6/5/17

MODULE :- 5

BACK TRACKING

The principal idea is to construct solutions through one component at a time and evaluate such partially constructed candidate sol^{n}.

If the partially constructed sol^{n} can be developed further without violating the problem constraints it then done through remaining legitimate option for the next component.

If there is no legitimate option for the next component no alternative for any relative for any remaining component, then algorithm backtracks to replace the last component solution with the next available option.

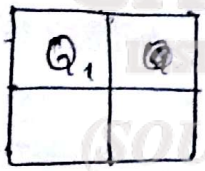
It is generally represent by ~~space~~ State - space - tree.

n - Queens problem

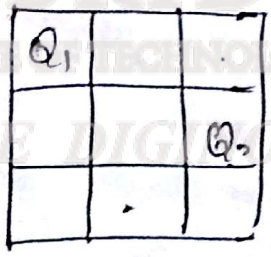
n x n - chess board / Cross - board

n - queens.

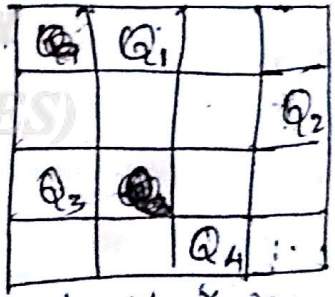
Objective :- place all queens in non-attacking position.



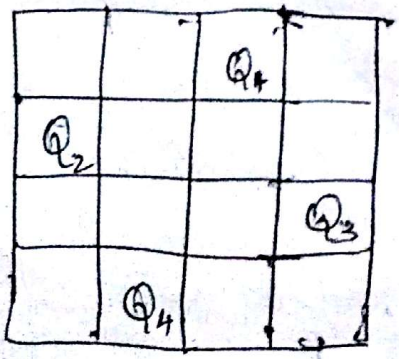
no - solution
2 x 2



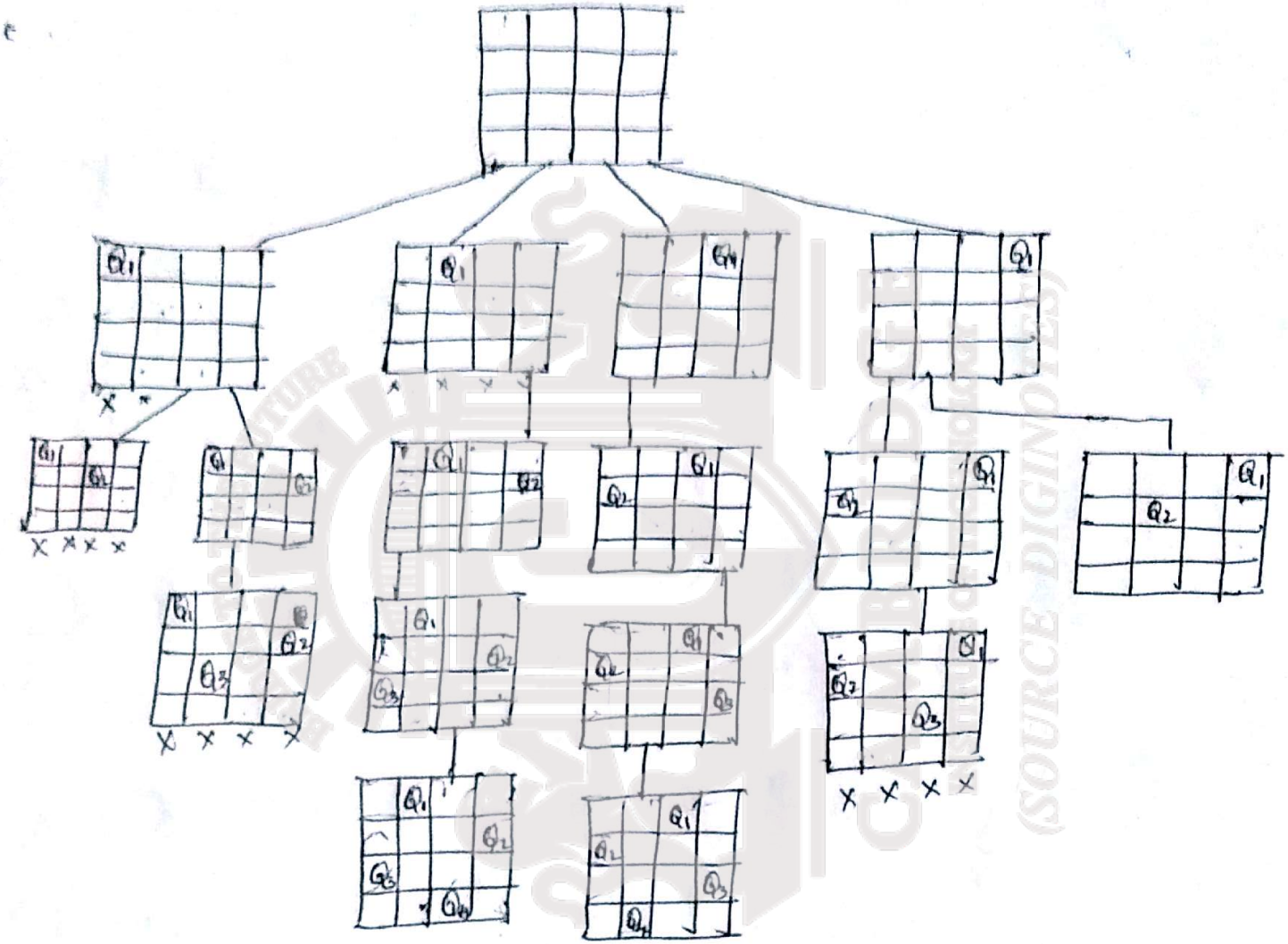
x x x
No - solution for 3x3



x x x x
~~No - solution for~~ 4x4



State space tree



Sub-set sum problem

Set $S = \{s_1, s_2, \ldots s_n\}$ where $s_1 < s_2 < \ldots < s_n$

d = sum of sub set

Ex :- $S = \{1, 2, 3, 5, 7\}$

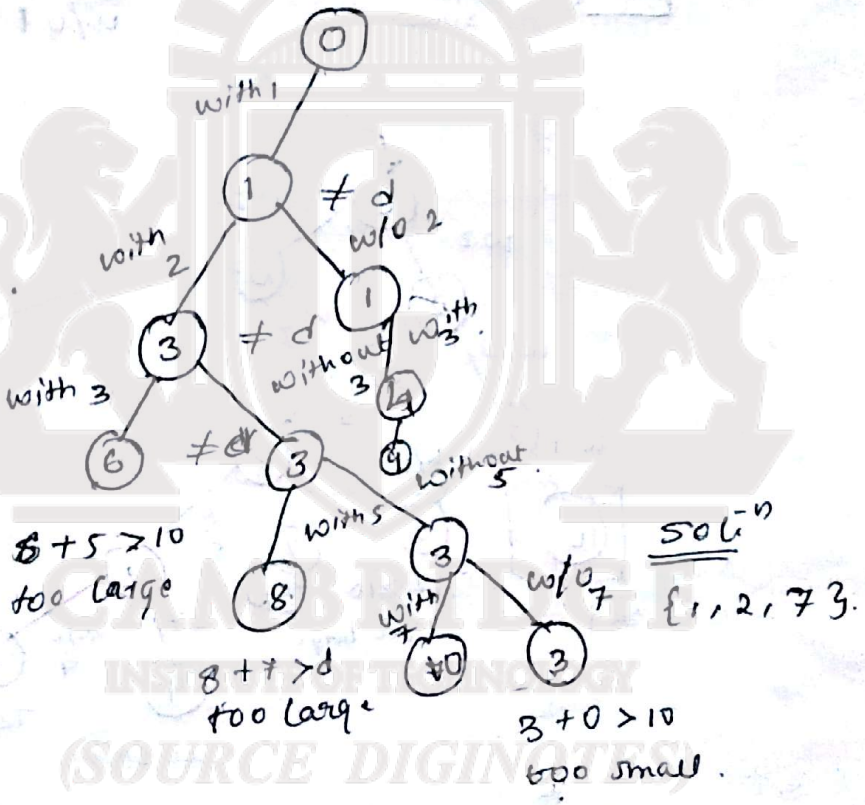
$d = 10$: $\{2,3,5\}, \{1,2,7\}, \{3,7\}$

Branch Terminating condition.

$s' + s_{i+1} > d \Rightarrow$ too Large.

$s' + \sum_{j=i}^{n} s_j < d \Rightarrow$ too Small

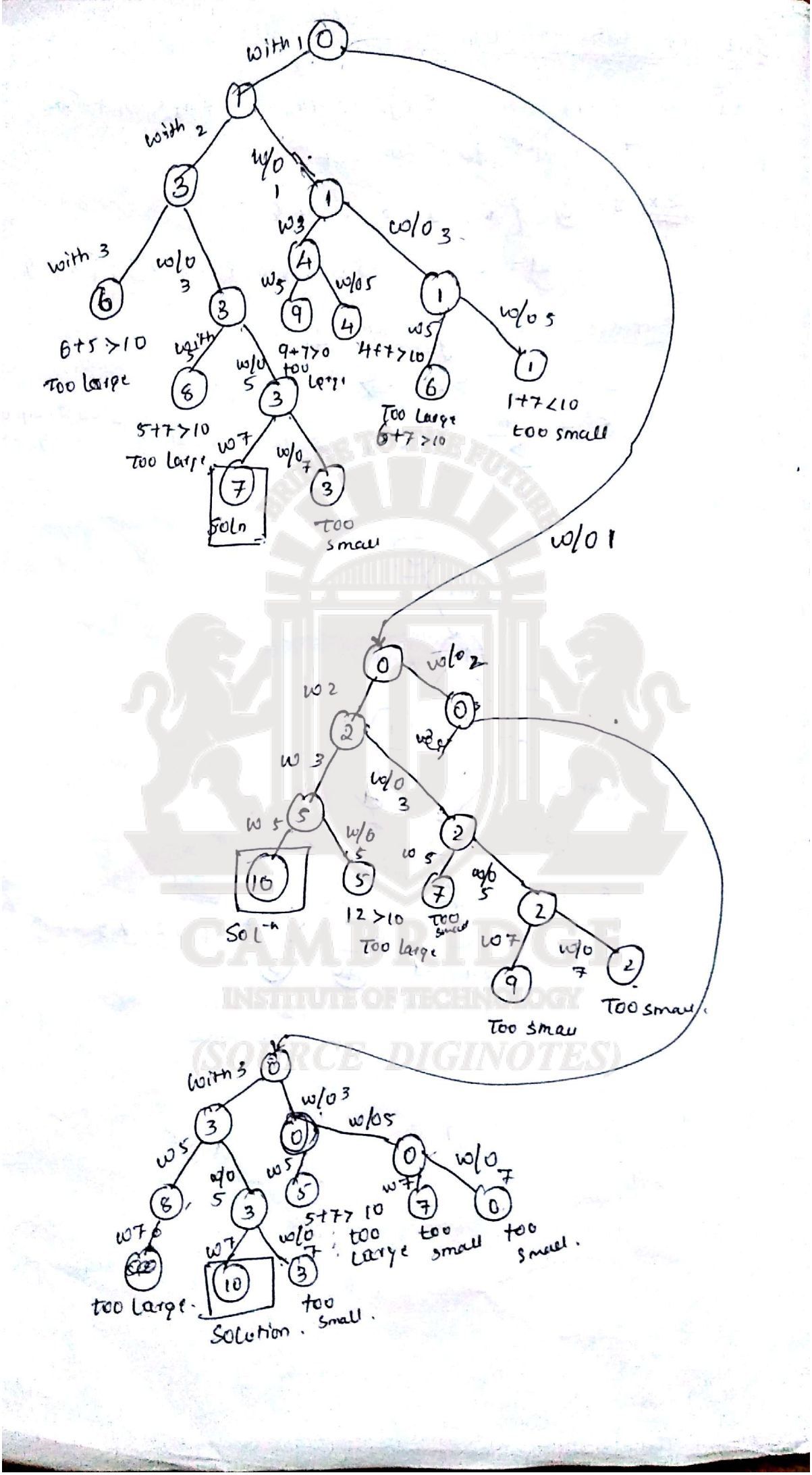
$s' \rightarrow$ Tempo
-ry sum

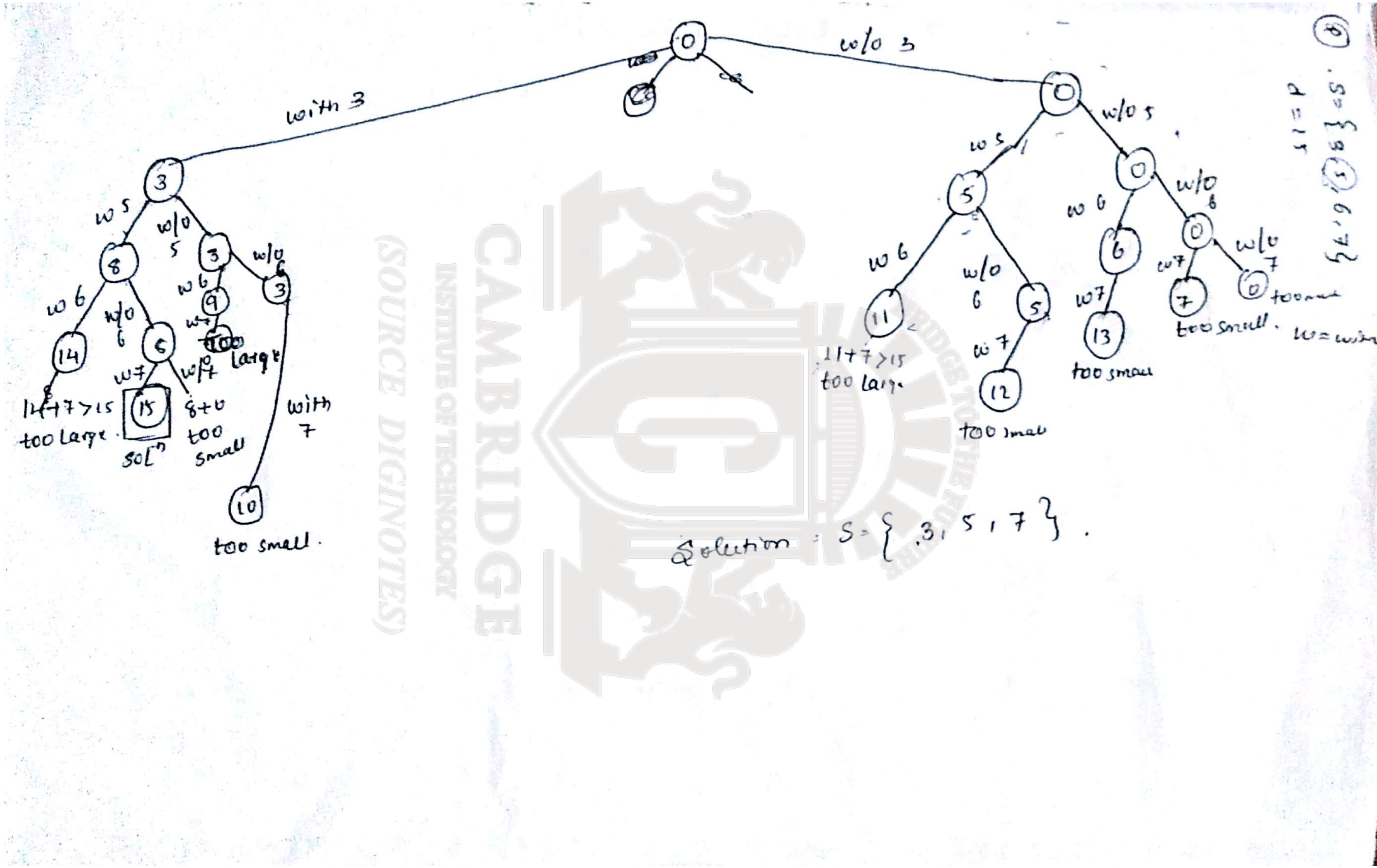


Soln

$\{1, 2, 7\}$.

P.T.O.



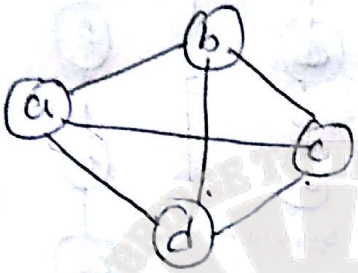


Hamiltonian Circuit Problem.

G{V, E}

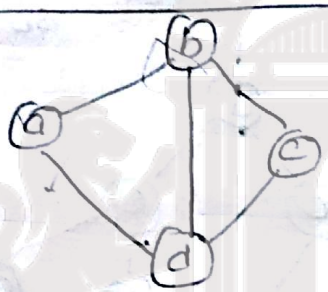
$u \rightarrow v_1 \rightarrow v_2 \cdots \rightarrow u$ → cyclic path.

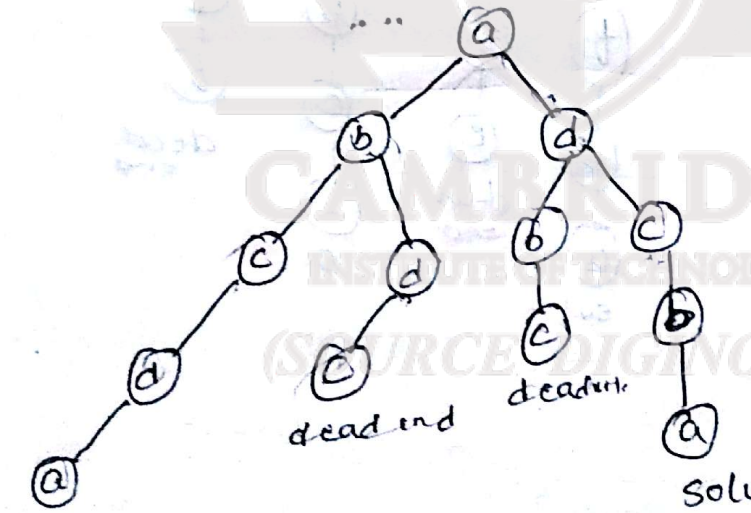
reaching all the vertices once and back to source vertix (u).



a → b → c → d → a.
a → d → c → b → a
a → b → d → c → a.
a → d → b → c → a.
a → c → b → d → a,
a → c → d → b → a.

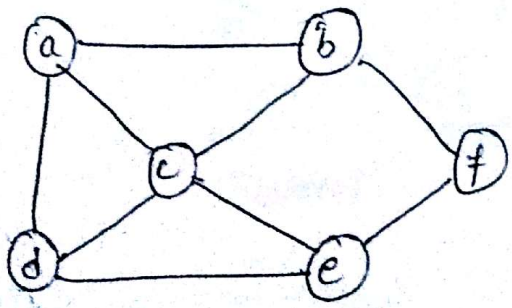
1.

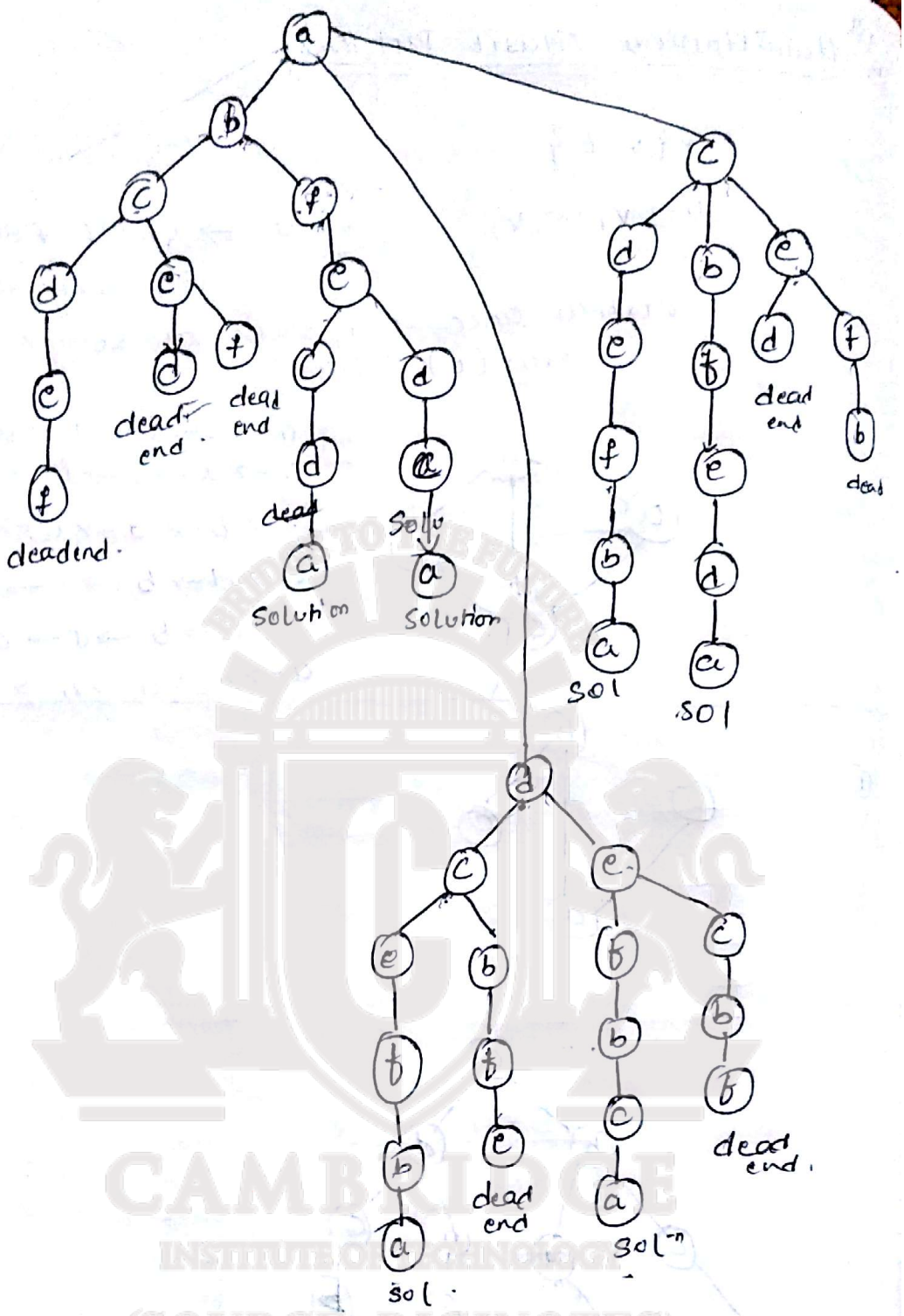




Solution.

3





Branch and - Bound

→ Solution for limitations of Algorithm
→ Applied to optimization problems.
→ By calculating upper bound (maximization) / Lower bound (minimization)
→ Branch and - Bound provides best sol" found so far
→ It bounds on the best value of the objective function.

→ Job Assignment problem
→ Knapsack problem.
→ TSP

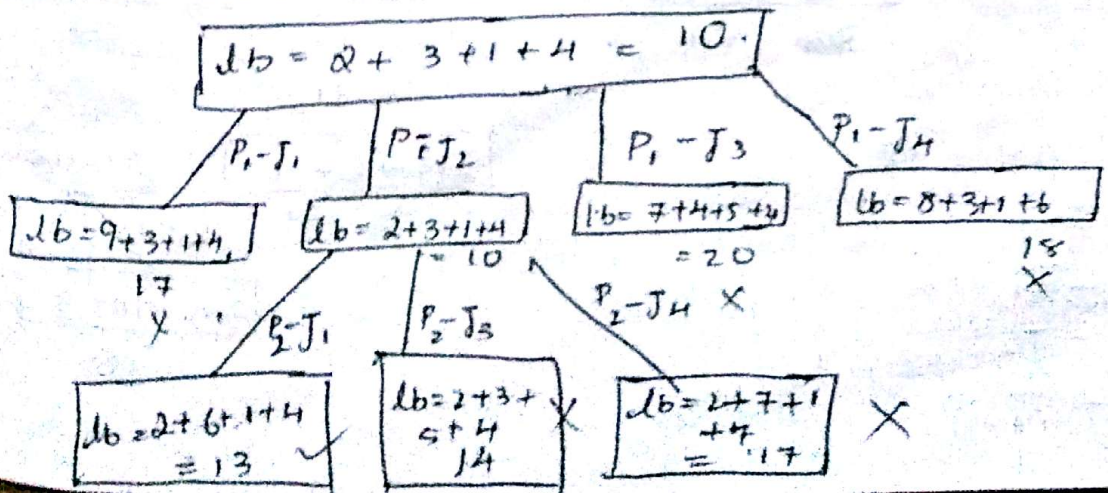
① Job Assignment Problem :-

→ n - jobs and n - people
→ Every person can perform all the jobs with diffrent cost
Feasible Constraint :- Assign one job to one person.
Optimal solution :- least cost job assignment.

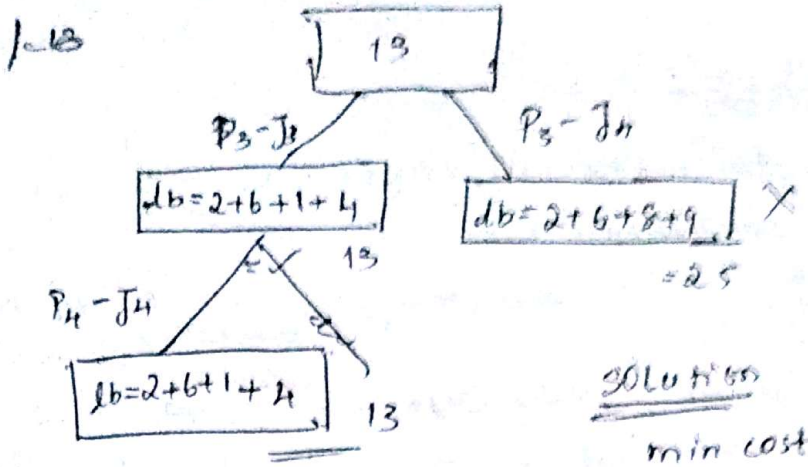
Ex

	J_1	J_2	J_3	J_4
P_1	9	[2]	7	8
P_2	[6]	4	3	7
P_3	5	8	[1]	8
P_4	7	6	9	[4]

lower bound = $\min P_1 + \min P_2 + \min P_3 + \min P_4$



1-13



Solution

min cost = 13

$P_1 - J_2$

$P_2 - J_1$

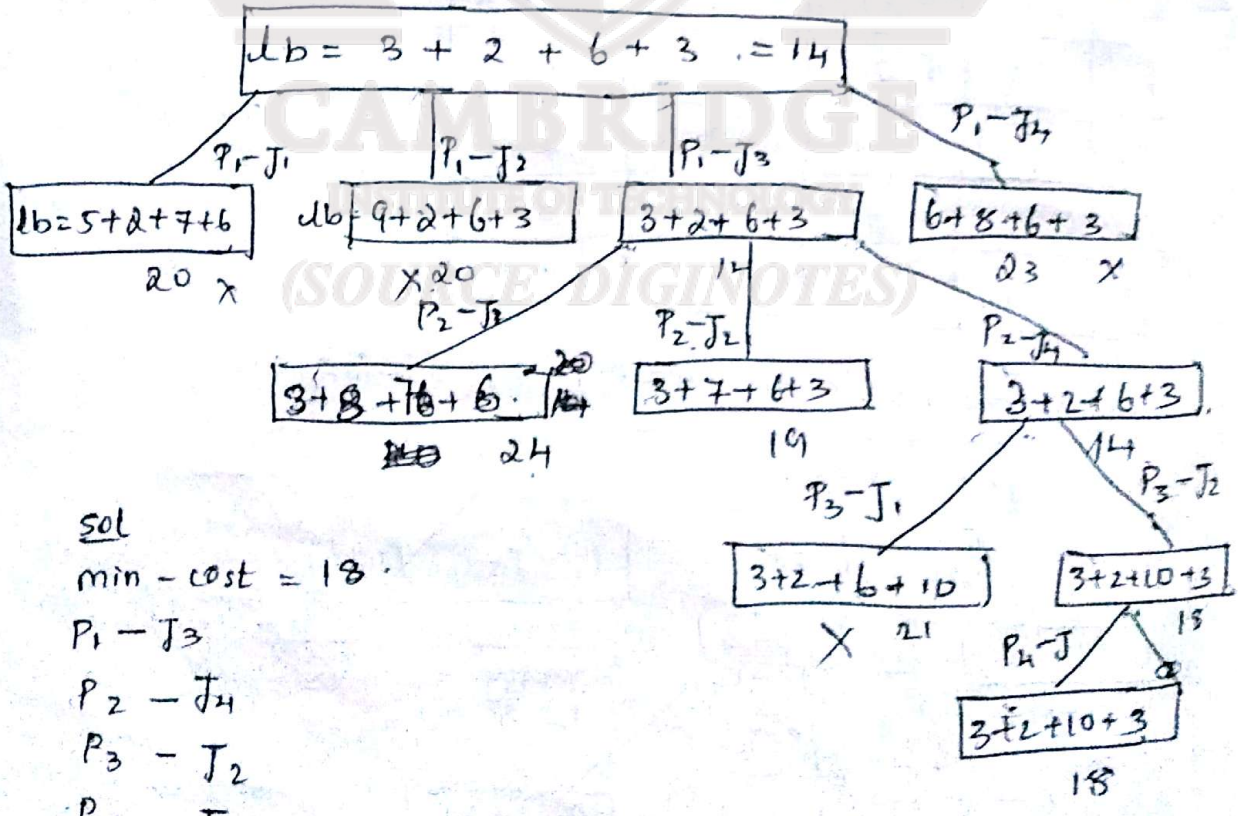
$P_3 - J_3$

$P_4 - J_4$

Example 2

	J_1	J_2	J_3	J_4
P_1	5	9	[3]	6
P_2	8	7	8	[2]
P_3	6	[10]	12	7
P_4	[3]	10	8	6

lower bound = $\min P_1 + \min P_2 + \min P_3 + \min P_4$



Sol

min - cost = 18

$P_1 - J_3$

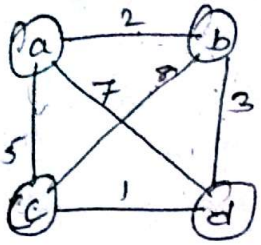
$P_2 - J_4$

$P_3 - J_2$

$P_4 - J_1$

Q3 | 05 | 17

Travelling salesperson problem [using branch & bound]



$$lb = \sum_{i=1}^{n} \frac{[\text{incoming edge of } v_i + \text{outgoing edge of } v_i]}{2}$$

divide by 2 because 2 vertices share the same edge.

$lb = \frac{(2+5)+(2+3)+(1+3)+(1+5)}{2} = 11$

a-b

$lb = \frac{(2+5)+(2+3)+(5+1)+(1+3)}{2} = 11$

b-c

$lb = \frac{(2+5)+(2+3)+(1+3)+(3+1)}{2} = 15$

a-c

$lb = \frac{(5+2)+(2+3)+(5+1)+(1+3)}{2} = 11$

c-d

$lb = \frac{(2+5)+(2+3)+(1+5)+(3+1)}{2} = 11$

Soln

c-a

$lb = \frac{(2+5)+(5+2)+(1+3)}{2} = 17$

a-d

$lb = \frac{(7+2)+(2+3)+(5+1)+(7+1)}{2} = 14$

$lb = \frac{(5+2)+(2+3)+(1+3)}{2} = 11$

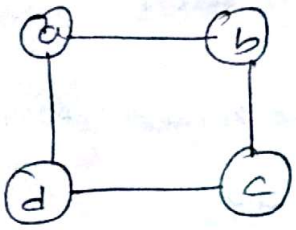
SOL.

Graph coloring problem (Back tracking method)

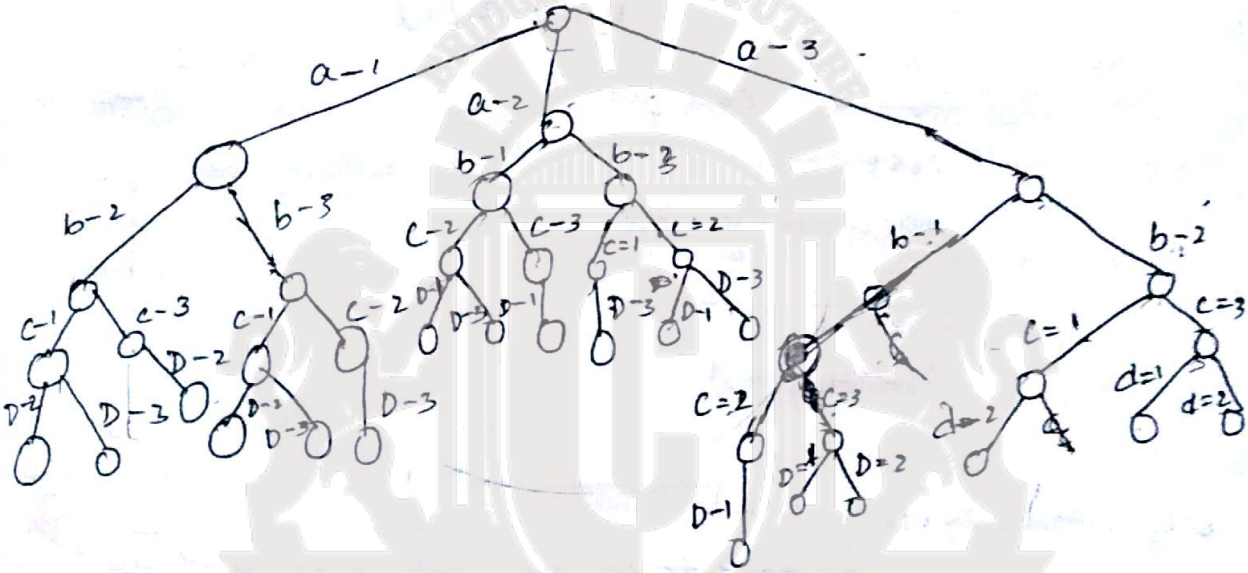
Graph $G = \{V, E\}$

m - colours.

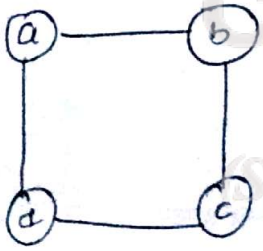
coloring vertices → no two adjacent vertices should have same color.



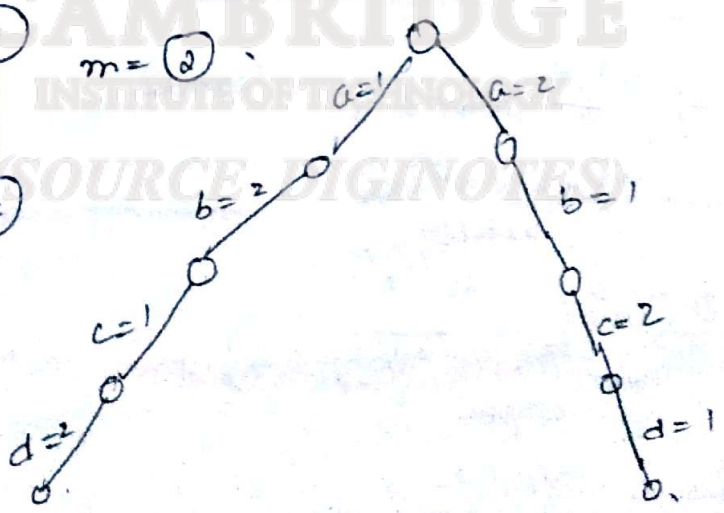
m = 3 ie 1, 2, 3.



Ex 2



m = 2.



```
Algorithm mcoloring(k).
//Input : Vertex k - need to be coloured which
        varies from coloured[1...n]
//output: colour assigned to vertex k in x[1..n]
Repeat forever
        nextvalue(k)
        if(x[k] = 0) then    //no new color is possible
                break.
        if(k=n) then    //All the nodes are coloured
                for i← to n do
                        print x[i]
                end for
        else
        mcoloring(k+1)
        end if
        endrepeat.

Algorithm nextvalue(k)
//Input: vertex k need to be assigned with a color
//output: Assigned colour for k in x[k]
                m = number of colours.
Repeat
        x[k] ← (x[k]+1) % (m+1)
        if(x[k]=0) then
                return
        for j←1 to n
                if(G[k][j] = 1 and x[k]=x[j]) then
                        break.
                end if
        end for
        if j=n+1 then
                return
        else
        end if
```

Knapsack problem [Branch-and-Bound]

$ub = v + (m - w) * \frac{v_{i+1}}{w_{i+1}}$ → value to weight ratio of next item.

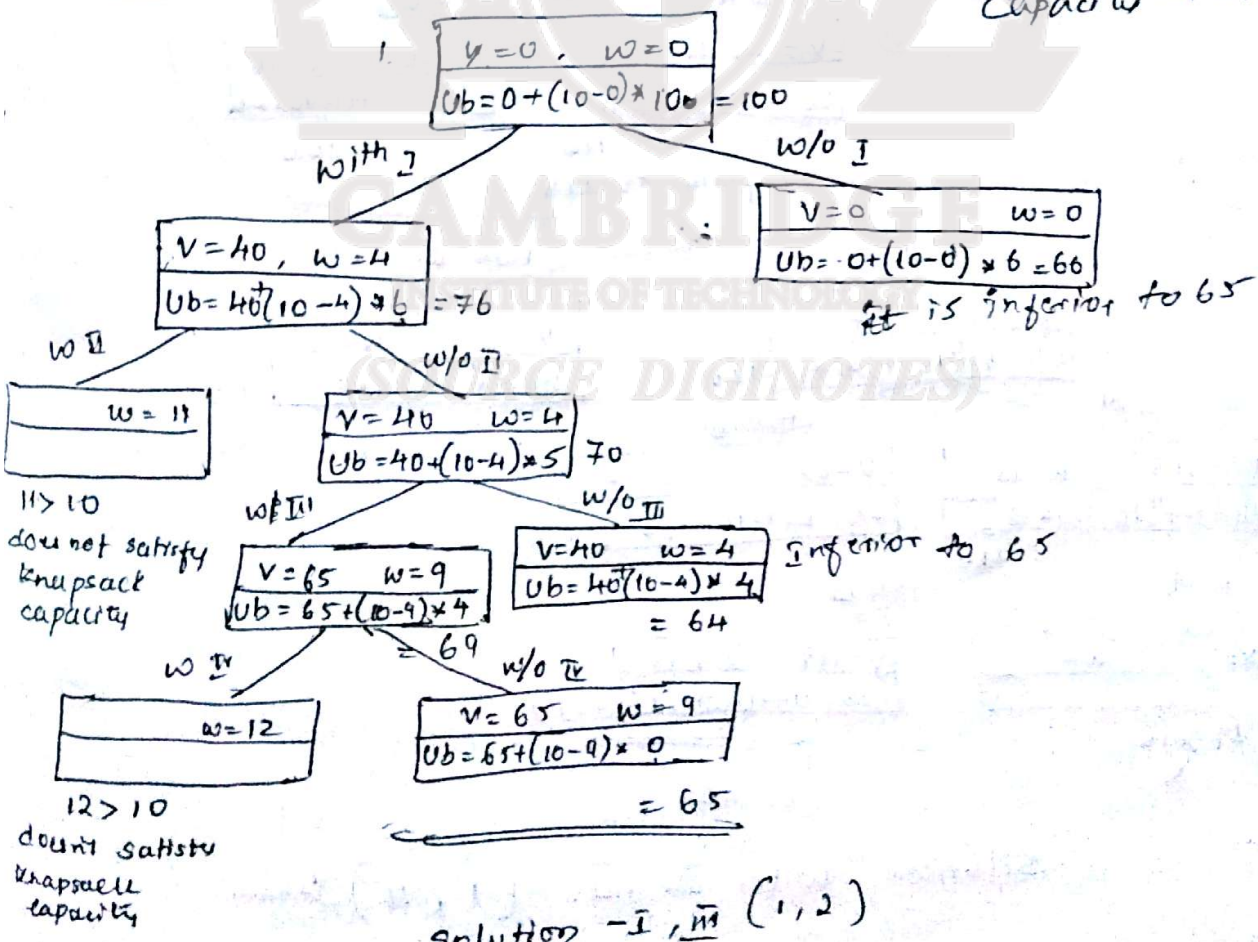
- v → Current value of Knapsack
- m → Capacity of Knapsack.
- w → occupied capacity in Knapsack.

1. Reorder items based value/weight ratio (Descending)
2. Calculate Ub based on equation.
3. Proceed with the branch which has larger upper bound.

Example: $m = 10$

Item	W weight	V value	v_i/w_i	
1	4	40	10	→ I
2	5	25	5	→ III
3	7	42	6	→ II
4	3	12	4	→ IV

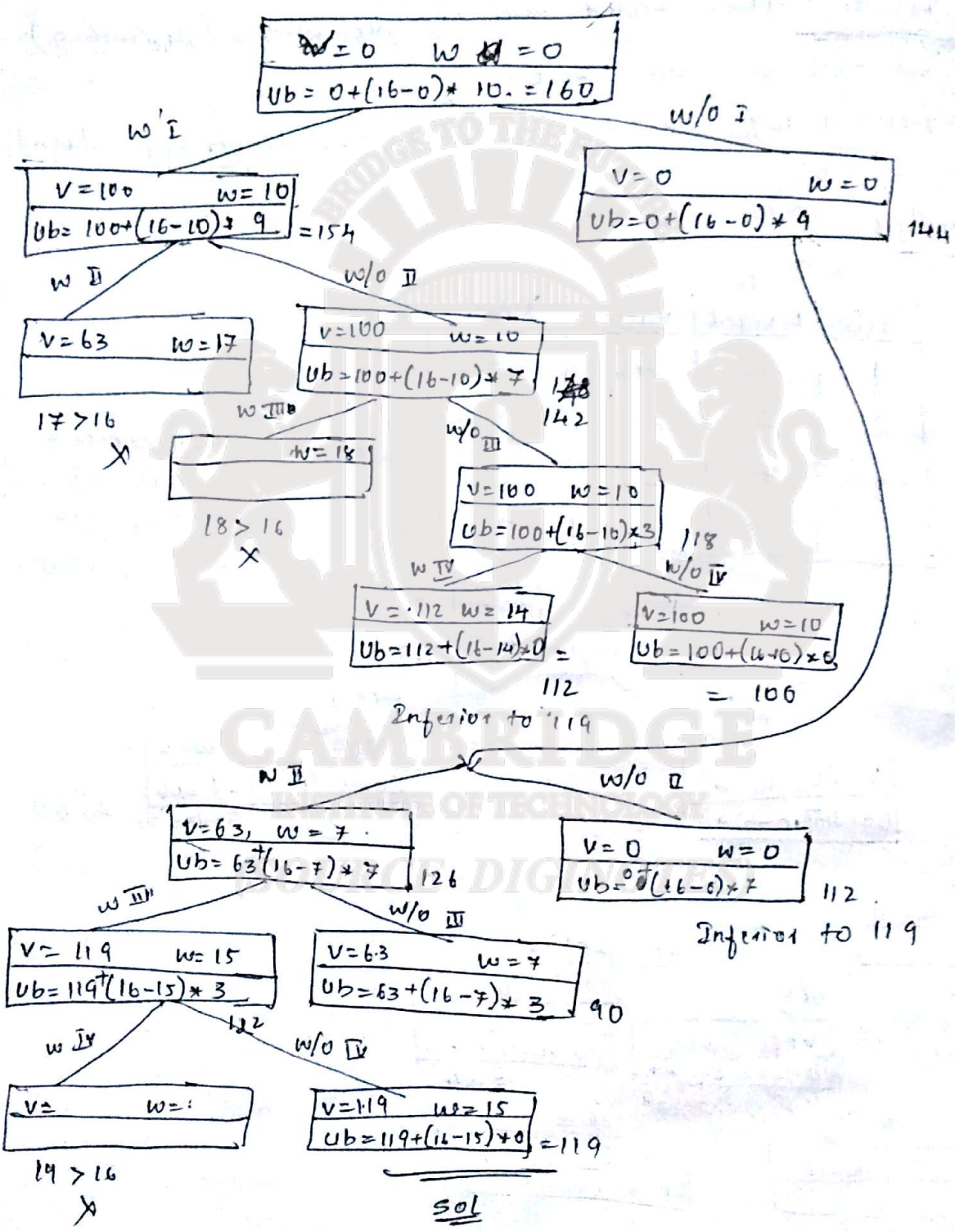
v → current value of Knapsack.
Capacity → m.



Solution - I, III (1, 2)

Example 2 . $m = 16$

item	weight	value	v_i/w_i	
1	8	56	7	IV
2	10	100	10	I
3	4	12	3	VI
4	7	63	9	II



Solution with III, II (1, 4) Items.