20)$

$\Rightarrow 360 + 2400 = 2760$

$A_2 (A_3 \times A_4)$

$\Rightarrow 0 + 720 + (10 \times 3 \times 20)$

$\Rightarrow 720 + 600 \Rightarrow 1320$

$A_2 \times A_4 = 1320$; Parenthesis $\Rightarrow A_2 (A_3 \times A_4)$

$A_3 \times A_5 \Rightarrow (A_3 \times A_4) A_5$

$\Rightarrow 720 + 0 + (3 \times 20 \times 7)$

$\Rightarrow 720 + 420 = 1140$

AND

$\Rightarrow A_3 (A_4 \times A_5)$

$\Rightarrow 0 + 1680 + (3 \times 12 \times 7)$

$\Rightarrow 1680 + 252 \Rightarrow 1932$

$A_3 \times A_5 = 1140$; Parenthesis $= (A_3 \times A_4) A_5$

$A_2 \times A_5 \Rightarrow A_2 (A_3 \times A_4 \times A_5)$

$0 + 1140 + (10 \times 3 \times 7)$

$\Rightarrow 1140 + 210$

$\Rightarrow$ ~~650~~ 1350

$(A_2 \times A_3 \times A_4) A_5$

$\Rightarrow 1320 + 0 + (10 \times 20 \times 7)$

$\Rightarrow 1320 + 1400$

$\Rightarrow 2720$

$(A_2 \times A_3)(A_4 \times A_5) \Rightarrow 360 + 1680 + (10 \times 12 \times 7)$

$\Rightarrow 360 + 1680 + 840$

$\Rightarrow 2880$

~~$A_3 \times A_5 \Rightarrow 1350$~~ ; Parenthesis $\Rightarrow A_3 (A_3 \times A_4 \times A_5)$

$A_1 \times A_4 \Rightarrow A_1 (A_2 \times A_3 \times A_4)$

$\Rightarrow 0 + 1320 + (4 \times 10 \times 20)$

$\Rightarrow 1320 + 800$

$\Rightarrow 2120$

$(A_1 A_2)(A_3 \times A_4)$

$(4 \times 3 \times 20) + 120 + 720$

$\Rightarrow 240 + 840$

$\Rightarrow 1080$

$(A_1 \times A_2 \times A_3) A_4$

$\Rightarrow 114 + 0 + (4 \times 12 \times 20)$

$\Rightarrow 114 + 960$ This is wrong.

$\Rightarrow 1074 \rightarrow$ ~~1080~~ 1224

$\therefore A_1 \times A_4 = 1074$

$\therefore$ Parenthesis $\Rightarrow A_1 (A_2 \times A_3 \times A_4)$

$\therefore$ Ans is 1080

$\therefore A_1 \times A_5 \Rightarrow m_1\ m_2\ m_3\ m_4\ m_5$ can be solved in following paths :-

(i) $(m_1 * m_2 * m_3 * m_4) * m_5$

(ii) $m_1 (m_2 * m_3 * m_4 * m_5)$

(iii) $(m_1 * m_2 * m_3)(m_4 * m_5)$

(iv) $(m_1 * m_2)(m_3 * m_4 * m_5)$

$\therefore$ (i) $(m_1 * m_2 * m_3 * m_4)\ m_5$

$\therefore (m_1 \times m_4) + m_5 + [P_0 * P_4 * P_5]$

$\Rightarrow 1074 + 0 + (4 \times 20 \times 7)$

$\Rightarrow 1074 + 560$

$\Rightarrow 1634$

(ii) $m_1 (m_2 * m_3 * m_4 * m_5)$

$\therefore m_1 + 1350 + (4 \times 10 \times 7)$

$\therefore 0 + 1350 + 280$

$\Rightarrow 1630$

(iii) $(m_1 * m_2 * m_3)(m_4 * m_5)$

$\Rightarrow 144 + 1680 + (4 \times 12 \times 7)$

$\Rightarrow 1824 + 336$

$\Rightarrow 2160$

(iv) $(m_1 * m_2)(m_3 * m_4 * m_5)$

$\Rightarrow 120 + 1140 + (4 \times 3 \times 7)$

$\Rightarrow 1260 + 84$

$\Rightarrow 1344$

$\therefore A_1 \times A_5 \Rightarrow 1344$;

Parenthesis $\Rightarrow (m_1 * m_2)(m_3 * m_4 * m_5)$

* 0/1 Knapsack Problem :-

→ val [0 ... N-1] & wt [0 ... N-1]

→ Either pick the complete item or don't pick it (0-1 property).

→ A knapsack (kind of shoulder bag) with limited weight capacity,

→ few items each having some weight and value.

* The problem states :-

→ Which items should be placed into the knapsack such that –

- The value or profit obtained by putting the items into the knapsack is maximum,
- And the weight limit of the knapsack does not exceed.

* Knapsack Problem Variants –

- Has two variants :-
  1. fractional Knapsack Problem
  2. 0/1 Knapsack Problem

* 0/1 Knapsack Problem Using Dynamic Programming :-

- Knapsack weight capacity = w
- No of items each having some weight and value = n

$$T(i,j) = \max\{T(i-1,j), value_i + T(i-1, j-weight_i)\}$$

=) Eg 1

Item	Weight	Value
1	2	3
2	3	4
3	4	5
4	5	6

$\therefore (n+1) = 4+1 = 5$ no. of rows

$(w+1) = 5+1 = 6$ no. of columns

* Solution

	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	3	3	3	3
2	0	0	3	4	4	7
3	0	0	3	4	5	7
4	0	0	3	4	5	7

① $T(1,1)$ => where $i=1$; $j=1$; $Value_i = 3$; $Weight_i = 2$

$\therefore T(i,j) = \max\{T(i-1,j), value_i + T(i-1, j-weight_i)\}$

=) $T(1,1) = \max\{T(1-1,1), 3 + T(1-1, 1-2)\}$

=) $= \max\{T(0,1), 3 + T(0,-1)\}$

$= \max\{T(0,1), Ignore\}$

$\therefore T(1,1) = 0$

(2) $T(1,2) \Rightarrow i=1; j=2;$ Value$_i = 3$; Weight$_i = 2$;

$T(i,j) = \max\{T(i-1,j), \text{Value}_i + T(i-1, j-\text{weight}_i)\}$

$= \max\{T(1-1,2), 3 + T(1-1, 2-2)\}$

$= \max\{T(0,2), 3 + T(0,0)\}$

$= \max\{0, 3\} \Rightarrow 3$

$T(1,3) \Rightarrow i=1; j=3;$ Value$_i = 3$; Weight$_i = 2$;

~~$T(i,j) = \max\{T(i,j-1),$~~

$\Rightarrow T(i,j) = \max\{T(i-1,j), \text{value}_i + T(i-1, j-\text{weight}_i)\}$

$\Rightarrow T(1,3) = \max\{T(1-1,3), 3 + T(1-1, 3-2)\}$

$= \max\{T(0,3), 3 + T(0,1)\}$

$= \max\{0, 3\} \Rightarrow 3$

$\Rightarrow T(1,4) \Rightarrow \max\{T(1-1,4), 3 + T(1-1, 4-2)\}$

$\Rightarrow \max\{T(0,4), 3 + T(0,2)\}$

$\Rightarrow \max\{0, 3\} \Rightarrow 3$

$T(1,5) \Rightarrow \max\{T(1-1,5), 3 + T(1-1, 5-2)\}$

$\Rightarrow \max\{T(0,5), 3 + T(0,3)\}$

$\Rightarrow \max\{3\} \Rightarrow 3$

$T(2,1) \Rightarrow \max\{T(2-1,1), 4 + T(2-1, 1-3)\}$

$\Rightarrow \max\{T(1,1), 4 + T(1,-2)\}$

$\Rightarrow \max\{0\} = 0$

$T(2,2) \rightarrow i=2; j=2; V_i = 4; W_i = 3;$

$\Rightarrow \max\{T(2-1,2), 4 + T(2-1, 2-3)\}$

$= \max\{T(1,2), 4 + T(1,-1)\}$

$\Rightarrow 3$

$T(2,3)$; $i=2$; $j=3$; $V_i=4$; $W_i=3$

$\Rightarrow \max\{T(2-1,3), 4+T(2-1,3-3)\}$

$\Rightarrow \max\{T(1,3), 4+T(1,0)\}$

$\Rightarrow \max\{3, 4\} \Rightarrow 4$

$T(2,4) \Rightarrow \max\{T(2-1,4), 4+T(2-1,4-3)\}$

$\Rightarrow \max\{T(1,4), 4+T(1,1)\}$

$\Rightarrow \max\{3, 4\} = 4$

$T(2,5) \Rightarrow \max\{T(2-1,5), 4+T(2-1,5-3)\}$

$\Rightarrow \max\{T(1,5), 4+T(1,2)\}$

$\Rightarrow \max\{3, 7\} \Rightarrow 7$

$T(3,1) \Rightarrow i=3; j=1; V_i=5; W_i=4$

$\Rightarrow \max\{T(3-1,1), 5+T(3-1,1-4)\}$

$\Rightarrow \max\{T(2,1), 5+T(2,-3)\}$

$\Rightarrow 0 \Rightarrow 0$

$T(3,2) \Rightarrow \max\{T(3-1,2), 5+T(3-1,2-4)\}$

$\Rightarrow \max\{T(2,2), 5+T(2,-2)\} \Rightarrow 3$

$T(3,3) \Rightarrow \max\{T(3-1,3), 5+T(3-1,3-4)\}$

$\Rightarrow \max\{T(2,3), 5+T(2,-1)\} \Rightarrow 4$

$T(3,4) \Rightarrow \max\{T(3-1,4), 5+T(3-1,4-4)\}$

$\Rightarrow \max\{T(2,4), 5+T(2,0)\}$

$\Rightarrow \max\{4, 5\} \Rightarrow 5$

$T(3,5) \Rightarrow \max\{T(3-1,5), 5+T(3-1,5-4)\}$

$\Rightarrow \max\{T(2,5), 5+T(2,1)\}$

$\Rightarrow \max\{7, 5\} \Rightarrow 7$

$T(4,1) \Rightarrow i=4; j=1; V_i=6 \; ; \; W_i=5$

$\therefore$ ~~MAX~~ $= T(4,1) \rightarrow \max\{T(4-1,1), 6+T(4-1,1-5)\}$

$\Rightarrow \max\{T(3,1), 6+T(3,-4)\} \Rightarrow \max\{0\} = 0$

$T(4,2) \Rightarrow \max\{T(4-1,2), 6+T(4-1,2-5)\} \Rightarrow \max\{T(3,2), 6+T(3,-3)$

$\Rightarrow 3$

$T(4,3) \Rightarrow \max\{T(4-1,3), 6+T(4-1,3-5)\} \Rightarrow \max\{T(3,3), 6+T(3,-2)\}$

$\Rightarrow 4$

$T(4,4) \Rightarrow \max\{T(4-1,4), 6+T(4-1,4-5)\} \Rightarrow \max\{T(3,4), 6+T(3,-1)\} = 5$

$T(4,5) \Rightarrow \max\{T(4-1,5), 6+T(4-1,5-5)\} \Rightarrow \max\{T(3,5), 6+T(3,0)\} \Rightarrow$

$\Rightarrow \max\{7, 6\} = 7$

Eg (2)

~~Item Weight Value~~

Input $\Rightarrow N=3 \; ; \; W=4$

Values :- $\{1,2,3\}$; Weight $\Rightarrow \{4,5,1\}$

Item ~~Output~~	Value	Weight
1	1	4
2	2	5
3	3	1

$N+1 \Rightarrow 3+1=4 \rightarrow 4$ rows

$W+1 \Rightarrow 4+1=5 \rightarrow 5$ columns

5 columns ↓ ; rows →

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

$T(1,1) \Rightarrow i=1; j=1; W_i=4; V_i=1$

$\Rightarrow \max\{T(1-1,1), 1+T(1-1,1-4)\}$

$\Rightarrow \max\{T(0,1), 1+T(0,-3)\}$

$\Rightarrow \max\{0\} \Rightarrow 0$

$T(1,2) \Rightarrow \max\{T(1-1,2), 1+T(1-1,2-4)\} \Rightarrow \max\{T(0,2), 1+T(0,-2)\}$

$\Rightarrow \max\{0\} \Rightarrow 0$

$T(1,3) \Rightarrow \max\{T(1-1,3), 1+T(1-1,3-4)\} \Rightarrow \max\{T(0,3), 1+T(0,-1)\}$

$\Rightarrow \max\{0\} \Rightarrow 0$

$T(1,4) \Rightarrow \max\{T(1-1,4), 1+T(1-1,4-4)\} \Rightarrow \max\{T(0,4), 1+T(0,0)\}$

$\Rightarrow \min(0,1] \Rightarrow 1$

~~T(1,1)~~

$T(2,1) \Rightarrow i=2 ; j=1 ; V_i = 2 \quad W_i = 5$

$T(2,1) \Rightarrow \max\{T(2-1,1), 2+T(2-1, 1-5)\}$

$\Rightarrow \max\{T(1,1), 2+T(1,-4)\} \Rightarrow 0$

$T(2,2) \Rightarrow \max\{T(2-1,2), 2+T(2-1, 2-5)\}$

$\Rightarrow \max\{T(1,2), 2+T(1,-3)\} = 0$

$T(2,3) = \max\{T(2-1,3), 2+T(2-1, 3-5)\}$

$\Rightarrow \max\{T(1,3), 2+T(1,-2)\}$

$\Rightarrow 0$

$T(2,4) \Rightarrow \max\{T(2-1,4), 2+T(2-1, 4-5)\}$

$\Rightarrow \max\{T(1,4), 2+T(1,-1)\}$

$\Rightarrow 1$

$T(3,1) \Rightarrow i=3 ; j=1 ; V_i = 3 \quad W_i = 1$

$\Rightarrow \max\{T(3-1,1), 3+T(3-1, 1-1)\} \Rightarrow$

$= \max\{T(2,1), 3+T(2,0)\}$

$\Rightarrow \max\{0, 3\} = 3$

$T(3,2) \Rightarrow \max\{T(3-1,2), 3+T(3-1, 2-1)\}$

$\Rightarrow \max\{T(2,2), 3+T(2,1)\}$

$\Rightarrow \max\{0, 3\} \Rightarrow 3$

$T(3,3) \Rightarrow \max\{T(3-1,3), 3+T(3-1, 3-1)\}$

$\Rightarrow \max\{T(2,3), 3+T(2,2)\}$

$\Rightarrow \max\{0, 3\} \Rightarrow 3$

$T(3,4) \Rightarrow \max\{T(3-1,4), 3+T(3-1, 4-1)\}$

$\Rightarrow \max\{T(2,4), 3+T(2,3)\}$

$\Rightarrow \max\{1, 3\} \Rightarrow 3$

UNIT-4

BRANCH AND BOUND

13 Aug 23

Types of Branch and Bound Search Techniques :-

1. FIFO B & B Search / BFS technique.

Queue - BFS like state space tree

→ Frontend - used for deletion ; front++

→ Rearend - used for insertion

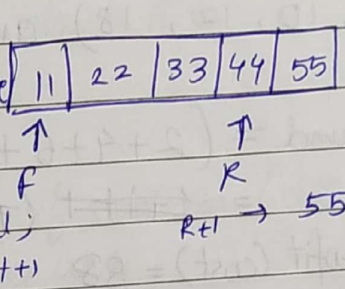
→ ① Rear + 1

② Increase a value

① Deletion from queue

② front = front + 1

NOTE :- Get the list of delete the list.



2. LIFO B & B Search :-

STACK - DFS.

→ Used for minimization?

3. LC - B & B Search :-

→ Called as least value customer.

Any one can except the ~~reduced~~

→ Working for minimization & avoiding generation of tree.

★ Branch & Bound :-

→ Backtracking based on DFS.

→ Branch & Bound method based on BFS.

→ In Branch & Bound we calculate lower and upper bound function, for each & every node in state of space tree.

→ Lower bound (x) → c(x) → we allow fraction parts means (eg - 1/3rd of object).

→ Upper bound (x) → u(x) → We don't allow fractions.

→ for Eg → In knapsack problem;

n = 4 , (w1, w2, w3, w4) = (2, 4, 6, 9) and

⇒ (v1, v2, v3, v4) = (10, 10, 12, 18) and m = 15 (knapsack capacity)

Fractional knapsack

Lower bound = (2+4+6+3/9)

= (1, 1, 1, 3/9) ← quantity

3/9 ⇒ 9/3 ⇒ 18/3 = 6

max, profit (cost) = 38

0/1 knapsack

Upper bound = (2 + 4+6+0)

= (1+1+1, 0) ← quantity (either take it or not)

max profit (Cost) = 32

* In FIFO, LIFO, LC which one is the E-Node

76	23	56	11	59
N_1	N_2	N_3	N_4	N_5

FIFO - N_1 ← Active nodes bec.

LIFO - N_5

LC - N_4

27/Aug

LC B & B search [Least cost Branch & Bound]
↳ Live nodes created based on bound function of least cost.

→ To calculate the cost of node x :-
$C(x) = f(x) + g(x)$

$c(x)$ ← Lower Bound
$u(x)$ ← Upper Bound
$g(x)$ ← No. of non blank tiles

★ Bounding function :-
→ Used to avoid the generation of subtrees (children) that does not contain the answer node.
→ We are going to kill the node means; $C(x)$ > upper
→

★ Least Cost (LC) search :-
→ The node x is assigned a rank using -
$c(x) = f(n(x)) + g(x)$

★ 0/1 Knapsack using Branch & Bound :-
→ Left Branch → Inclusion
→ Right Branch → Exclusion
→ Upper Bound →

$ub = v + (W - w)(V_{i+1}/W_{i+1})$

where,
v → value/profit associated with selected items from the first i items,
W → Capacity of the knapsack
w → weight of selected items from first i items

Not given in exam

Eg →

Item	Weight	Value	Value/Weight
1	4	40	10
2	7	42	6
3	5	25	5
4	3	12	4

Knapsack weigh = 10

→ Knapsack is maximization problem of optimization and we are using LC cost ie minimizing
→ Using Branch & Bound technique
→ Left Branch is known as Inclusion
→ Most Branch is known as Exclusion
→ We cannot apply LC Bound technique
→ The first step is to solve value/weight

∴

Weight	Value
ub	

c(x) ~~upper~~ lower bound (with fraction)
u(x) - upper bound (without fraction)

where $ub = v + (W - w)(v_i + 1 / w_i + 1)$

∴ Steps :-

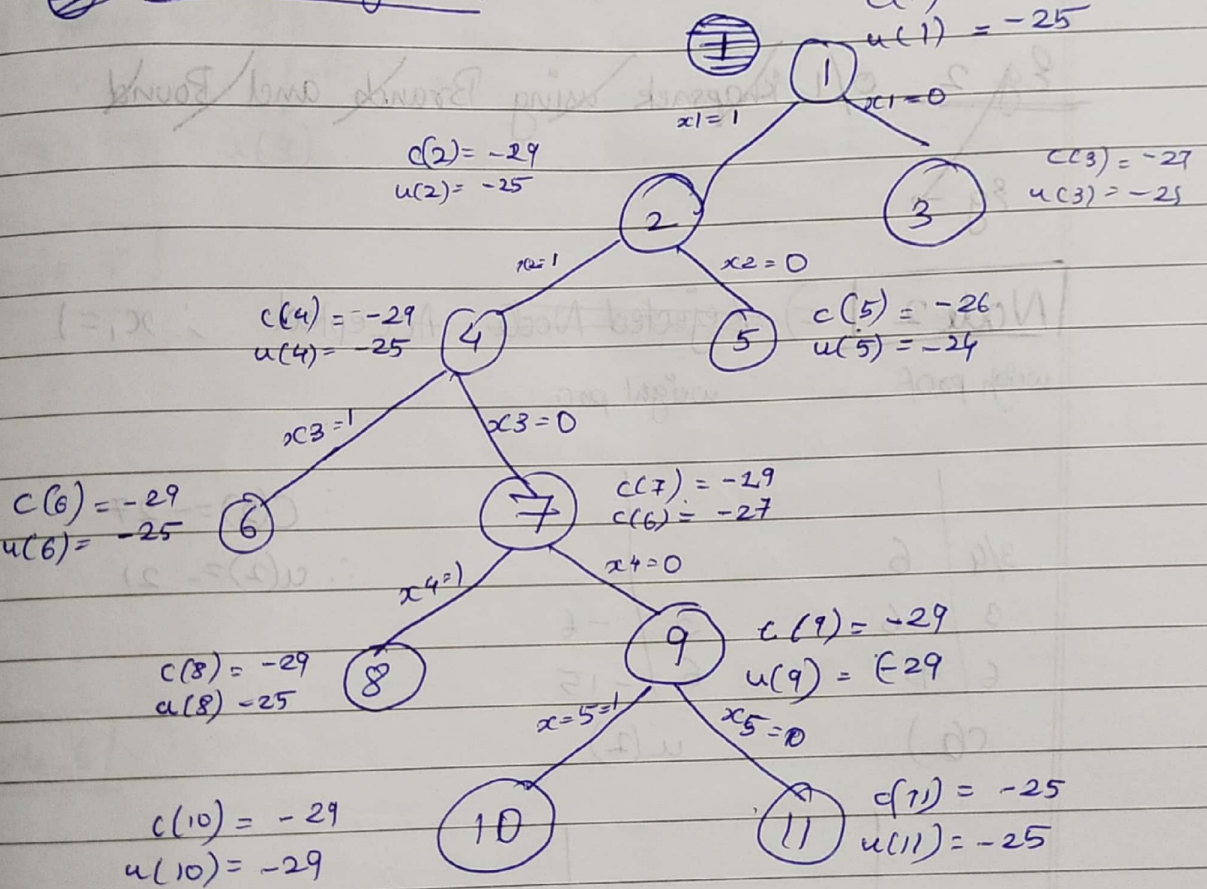
① Derive state-space ~~Table~~ Tree (SST)

* Example :- 0/1 Knapsack using Branch and Bound.

Eg → P = (10, 15, 6, 8, 4) ∴ P = profit
W = (4, 6, 3, 4, 3)
M = (12)

① negative the P : to make the problem of minimization, make all profits negative
P = (-10, -15, -6, -8, -4)

② Ulta diagram



∴ $(x_1, x_2, x_3, x_4, x_5)$ = 1 1 0 0 1
∴ Maximum Profit = 10 + 15 + 4 = 29

$\frac{3}{4} \times 8 = 2$ 3

Maximum capacity (M) = 12

(1) Node (1)

weight =	4	6	2/3 =
profit =	10	15	4 -29

c(x) => c(1) => u(x)

Always put - on c(x) & u(x)

weight	Profit
4	10
6	15
∴ 10	25

P = (10, 15, 6, 8, 4)
W = (4, 6, 3, 4, 3)
M = (12)

∴ u(x) = 25 & c(x) = -29

Eg 2 0/1 ~~Knapsack using~~ Branch and ~~Bound~~

Eg →

Node 2 => ~~Rejected Node~~ Accepted ; $x_1 = 1$

weigh prof weight pro

3/4	6
3	-6
6	-15

c(2)

3	-6
6	-15

u(2)

∴ C(2) = -27

∴ u(2) = -21

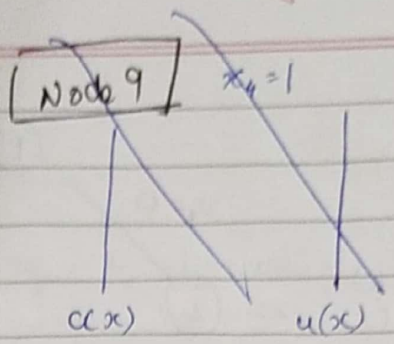
c(x) u(x)

weight / profits

Eg 2 =)

	W	P
x_1	2	10
x_2	4	10
x_3	6	12
x_4	9	18

max = 15



Node 1

3/9	-6		
6	-12	6	-12
4	-10	4	-10
2	-10	2	-10
	c(x)		u(x)

c(x) = -38

u(x) = -32

$x_1 = 0$ **Node 2**

5/9	-10		
6	-12	6	-12
4	-10	4	-10
	c(x)		u(x)

c(x) = -32

u(x) = -22

Node 3 $x_1 = 1$

3/9	-6		
6	-12	6	-12
4	-10	4	-10
2	-10	2	-10
	c(x)		u(x)

c(x) = -38

u(x) = -32

Node 4 $x_1 = 1$, $x_2 = 0$

7/9	-14		
6	-12	6	-12
2	-10	2	-10
	c(x)		u(x)

c(x) = -36

u(x) = -22

Node 5 $x_1 = 1$, $x_2 = 1$

3/9	-6		
6	-12	6	-12
4	-10	4	-10
2	-10	2	-10
	c(x)		u(x)

c(x) = -38

u(x) = -32

Node 6 $x_1 = 1$, $x_2 = 1$, $x_3 = 0$

9	-18	9	-18
4	-10	4	-10
2	-10	2	-10
	c(x)		u(x)

c(x) = -38

u(x) = -38

Node 7 $x_3 = 1$, $x_1 = 1$, $x_2 = 1$

3/9	-6		
6	-12	6	-12
4	-10	4	-10
2	-10	2	-10
	c(x)		u(x)

c(x) = -38

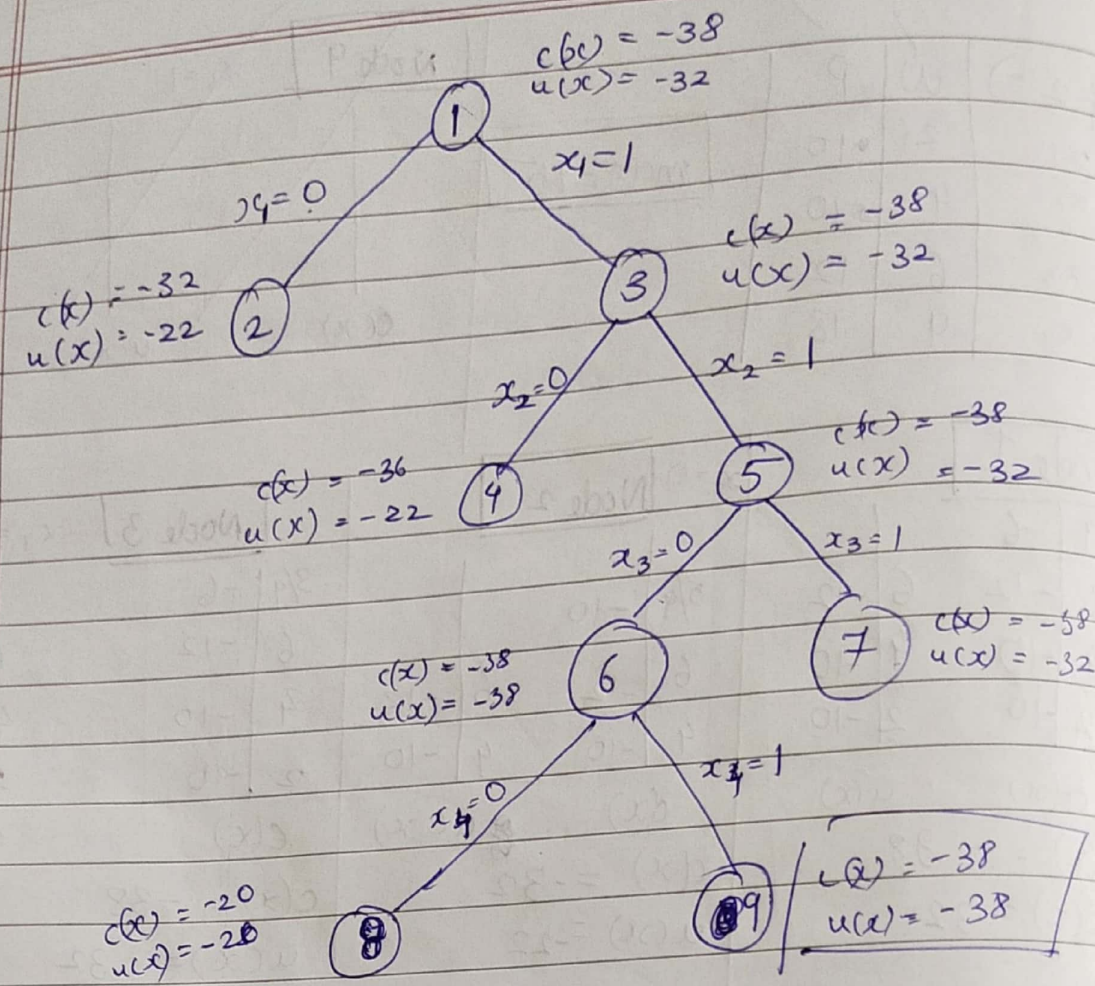
u(x) = -32

Node 8 $x_4 = 0$

6	-12	6	-12
4	-10	4	-10
2	-10	2	-10
	c(x)		u(x)

$\therefore$ c(x) = -32

$\therefore$ u(x) = -32



Node 8

$x_1 = 1$; $x_2 = 1$

$x_3 = 0$; $x_4 = 0$

4	-10	4	-10
2	-10	2	-10
c(x)		u(x)	

$c(x) = -20$

$u(x) = -20$

Node 9

$x_1 = 1$; $x_2 = 1$

$x_3 = 0$; $x_4 = 1$

9	-18	9	-18
4	-10	4	-10
2	-10	2	-10
c(x)		u(x)	

$c(x) = -38$

$u(x) = -38$

$\therefore (x_1, x_2, x_3, x_4) = 1\ 1\ 0\ 1$

Maximum Profit $= 10 + 10 + 18 = -38$

Convert it into maximization $= -38$

3 Sept 23

classmate
Date
Page

UNIT- Complexities Analysis

→ Complexity analysis is a technique to characterize the time taken by an algorithm with respect to input size.

10 Sept 23

★ P-(Polynomial) Class

- Definition of P class Problem
- Definition of Polynomial Time
- Definition of Non-Polynomial Time
- Features [next slide]
- This class contains many natural problems.

★ NP (Non-deterministic Polynomial Time) Class

- The class NP consists of those problems that are

- Definition of NP class -

★ NP-Hard :-

Definition of NP-Hard → A problem is NP-Hard if all problems NP are polynomial time reducible to it.

Eg → Hamiltons Cycle.

* NP - Complete :-

→ It is called NP - Complete if it is in
  - NP
  - NP hard

→ Definition of NP - Completeness

* TSP in NP - Complete

* Reduction

* NP - Hard & NP - Complete.

→ If R is polynomial time reducible to Q ; denoted by $R \leq_p Q$

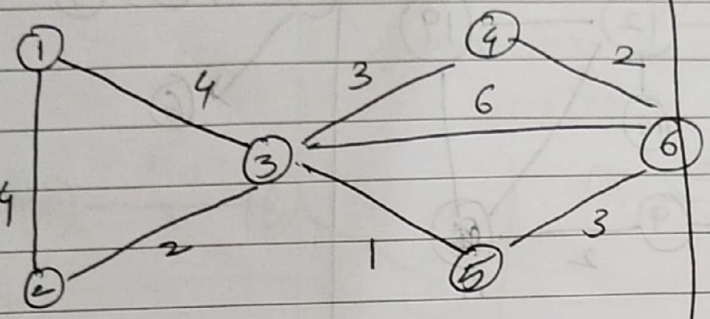
→ Defi of NP Hard & NP - Complete
  If all probs R

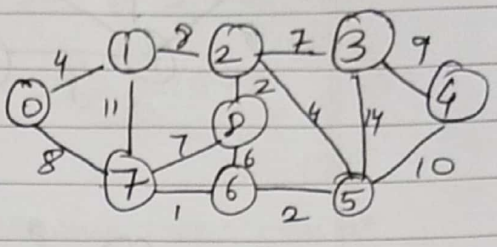
24/Sept/23

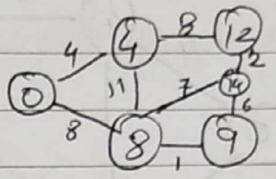
Unit 6 - Concurrent Algorithms

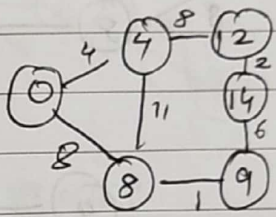
* Mutual Exclusion Problem

* Dijkstras' Algorithm :-

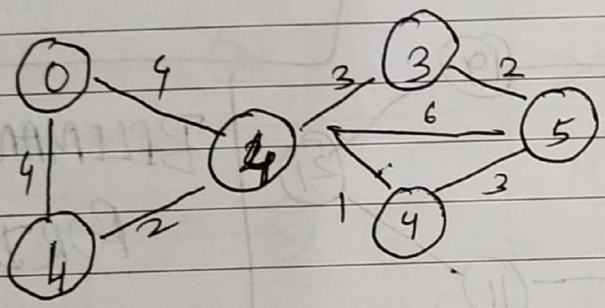


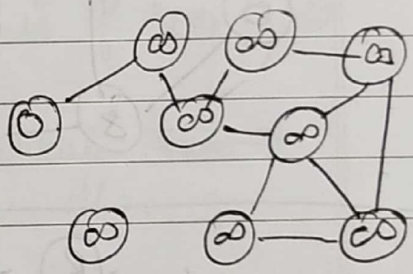


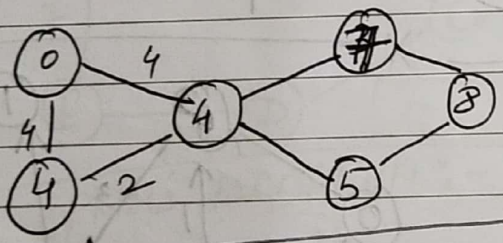




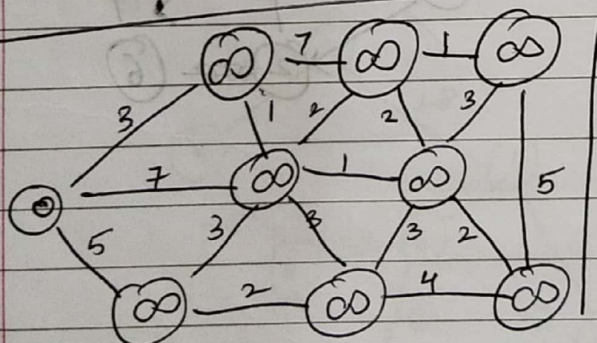
X

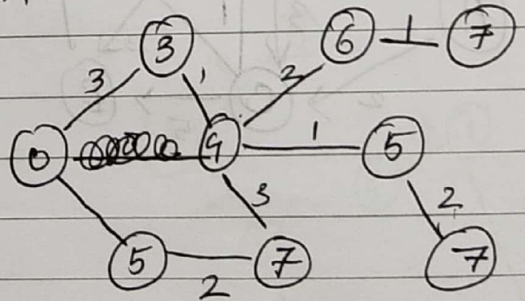


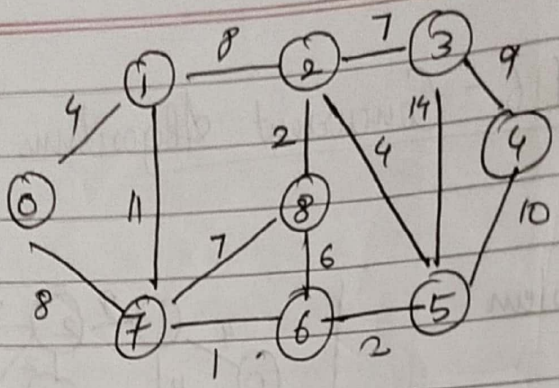


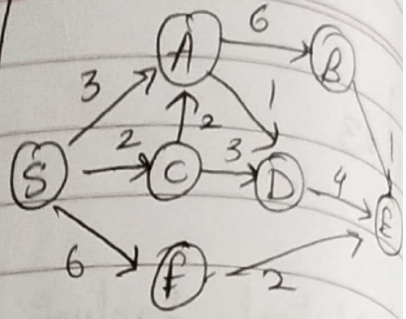


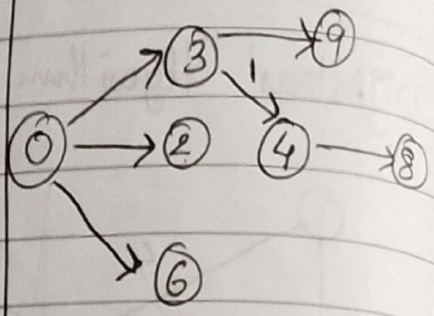
X

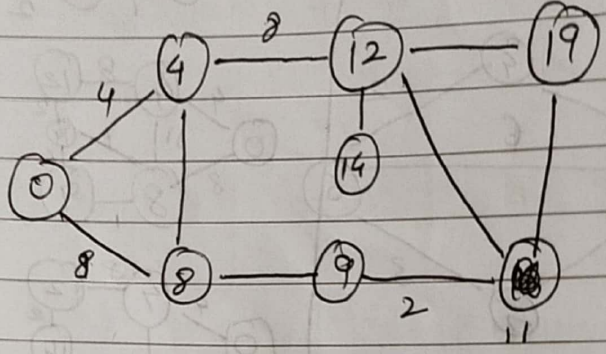


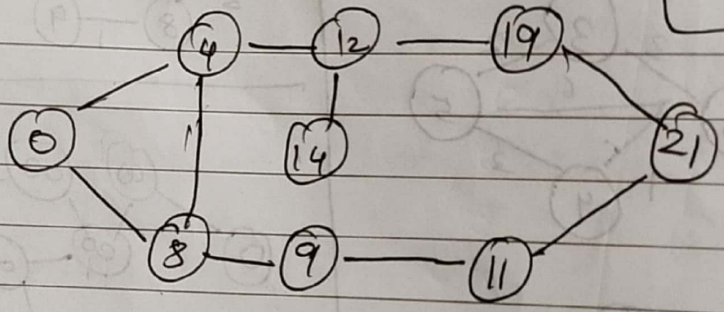






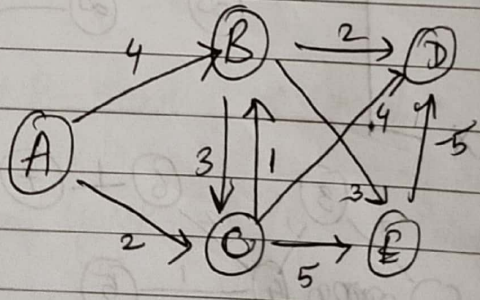


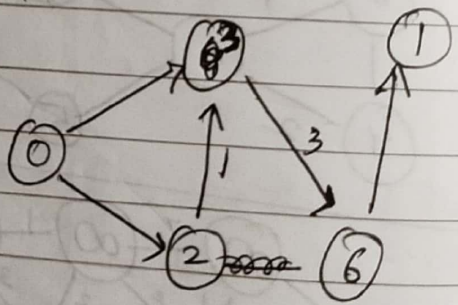




BELLMAN'S FORD ↓

X

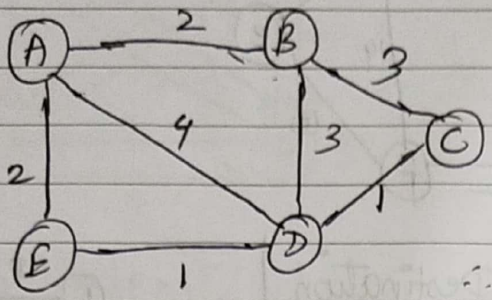




$T_{ij} = \max\{T(i-1,j), value_i + T(i-1, j - weight_i)\}$

— edges
○ vertices

Spanning Tree :-

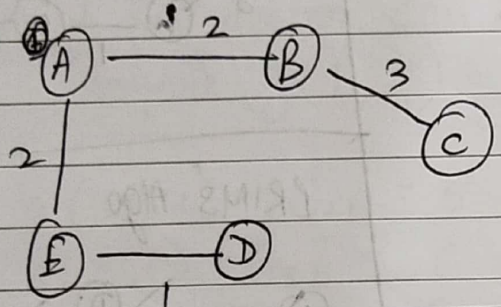


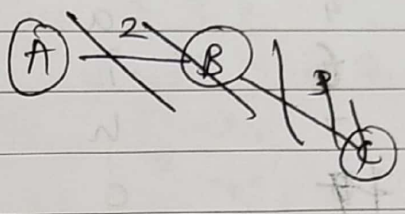
∴ no of edges will be $(n-1) = 5-1$
$= 4$ edges

$n = 5$; here n is vertices

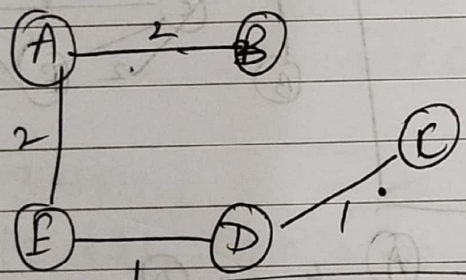
$\therefore n^{(n-2)} = 5^{(5-2)} = 5^{(3)}$
$= 125$

∴ 125 spanning trees can be formed from a complete graph of 5 vertices





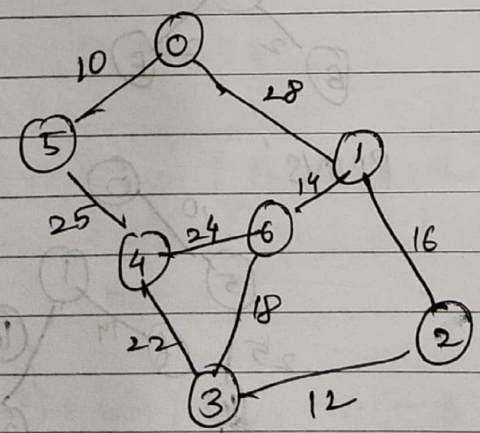
DO step wise

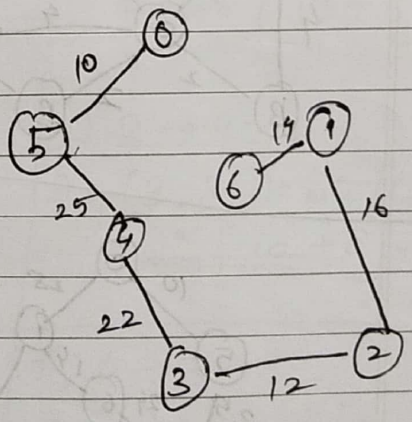


MST is of two types

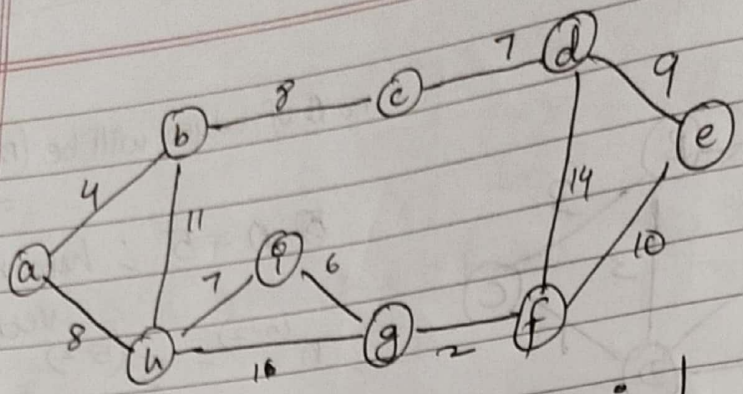
- Kruskal's
- Prim's Algo

PRIM'S ALGORITHM :-

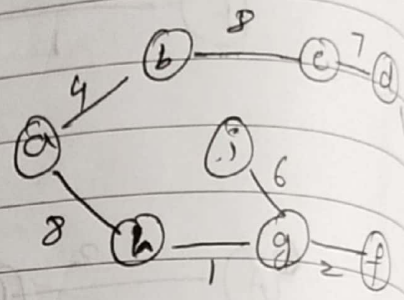




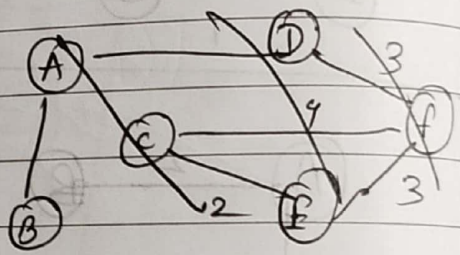
$T_{ij} = \max\{T(i-1,j)$

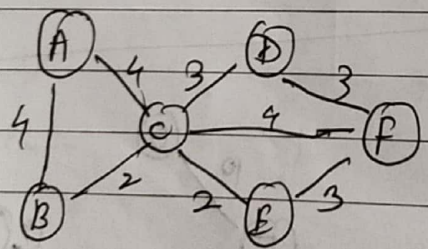


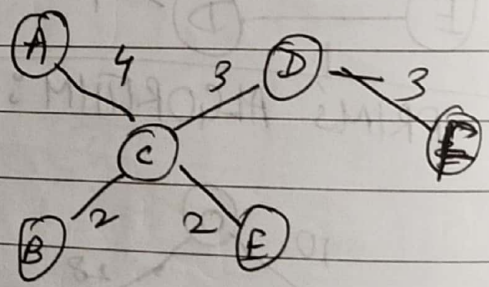
Weight	Source	Destination
1	h	g
2	g	f
4	a	b
6	i	g
7	h	i
~~7~~	c	d
8	a	h
8	b	c
9	d	e
10	f	e
11	b	h
14	d	f



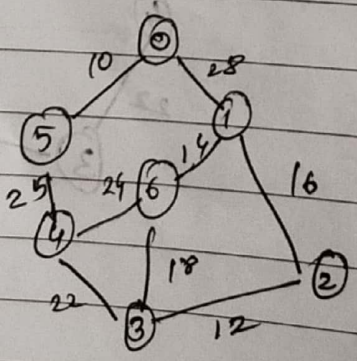
PRIMS' Algo

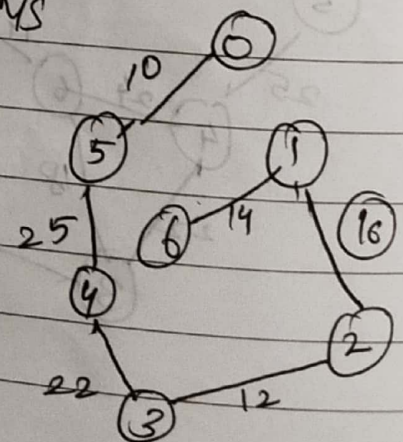




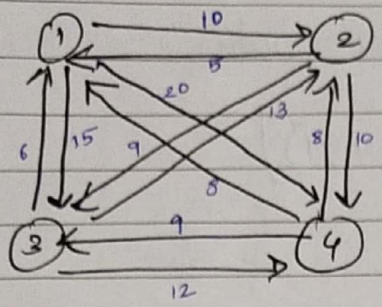


PRIMS'





$T_{ij} = \max\{T(i-1, j), value_i + T(i-1, j - weight_i)\}$

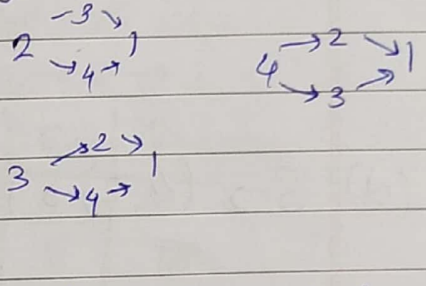


	1	2	3	4
1	0	10	15	20
2	5	0	9	10
3	6	13	0	12
4	8	8	9	0

① Starting Vertex ⇒ 1
Ending Vertex ⇒ 1

② Second last vertex can be either 2, 3, 4;
Compute $g(i, \phi) = C_{ij}$ $\phi = j = 1$; $i = 2, 3, 4$

ie $g(2, 1) = C_{21} = 5$
ie $g(3, 1) = C_{31} = 6$
ie $g(4, 1) = C_{41} = 8$



③ Three ~~vertex~~ last vertex can have the following path :-

(a) $2 \to 3 \to 1$
$\therefore g(2, \{3\}) = \min(C_{23} + g(3, 1))$
$= \min(9 + 6)$
$= 15$

(b) $2 \to 4 \to 1$
$g(2, \{4\}) = \min[C_{24} + g(4, 1)]$
$= \min[10 + 8]$
$= 18$

(c) $3 \to 2 \to 1$
$g(3, \{2\}) = \min[C_{32} + g(2, \phi)]$
$= \min[13 + 5] = 18$

(d) $3 \to 4 \to 1$
$\Rightarrow g(3, \{4\}) = \min[C_{34} + g(4, \phi)]$
$= \min[12 + 8] = 20$

(e) $4 \to 2 \to 1$
$g(4, \{2\}) = \min[C_{42} + g(2, \phi)]$
$= \min[8 + 5]$
$= 13$

(f) $4 \to 3 \to 1$
$g(4, \{3\}) = \min[C_{43} + g(3, \phi)]$
$= \min[9 + 6] = 15$

(4) Now the last 4 vertex can have paths :-

(a) $2 \to (3,4) \to 1 \Rightarrow \min [C_{23} + g(3,4,1)]$

$= \min [9 + 20]$

$= 29$

(b) $2 \to (4,3) \to 1 \Rightarrow \min [C_{24} + g(4,3,1)]$

$\Rightarrow \min [10 + 15]$

$\Rightarrow 25$

(c) $3 \to (2,4) \to 1 \Rightarrow \min [C_{32} + g(2,4,1)]$

$\Rightarrow \min [13 + 18]$

$= 31$

(d) $3 \to (4,2) \to 1 \Rightarrow \min [C_{34} + g(4,2,1)]$

$\Rightarrow \min [12 + 13]$

$\Rightarrow 25$

(e) $4 \to (2,3) \to 1 \Rightarrow \min [C_{42} + g(2,3,1)]$

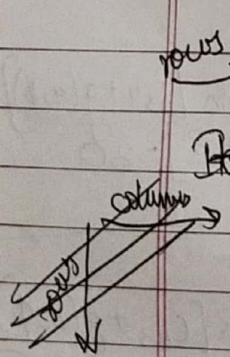
$\Rightarrow \min [8 + 15]$

$\Rightarrow 23$

(f) $4 \to (3,2) \to 1 \Rightarrow \min [C_{43} + g(3,2,1)]$

$\Rightarrow \min [9 + 18]$

$\Rightarrow 27$



Input $= \{1, 2, 3\}$

weight $= \{4, 5, 1\}$

$N = 3$

$N = 3 + 1 = 4$ rows

$W = 4 + 1 = 5$ columns

∴ Thus the final path can be found as

$1 \rightarrow \{2,3,4\} \rightarrow 1$

$g(1,\{2,3,4\}) =$

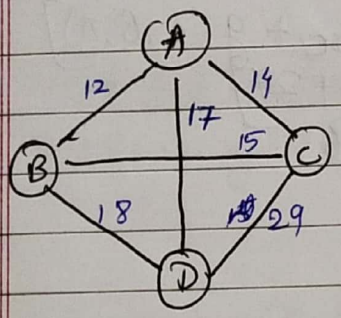
$= \min \begin{cases} C_{12} + g(2,3,4,1) \\ C_{13} + g(3,4,2,1) \\ C_{14} + g(4,3,2,1) \end{cases}$

$= \min \begin{cases} 10 + 25, \\ 15 + 25, \\ 20 + 23) \end{cases} \Rightarrow \min(35, 45, 43)$

$\therefore 35 \therefore$

The shortest path is

$1 \rightarrow 2 \rightarrow 4 \rightarrow 3 \rightarrow 1$



	A	B	C	D
A	0	12	14	17
B	12	0	15	18
C	14	15	0	29
D	17	18	29	0

① Starting vertex $\Rightarrow$ A
Ending vertex $\Rightarrow$ A

② Second last vertex can either be ~~A~~ B, C, D
$\therefore g(i,\phi) = C_{ij}$; $\phi = j = A$; $i = B, C, D$
$\therefore g(B,\phi) = C_{BA} = 12$
$\therefore g(C,\phi) = C_{CA} = 14$
$\therefore g(D,\phi) = C_{DA} = 17$

∴ Thus the third last vertex can have the following path :-

(a) ~~$2 \rightarrow 3 \rightarrow 1 = g \min [C_{23} + g$~~ ∴ ~~$2 \rightarrow 3 \rightarrow 1$~~

(a) $B \rightarrow C \rightarrow A \Rightarrow g(B,\{C\}) = \min [C_{BC} + g(C,\phi)] = \min [15 + 14] = 29$
(b) $B \rightarrow D \rightarrow A \Rightarrow g(B,\{D\}) = \min [C_{BD} + g(D,\phi)] = \min [18 + 17] = 35$
(c) $C \rightarrow B \rightarrow A \Rightarrow g(C,\{B\}) = \min [C_{CB} + g(B,\phi)] = \min [15 + 12] = 27$
(d) $D \rightarrow B \rightarrow A \Rightarrow g(D,\{B\}) = \min [C_{DB} + g(B,\phi) = \min [18 + 12] = 30$

(e) $D \to C \to A = g(D, \{C\}) = \min [C_{DC} + g(C, \phi)] = \min [29 + 14] = 43$

(f) $C \to D \to A = g(C, \{D\}) = \min [C_{CD} + g(D, \phi)] = \min [29 + 17] = 46$

(g) Thus the last 4 vertex can have the following path :-

(a) $B \to (C, D) \to A$
$\Rightarrow \min [C_{BC} + g(C, D, A)]$
$\Rightarrow \min [15 + 46] = 61$

(b) $B \to (D, C) \to A$
$\Rightarrow \min [C_{BD} + g(D, C, A)]$
$\Rightarrow \min [18 + 43] = 61$

(c) $C \to (B, D) \to A$
$= \min [C_{CB} + g(B, D, A)] =$
$\Rightarrow \min [15 + 35] = 50$

(d) $C \to (D, B) \to A = \min [C_{CD} + g(D, B, A)]$
$\Rightarrow \min [29 + 30] = 59$

(e) $D \to (B, C) \to A$
$\Rightarrow \min [C_{DB} + g(B, C, A)]$
$\Rightarrow \min (18 + 29)$
$= 47$

(f) $D \to (C, B) \to A$
$\Rightarrow \min [C_{DC} + g(C, B, A)]$
$\Rightarrow \min [29 + 27]$
$= 56$

∴ Thus the final path can be :-

$A \to \{B, C, D\} \to A$

$g(A, \{B, C, D\}) \Rightarrow \min \{ C_{AB} + g(B, C, D, A)$
$C_{AC} + g(C, B, D, A)$
$C_{AD} + g(D, B, C, A) \}$

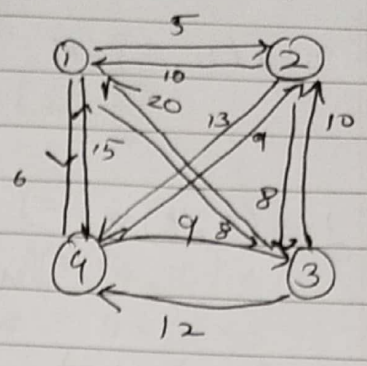
$\Rightarrow \min \{ 12 + 61$
$14 + 50$
$17 + 47 \}$

$\Rightarrow \min \{73, 64, 64\}$

∴ There are two paths :-

$A \to C \to B \to D \to A$ &
$A \to D \to B \to C \to A$

	1	2	3	4
1	0	5	8	6
2	10	0	8	13
3	20	10	0	12
4	15	9	9	0



① Starting Vertex = 1
Ending Vertex = 2

② Second last vertex can either be 2, 3 or 4
∴ Compute $g(i, \phi) = C_{ij}$ ⇒ $i = 2, 3, 4$; $\phi = j = 1$
$g(2,1) = C_{21} = 10$
$g(3,1) = C_{31} = 20$
$g(4,1) = C_{41} = 15$

③ Third last vertex can have the following path :-
(a) $2 \to 3 \to 1$; $g(2,\{3\}) = \min[C_{23} + g(3,\phi)]$
$= \min[8 + 20]$
$= 28$

(b) $2 \to 4 \to 1$; $g(2,\{4\}) = \min[C_{24} + g(4,\phi)]$
$= \min[13 + 15] = 28$

(c) $2 \to 2 \to 1$; $g(3,\{2\}) = \min[C_{32} + g(2,\phi)]$
$= \min[10 + 10] = 20$

(d) $3 \to 4 \to 1$; $g(3,\{4\}) = \min[C_{34} + g(4,\phi)] = \min[12 + 15] = 27$

(e) $4 \to 2 \to 1$; $g(4,\{2\}) = \min[C_{42} + g(2,\phi)] = \min[9 + 10] = 19$

(f) $4 \to 3 \to 1$; $g(4,\{3\}) = \min[C_{43} + g(3,\phi)] = \min[9 + 20] = 29$

④ Third last vertex ch + fp :-

(a) $2 \to (3,4) \to 1$; $= \min[C_{23} + g(3,4;1)]$
$= \min[8 + 27] = 35$

(b) $2 \to (4,3) \to 1$: $\min [C_{24} + g(4,3,1)]$
$\Rightarrow \min [13 + 29]$
$\therefore 42$

(c) $3 \to (2,4) \to 1 \Rightarrow \min [C_{32} + g(2,4,1)]$
$\Rightarrow \min [10 + 28]$
$\Rightarrow 38$

(d) $3 \to (4,2) \to 1 \Rightarrow \min [C_{34} + g(4,2,1)]$
$\Rightarrow \min [12 + 19] = 31$

(e) $4 \to (3,2) \to 1 \Rightarrow \min [C_{43} + g(3,2,1)]$
$\Rightarrow \min [9 + 20] = 29$

(f) $4 \to (2,3) \to 1 \Rightarrow \min [C_{42} + g(2,3,1)]$
$\Rightarrow \min [9 + 28]$
$\Rightarrow 37$

(g) : The final path can be found as :-

$1 \to \{2,3,4\} \to 1$

$g(1\{2,3,4\}) = \min (C_{12} + g(2,\{3,4,1\}),$
$C_{13} + g(3\{2,4,1\}),$
$C_{14} + g(4\{2,3,1\}) \}$

$= \min (5 + 35$
$8 + 31 \Rightarrow \min [40, 39, 35]$
$6 + 29)$

$\therefore$ The final path can be $\Rightarrow 1 \to 4 \to 3 \to 2 \to 1$

$4 \times 3 \times 12$
36×4
$\Rightarrow 144$

21×12
42
21
252

$8 \times 2 = 16$

* Matrix Chain Multiplication :-

$\therefore N = 6$; arr $[4, 10, 3, 12, 20, 7]$

$A_1 = [4, 10]$ $A_3 = [3, 12]$ $A_5 = [20, 7]$

$A_2 = [10, 3]$ $A_4 = [12, 20]$

	1	2	3	4	5	
1	0	120	264	1080	1344	1
2		0	360	1320	1350	2
3			0	720	1140	3
4				0	1680	4
5					0	5

$A_1 * A_2 = [4, 10] * [10, 3]$
$= 4 \times 10 \times 3$
$= 120$

$A_2 * A_3 \Rightarrow [10, 3] * [3 * 12]$
$\Rightarrow 10 * 3 * 12$
$\Rightarrow 360$

$A_3 * A_4 \Rightarrow [3, 12] * [12, 20]$
$\Rightarrow [3 * 12 * 20]$
$\Rightarrow 720$

$A_4 * A_5 = [12, 20] * [20, 7]$
$\Rightarrow 12 * 20 * 7$
$\Rightarrow 1680$

$A_1 * A_3 \Rightarrow (A_1 * A_2) * A_3$
$120 * [4 * 3 * 12] * 1$
$120 * [144] \Rightarrow 17280$
264 ; Parenthesis $\Rightarrow (A_1 * A_2) A_3$

OR

$A_1 * (A_2 * A_3)$
$0 + 360 + [4 * 10 * 12]$
$\Rightarrow 360 + 480$
$\Rightarrow 840$

$A_2 * A_4 \Rightarrow (A_2 * A_3) A_4$
$\therefore 360 + 0 + (10 * 12 * 20]$
$\therefore 360 + (2400)$
$\therefore \Rightarrow 2760$

$A_2 * A_4 \Rightarrow A_2 (A_3 * A_4)$
$\Rightarrow 0 + 720 + (10 * 3 * 20)$
$\Rightarrow 720 + 600$
$\Rightarrow 1320$

$A_3 * A_5$
$(A_3 * A_4) A_5$
$720 + 0 + (3 * 20 * 7)$
$720 + 420$
$\therefore 1140$

OR

$A_3 * A_5$
$A_3 (A_4 * A_5)$
$0 + 1680 + (3 * 12 * 7)$
$\Rightarrow 1680 + 252$
$\Rightarrow 1932$

$A_1 * (A_2 * A_3)$
$A_1 * (10 * 3 * 12)$
$[4 * 10] [10 * 12]$
$[4 * 10 * 12]$

12
× 8
96

$A_1 * A_4 \Rightarrow A_1 (A_2 * A_3 * A_4)$
$0 + 1320 + (4 * 10 * 20)$
$\Rightarrow 1320 + 800$
$\Rightarrow 2120$

$(A_1 * A_2 * A_3) A_4$
$\Rightarrow 264 + 0 + (4 * 12 * 20)$
$\Rightarrow 264 + 960$
$\Rightarrow 1224$

$(A_1 * A_2) * (A_3 * A_4)$
$(120 + 720) + (4 * 3 * 20)$
$840 + 240$
1080

$T_{ij} = \max\{T(i-1$

$\therefore A_1 * A_5$

$(m_1 * A_2 * m_3 * m_4) m_5$
$m_1 (m_2 * m_3 * m_4 * m_5$
$(m_1 * m_2 * m_3)(m_4 * m_5)$
$(m_1 * m_2)(m_3 * m_4 * m_5)$

$m_1 (m_2 * m_3 * m_4 * m_5)$
$0 + 1350 + (4 * 10 * 7)$
$\Rightarrow 1350 + 280$
$\Rightarrow 1630$

$\therefore (m_1 * m_2 * m_3 * m_4) m_5$
$1080 + 0 + (4 * 20 * 7)$
$\Rightarrow 1080 + 560$
$\Rightarrow 1640$

* 0/1 Knapsack Problem :-

$T(i,j) = \max[T(i-1,j), \text{value}_i + T(i-1, j - \text{weight}_i)]$

$(A_2 * A_5) \Rightarrow A_2(A_3 * A_4 * A_5)$

$0 + 1140 + (10 * 3 * 7)$

$\Rightarrow 1140 + 210 \Rightarrow 1350$

$(A_2 * A_3) * (A_4 * A_5)$

$360 + 1680 + (10 * 12 * 7)$

$360 + 1680 + 840$

2880

$(A_2 * A_3 * A_4) A_5$

$1320 + (10 * 20 * 7)$

$\Rightarrow (1320 + 1400)$

$\Rightarrow 2720$

Item $\Rightarrow \{1, 2, 3, 4\}$ $N = 4 + 1 = 5$ Rows

weight $\Rightarrow \{2, 3, 4, 5\}$ $W = 5 + 1 = 6$ Columns.

	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0					
2	0					
3	0					
4	0					

Item	weight
1	2
2	3
3	4
4	5

$T_{ij} \Rightarrow$ where $i = 1, j = 1$

$T_{ij} = \max\{T(i-1, j), value_i + T(i-1, j - weight_i)\}$

$T_{11} = \max\{T(1-1, 1),$

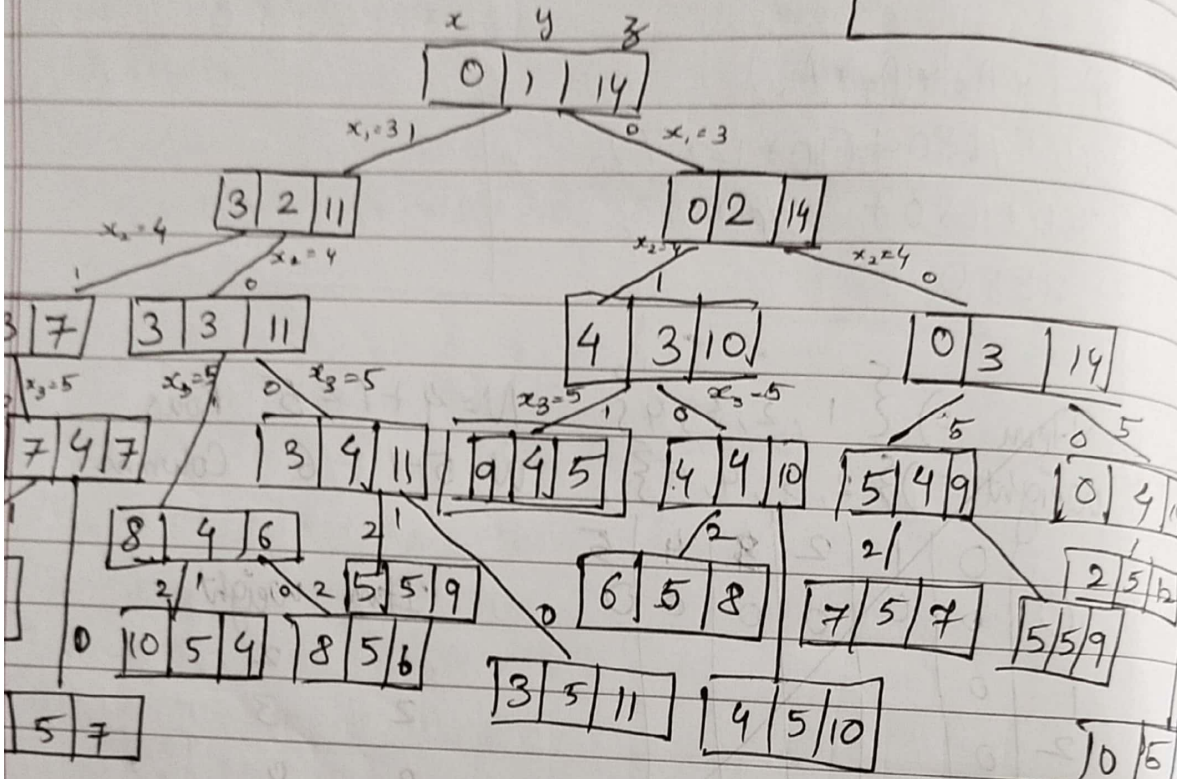
Sum of Subsets :- arr = [3, 4, 5, 2]

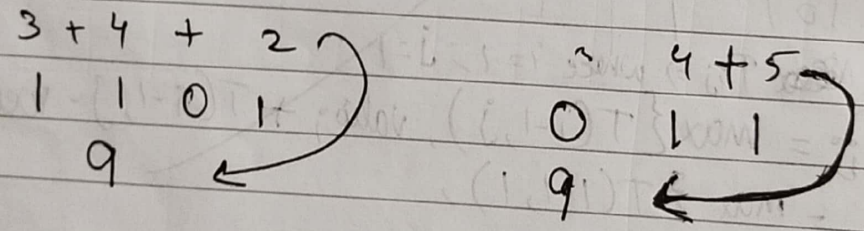
sum = 9

Here m = 9

Sum $\sum_{i=1}$ = 3+4+5+2 = 14 = solution

Conditions to
x > sum (or m
x = sum (solutio
y > n

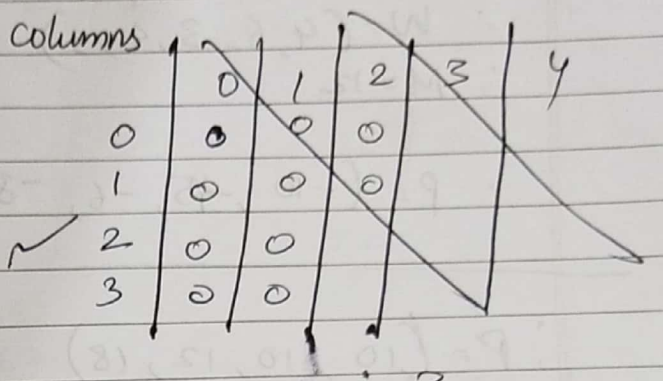




0/1 Knapsack Problem

Input $N = 3 \Rightarrow$ 4 rows
$W = 4 \Rightarrow$ 5 columns

Item	values	Weight
1	1	4
2	2	5
3	3	1



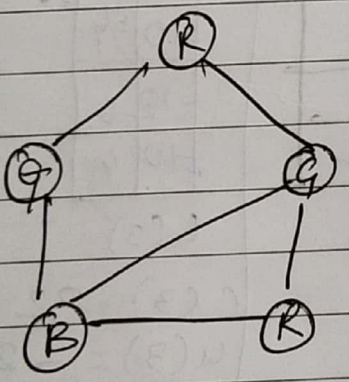
$i = 1 ; j = 1 ;$

$T_{ij} = \max \{ T(i-1, j), value_i + T(i-1, j - weight_i) \}$

$= \max \{ T(1-1, 1), 1 + T(1-1, 1-4) \}$

$= \max \{ T(0, 1), 1 + T(0, -3) \}$

$= \max \{ 0 \} = 0$



★ Hamiltonian Cycle

* 0/1 Knapsack ; Branch & Bound

∴ P = (10, 15, 6, 8, 4)
∴ W = (4, 6, 3, 4, 3)
∴ M = 12

[To make the problem of minimization make all profits negative]

∴ P = (-10, -15, -6, -8, -4) ∝

P = (10, 10, 12, 18)
W = (2, 4, 6, 9)
M = 15
x_1 x_2 x_3 x_4

(i) Node 1

-6	3/9
-12	6
-10	4
-10	2

LB 1
c(1)

-12	6
-10	4
-10	2

UB 1
u(1)

∴ c(1) = -38
u(1) = -32

(ii) Node 2 $x_1 = 1$

c(2)

	6
	4
-10	2

u(2)

∴ c(2) = -38
∴ u(2) = -32

(iii) Node 3 $x_1 = 0$

-10	5/9
-10	6
-10	4

L(3)

-12	6
-10	4

u(3)

c(3) = -32
u(3) = -22

(iv) Node 5 ; $x_1 = 1$, $x_2 = 0$

-14	7/9
-12	6
-10	2

LB (5)

~~-14~~	
-12	6
-10	2

U(5)

c(5) = -36
u(5) = -22

(v) Node 7 $x_1 = 1$, $x_2 = 1$, $x_3 = 0$

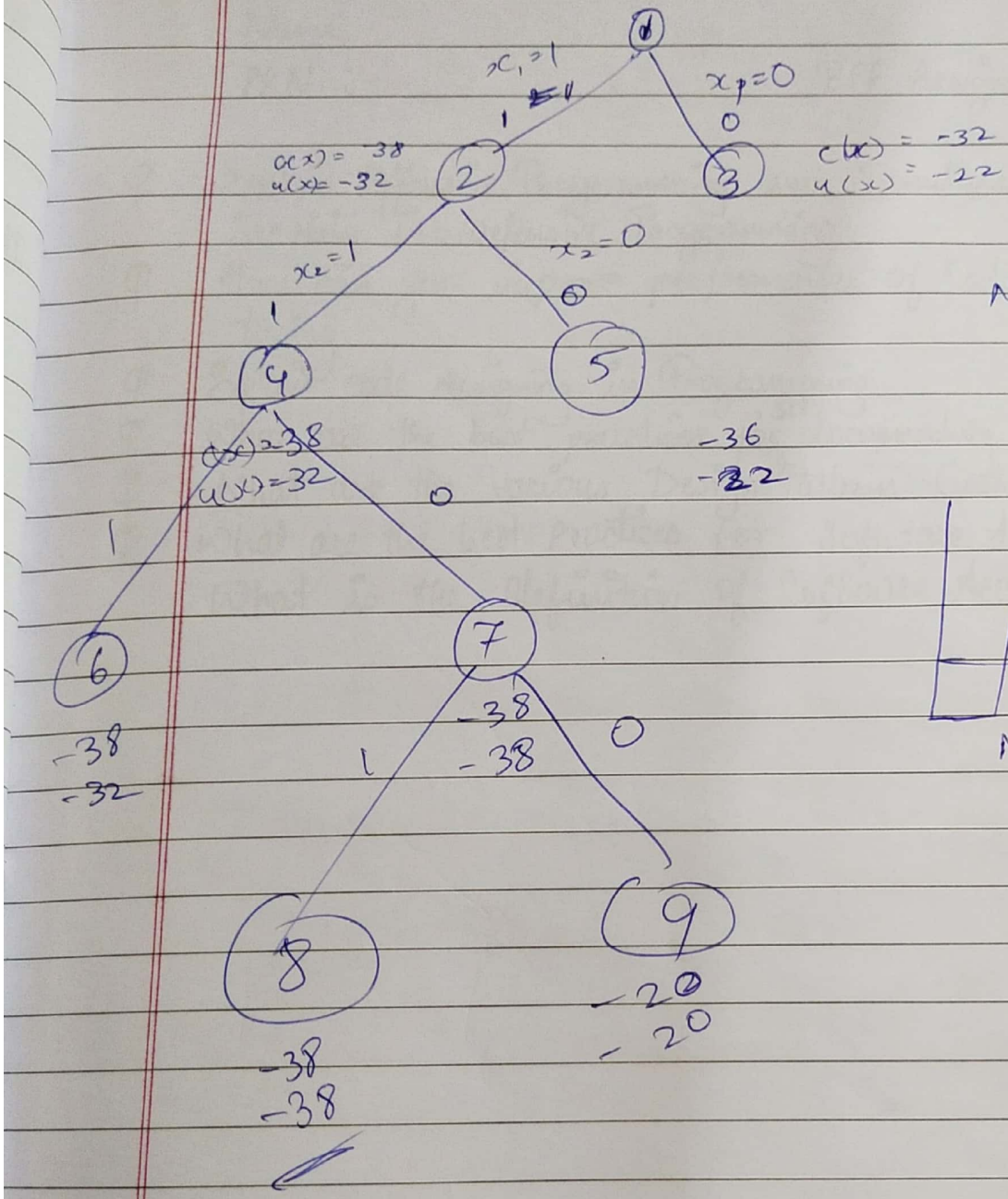
-18
-10
-10

L(7) = L(5)

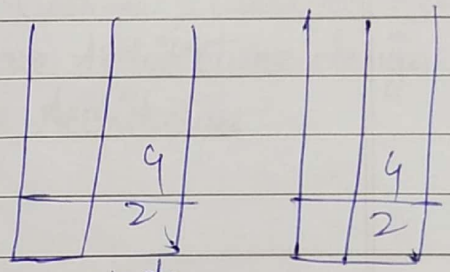
-18	9
-10	4
-10	2

U(7) = U

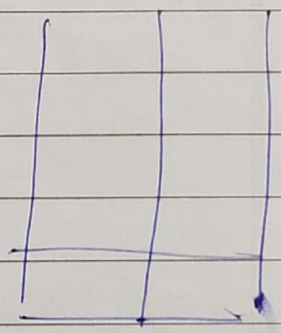
c(7) = -38
u(7) = -38



Node 4
$x_1 = 1$
$x_2 = 1$
$x_3 = 0$
$x_4 = 0$



Node



1 1 0 1

-10 -10 -18 $\Rightarrow$ 38 $\Leftarrow$ Total Profit

Hamilton's Cycle

~~SN~~ NP Hard