Unit-③

Part-① :- Branch and Bound

~~General Method~~

→ Branch and Bound is an approach/technique for solving optimization Problems.

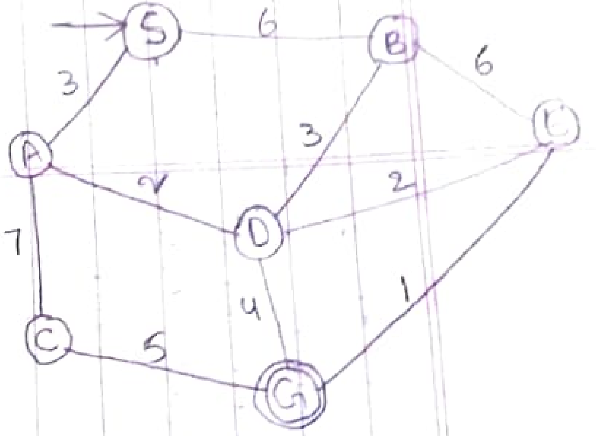
→ The optimum soln is found by, exploring all possible solutions to a problem by dividing the Problem into Sub-problems.

→ It is similar to Backtracking, but, Backtracking uses DFS (Depth-first-search) approach, whereas, Brand and Bound uses Breadth-first-Search (BFS) approach.

→ Branching refers to the Process of exploring all possibilities of Solving a Problem (or) Sub-Problem.

→ Bounding refers to the Process of Selecting and sticking to the best soln from all possibilities, and then exploring only that Sub-Problem further.

i.e., the Possibilities which are not better than the current best soln, are ignored and not explored further.

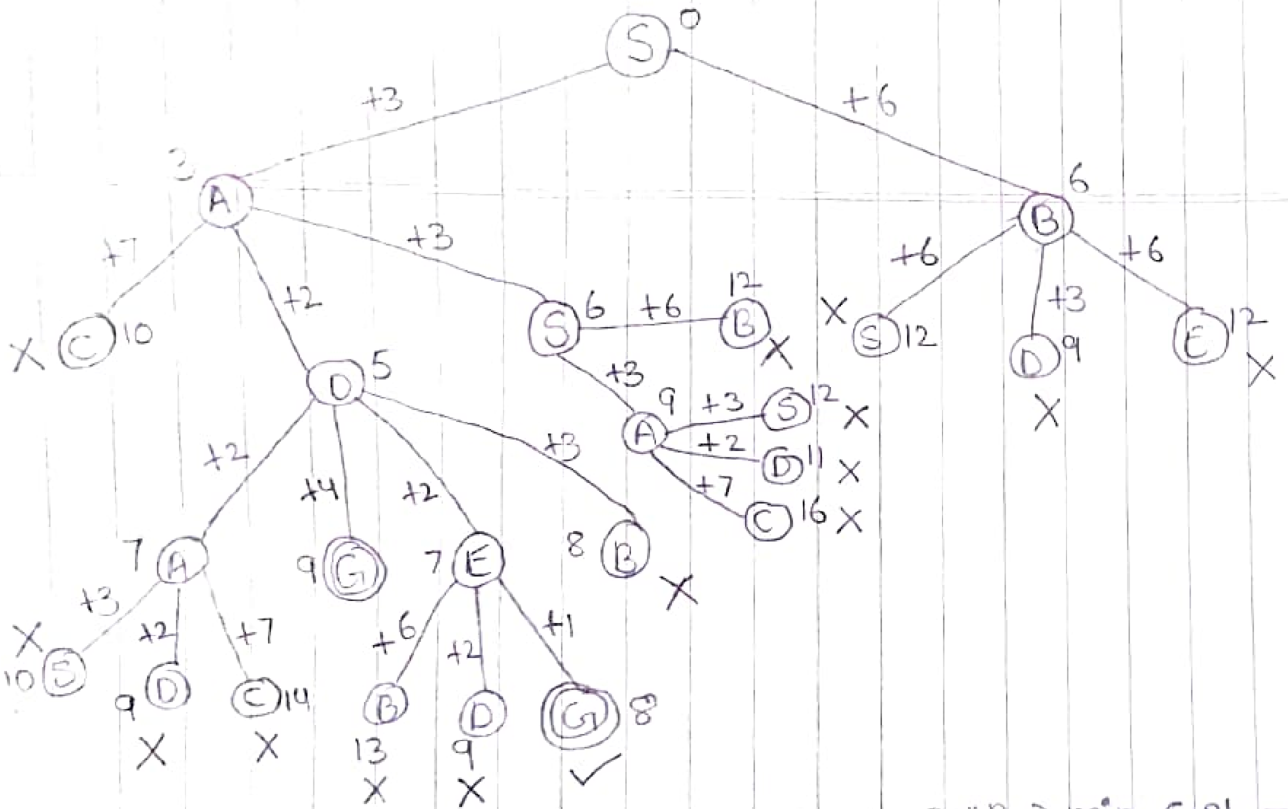


(Q) Start from (S) node and find the best soln (min cost) to reach (G) node.

Possible Solutions

1) Cost = 9

2) Cost = 8



∴ Best soln ⇒ min_cost = 8

i.e., min_cost_path is:–

$\boxed{S \rightarrow A \rightarrow D \rightarrow E \rightarrow G}$

→ when first soln (cost = 9) is found, we ignore those paths with greater than the Current best cost and Explore only those paths with (Cost <= Curr_best_cost).

[if a node's cost is Equal to Current best cost and it is not a goal node then ignore it anyway.]

→ Branch and Bound technique can be applied using (3) approaches, viz:—

① FIFO (Queue)
② LIFO (Stack)
③ LC (Least Cost/Least Count)

① **FIFO** (Queue):—

The nodes are inserted into a queue datastructure and Explored in FIFO order.

i.e., while Exploring a node, all of its adjacent nodes are Enqueued. And if a dequeued node doesnt meet a certain criteria, then it is rejected.

② **LIFO** (Stack):—

The nodes are inserted into a Stack datastructure and Explored in LIFO order.

i.e., while Exploring a node, all of its adjacent nodes are pushed into Stack. And if a popped node doesn't meet a certain criteria, then it is rejected.

③ LC (Least Cost / Least Count) :-

→ This approach is suitable for weighted graph-based problems.

→ While Exploring a node, all of its adjacent nodes' costs are Examined and only those nodes are Explored further, which have minimum cost (from the root node to current node).

[The Above Example is of LC approach]

Applications of Branch and Bound :-

BnB can be applied to following Problems :-

① Travelling Sales person Problem

② 0/1 Knapsack problem

- → LC Approach
- → FIFO Approach

① Travelling Sales Person Problem :-

Problem :- A Salesman is given a set of cities, and the task is to find the shortest possible tour (path) that visits each city exactly once and returns to the starting city.

→ i.e., Given a bi-directional graph (each node represents a city) and a Matrix (each value at (i,j) represents, the

Cost from ith node to jth node), the task is to find the Closed loop which covers all the nodes of the graph such that the total cost of edges is minimum.

Solving Procedure

=> A Reduced Matrix is a Matrix in which every Row and every Column have atleast one zero(0)

→ Step①:- Find the reduced Cost matrix from a given Cost matrix.
This is obtained by:-
(i) Row Reduction
(ii) Column Reduction.

(i) Row Reduction:- Take min. element of first row, Subtract that element from first row, Apply this procedure for all rows.

(ii) Column Reduction:- Take min. element of first column, Subtract that element from first column, Apply this procedure for all columns.

→ Step②:- Determine the Cumulative Reduction (γ).

$$\gamma = (\text{Row-wise Reduction Sum}) + (\text{Column-wise Reduction Sum})$$

where,

Row-wise Reduction Sum ⇒ Sum of elements subtracted from all rows.

Column wise Reduction Sum ⇒ Sum of elements subtracted from all columns.

→ Step ③ :-

∴ Cumulative Reduction (r) is nothing but the minimum cost of tour.
Hence, for starting node, consider (r) as lower bound and (∞) as the upper bound. [i.e., $L = r$, $U = \infty$]
And, find the cost to every adjacent node.
Steps to find the cost of an edge is :-

(i) For path (i, j), change the entries in row (i) and column (j) as (∞), in the Reduced Matrix.

(ii) Also change (j, 1) to (∞). Because person after reaching (j), the person must not go back to starting node (1).

(iii) Reduce the Matrix if it is not in Reduced state.
i.e., determine (r) again.
if it is already in reduced state, then, ($r = 0$).

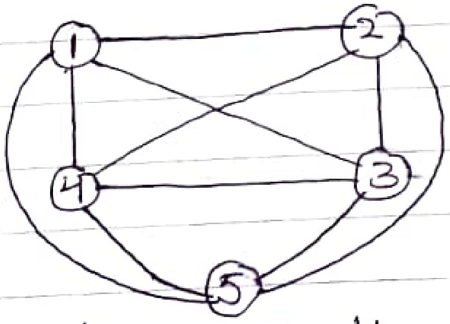
∴ Cost of an Adjacent node (Adj) is:-

$$\boxed{C(Adj) = A(i,j) + C(P) + r}$$

(A ⇒ original Cost matrix)
(C(P) ⇒ Cost of (Parent node)

→ Step ④:-
After finding Costs for all adjacent nodes, select the node with minimum cost and follow the same procedure from step ①, for that node.

Example:-
find the minimum Cost Path from vertex (1) to itself, by using TSP Approach.



$$\text{Cost Matrix} = \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & \infty & 20 & 30 & 10 & 11 \\ 2 & 15 & \infty & 16 & 4 & 2 \\ 3 & 3 & 5 & \infty & 2 & 4 \\ 4 & 19 & 6 & 18 & \infty & 3 \\ 5 & 16 & 4 & 7 & 16 & \infty \end{array} = A$$

Sol: By Reducing the Cost Matrix, we get:-

→ Row Reduction

$$C = \begin{bmatrix} \infty & 10 & 20 & 0 & 1 \\ 13 & \infty & 14 & 2 & 0 \\ 1 & 3 & \infty & 0 & 2 \\ 16 & 3 & 15 & \infty & 0 \\ 12 & 0 & 3 & 12 & \infty \end{bmatrix} \begin{array}{c} 10 \\ 2 \\ 2 \\ 3 \\ 4 \\ \hline 21 \end{array}$$

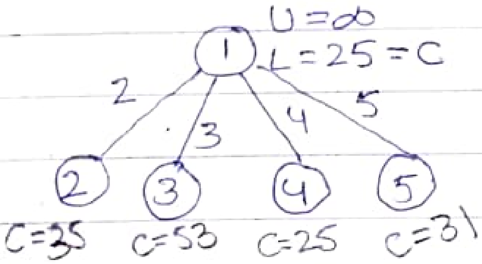
→ Colum Reduction

$$C = \begin{bmatrix} \infty & 10 & 17 & 0 & 1 \\ 12 & \infty & 11 & 2 & 0 \\ 0 & 3 & \infty & 0 & 2 \\ 15 & 3 & 12 & \infty & 0 \\ 11 & 0 & 0 & 12 & \infty \end{bmatrix}$$
$$\begin{array}{ccccc|c} 1 & 0 & 3 & 0 & 0 & 4 \end{array}$$

$\therefore \gamma = 21 + 4 = 25$

$\therefore U = \infty$

$L = \gamma = 25$



* for path (1,2), the cost is:-

By changing Row (1) and Column (2) entries and C(2,1), to (∞), we get:-

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 11 & 2 & 0 \\ 0 & 3 & \infty & 0 & 2 \\ 15 & 3 & 12 & \infty & 0 \\ 11 & 0 & 0 & 12 & \infty \end{bmatrix}$$

∵ the matrix is already reduced, then

$\gamma = 0$

∴ Cost to node(2) is:-

$$C(2) = A(1,2) + r + C(1)$$
$$= 10 + 0 + 25$$
$$= 35$$

* for path (1,3), the cost is:-

By changing Row(1) and Column(3) and C(3,1) to (∞), we get:-

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 12 & \infty & \infty & 2 & 0 \\ \infty & 3 & \infty & 0 & 2 \\ 15 & 3 & \infty & \infty & 0 \\ 11 & 0 & \infty & 12 & \infty \end{bmatrix}$$

By Reducing, we get:-

row-reduction = 0 (∵ Already reduced)

Column-reduction:-

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 1 & \infty & \infty & 2 & 0 \\ \infty & 3 & \infty & 0 & 2 \\ 4 & 3 & \infty & \infty & 0 \\ 0 & 0 & \infty & 12 & \infty \end{bmatrix}$$
$$11 \quad 0 \quad 0 \quad 0 \quad 0 \qquad \overline{\underline{11}}$$

$$\therefore r = 11 + 0 = 11$$

∴ Cost at node(3) is:-

$$C(3) = A(1,3) + r + C(1)$$
$$= 17 + 11 + 25 = 53$$

* for path (1,4), the cost is:-

By Changing Row (1) and Column (4) and C(4,1) to (∞), we get:-

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 12 & \infty & 11 & \infty & 0 \\ 0 & 3 & \infty & \infty & 2 \\ \infty & 3 & 12 & \infty & 0 \\ 11 & 0 & 0 & \infty & \infty \end{bmatrix}$$

∵ the matrix is already reduced,
then :- $\gamma = 0$.

∴ Cost at node (4) is:-

$$C(4) = A(1,4) + \gamma + C(1)$$
$$= 0 + 0 + 25 = 25$$

for path (1,5), the cost is:-
By Changing Row(1) and Column (5) and C(5,1) to (∞), we get:-

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 12 & \infty & 11 & 2 & \infty \\ 0 & 3 & \infty & 0 & \infty \\ 15 & 3 & 12 & \infty & \infty \\ \infty & 0 & 0 & 12 & \infty \end{bmatrix}$$

∵ the Matrix is Column wise reduced but not Row-wise Reduced.

∴ By Row-wise Reducing, we get:-

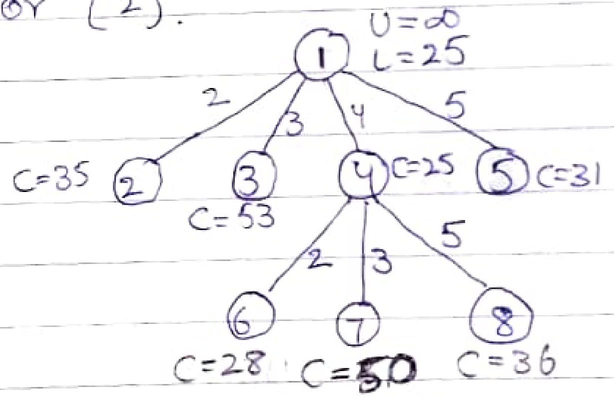
$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 10 & \infty & 9 & 0 & \infty \\ 0 & 3 & \infty & 0 & \infty \\ 12 & 0 & 9 & \infty & \infty \\ \infty & 0 & 0 & 12 & \infty \end{bmatrix} \begin{matrix} 0 \\ 2 \\ 0 \\ 3 \\ 0 \\ \hline 5 \end{matrix}$$

$$\therefore \gamma = 5 + 0 = 5$$

∴ Cost at node (5) is:-

$$C(5) = A(1,5) + r + C(1)$$
$$= 1 + 25 + 5 = 31$$

Now, Select the Node with least Cost [Least Cost Branch and Bound],
i.e., node (4) [C=25](min)
and Explore the node (4) further.
∴ from (4), we Can go to node (5) or (3) or (2).



Now, the Cost Matrix for node (4) is:-

$$C = \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 12 & \infty & 11 & \infty & 0 \\ 0 & 3 & \infty & \infty & 2 \\ \infty & 3 & 12 & \infty & 0 \\ 11 & 0 & 0 & \infty & \infty \end{bmatrix}$$

\+ for path (4,2) :-
make (4)th row and (2)nd Column and $C(2,1)$ as (∞)
because the Person must not go back from (2) to (1).

$$\begin{array}{cc}
\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 11 & \infty & 0 \\ 0 & \infty & \infty & \infty & 2 \\ \infty & \infty & \infty & \infty & \infty \\ 11 & \infty & 0 & \infty & \infty \end{bmatrix} & \begin{matrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{matrix} \\
\begin{matrix} 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 \end{matrix} &
\end{array}$$

$\Rightarrow \gamma = 0$

$\therefore C(6) = A(4,2) + C(4) + \gamma$
$= 3 + 25 + 0 = 28$

* for path (4, 3) :-

$$\begin{array}{cc}
\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 12 & \infty & \infty & \infty & 0 \\ \infty & 3 & \infty & \infty & 2 \\ \infty & \infty & \infty & \infty & \infty \\ 11 & 0 & \infty & \infty & \infty \end{bmatrix} & \begin{matrix} 0 \\ 0 \\ 2 \\ 0 \\ 0 \end{matrix} \\
\begin{matrix} 11 & \quad 0 & \quad 0 & \quad 0 & \quad 0 \end{matrix} &
\end{array}$$

$\Rightarrow \gamma = 11 + 2 = 13$

$\therefore C(7) = A(4;3) + C(4) + \gamma$
$= 12 + 25 + 13 = 50$

* for path (4, 5) :-

$$\begin{array}{cc}
\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 12 & \infty & 11 & \infty & \infty \\ 0 & 3 & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & 0 & 0 & \infty & \infty \end{bmatrix} & \begin{matrix} 0 \\ 11 \\ 0 \\ 0 \\ 0 \end{matrix} \\
\begin{matrix} 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 \end{matrix} &
\end{array}$$

$\Rightarrow \gamma = 11 + 0 = 11$

$\therefore C(8) = A(4,5) + C(4) + r$
$= 0 + 25 + 11 = 36$

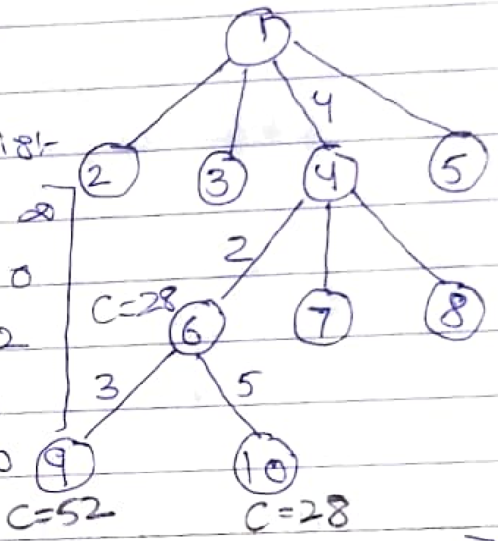
∵ Node (6) has min Cost.
∴ Explore (6), [i.e.,(2)]

from (2), we can go to (3), or (5).

[illegible]

∵ the 6th Node matrix is:-

$$C = \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 11 & \infty & 0 \\ 0 & \infty & \infty & \infty & 2 \\ \infty & \infty & \infty & \infty & \infty \\ 11 & \infty & 0 & \infty & \infty \end{bmatrix}$$



* for path (2,3):-

$$\left[\begin{array}{ccccc} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & 2 \\ \infty & \infty & \infty & \infty & \infty \\ 11 & \infty & \infty & \infty & \infty \end{array}\right] \begin{array}{c} 0 \\ 0 \\ 2 \\ 0 \\ 0 \end{array}$$
$$\begin{array}{ccccc} 11 & 0 & 0 & 0 & 0 \end{array}$$

$\therefore C(9) = A(2,3) + C(6) + r$
$= 11 + 28 + 13$
$= 52$

$\Rightarrow r = 2 + 11$
$= 13$

* for path (2,5):-

$$\left[\begin{array}{ccccc} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 0 & \infty & \infty \end{array}\right] \begin{array}{c} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{array}$$
$$\begin{array}{ccccc} 0 & 0 & 0 & 0 & 0 \end{array}$$

$\therefore C(10) = A(2,5) + C(6) + r$
$= 0 + 28 + 0 = 28$

$\Rightarrow r = 0$

Now, we select Node (10) i.e. [(5)] because its cost (c=28) is min.

∴ from (5), we can only go to (3). i.e., unvisited node.

∴ For path (5,3) :-

$$\therefore C(11) = A(5,3) + C(10) + r$$
$$= 0 + 28 + 0 = 28.$$

∴ [illegible]

∴ All nodes are visited.

∴ min_cost = 28 = C(11) //

min_cost_path ⇒ 1→4→2→5→3→1 //

0/1 Knapsack Problem:-

Problem:- Given (N) items where each item has some weight (w_i) and Profit (P_i) associated with it, and also given a bag with capacity (W) [i.e., the bag can hold atmost (W) weight in it].

→ The task is to put the items into the bag such that the sum of profits associated with them is the maximum possible.

Constraint:- We can either put an item completely into the bag (or) cannot put it at all.

(or) Total weight of items on the knapsack must be less than or equal to the bag capacity.

~~i.e.,~~

i.e., $\boxed{\sum_{i=1}^{N} w_i \leq W}$

Example: ④ (i) Using LC Branch and Bound

$W = 15$
$N = 4$
$P_i = \{10, 10, 12, 18\}$
$w_i = \{2, 4, 6, 9\}$

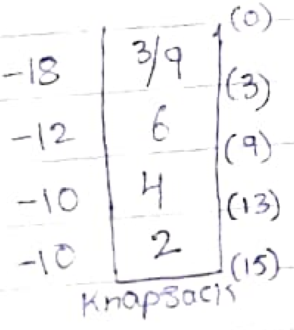
Sol: ∵ the branch and bound is for Minimization Problems, but Knapsack is the maximization Problem. Hence we Convert the Problem to Minimization, by taking the Profits as negative, so that Max value becomes Min value and vice-versa.

∴ $P_i = \{-10, -10, -12, -18\}$

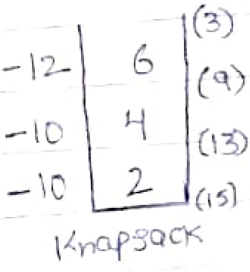
Now, we start building the tree and Exploring the nodes by determining upper and lower bounds at each node.

* for Node(1) [Root]:- (including all items)

→ The ~~upper~~ bound is:-



→ The upper bound is:-



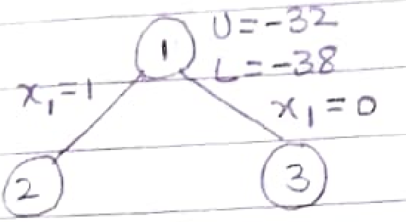
$\boxed{L = U = \text{Sum of Profits}}$

$L = -10-10-12-\overset{2}{\cancel{18}} \times \frac{3}{\cancel{9}}$

$= -38$

$U = -10-10-12$

$= -32$



$x_i = 0$ ⇒ Exclude item(i)
$x_i = 1$ ⇒ include item(i)

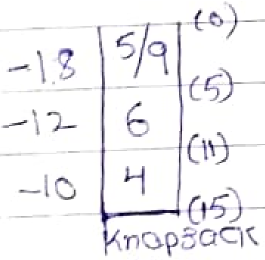
* for Node (2) :- ($x_1 = 1$ and all items)

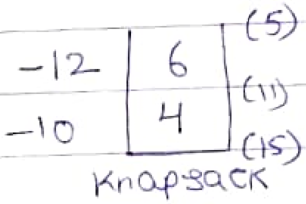
∵ Same as for Node (1).

∴ U = -32 , L = -38

* for Node (3) :- ($x_1 = 0$ and all items)

→ lower bound is:- →upper bound is:-



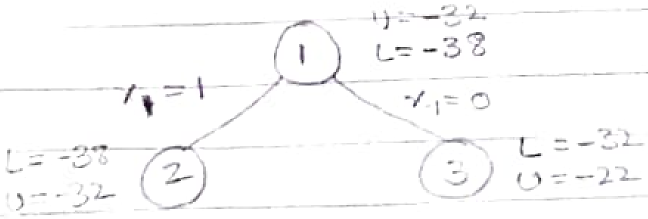


$L = -10-12-\overset{2}{\cancel{18}} \times \frac{5}{\cancel{9}}$

$= -32$

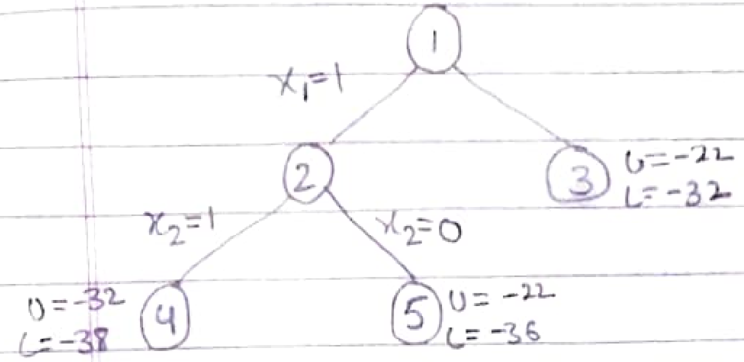
$U = -10-12$

$= -22$



Now, Choose the node with least lower bound and Explore further.

∵ (2) has the lowest lower bound



★ for Node (4) :- $(x_1=1, x_2=1)$

∵ same as Node (1). and (2)

∴ $L=-38$, $U=-32$

★ for Node (5) :- $(x_1=1, x_2=0)$

→ lower bound is :-

		(0)
-18	7/9	(7)
-12	6	(13)
-10	2	(15)

→ upperbound is :-

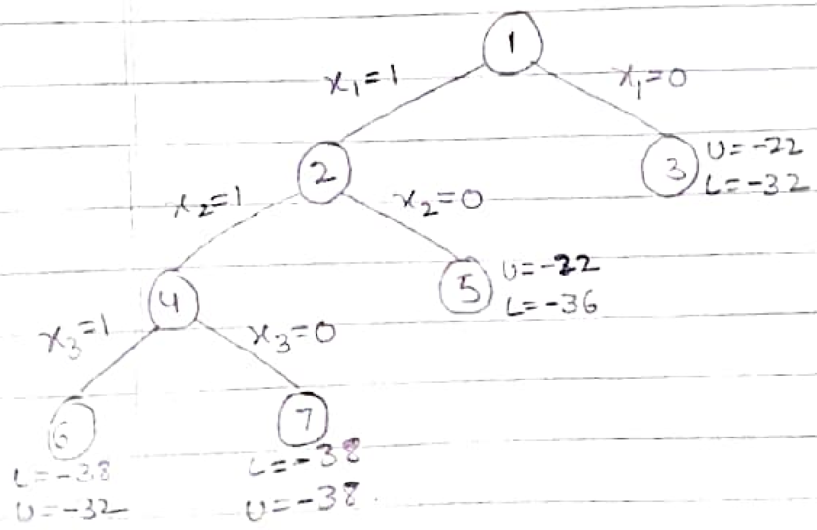
		(7)
-12	6	(15)
-10	2	(15)

$\therefore L = -10-12-18\times\frac{7}{9}$

$= -36$

$U = -10-12 = -22$

Now, out of (3), (4), (5) Nodes, Node (4) has the least lower bound, hence we will explore the Node (4) further.



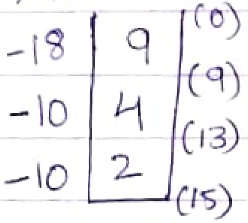
* for Node (6) :- ($x_1=1, x_2=1, x_3=1$)

∵ Same as Node(1)

∴ L = -38 , U = -32

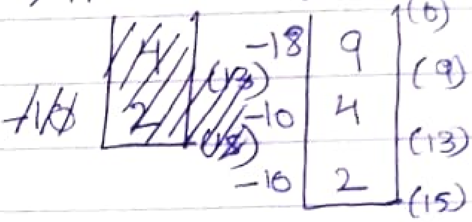
* for Node (7) :- ($x_1=1, x_2=1, x_3=0$)

->lowe bound is:-



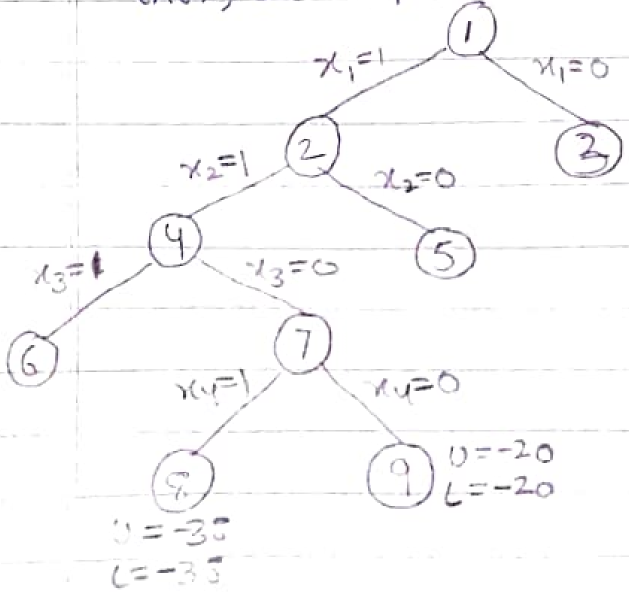
L = -16-10-18
= -38

->upper bound is:-



U = -10-10-18
= -38.

Now, the Nodes with least lowed bound are (6) and (7), but we must Explore only one node, Hence we Consider the choosing criteria based on the upper bound, instead of lower bound.
∴ Since, the node (7) has least upper bound then, we Explore node(7) further.



*for Node (8) :- $(x_1=1, x_2=1, x_3=0, x_4=1)$

∵ Same as Node (7)

∴ $L=-38$, $U=-38$.

* for Node (9):-

->Lower bound is:-

(9)

-10 | 4 |

(13)

-10 | 2 |

(15)

$L=-20$

->Upper bound is:-

$U=-20$.

Now, ∵ Node(8) has least Lower bound, we Consider Node (8). But all the items are already Considered and Explored. ∴ The path from Root Node to the Node(8) is the final answer.

∴ Optimum path ⇒ 1 → 2 → 4 → 7 → 8

which includes the objects/items :-

x_1, x_2, x_4 (which gives max profit) //

∴ Max. Profit = $P_1 + P_2 + P_4 = 10+10+18$

$= 38$ //

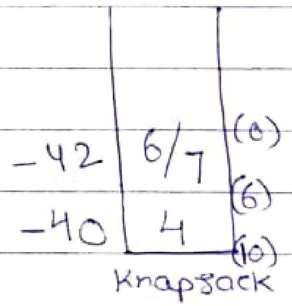
~~(ii) Using FIFO Branch and Bound~~:-

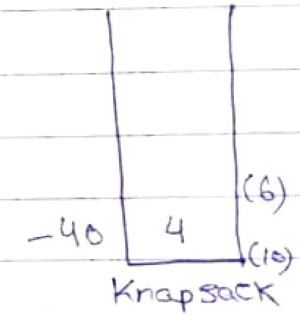
Example② :- Solve the instance of Knapsack problem using branch and bound algorithm. The Knapsack Capacity $W=10$.

Item	1	2	3	4
Weight	4	7	5	3
Profit	40	42	25	12

By Converting the Profits values to negative, we get the minimization Problem
i.e, Profit = {-40, -42, -25, -12}

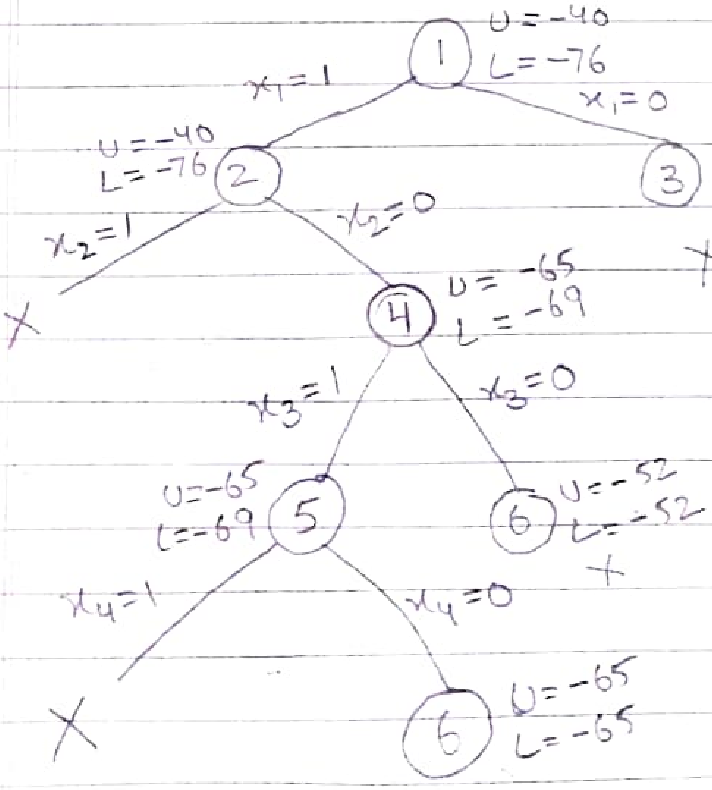
Now, we Start building the tree and Explore
* for Root Node, [Node ①] :- (including all nodes)
→ lower bound is :-   → upper bound is :-





Lower bound = $-40 - 42 \times \frac{6}{7}$
= -76

upper bound = -40



* for Node(2) :- [Including item(1)]

we get same results as that for Node(1)

* for Node (3) :- [Excluding item(1)]

→ lower bound is:-

		(0)
-25	3/5	(3)
-42	7	(10)

$L = -42 - 25 \times \frac{3}{5}$

$= -57$

→ upper bound is:-

		(3)
-42	7	(10)

$U = -42$

∵ Node (2) has least lower bound

∴ Explore Node(2)

~~[illegible]~~

~~[illegible]~~

* for Node (4) :- $[x_1 = 1, x_2 = 0]$

		(0)
-12	1/3	(1)
-25	5	(6)
-40	4	(10)

$L = -40 - 25 - 12 \times \frac{1}{3}$

$= -69$

		(1)
-25	5	(6)
-40	4	(10)

$U = -40 - 25 = -65$

∵ Node (4) has lowest lower bound

∴ Explore Node (4).

* for Node (5) :- $[x_1 = 1, x_2 = 0, x_3 = 1]$

we get same results as that for Node(4)

* for Node (6) :- $[x_1 = 1, x_2 = 0, x_3 = 0]$

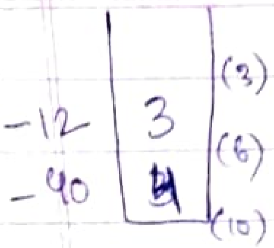
~~-42~~	~~6/7~~	~~(6)~~
~~-40~~	~~4~~	~~(10)~~

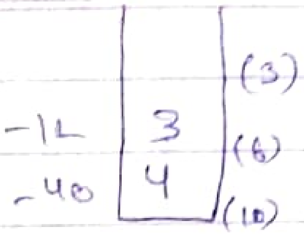
~~$L = -40 - 42 \times \frac{6}{7}$~~

~~$= -76$~~

		~~(6)~~
~~-40~~	~~4~~	~~(10)~~

~~$U = -40$~~





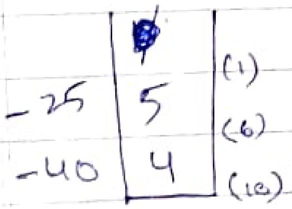
L = -40-12
= -52

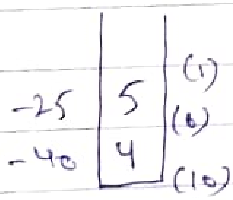
U = -40-12
= -52

∵ Node(5) has least lower bound.
∴ we Explore Node(5)

∵ If we include (x_4) (item 4), the knapsack capacity is exceeding, therefore we cannot include item (4).

+ for Node (6) :- [$x_1=1$, $x_2=0$, $x_3=1$, $x_4=0$]





=> L = -40-25
= -65

U = -40-25
= -65

∵ No other leaf node has less than (L=-65), we kill other Nodes

∴ optimum path = 1 → 2 → 4 → 5 → 6

item included = 1, 3 //

Profit = 40+25 = 65 //