UNIT-V

BACKTRACKING

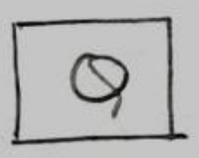
Terms:- ↳ generates solution in DFS manner.

- promising nodes:-
  - Nodes that can lead to a solution are calle promising nodes.
- non-promising nodes:-
  - Nodes that violate any of the constraints
- optional solution and non-promising leaf nodes.

N-QUEENS PROBLEM:-

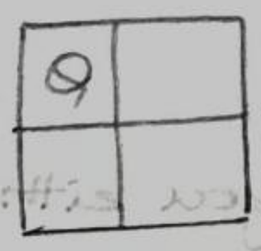
For given number of queens, say N, place the queens on a N X N chess board such that they donot attack each other. Therefore two queens can not be placed in the same row, column or diagonal.

For n = 1:-

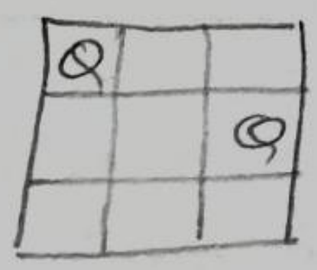


Trivial Solution

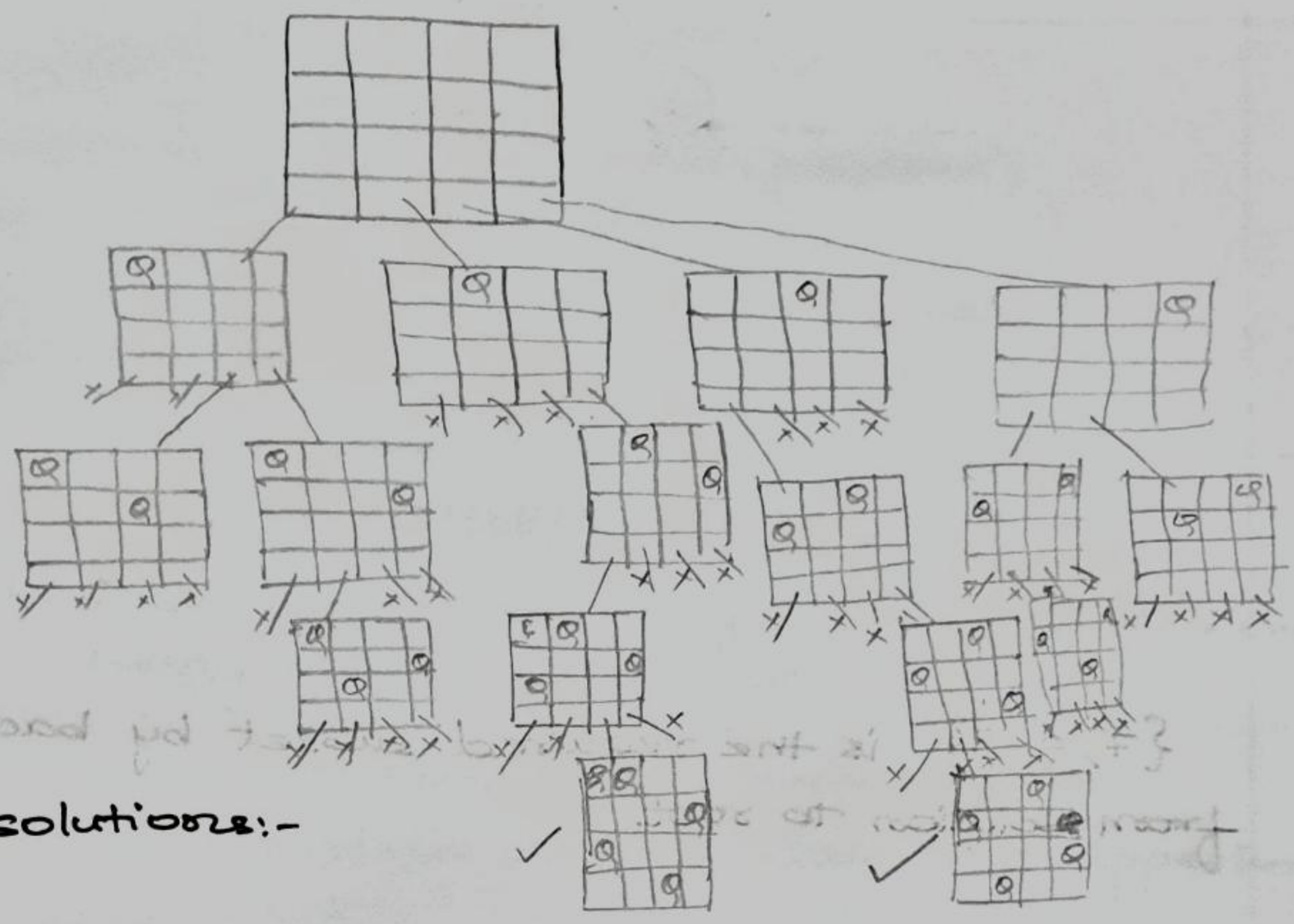
For n = 2 and n = 3:-



We can not place the second queen



For n = 4 :- State-space tree for 4-Queen



Two solutions:-

SUBSET SUM PROBLEM:-

Given a set of numbers and a sum value, find the possible solutions such that the subset of elements derived adds up to give the sum value.

1. $S = \{3, 5, 6, 7\}$, $d = 15$

The state space tree constructed will always be a binary tree.

Stopping condition:-

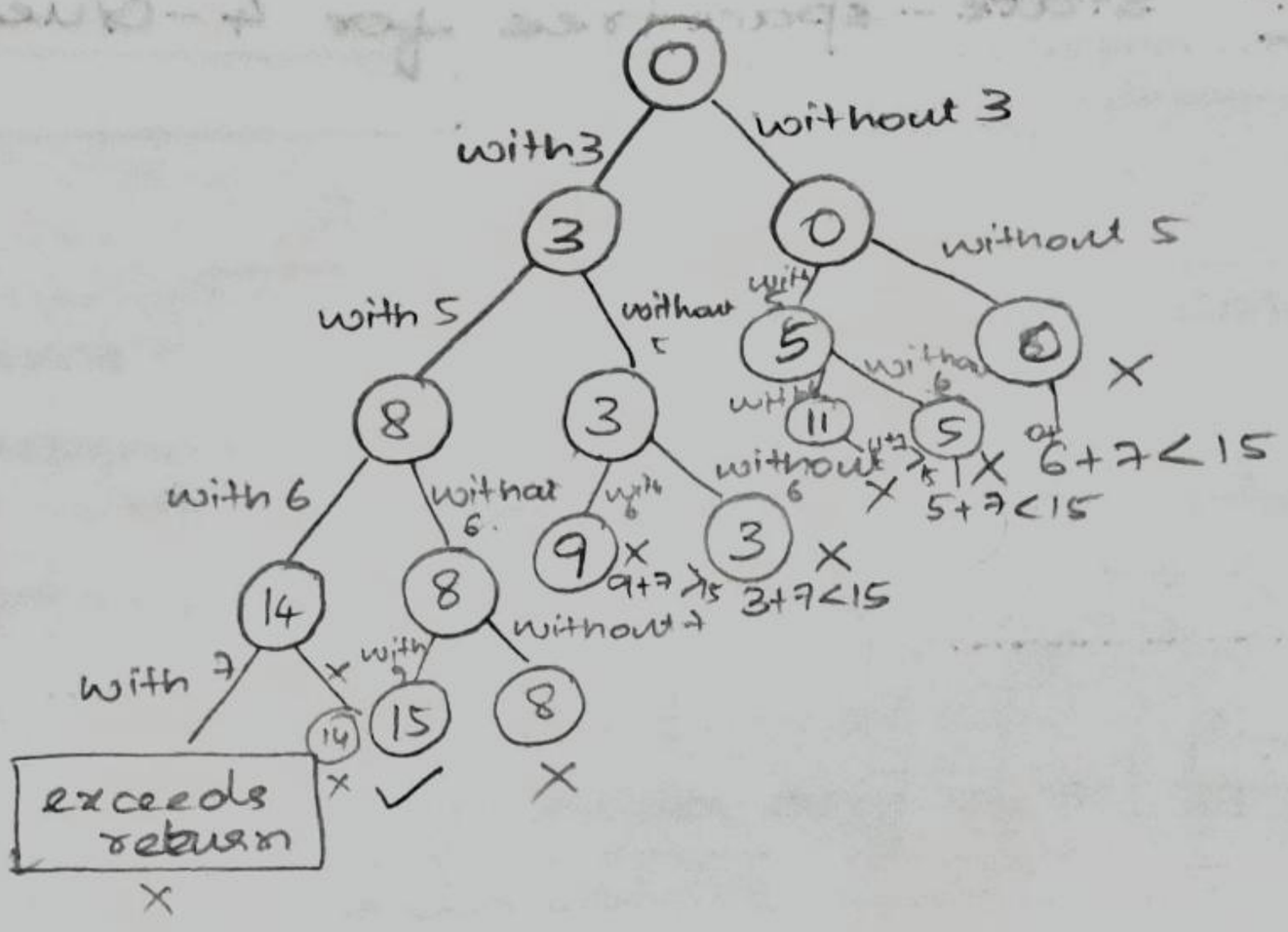
↳ when current sum value and next element to be added creates a larger sum that d.

i.e. $s + a_{i+1} > d$

↳ $s + \sum_{j=i+1}^{n} a_j < d$ when sum value is too small

The set elements should be sorted.

$S = \{3, 5, 6, 7\}$, $d = 15$

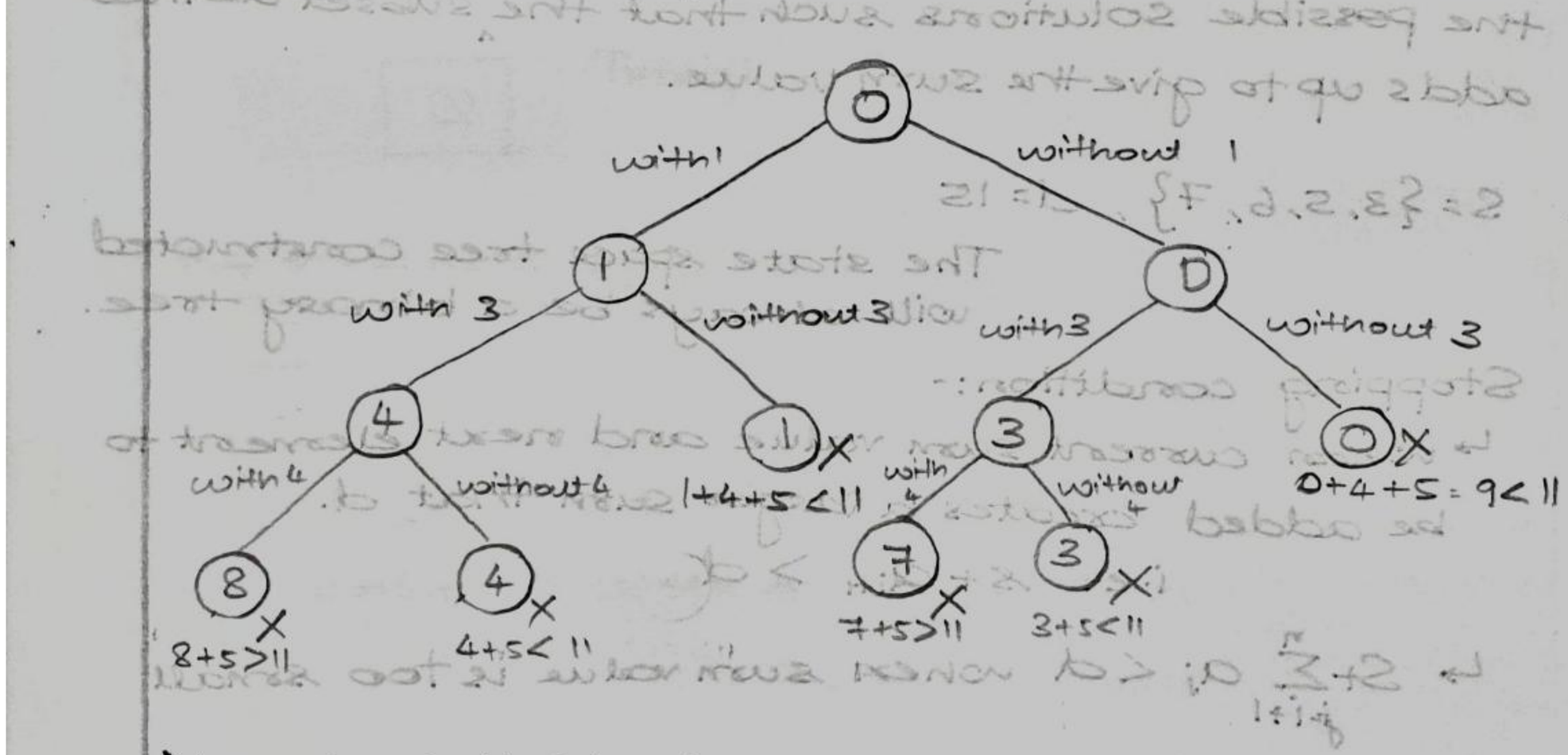


{7, 5, 3} is the required subset by backtracking from solution to root.

2. S = {3, 1, 5, 4} d = 11

arranging S in ascending order.

S = {1, 3, 4, 5}



No subset that gives the sum of 11.

BRANCH AND BOUND

Optimization problem:-

↳ maximization (knapsack problem)

↳ minimization

Solutions:-

→ feasible and optimal

KNAPSACK PROBLEM:-

↳ State space tree is generated in BFS manner using branch and bound.

we calculate upperbound for maximization.

For knapsack problem we calculate the upperbound using $\frac{\text{Value}}{\text{weight}}$ ratio.

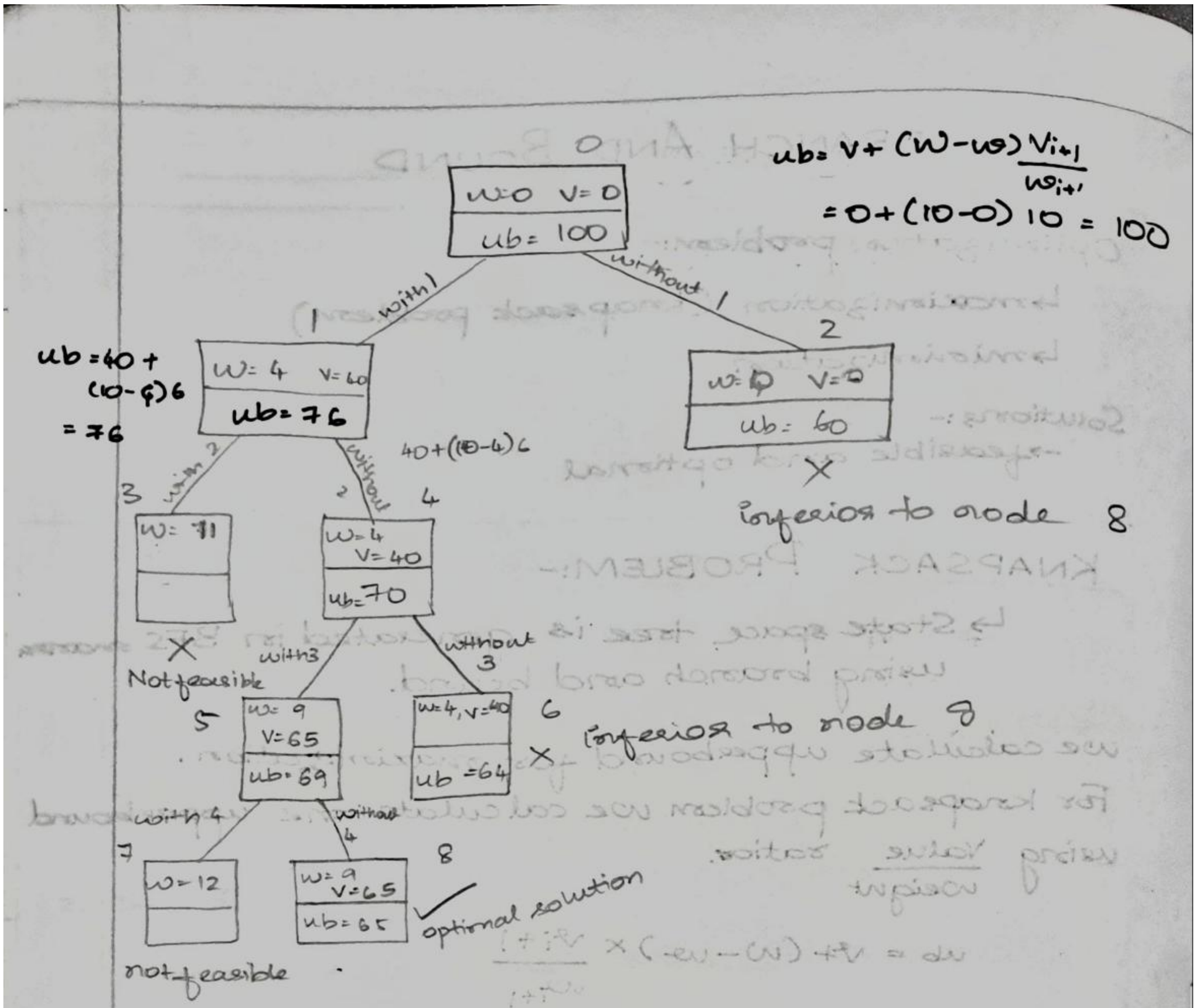
$$ub = v + (W - w) \times \frac{v_{i+1}}{w_{i+1}}$$

PROBLEM:- W=10

Item:-	weight	value	value/weigh
1	4	$40	10
2	7	$42	6
3	5	$25	5
4	3	$12	4

arrange in descending order

State space tree →



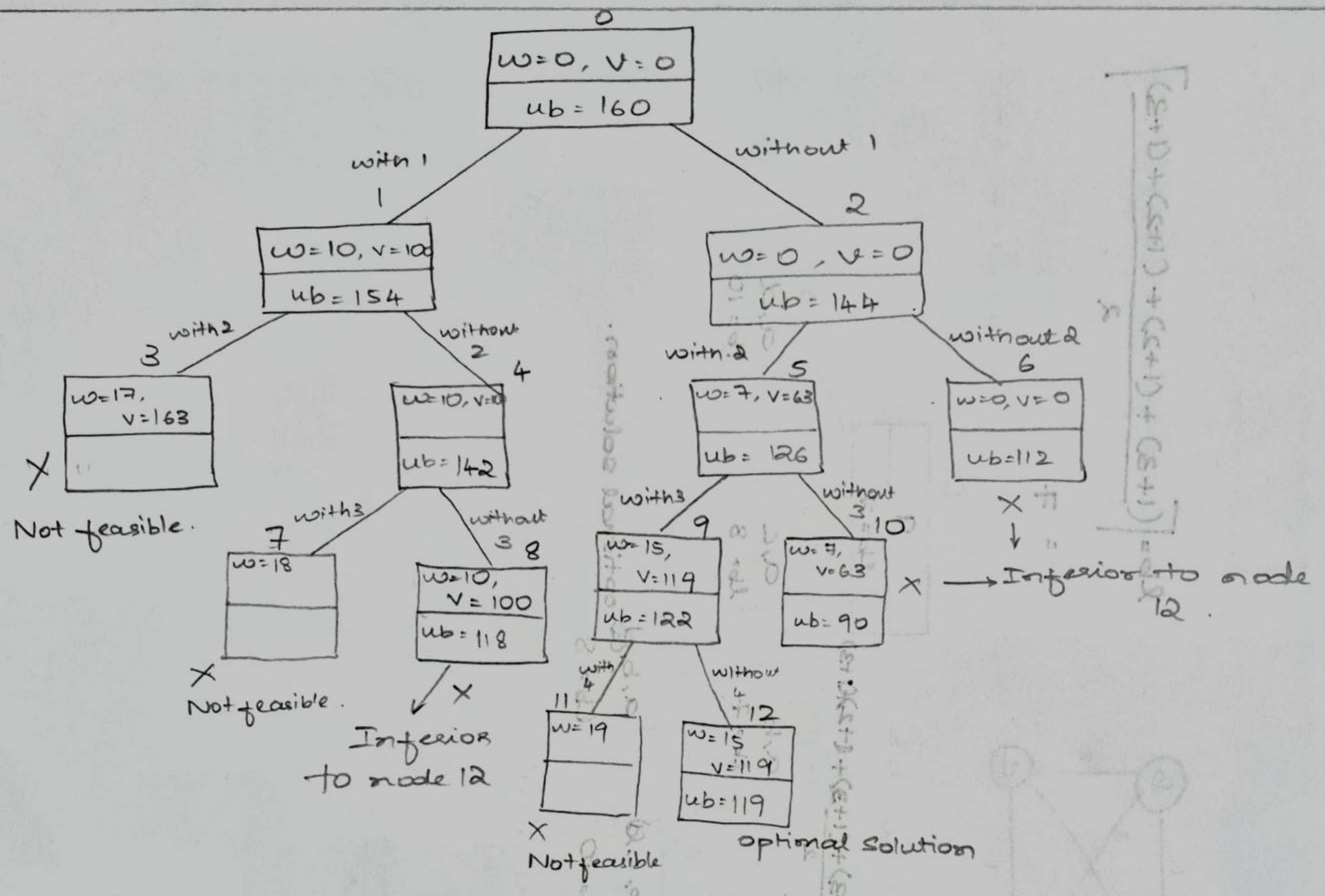
Item 1 and 3 are added, Total weight = 9

Total value = 65

2.

Item	weight	value	value/weight
1	10	100	10
2	7	63	9
3	8	56	7
4	4	12	3

W=16



∴ v = $119

s = {2, 3}

total weight = 15

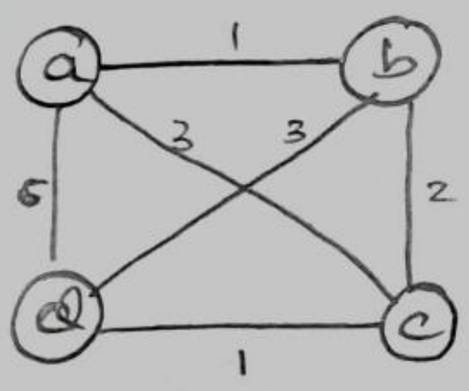
Item	weight	value	value/weight
1	10	100	10
2	7	63	9
3	8	56	7
4	4	12	3

TRAVELLING SALESMAN PROBLEM (TSP) :-

↳ ~~minimization~~ zation problem.

By using exhaustive search (brute force) complexity is in the order of $n!$ for a graph with n vertices.

1.



$$lb = \left\lceil \frac{(1+3)+(1+2)+(1+2)+(1+3)}{2} \right\rceil$$

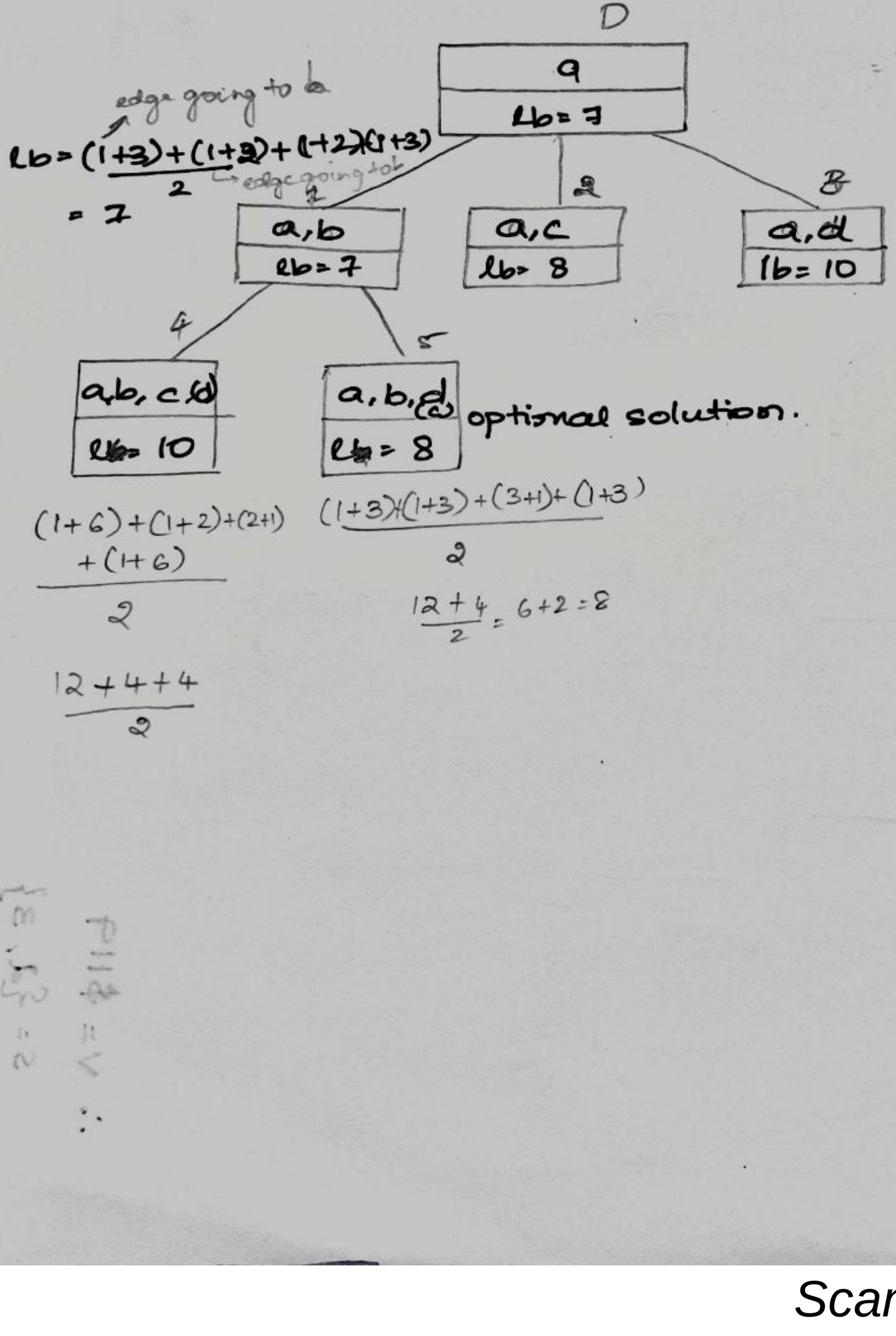
$= 7$

$$\frac{1+3+1+2+3+1+1+3}{2}$$

$= \lceil 15 \rceil = 8$

$$\frac{6+1+1+2+1+2+6+1}{2}$$

$$\frac{12+4+4}{2} = 10$$

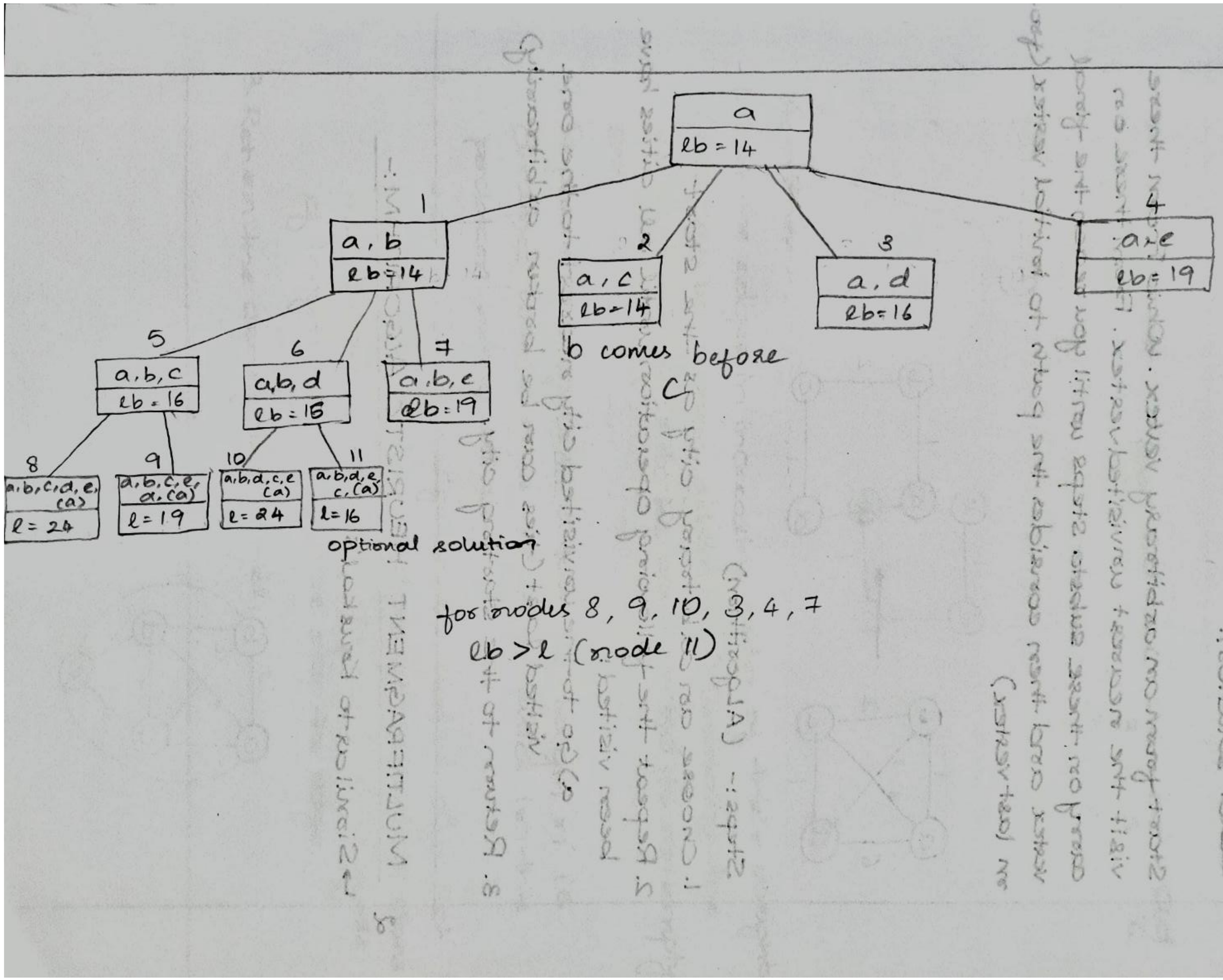


$$\frac{(1+6)+(1+2)+(2+1)+(1+6)}{2}$$

$$\frac{12+4+4}{2}$$

$$\frac{(1+3)+(1+3)+(3+1)+(1+3)}{2}$$

$$\frac{12+4}{2} = 6+2 = 8$$



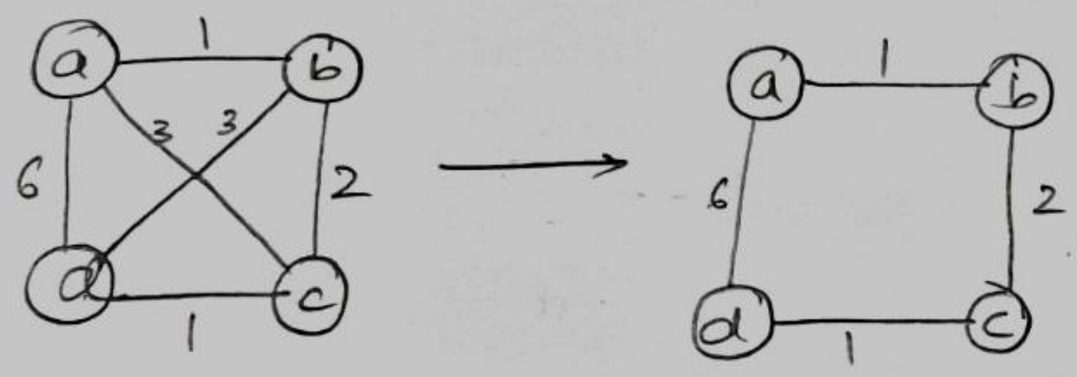
APPROXIMATION ALGORITHMS

1. NEAREST NEIGHBOR ALGORITHM

Follows greedy technique.

Note:-
With approximation algorithm we only get an approximate solution which need not be the exact solution.

Start from an arbitrary vertex. From there visit the nearest unvisited vertex. From there on carry on these steps until you reach the final vertex and then consider the path to initial vertex (from last vertex)

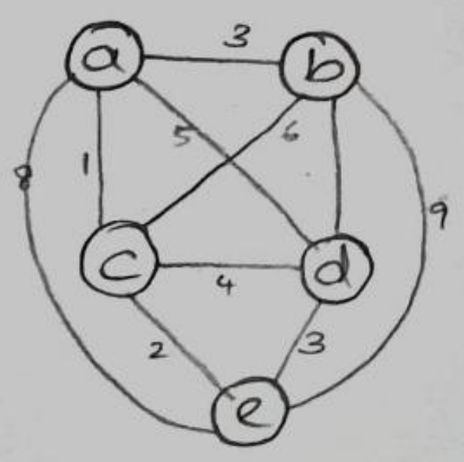


Steps:- (Algorithm)

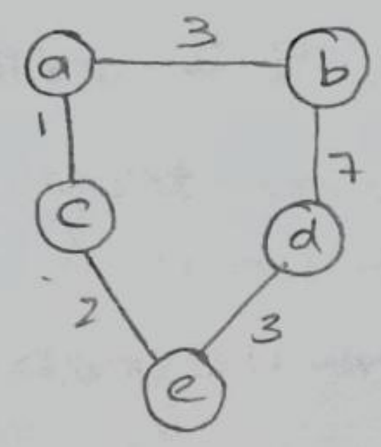
1. Choose an arbitrary city as the start
2. Repeat the following operation until all cities have been visited:
   a) Go to the unvisited city nearest to the one visited last (ties can be broken arbitrarily)
3. Return to the starting city.

2 MULTIFRAGMENT HEURISTIC ALGORITHM:-

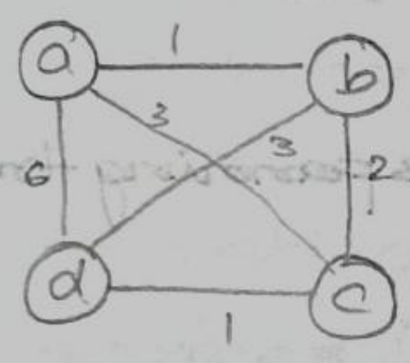
↳ Similar to Kruskal's



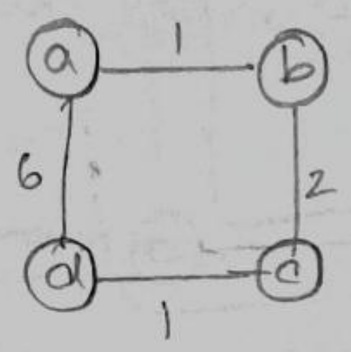
ac	ce	ed	ab	cd	ad	bc	bd	ae
1	2	3	3	4	5	6	7	8
✓	✓	✓	✓	×	×	×	✓	be 9



cost = 16



ab	cd	bc	ac	bd	ad
1	1	2	3	3	6
			×	×	



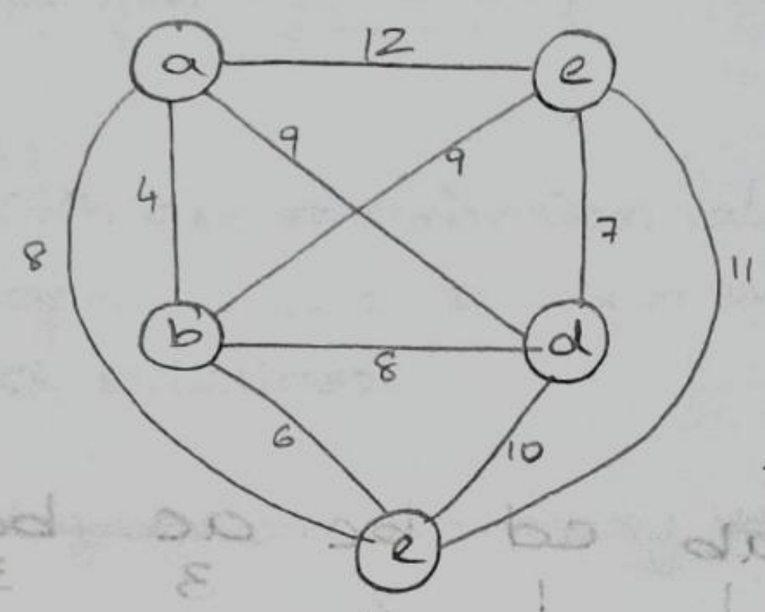
Algorithm:-

1. Sort the edges in increasing order of their weights (ties can be broken arbitrarily). Initialize the set of tour edges to be constructed to an empty set.
2. Repeat this step until a tour of length 'n' is obtained where n is the n of cities in the problem being solved:-
   ↳ add the next edge in the sorted edge list to the set of tour edges provides this does not create a vertex of degree 3 or a cycle of length < n. otherwise skip the edge.
3. Return the set of tour edges.

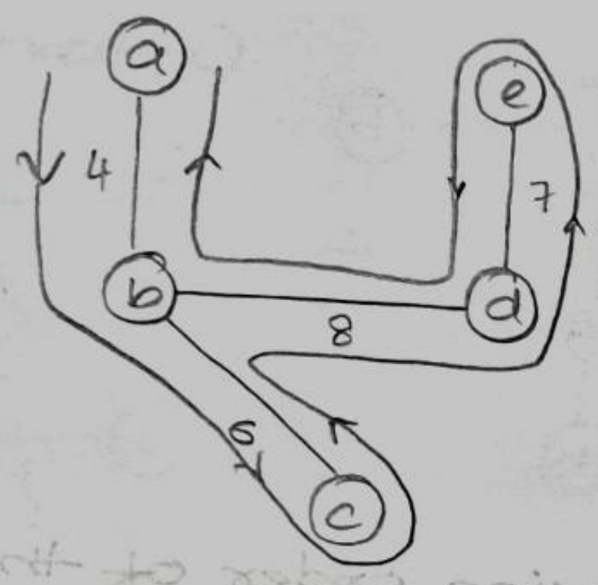
MINIMUM SPANNING TREE BASED ALGORITHM:-

1. Twice - around the tree algorithm.

1.



Using Kruskal's finding minimum spanning tree:-



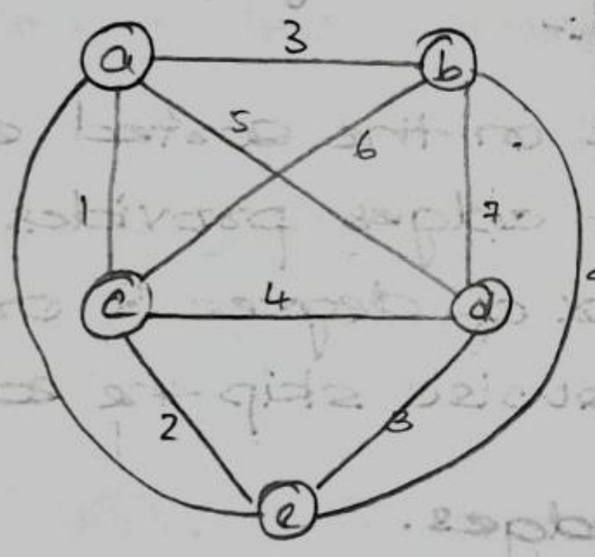
walk → a-b-c-~~b~~-d-e-~~d~~-~~b~~-a

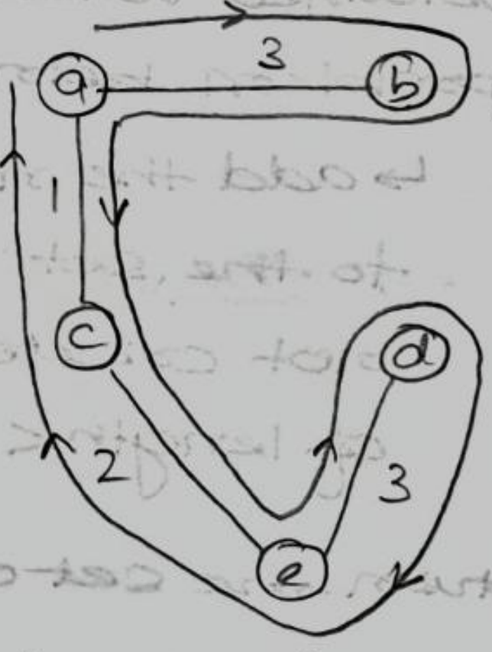
cancelling repeated vertices we get

a → b → c → d → e

total cost = 39

2.





walk → a → b → ~~c~~ → c → e → d → ~~e~~ → ~~c~~ → a

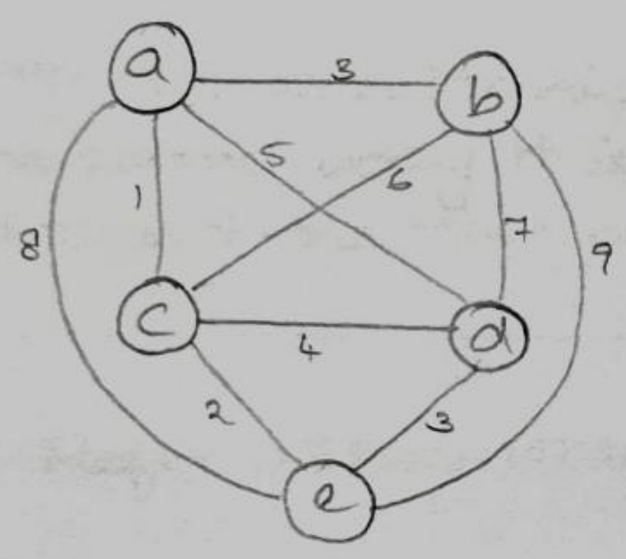
a → b → c → e → d → a

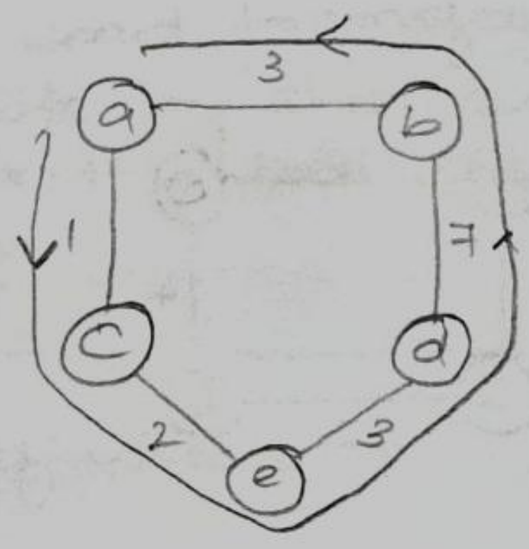
total cost = 19

Algorithm :-

1. Construct a minimum spanning tree of the graph corresponding to a given instance of the travelling S.P.
2. Starting at an arbitrary vertex perform a walk around the minimum spanning tree, recording all the vertices passed by.
3. Scan the vertex list obtained in step 2 and eliminate from it all repeated occurances of the same vertex except for the starting one at the starting of the list. This step is equivalent to making shortcuts. The vertices remaining form an hamiltonian circuit

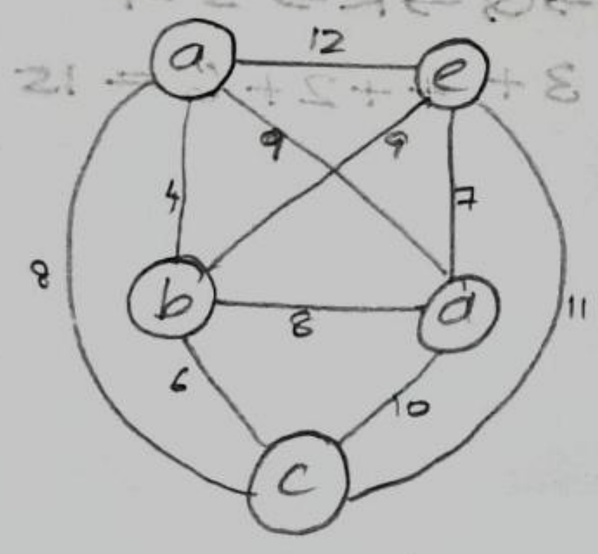
2. <u>Christofides Algorithm :-</u>

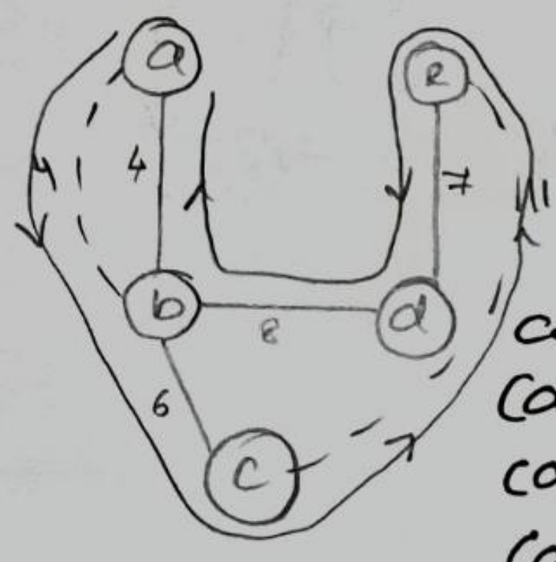




a → c → e → d → b → a

total cost = 16





vertices with odd degree

a, b, c, e

combinations:-

(a,b) (c,e) – 15

(a,c) (b,e) – 17

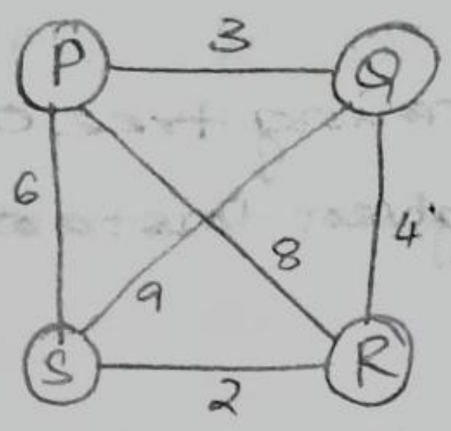
(a,e) (b,c) – 18

Consider lowest cost

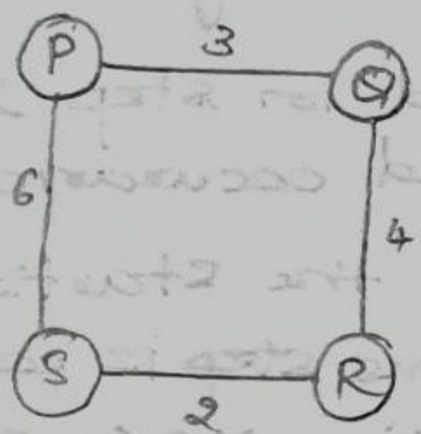
a → b → c → ~~e~~ → d → ~~b~~ → a

a → b → c → e → d → a

cost = 37



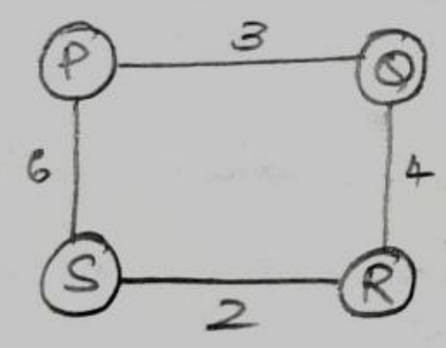
Nearest neighbour



cost = 15

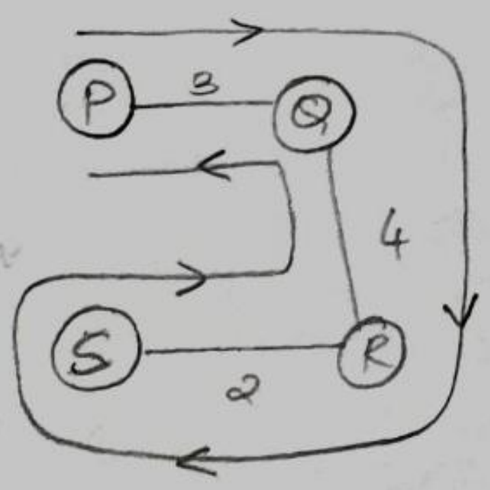
Multifragoneort heuristic

SR	PQ	QR	PS	PR	QS
2	3	4	6	8	9



cost = 15

Twice around the tree:-

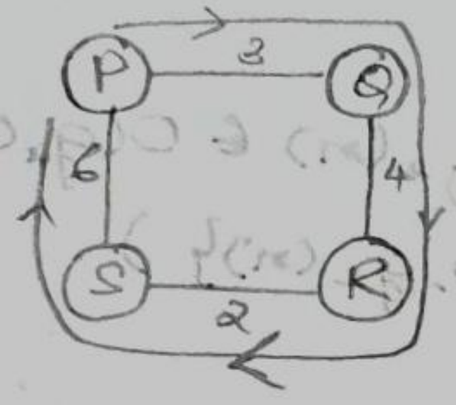


Walk → P → Q → R → S → ~~R~~ → ~~Q~~ → P

P → Q → R → S → P

3 + 4 + 2 + 6 = 15

Christofides :-



P → Q → R → S → P

cost = 15

Algorithm

This method also using minimum S.T. but is better than twice around the tree. It uses the concept of Eulerian circuit in a multigraph. A Eulerian circuit exists in a connected multigraph if an only if all its vertices have even degrees. The christofides algorithm obtains such a multigraph by adding to the graph the edges of a minimum weight matching of all odd degree vertices in its MST. Then the algorithm finds an Eulerian circuit in the multigraph and transforms it into a hamiltonian circuits by shortcuts exactly the same way it is done in the last step of twice around the tree algorithm.

Definitions of :-

- class P problems.
- class NP problems
- NP complete problems.
- non-deterministic algorithms.
- polynomially reducible problems.