Design and Analysis of Algorithms

Dr. Roshan Fernandes
Associate Professor
Dept. of CS&E
NMAMIT, Nitte

BACKTRACKING

Back Tracking.

- The main idea behind backtracking is to construct solutions one component at a time and evaluate such partially constructed candidates.

- If a partially constructed solution can be developed further without violating the problem's constraints, it is done by taking the first remaining legitimate option for the next component.

If there is no legitimate option for the next component, no alternatives for any remaining component need to be considered.

In this case, the algorithm backtracks to replace the last component of the partially constructed solution with its next option.

n - Queens Problem :

The problem is to place n queens on an n-by-n (n×n) chessboard so that no two queens attack each other by being in the same row or in the same column or on the same diagonal.

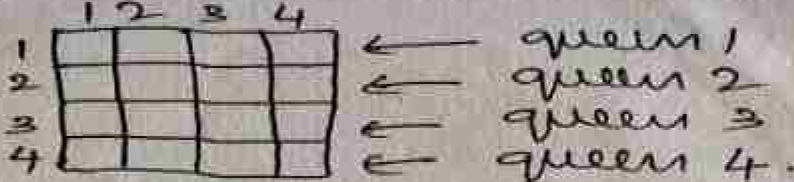
<u>Note</u> :

$n = 1 \rightarrow$ trivial soln.

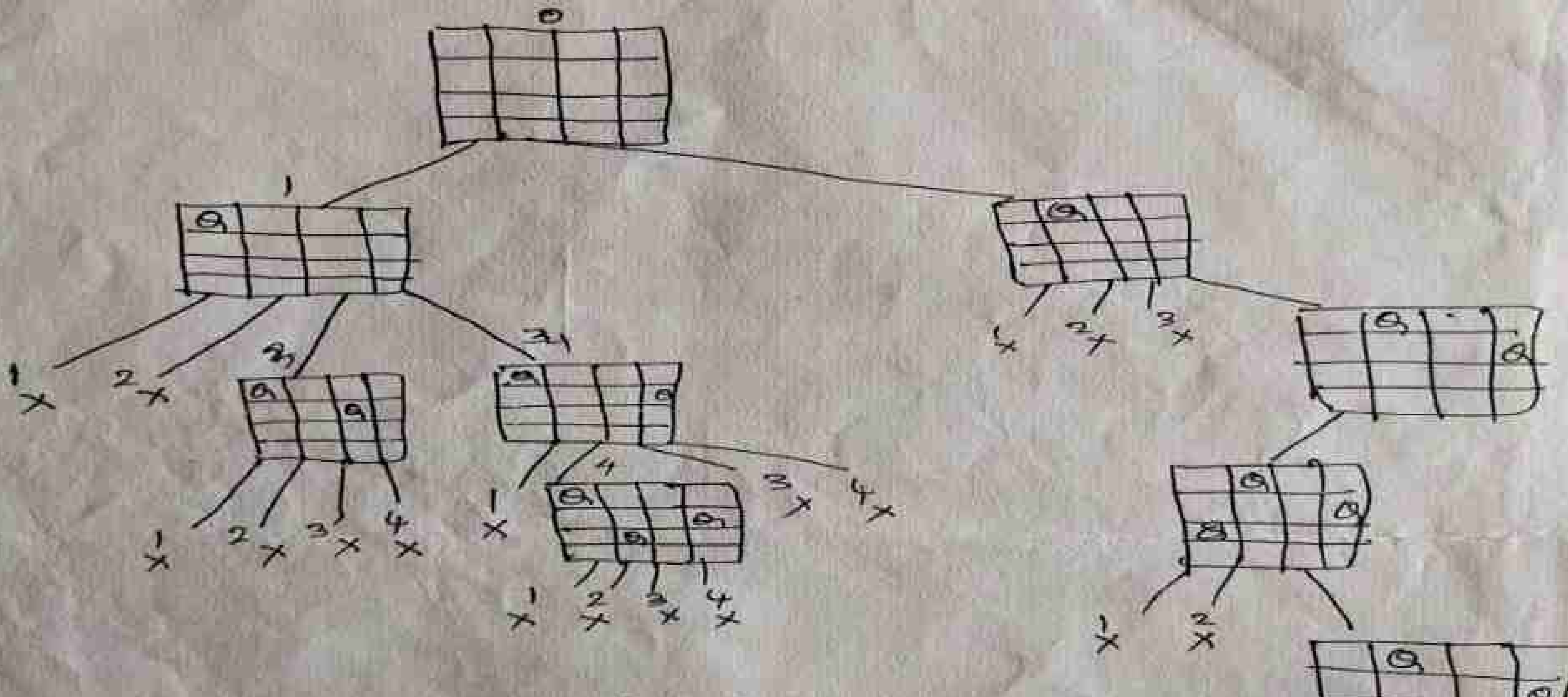
$n = 2, 3 \rightarrow$ No soln.

$n = 4$ there is a solution.

The state-space tree for 4-queens problem given below:

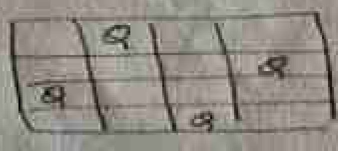


– board for the 4-queens problem –.



state-space tree of solving 4-queens problem.

2 solns ①



②

Implementation of n-Queens problem:

The array A is used to represent the Queen Numbers & its column position.

Initially array A is 0.

Ex:

	1	2	3	4
column No. →	0	0	0	0

→ Queen No (Row No)

Meaning Queen 1 placed in Row 1 & column 0.
" 2 " " " 2 & " 0.
" 3 " " " 3 & " 0.
" 4 " " " 4 & " 0.

```
main()
{ int n;
  read n.
  if (n <= 3  && n > 1)
       print No solution.
  else
       nqueen(n);
}
void  nqueen (int n)
{
   int  k = 1;     // Queen 1, in Row 1.
   int    i, count = 0;
   int    A[k] = 0;     // Queen 1, in Row 1, column 0.
                        // back track till k == 1.
   while (k != 0)
   {
       A[k] = A[k] + 1;   // Queen k, in next column.
       while (A[k] <= n  && (place(k) == 0))
        {
            A[k] ++;
        }
```

```
if (A[k] <= n)
{
    if (k == n)
    {
        printf ("soln No %d \n", ++count);
        for (i=1 ; i<=n ; i++)
        {
            printf ("%d Queen is placed on
                %d row & %d column \n",
                    i, i, A[i]) ;

        }
    }else
        {   k++ ;
            A[k] = 0 ;
        }

}
else
{   k-- ;
}
}
}.
```

```
int place (int k)
{
  int i;
  for (i=1; i<k ; i++)
  {
    if ((A(k) == A[i]) || (abs(i-k) == abs(A[i] - A[k])
      return 0;
  }
} return 1;
```

Subset - Sum Problem:

- Find a subset of a given set $S = \{s_1, \ldots, s_n\}$ of n positive integers whose sum is equal to a given +ve integer d.

Example: $S = \{1, 2, 5, 6, 8\}$ and $d = 9$.

There are 2 solutions: $\{1, 2, 6\}$ & $\{1, 8\}$.

- Some instances of this problem may have no solution.

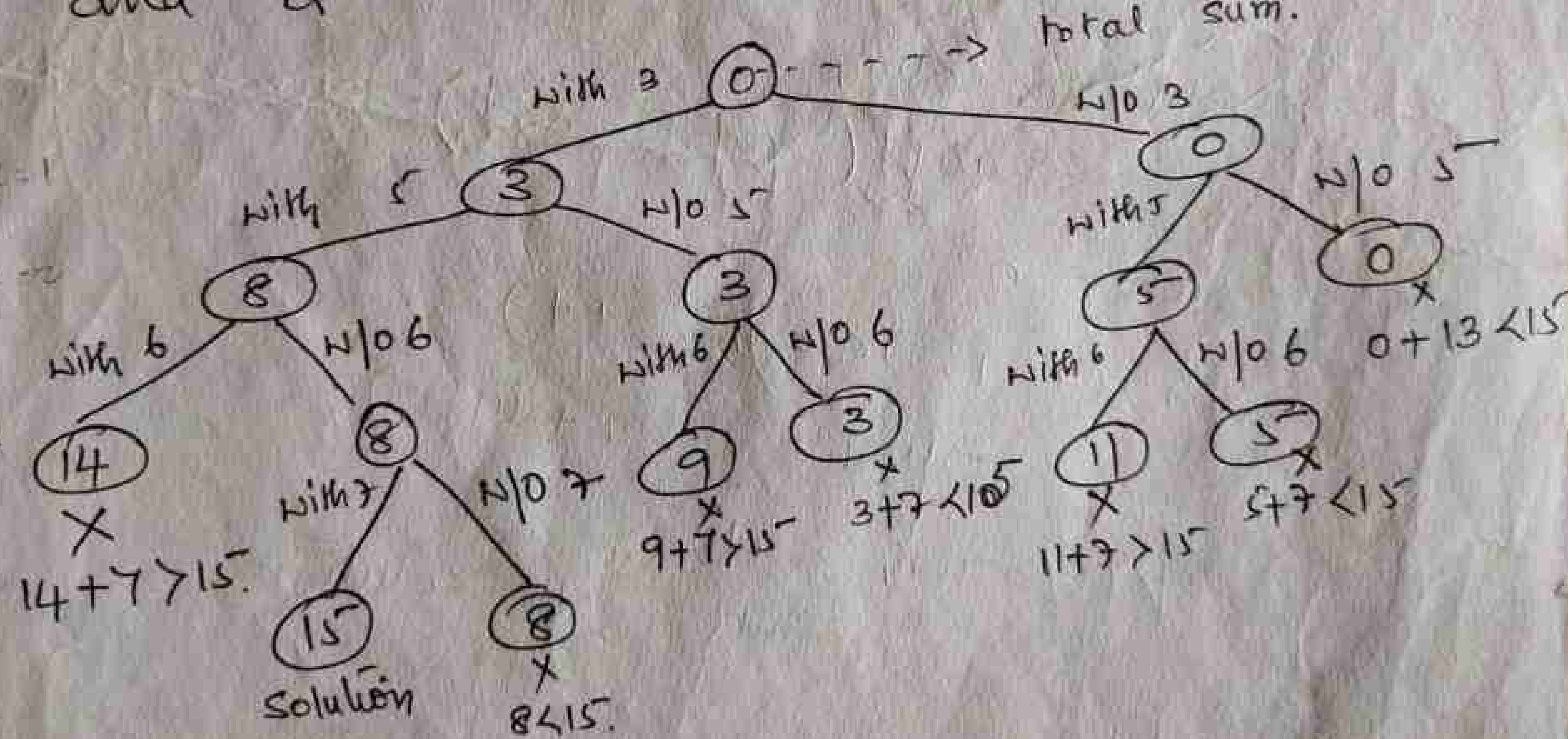
... in increasing order

<u>Note</u>; sort the set's elements in increasing order

$s_1 \leq s_2 \leq \cdots \leq s_n$.

state-space tree for the problem $S = \{3, 5, 6, 7\}$

and $d = 15$ is given below;



Implementation of Subset Sum problem:

Note: 1. Array W → set of elements in ascending order.

2. Array X → initially all 0, indicating None of the elements are added to the soln.
If i^{th} ~~item~~ item is added, corresponding $X[i]$ is made 1.

```
int n, x[20], w[20], count=0, d;
main()
{
    Read no. of elements n.
    for (i= 1; i<= n; i++)
        x[i] = 0;
    Enter elements in ascending order.
    for (i= 1; i<= n; i++)
        scanf("%d", &w[i]);
    Enter the sum.
        scanf("%d", &d);

    sum = 0;
    for (i=1; i<=n; i++)
        sum = sum + w[i];

    if (sum < d)
        print "No solution".

    else
        subsetsum(0, 1, sum);

}
```

```
void subsetsum(int s, int k, int r)
{
    int i;
    x[k] = 1;
    if (s + w[k] == d)
    {
        count++;
        printf(" soln No. %d\n", count);
        for (i=1; i<=k; i++)
        {      if (x[i] == 1)
                    printf("%d\t", w[i]);
        }
    }
    else
    {
        if (s+w[k]+w[k+1] <= d)
        {      subsetsum(s+w[k], k+1, r-w[k]);
        }
    }
    if ((s+(r- w[k])) >= d && (s+w[k+1]) <= d)
    {
        x[k] = 0;
        subsetsum(s, k+1, r-w[k]);
    }
}
```

s → ~~initial~~ sum so far

k → next ~~initial~~ item No.

r → remaining sum

(8) Roshan Fernandes.

Branch - and - Bound.

Compared to backtracking, branch-and-bound requires two additional items:

- A way to provide, for every node of a state-space tree, a bound on the best value of the objective function, on any solution that can be obtained by adding further components to the partial solution represented by the node.
- The value of the best solution seen so far

* This bound should be a lower bound for a minimization problem and an upper bound for a maximization problem.

We terminate a search path at the current node in a state-space tree of a branch-and-bound algorithm for any one of the following 3 reasons:

- The value of the node's bound is not better than the value of the best solution seen so far.
- The node represents no feasible solution because the constraints of the problem is/are completely violated.
- The subset of feasible solutions represented by the node consists of a single point & hence no further choices can be made.

Assignment Problem:

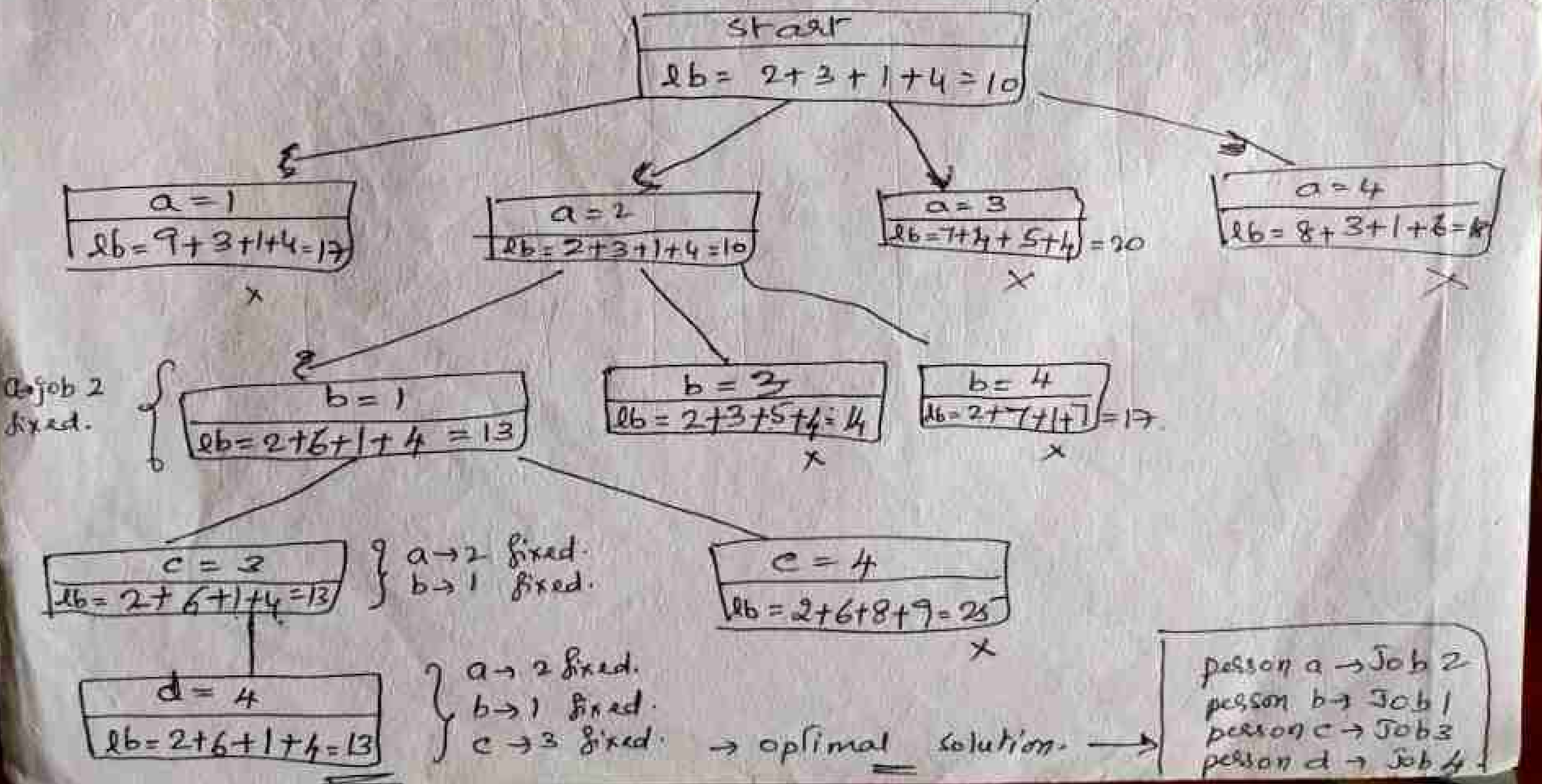
$$
C = \begin{array}{c} \\ \\ \\ \\ \\ \end{array}
\begin{array}{cccc}
\text{Job1} & \text{Job2} & \text{Job3} & \text{Job4} \\
\end{array}
$$

$$
C = \begin{bmatrix} 9 & 2 & 7 & 8 \\ 6 & 4 & 3 & 7 \\ 5 & 8 & 1 & 8 \\ 7 & 6 & 9 & 4 \end{bmatrix}
\begin{array}{l} \text{Person a} \\ \text{Person b} \\ \text{Person c} \\ \text{Person d.} \end{array}
$$

- Calculate the lower bound. So the optimal soln can not be less than lower bound.
- Lower bound is calculated by taking minimum value in each Row.

i.e. $2 + \underbrace{3 + 1}_{\text{same column.}} + 4 = \underline{\underline{10}}$.

<u>Note</u>: In the above example observe that min. values taken 3 & 1 are from same column which is not permitted for solving the assignment problem. But to calculate lower bound, it is fine.



Knapsack Problem;

- is a maximization problem so calculate the upper bound.

Calculating upper bound (ub) to knapsack problem;

$$\boxed{ub = v + (W - w) \cdot \left(v_{i+1} / w_{i+1} \right)}$$

Add to v (the total value of the items already selected), the product of the remaining capacity of the knapsack $W - w$ and the best per unit payoff among the remaining items, which is v_{i+1} / w_{i+1}.

Example;

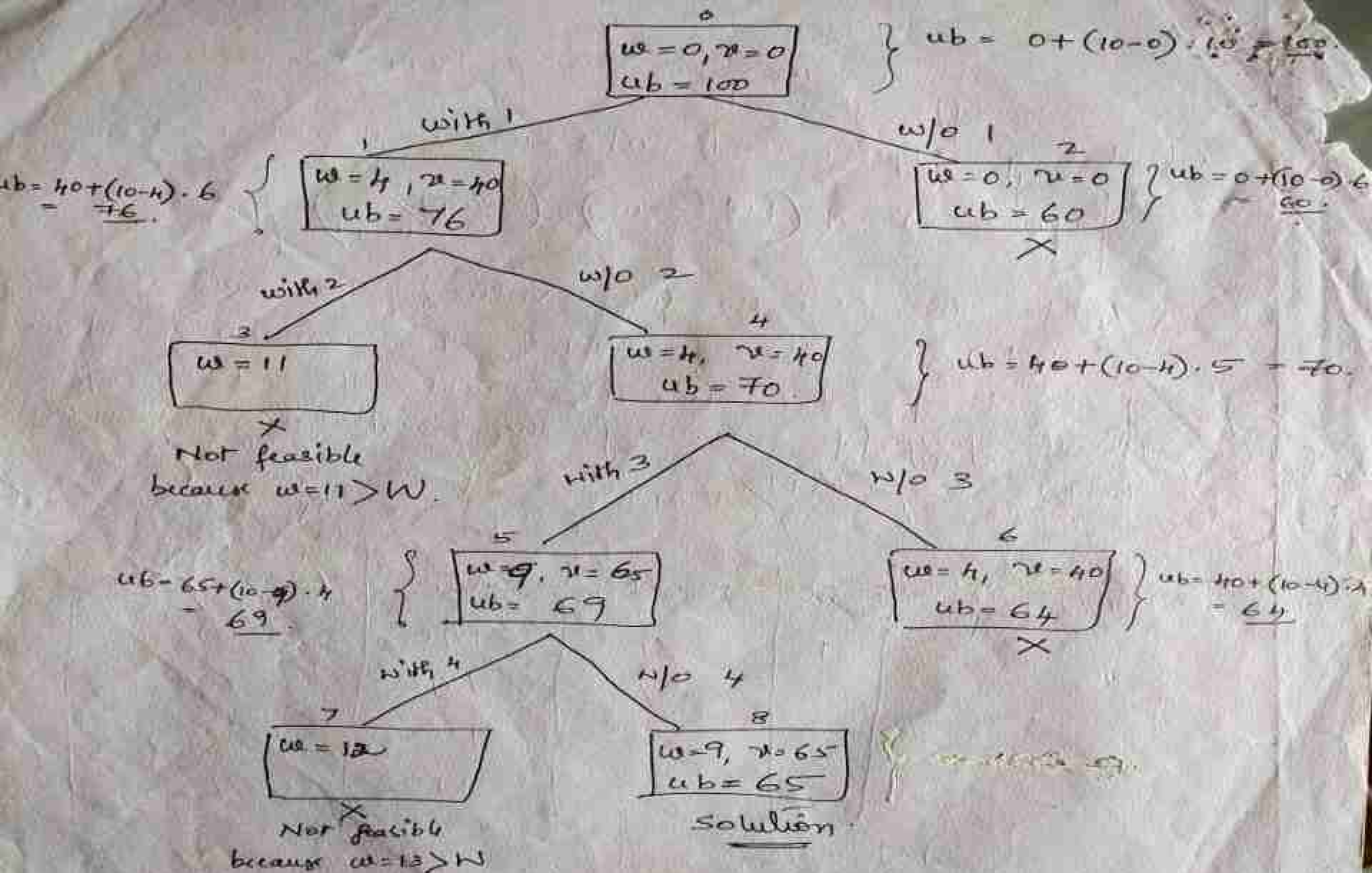
item	weight	Value	value/weight
1	4	$40	10
2	7	$42	6
3	5	$25	5
4	3	$12	4

40/4
42/7
25/5
12/3

W = 10.

This will be given in the problem.

we have to calculate this column

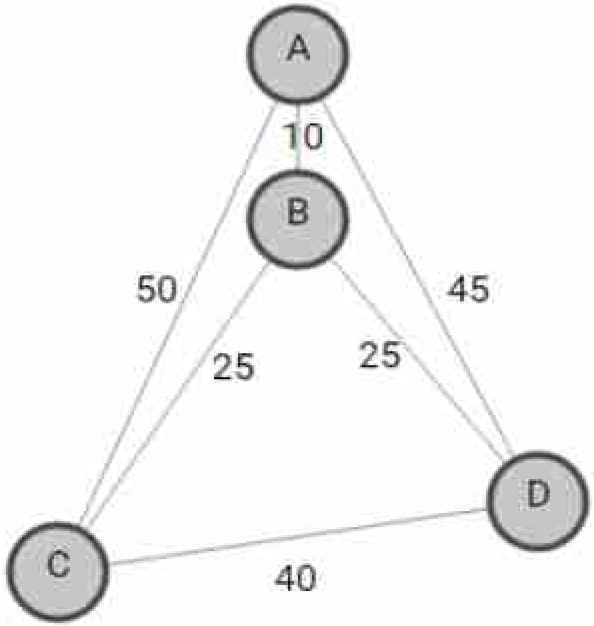


Note: At node 8, as there are no additional items to consider, the upper bound is simply equal to the total value added so far. i.e. Since $v = 65$, $ub = 65$.

Traveling Salesman Problem

Given a set of cities and the distance between every pair of cities, the problem is to find the shortest possible route that visits every city exactly once and returns to the starting point.

For example, consider the following graph. A TSP tour in the graph is `A -> B -> C -> D -> B -> A`. The cost of the tour is `10 + 25 + 40 + 25 + 10 = 100`.

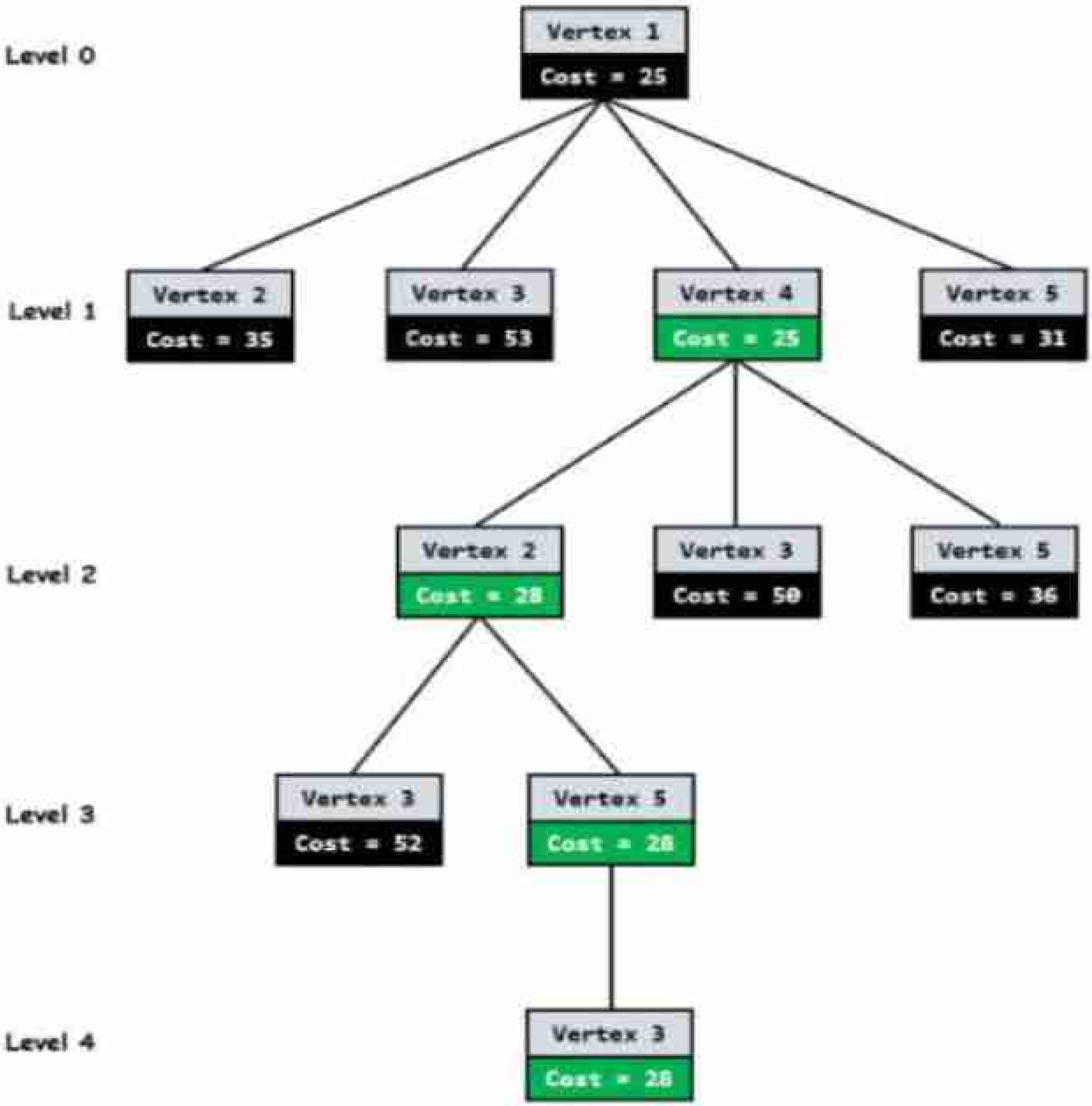


```
C(i, j) = W(i, j), if there is a direct path from Ci to Cj
        = INFINITY, if there is no direct path from Ci to Cj
```

For example, consider the following cost matrix `M`,

```
     C0   C1   C2  C3  C4
C0  INF   20   30  10  11
C1  15   INF   16   4   2
C2  3      5  INF   2   4
C3  19     6   18 INF   3
C4  16     4    7  16 INF
```

Following is the state-space tree for the above TSP problem, which shows the optimal solution marked in green:



Now, how do we calculate the lower bound of the path starting at any node?

In general, to get the lower bound of the path starting from the node, we reduce each row and column so that there must be at least one zero in each row and Column. We need to reduce the minimum value from each element in each row and column.

Let's start from the root node.

We reduce the minimum value in each row from each element in that row. The minimum in each row of cost matrix `M` is marked by blue `[10 2 2 3 4]` below.

```
INF  20  30  10  11
15  INF  16   4   2
 3    5 INF   2   4
19    6  18 INF   3
16    4   7  16 INF
```

After reducing the row, we get the below reduced matrix.

```
INF  10  20   0   1
 13 INF  14   2   0
  1   3 INF   0   2
 16   3  15 INF   0
 12   0   3  12 INF
```

We then reduce the minimum value in each column from each element in that column. Minimum in each column is marked by blue `[1 0 3 0 0]`. After reducing the column, we get below the reduced matrix. This matrix will be further processed by child nodes of the root node to calculate their lower bound.

```
INF  10  17   0   1
 12 INF  11   2   0
  0   3 INF   0   2
 15   3  12 INF   0
 11   0   0  12 INF
```

The total expected cost at the root node is the sum of all reductions.

```
Cost = [10 2 2 3 4] + [1 0 3 0 0] = 25
```

Since we have discovered the root node `C0`, the next node to be expanded can be any node from `C1`, `C2`, `C3`, `C4`. Whichever node has a minimum cost will be expanded further. So, we have to find out the expanding cost of each node.

The parent node `C0` has below reduced matrix:

```
INF  10  17   0   1
 12 INF  11   2   0
  0   3 INF   0   2
 15   3  12 INF   0
 11   0   0  12 INF
```

Let's consider an edge from `0 -> 1`.

1. As we add an edge `(0, 1)` to our search space, set outgoing edges for city `0` to `INFINITY` and all incoming edges to city `1` to `INFINITY`. We also set `(1, 0)` to `INFINITY`.

So in a reduced matrix of the parent node, change all the elements in row `0` and column `1` and at index `(1, 0)` to `INFINITY` (marked in red).

```
INF  10  17   0   1
 12 INF  11   2   0
  0   3 INF   0   2
 15   3  12 INF   0
 11   0   0  12 INF
```

The resulting cost matrix is:

```
INF INF INF INF INF
INF INF  11   2   0
  0 INF INF   0   2
 15 INF  12 INF   0
 11 INF   0  12 INF
```

2. We try to calculate the lower bound of the path starting at node `1` using the above resulting cost matrix. The lower bound is `0` as the matrix is already in reduced form, i.e., all rows and all columns have zero value.

Therefore, for node 1, the cost will be:

```
Cost = cost of node 0 +
        cost of the edge(0, 1) +
        lower bound of the path starting at node 1
    = 25 + 10 + 0 = 35
```

Let's consider an edge from `0 -> 2`

1. Change all the elements in row `0` and column `2` and at index `(2, 0)` to `INFINITY` (marked in red).

```
INF  10  17   0   1
 12 INF  11   2   0
  0   3 INF   0   2
 15   3  12 INF   0
 11   0   0  12 INF
```

The resulting cost matrix is:

```
INF INF  INF  INF  INF
12  INF  INF    2    0
INF  3   INF    0    2
15   3   INF  INF    0
11   0   INF   12  INF
```

2. Now calculate the lower bound of the path starting at node 2 using the approach discussed earlier. The resultant matrix will be:

```
INF INF INF INF INF
  1 INF INF   2   0
INF   3 INF   0   2
  4   3 INF INF   0
  0   0 INF  12 INF
```

Therefore, for node 2, the cost will be

```
Cost = cost of node 0 +
       cost of the edge(0, 2) +
       lower bound of the path starting at node 2
     = 25 + 17 + 11 = 53
```

Let's consider an edge from `0 -> 3`.

1. Change all the elements in row `0` and column `3` and at index `(3, 0)` to `INFINITY` (marked in red).

```
INF  10  17   0   1
 12 INF  11   2   0
  0   3 INF   0   2
 15   3  12 INF   0
 11   0   0  12 INF
```

The resulting cost matrix is:

```
INF  INF  INF INF  INF
 12  INF  11  INF   0
  0   3  INF  INF   2
 INF  3   12  INF   0
 11   0    0  INF  INF
```

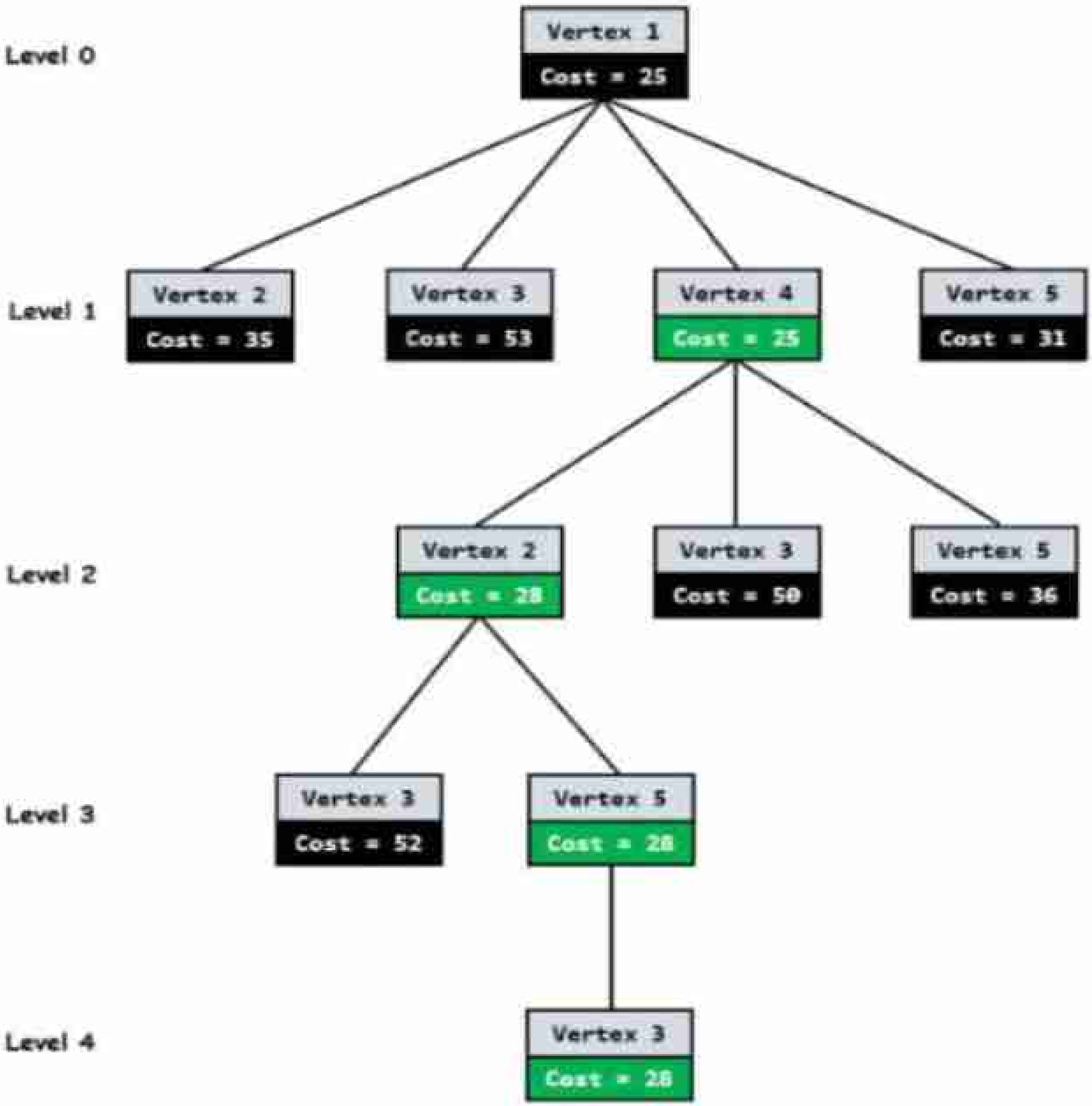
2. Now calculate the lower bound of the path starting at node `3` using the approach discussed earlier. The lower bound of the path starting at node `3` is `0` as it is already in reduced form, i.e., all rows and all columns have zero value.

Therefore, for node 3, the cost will be

```
Cost = cost of node 0 +
       cost of the edge(0, 3) +
       lower bound of the path starting at node 3
     = 25 + 0 + 0 = 25
```

Similarly, we calculate the cost of `0 -> 4`. Its cost will be `31`.

Following is the state-space tree for the above TSP problem, which shows the optimal solution marked in green:



Now find a live node with the least estimated cost. Live nodes 1 , 2 , 3 , and 4 have costs 35 , 53 , 25 , and 31 , respectively. The minimum among them is node 3 , having cost 25 . So, node 3 will be expanded further, as shown in the state-space tree diagram. After adding its children to the list of live nodes, find a live node with the least cost and expand it. Continue the search till a leaf is encountered in the space search tree. If a leaf is encountered, then the tour is completed, and we will return to the root node.

Initial Cost Matrix :

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

Reduce the matrix by finding & subtracting min in row & then in column.

$$\begin{bmatrix} \infty & 10 & 20 & 0 & 1 \\ 13 & \infty & 14 & 2 & 0 \\ 1 & 3 & \infty & 0 & 2 \\ 16 & 3 & 15 & \infty & 0 \\ 12 & 0 & 3 & 12 & \infty \end{bmatrix}$$

$$\begin{array}{ccccc} 1 & 0 & 3 & 0 & 0 \end{array}$$

Now reduce column.

$$\begin{bmatrix} \infty & 10 & 17 & 0 & 1 \\ 12 & \infty & 11 & 2 & 0 \\ 0 & 3 & \infty & 0 & 2 \\ 15 & 3 & 12 & \infty & 0 \\ 11 & 0 & 0 & 12 & \infty \end{bmatrix}$$

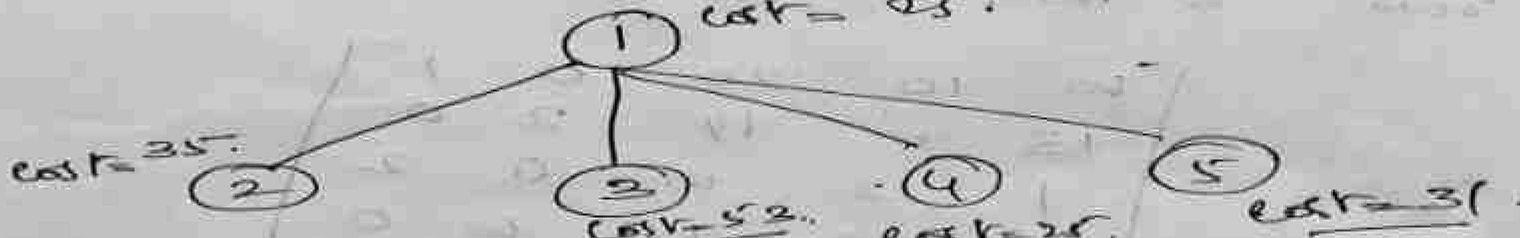
Cost of reduce initially $= (10+2+2+3+4) + (1+0+3+0+0)$

$= \underline{\underline{25}}$.

Reduced matrix at 1

$$\begin{bmatrix} \infty & 10 & 17 & 0 & 1 \\ 12 & \infty & 11 & 2 & 0 \\ 0 & 3 & \infty & 0 & 2 \\ 15 & 3 & 12 & \infty & 0 \\ 11 & 0 & 0 & 12 & \infty \end{bmatrix} \text{ ——— } (1)$$

with reduce cost at Node 1 = 25



Consider 1 → 2 :

Consider the reduced matrix (eq 1).

make 1st row ∞ & 2nd row ∞

& 2 → 1 → ∞ .

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ \\ \end{array}\begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 11 & 2 & 0 \\ 0 & \infty & \infty & 0 & 2 \\ 15 & \infty & 12 & \infty & 0 \\ 11 & \infty & 0 & 12 & \infty \end{bmatrix} \\ \begin{array}{ccccc} 0 & & 0 & 0 & 0 \end{array} \end{array}\begin{array}{c} \\ \\ - 0 \\ - 0 \\ - 0 \\ - 0 \\ \\ \end{array}$$

reduced matrix at 2 with R.C. = 0.

(already reduced & reduce cost = 0.

cost at 2 = reduce cost at ① + C(1,2) + reduce cost at ②

= 25 + 10 + 0 = 35.

Consider 1 → 3.

For matrix ① make row 1 ∞ & col 3 ∞ & 3 → 1 ∞.

$$
\begin{array}{c|ccccc|l}
 & 1 & 2 & 3 & 4 & 5 & \\
\hline
1 & \infty & \infty & \infty & \infty & \infty & \to 0 \\
2 & 12 & \infty & \infty & 2 & 0 & \to 0 \\
3 & \infty & 3 & \infty & 0 & 2 & \to 0 \\
4 & 15 & 3 & \infty & \infty & 0 & \to 0 \\
5 & 11 & 0 & \infty & 12 & \infty & \to 0 \\
\hline
 & 11 & 0 & & 0 & 0 &
\end{array}
$$

cost = 0+11 = 11.

Reduce it:

$$
\begin{array}{c|ccccc|}
 & 1 & 2 & 3 & 4 & 5 \\
\hline
1 & \infty & \infty & \infty & \infty & \infty \\
2 & 01 & \infty & \infty & 2 & 0 \\
3 & \infty & 3 & \infty & 0 & 2 \\
4 & 04 & 3 & \infty & \infty & 0 \\
5 & 0 & 0 & \infty & 12 & \infty
\end{array}
$$

Reduced matrix ③ at with R.C. = 11.

cost at 1 → 3. = 25 + 17 + 11 = 53.

Consider $1 \to 4$.

For matrix (1) make row 1 & col 4 $\to \infty$
& $4 \to 1$ as ∞.

$$
\begin{array}{c}
\\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ \\
\end{array}
\begin{array}{c}
\begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\
\left[\begin{array}{ccccc}
\infty & \infty & \infty & \infty & \infty \\
12 & \infty & 11 & \infty & 0 \\
0 & 3 & \infty & \infty & 2 \\
\infty & 3 & 12 & \infty & 0 \\
11 & 0 & 0 & \infty & \infty
\end{array}\right] \\
\begin{array}{ccccc} 0 & 0 & 0 & & 0 \end{array}
\end{array}
\begin{array}{c}
\\ \\ 0 \\ 0 \\ 0 \\ 0 \\ \\
\end{array}
$$

reduced matrix at (4) with r.c. = 0.

cost at 4 = 25 + 0 + 0 = 25

Consider 1 — 5
For matrix (1) make row 1 = ∞ & col 5 ∞
& 5 — 1 as ∞.

$$
\begin{array}{c}
\\ 1 \\ 2 \\ 3 \\ 4 \\ 5
\end{array}
\begin{array}{c}
\begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\
\left[\begin{array}{ccccc}
\infty & \infty & \infty & \infty & \infty \\
12 & \infty & 11 & 2 & \infty \\
0 & 3 & \infty & 0 & \infty \\
15 & 3 & 12 & \infty & \infty \\
\infty & 0 & 0 & 12 & \infty
\end{array}\right]
\end{array}
\begin{array}{c}
\\ \\ 2 \\ 0 \\ 3 \\ 0
\end{array}
$$

reduce:

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

$\xrightarrow{\text{reduced}}$ matrix at (5) with r.c. = 5

reduction cost = $\underline{5}$.

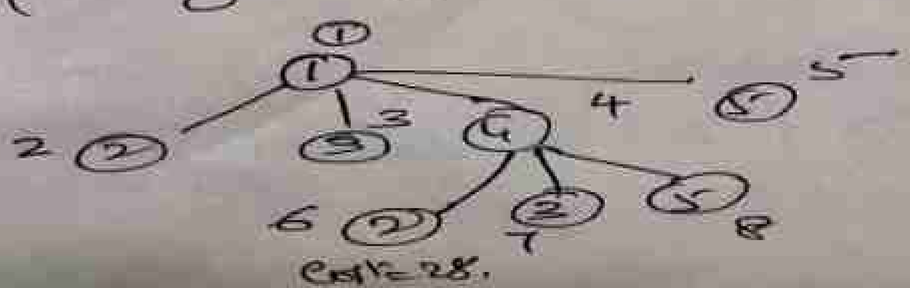
cost at 5 = 25 + 1 + 5 = $\underline{31}$.

Among 2, 3, 4, 5 min is (4).

So consider the reduction matrix of (4) with r.c. = 0.

$$\begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & \infty & \infty & \infty & \infty & \infty \\ 2 & 12 & \infty & 11 & \infty & 0 \\ 3 & 0 & 3 & \infty & \infty & 2 \\ 4 & \infty & 3 & 12 & \infty & 0 \\ 5 & 11 & 0 & 0 & \infty & \infty \end{array}$$

$\xrightarrow{\text{copy of}}$ matrix (4)



Consider $4 \to 2$. make 4^{th} row ∞

2^{nd} col. ∞ & 2 to 1 as ∞.

$$
\begin{array}{c}
1 \\ 2 \\ 3 \\ 4 \\ 5 \\ \\
\end{array}
\begin{array}{c}
\left[\begin{array}{ccccc}
\infty & \infty & \infty & \infty & \infty \\
\infty & \infty & 11 & \infty & 0 \\
0 & \infty & \infty & \infty & 2 \\
\infty & \infty & \infty & \infty & \infty \\
11 & \infty & 0 & \infty & \infty
\end{array}\right] \\
\begin{array}{ccccc}
0 & & 0 & & 0
\end{array}
\end{array}
\begin{array}{c}
\\ 0 \\ 0 \\ \\ 0 \\ \\
\end{array}
$$

reduced matrix at (2). (6)

with r.c. = 0.

cost $1 \to 4 \to 2$ = cost at 4 + $c(4,2)$ + red. cost at 2.

$= 25 + 3 + 0 = 28$

— x —

$4 \to 3 \quad \to \text{cost} = 50$.

$4 \to 5 \quad \to \text{cost} = 36$..

$1 \to 4 \to 2 \to 3$

make row 2 → ∞ Col 3 → ∞ for matrix ⑥

& 3 → 1 → ∞

$$
\begin{array}{c|ccccc|c}
 & 1 & 2 & 3 & 4 & 5 & \\
\hline
1 & \infty & \infty & \infty & \infty & \infty & \\
2 & \infty & \infty & \infty & \infty & 2 & 2 \\
3 & \infty & \infty & \infty & \infty & \infty & \\
4 & \infty & \infty & \infty & \infty & \infty & 11 \\
5 & 11 & \infty & \infty & \infty & \infty & \\
\end{array}
$$

$$
\begin{array}{c|ccccc|c}
1 & \infty & \infty & \infty & \infty & \infty & \\
2 & \infty & \infty & \infty & \infty & 0 & 0 \\
3 & \infty & \infty & \infty & \infty & \infty & \\
4 & \infty & \infty & \infty & \infty & \infty & 0 \\
5 & 0 & \infty & \infty & \infty & \infty & \\
\hline
 & 0 & & & & 0 & \\
\end{array}
$$

⑨ reduced matrix at vertex 3

reduction cost = $\underline{13}$.

cost = cost at node vertex 2 + c(2, 3) + r cost

= 28 + 11 + 13 = $\underline{52}$.

$1 \to 4 \to 2 \to 5$

Make row 2 & col 5 ∞ & 5 (6) 1 ∞

for matrix (6)

$$\begin{array}{c|ccccc|c}
 & 1 & 2 & 3 & 4 & 5 & \\
\hline
1 & \infty & \infty & \infty & \infty & \infty & \\
2 & \infty & \infty & \infty & \infty & \infty & \\
3 & 0 & \infty & \infty & \infty & \infty & 0 \\
4 & \infty & \infty & \infty & \infty & \infty & \\
5 & \infty & \infty & 0 & \infty & \infty & 0 \\
\hline
 & 0 & & 0 & & &
\end{array}$$

10

reduced matrix at vertex (5)

cost at (5) = 28 + 0 + 0 = 28.

Last → $1 \to 4 \to 2 \to 5 \to 3$

5 to 3 is 0.

So final answer = 28 + 0 + 0.

= 28.

Thank You!!!