SNS COLLEGE OF ENGINEERING

Coimbatore-107

COURSE NAME: ANALYSIS OF ALGORITHM

II YEAR/ IV SEMESTER

UNIT – V

BRANCH& BOUND ALGORITHM

Topic

Knapsack Problem

UNIT – 5

Knapsack Using Branch & Bound

This is solved by finding Upper Bound Value.

Example: Given:

S.No	Weight (w)	Value (v)	Value / Weight
1.	4	$40	10
2.	7	$42	6
3.	5	$25	5
4.	3	$12	4

Total weight capacity = $\boxed{W = 10}$

The max weight to be selected to fill the sack is solution.

Step 1:

Calculate Upper Bound Value:

$$ub = V + (W - w)(v_{i+1} / w_{i+1})$$

Step 2: Node 0: Root Node i = 0

$i = 0;\ w = 0;\ v = 0,\ \text{Total } W = 10$

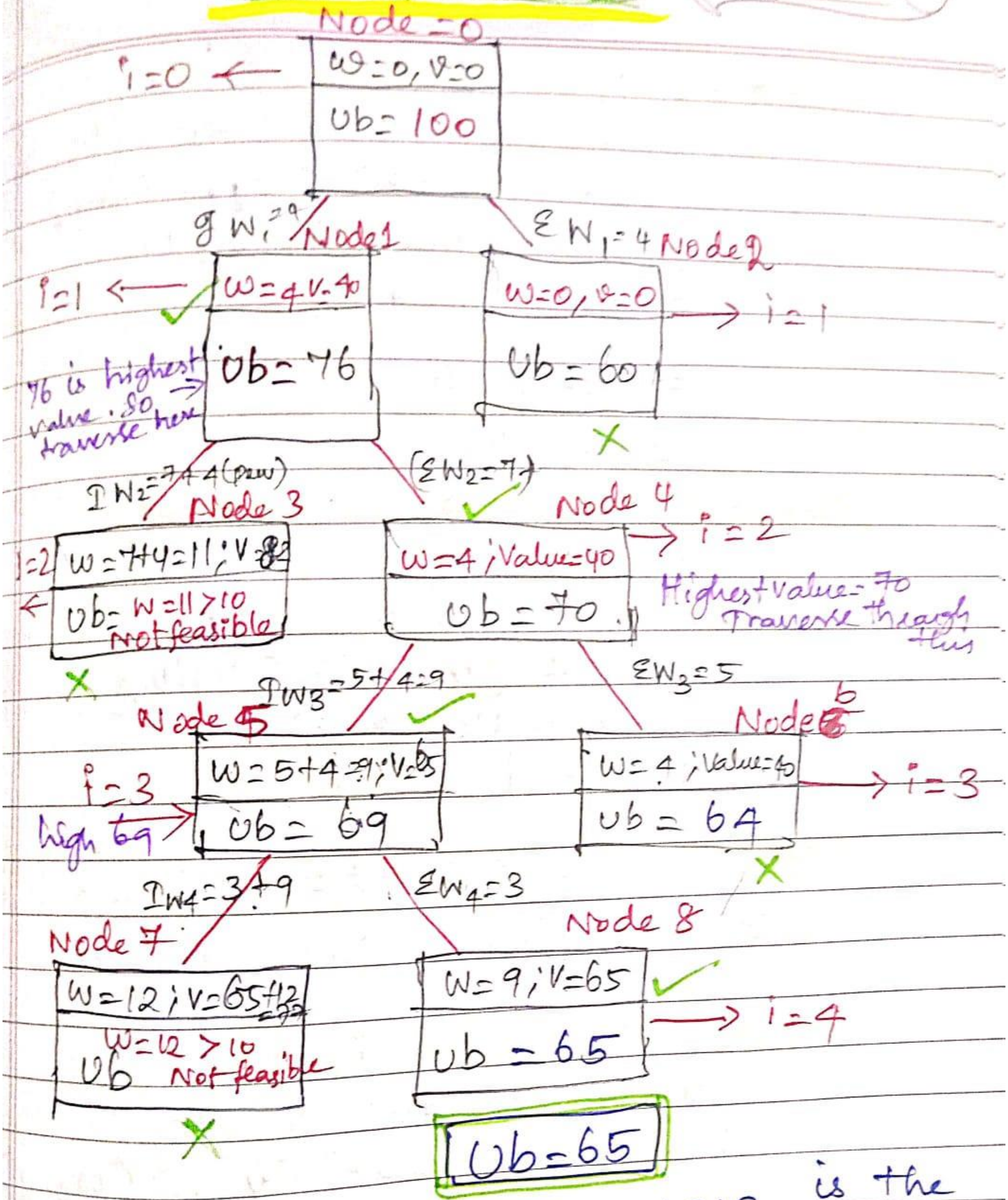
$$ub = V + (W - w)(v_{0+1} / w_{0+1})$$

$$= 0 + (10 - 0)(v_1 / w_1)$$ → 10 from table

$$= 0 + 10\ (10)$$

$$\boxed{ub = 100}$$

State Space Tree



is the solution

W_1 ✓ W_2 ✗ W_3 ✓ W_4 ✗ = 1010

Step 3: Node 1: i = 1.

$i = 1$; $w = 4$; $v = 40$; $\boxed{W = 10}$

$ub = v + (W - w)(v_{i+1} / w_{i+1})$

$= 40 + (10 - 4)(v_2 / w_2)$ → 6 from table

$= 40 + (6) * (6)$

$= 40 + 36$

$\boxed{ub = 76}$

Step 4: Node 2: i = 1; w = 0; v = 0

$i = 1$; $\boxed{W = 10}$

$ub = v + (W - w)(v_{i+1})(w_{i+1})$

$= 0 + (10 - 0)(v_2 / w_2)$ → 6 from Table

$= 0 + (10) * (6)$

$\boxed{ub = 60}$

Step 5: Node 3: $w = 4 + 7 = 11 >$ Total weight 10

So Solution Not feasible.

Step 6: Node 4: i = 2

$w = 4$; $v = 40$; $W = 10$

$ub = v + (W - w)(v_{2+1})(w_{2+1})$

$= 40 + (10 - 4)(v_3 / w_3)$

$= 40 + \underbrace{(6) * (5)}_{}$ 30

$\boxed{ub = 70}$

Page

step 7: Node 5: $i = 3$;

$w = 9$; $v = 25$; $W = 10$

$ub = v + (W - w)(v_{3+1} / w_{3+1})$

$= 65 + (10 - 9)(v_4 / w_4)$

$= 65 + (1) * (4)$

$= 65 + 4$

$\boxed{ub = 69}$

step 8: Node 6: $i = 3$;

$w = 4$; value $= 40$; $W = 10$

$ub = v + (W - w)(v_{3+1} / (w_{3+1}))$

$= 40 + (10 - 4)(v_4 / w_4)$

$= 40 + (6) * (4)$

$= 40 + 24$

$\boxed{ub = 64}$

step 9: Node 7: $w = 9 + 3 = 12 > 10$

So Not feasible solution.

Step 10: Node 8: $i = 4$;

$w = 9$; $V = 65$; $W = 10$.

$ub = \cancel{65}\, v + (W - w)(v_{4+1} / w_{4+1})$

$= 65 + (10 - 9)(v_5 / w_5)$ ↳ No value exist in Table

$\boxed{ub = 65}$

Algorithm:

{ while (front < rear)
v = queue [front ++]
if (v. bound > maxprofit)
{ v = {u.level + 1, v.profit + items[v.level][1]
u.weight + items[v.level][1]
if (v.weight <= capacity && v.profit > maxp
maxprofit = v.profit
if (v.bound > maxprofit) queue [rear++] = v;
v. weight = u.weight
v.profit = v.profit
v.bound = bound (v.level, v.profit, v.weight)
if(v.bound > maxprofit) queue (rear++] = v
} } }

Time complexity:

Worstcase: O(2^n) since explores all possible combinations.
Bestcase: O(n) [For I/p data, where branches are pruned early).

Space Complexity:

O(2^n) → Queue Store all nodes (sometimes). But in practice space is much smaller due to pruning.