BRANCH AND BOUND TECHNIQUE

⇒ Branch and bound is a state-space search method that can be visualized as an improved form of backtracking.

⇒ Suitable for solving combinatorial optimization problems. Eg. TSP, knapsack problem etc.
↳ These problems have innumerable but finite feasible solutions.

⇒ A set of feasible solutions / state space are partitioned and the subsets do not have optimal solutions are deleted from further consideration.

⇒ 2 steps.
① Branching } repeated till the problem is solved.
② Bounding

* Branching is the 1st step of Branch and bound. It divides given problem into two or more subproblems. These problems are exclusive and independent.

$f(x)$ ⇒ objective function.
S ⇒ state space tree
S ⇒ set of all solutions.
↳ also called feasible region.

BRANCH AND BOUND TECHNIQUE

⇒ Branch and bound is a state - space search method that can be visualized as an improved form of backtracking.

⇒ Suitable for solving combinatorial optimization problems. Eg. TSP, knapsack problem etc.
↳ These problems have innumerable but finite feasible solutions.

⇒ A set of feasible solutions / state space are partitioned and the subsets do not have optimal solutions are deleted from further consideration.

⇒ 2 steps.
① Branching } repeated till the problem is solved.
② Bounding }

* Branching is the 1st step of Branch and Bound. It divides given problem into two or more subproblems. These problems are exclusive and independent.

$f(x) \Rightarrow$ Objective function.
$S \Rightarrow$ State space Tree
$S \Rightarrow$ set of all solutions.
↳ also called feasible region.

(2) (s) This set is subdivided into feasible subregions S_i, such that the of all feasible subregions S_i (i ranges from to k) gives S back.

$$S = \bigcup_{i=1}^{k} S_i$$

→ The Constraints are both explicit and implicit.

→ In Branch and bound, we have to search the state space tree for finding the optimal solution.

(2) Bounding step limits the growth of state space tree exponentially.

* The best solution of the subproblems is identified and used as a lower bound of the given problem.

* Lower bound is the optimistic estimate of the best solution.

* For every node i of the state space tree lower bound is calculated.

* Similarly upper bound needs to be computed.

Upper Bound - Minimum amount of work is required to compute the result.

Lower bound for minimization problem
Upper Bound " Maximization " } To determine further action

In minimization problems, if the lower bound of i node is > its upper bound, then node i is said to be 'fathomed'

~~Upper~~ Lower bound should be lesser than upper bound.

⇒ Success of branch and bound algs. depends on the quality of the lower and upper bounds.

⇒ These alg. spend more time on the computations of bounds of the given problem.

∴ Implementation of bounding step is more difficult than branching step.

BACKTRACKING	BRANCH AND BOUND
① Uses only DFS technique	① Not limited to any search. BFS, DFS or least cost search can be used
② Explores state-space trees partially, potentially solutions may sometimes be ignored.	② Check the state space tree completely for an optimal solution. always potential solutions obtained
③ Can be used for all types of problems such as enumerative, decision, optimization.	③ Can be used only for optimisation problems.

(4)

SEARCH TECHNIQUES FOR BRANCH AND BOUND.

(1) Blind searches that give no preference to nodes. Alg's such as DFS, BFS and D-search use queue & stack are called blind searches.

(2). Intelligent searches.

Eg. Least Cost (LC) search. gives preference to nodes and process nodes that are promising.

[Suboptimal nodes are pruned]

KNAPSACK PROBLEM

⇒ It can be solved using branch and bound technique.

⇒ Each item is associated with a profit V_i and weight w_i.

⇒ Objective function is to achieve the maximum profit subjected to the constraint of the capacity of the knapsack.

Steps

1. Arrange the items $\frac{V_1}{w_1} \geq \frac{V_2}{w_2} \geq \frac{V_n}{w_n}$.

2. Construct state space tree. Branching to the left indicates that the item is included. Branching to the right indicates that the item is excluded.

3. Compute ~~lower~~ upper bound

$$ub = V + (W - w) + \frac{V_{i+1}}{W_{i+1}}.$$

W ⇒ Capacity of the knapsack.

w ⇒ weight of the item.

V_{i+1}, W_{i+1} ⇒ Value and weight of next item.

(6) problem

Items	w_i	V_i
1	2	8
2	3	6
3	2	4

with knapsack capacity $W = 6$.

$\frac{V_1}{w_1} = \frac{8}{2} = 4$ $\quad \frac{V_2}{w_2} = \frac{6}{3} = 2$ $\quad \frac{V_3}{w_3} = \frac{4}{2} = 2$

Item 1, 2, 3 → sorted order.

Upper Bound (Bounding fn.) $= V + (W - w) \times \frac{V_{i+1}}{W_{i+1}}$

step 1

$$UB = V + (W - w) \times \frac{V_{i+1}}{w_{i+1}}$$

$$= 0 + (6 - 0) \times \frac{V_1}{w_1} = 6 \times \frac{8}{2} = 24.$$

Nodes 2 and 3

Node 2 → Inclusion of item 1

Node 3 → Exclusion " " "

weight of item 1 is 2

profit " " " 8.

$$\therefore \; UB = 8 + (6 - 2) \times 6/3$$

$$= 16.$$

node 3.   3 to.

$$UB = 0 + (6-0) \times 6/3$$
$$= 12.$$

Node 4 (4 to 7 - Inclusion of item 2)

$$UB = 14 + (6-5) \times 4/2 = 16.$$

Node 5

$$UB = 8 + (6-5) \times 2 = 10.$$

Node 6

$$UB = 6 + (6-3) \times 2 = 12$$

Node 7

$$UB = 0 + (6-0) \times 2 = 12.$$

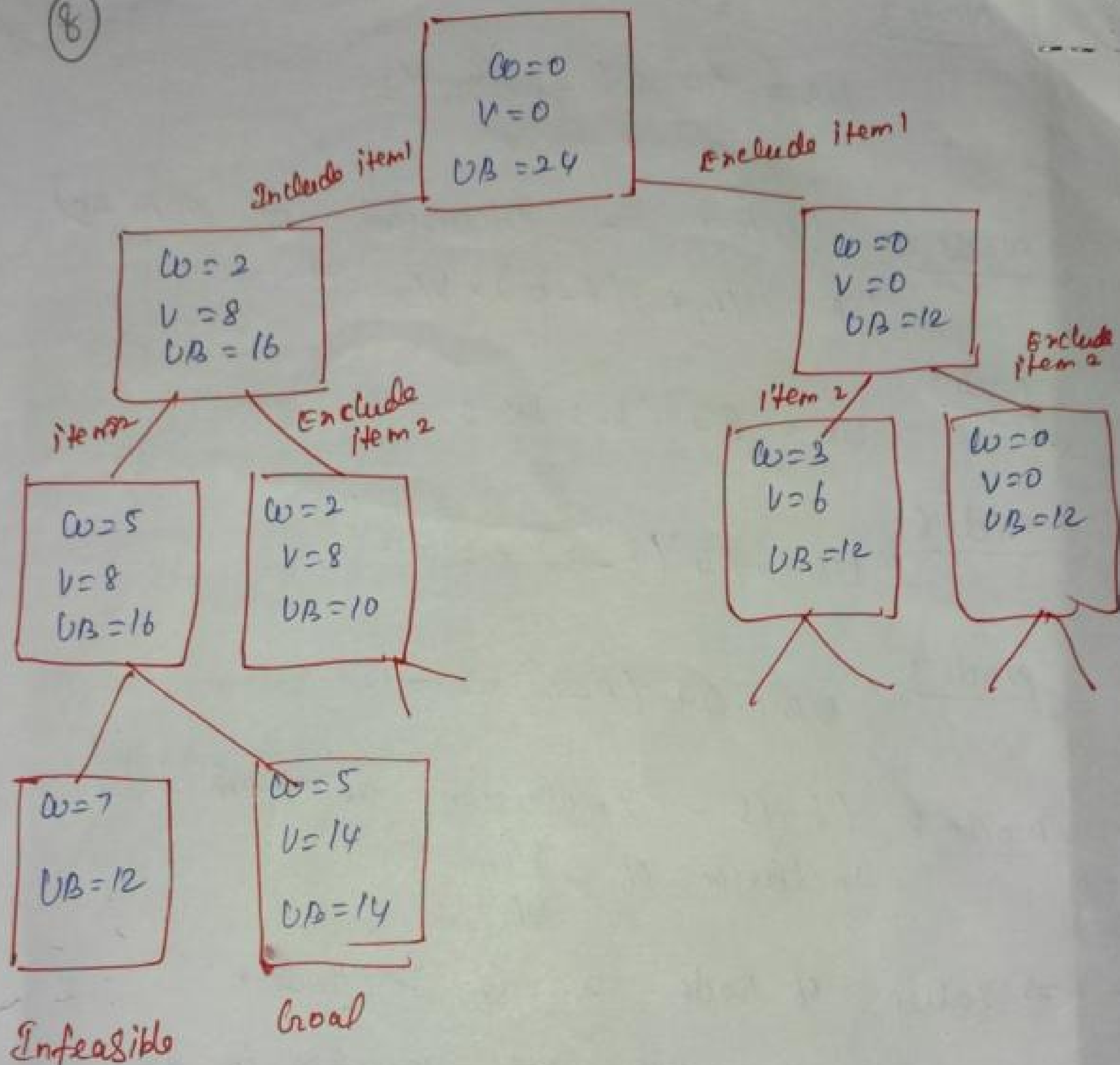
Node 8 (8-15 - Inclusion of item 3).
Inclusion of item 3.
UB =

⇒ value of node 2 is > node 3.

∴ node 3 does not require to be generated in state space tree.

Maximum profit can be achieved if the given i/p value is 14. ⇒ Including items 1 and 2 and discarding 3.

8


W=0
V=0
UB=24
Include item1
Exclude item1
W=2
V=8
UB=16
W=0
V=0
UB=12
item2
Exclude item2
item 2
Exclude item 2
W=5
V=8
UB=16
W=2
V=8
UB=10
W=3
V=6
UB=12
W=0
V=0
UB=12
W=7
UB=12
W=5
V=14
UB=14
Infeasible
Goal

31