# Computer Engineering - Sem IV

# NCMPC 41 : Design and Analysis of Algorithms

## Module - 5 : Backtracking & Branch and Bound (05 Hours)

Instructor : Mrs. Lifna C S

*Department of Computer Engineering, VESIT, Mumbai*

# Topics to be covered

- Backtracking Method
    - N-queen problem
    - Sum of subsets
    - Graph coloring
- Branch and Bound Method
    - 15 Puzzle problem
    - Traveling Salesperson problem

# Backtracking Method

- **Systematic trial-and-error** strategy used to solve **constraint satisfaction problems**. It builds the solution **incrementally**, and if a partial solution **violates constraints**, it **backtracks** (undoes the last step) and tries an alternative.

- Backtracking follows the **depth-first search (DFS)** paradigm:
  1. **Choose**: Make a choice from the available options.
  2. **Explore**: Recurse with the choice included.
  3. **Unchoose (Backtrack)**: If it leads to a dead-end, undo the choice and try the next option.

- Backtracking is used for problems that require:
  - Finding **all solutions**
  - Finding **any one solution**
  - Finding the **optimal solution** (sometimes, combined with bounding)

# Backtracking Method

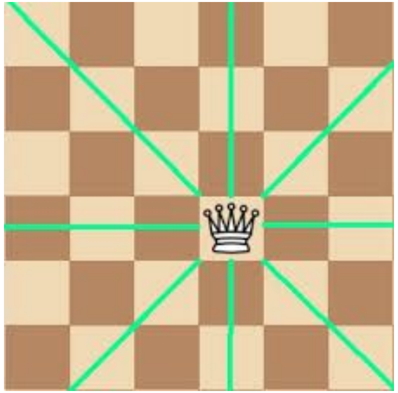
| Problem Type | Examples |
| --- | --- |
| Constraint Satisfaction Problems | N-Queens, Sudoku, Graph Coloring |
| Combinatorial Problems | Subsets, Permutations, Combinations |
| Optimization Problems | Knapsack, TSP (with pruning/bounding) |
| Puzzle Solving | Maze problems, Crossword filling |

# Topics to be covered

- Backtracking Method
  - N-queen problem
  - Sum of subsets
  - Graph coloring
- Branch and Bound Method
  - 15 Puzzle problem
  - Traveling Salesperson problem

# Backtracking Method - N Queens Problem

- Classic problem in computer science and artificial intelligence.
- Goal : Place **N queens on an N×N chessboard** such that **no two queens attack each other**.
    - No two queens can be in the same **row**
    - No two queens can be in the same **column**
    - No two queens can be on the same **diagonal**



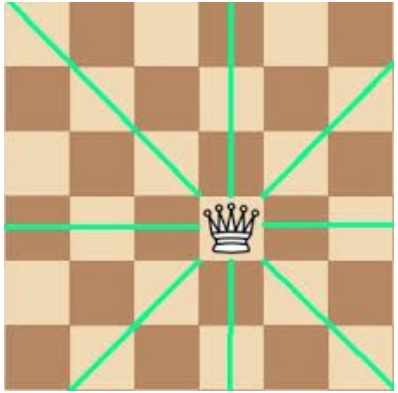
**Backtracking**

- General algorithm for finding all (or some) solutions to computational problems by incrementally building candidates and abandoning them ("backtrack") if they fail to satisfy the constraints.

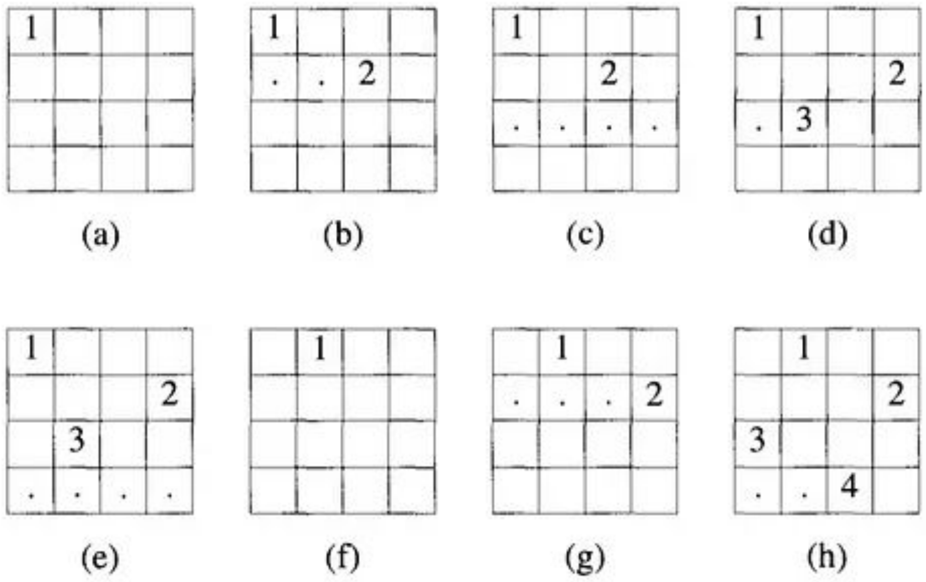
# Backtracking Method - N Queens Problem

**Approach**

- Place queens row by row.
- At each row, try placing a queen in each column.
- Check for validity (no attacks from previously placed queens).
- If valid, move to the next row.
- If not valid or stuck, **backtrack** and try a different position.



# Backtracking Method - 4 Queens Problem with Solution



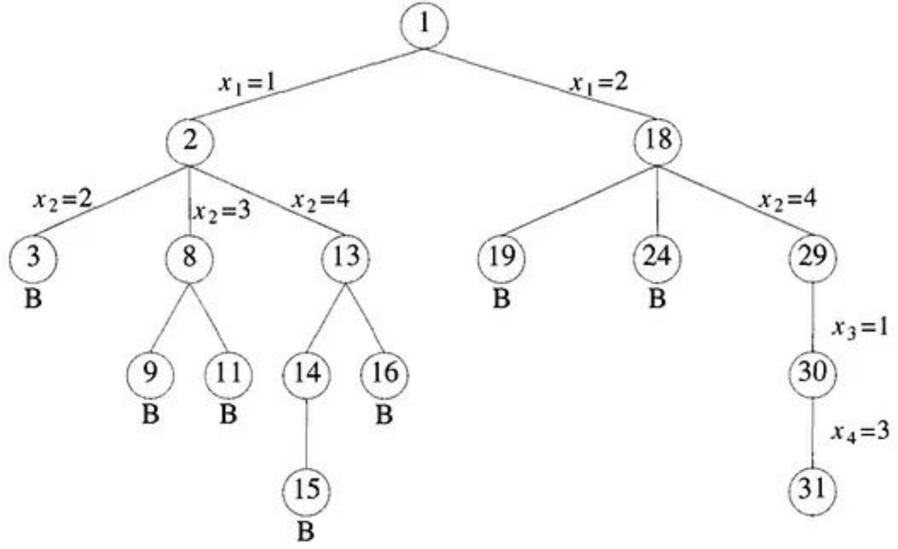
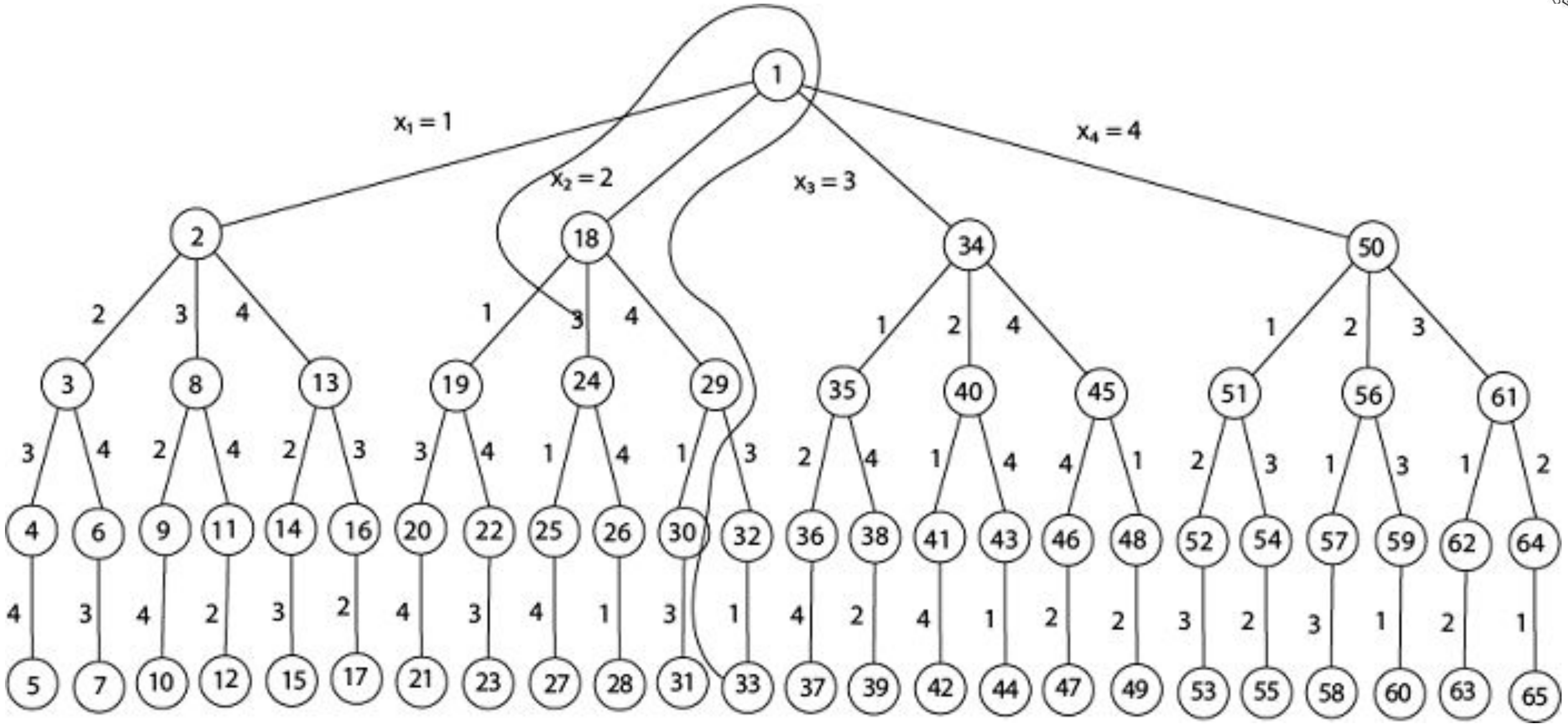



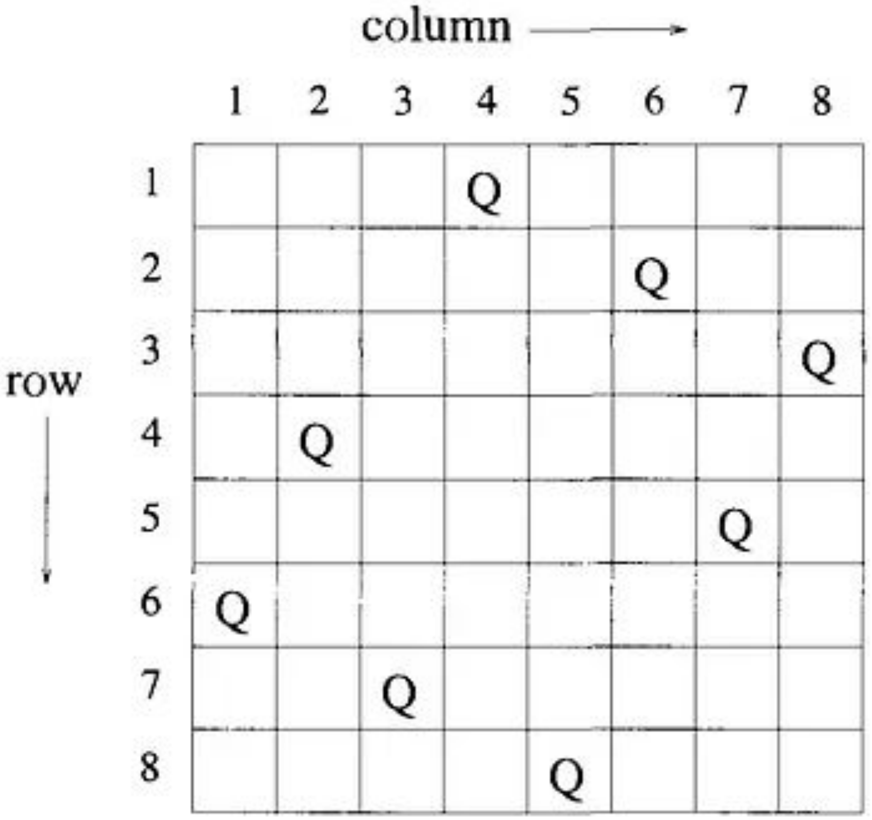

**Figure 7.6** Portion of the tree of Figure 7.2 that is generated during backtracking

# Backtracking Method - N Queens Problem



**4 - Queens solution space with nodes numbered in DFS**

# Backtracking Method - 8 Queens Solution



# Backtracking Method - N Queens Problem

⏱ **Time Complexity:**

- Worst-case: O(N!)
  - For the first row, N choices, then N-1, then N-2... hence N!
  - Backtracking prunes many invalid paths.

💾 **Space Complexity:**

- $O(N^2)$ for the board (can be optimized to O(N))
- O(N) for recursion stack

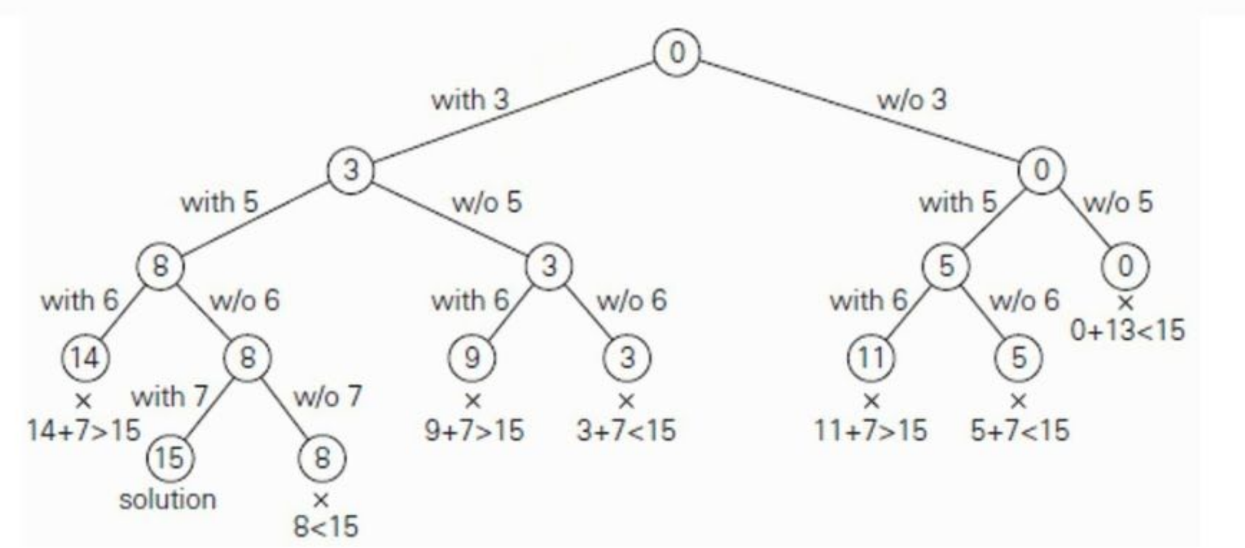
# Topics to be covered

- Backtracking Method
  - N-queen problem
  - Sum of subsets
  - Graph coloring
- Branch and Bound Method
  - 15 Puzzle problem
  - Traveling Salesperson problem

# Backtracking Method - Sum of Subsets Problem

- classic **backtracking** problem.
- **Input :** Given a set of **positive integers** and a target **sum M**,
- **Goal** : Find all **subsets** of the set whose **sum is equal to M**.

Given :
- Set = {3,5,6,7}
- Sum = 15




Courtesy : Medium - DAA

# Backtracking Method - Sum of Subsets Problem

1. Start with an empty subset and current sum = 0.
2. For each element, choose to either:
    a. Include it in the subset.
    b. Exclude it from the subset.
3. Repeat recursively for the next element.
4. If the current sum equals the target sum, record the solution.
5. If the current sum exceeds the target, backtrack (discard this path).

Given : Set = {3, 4, 5, 2}, Sum = 7

```
Start -> Include 3 -> sum=3
        └─> Include 4 -> sum=7 ✅ Solution: [3,4]
        └─> Exclude 4 -> Include 5 -> sum=8 ❌ (Backtrack)
                      -> Include 2 -> sum=5 ❌ (Backtrack)
        └─> Exclude 3 -> Include 4 -> sum=4
                      └─> Include 5 -> sum=9 ❌ (Backtrack)
                      └─> Include 2 -> sum=6 ❌ (Backtrack)
        └─> Include 5 -> sum=5
                      └─> Include 2 -> sum=7 ✅ Solution: [5,2]
```

Courtesy : [Medium - DAA]

# Backtracking Method - Sum of Subsets Problem

## ⏱ Time Complexity:

- In the **worst case**, each element has 2 choices (include or exclude) → **O(2^n)**
- If pruning is efficient (e.g., sorted input, early exits), it can reduce the actual runtime.

## 💾 Space Complexity:

- **O(n)** for the recursion stack
- **O(2^n)** for storing results in the worst case

**Solve the Following Problems**

1. Set = {1, 9, 7, 5, 18, 12, 20, 15} sum value = 35
2. Set = {5,10, 12, 13, 15, 18} and m = 30

Courtesy : Medium - DAA

# Topics to be covered

- Backtracking Method
  - N-queen problem
  - Sum of subsets
  - Graph coloring
- Branch and Bound Method
  - 15 Puzzle problem
  - Traveling Salesperson problem

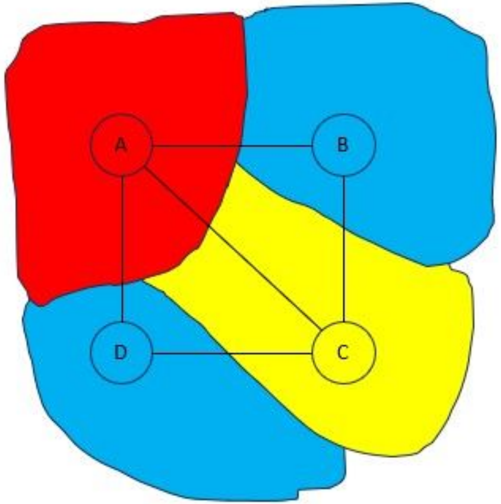
# Backtracking Method - Graph Coloring

**Input** : Given a graph with N vertices,

**Goal** : Determine if it's possible to **color the vertices using at most M colors** such that **no two adjacent vertices have the same color**.

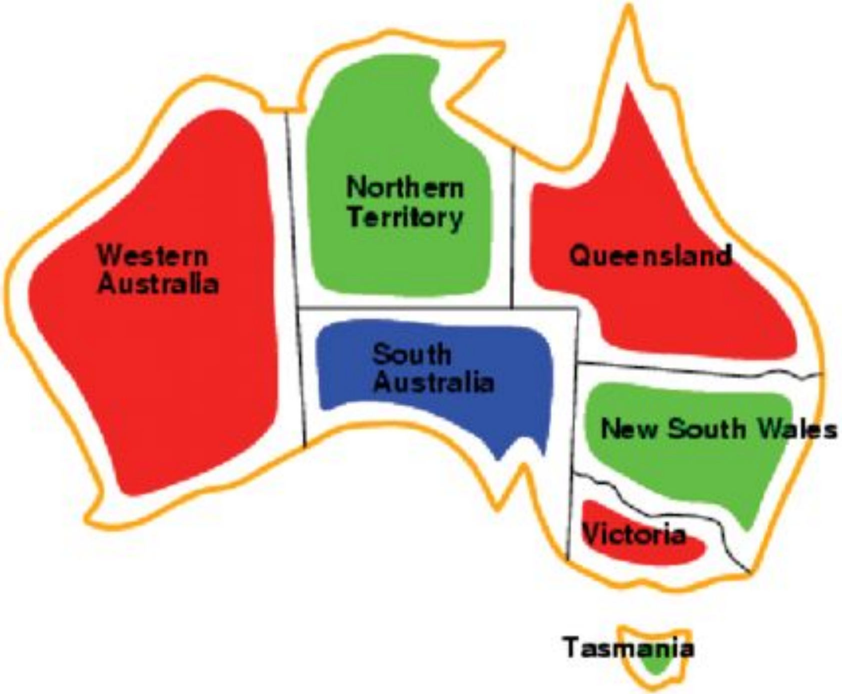
**Logic**

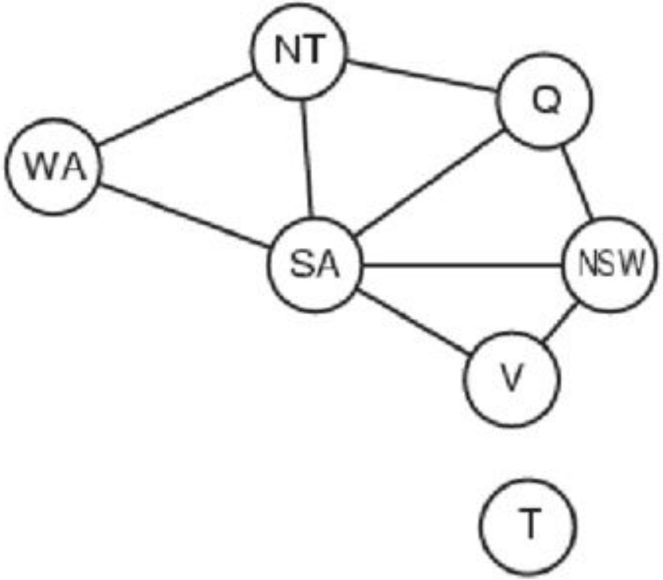
1. Try every possible color (from 1 to M).
2. Check if it's safe (i.e., no adjacent vertex has the same color).
3. If it's safe, assign the color and move to the next vertex.
4. If stuck, **backtrack** and try a different color.




Courtesy : Compgeek

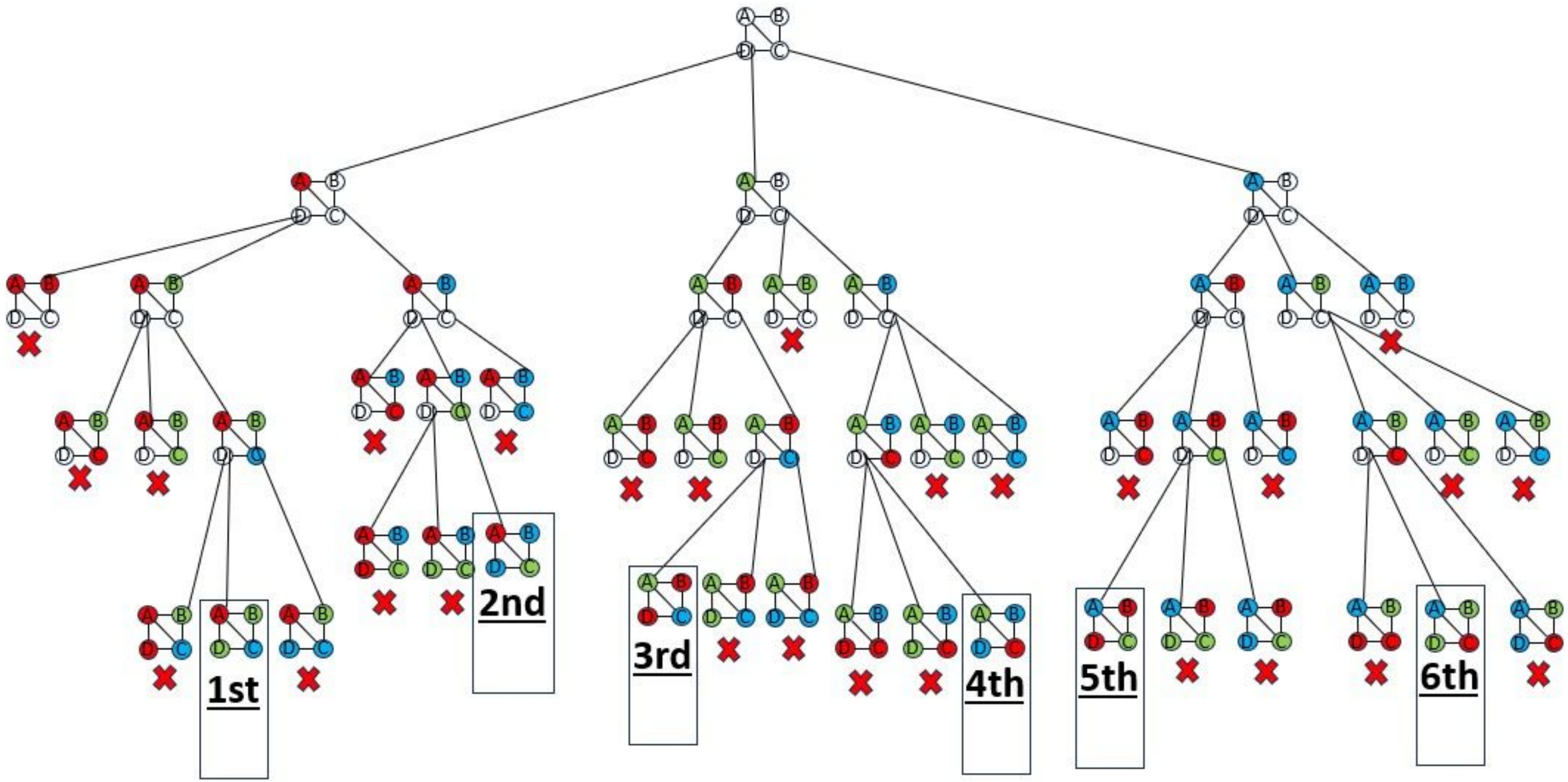
# Backtracking Method - Graph Coloring




Courtesy : [Compgeek]

# Backtracking Method - Graph Coloring




Courtesy : [Compgeek]

# Backtracking Method - Graph Coloring

## ⏱ Time Complexity:

- Worst-case: O(M^N)

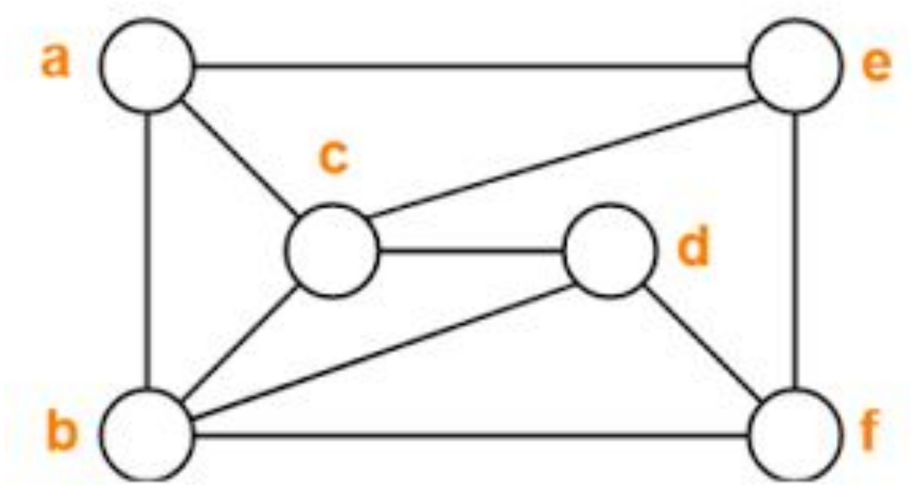
  (Each of the `N` vertices could have up to `M` color choices)

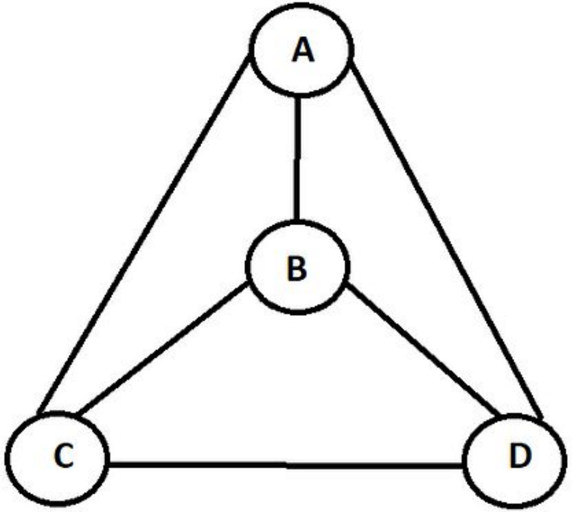
## 💾 Space Complexity:

- O(N) for color assignments
- O(N) for recursion stack

Courtesy : Compgeek

# Backtracking Method - Graph Coloring



# Topics to be covered

- Backtracking Method
  - N-queen problem
  - Sum of subsets
  - Graph coloring
- Branch and Bound Method
  - 15 Puzzle problem
  - Traveling Salesperson problem

# Branch and Bound

- General algorithm design paradigm used for **solving combinatorial optimization problems**, particularly those that involve **discrete decision-making**.
- Widely used when **A problem is too large for brute-force,**
- The search space can be represented as a **state space tree**, and
- We can **prune** parts of the tree that cannot contain better solutions than those already found.

Branch and Bound is used to solve problems such as:

- Traveling Salesperson Problem (TSP)
- 15 Puzzle Problem
- 0/1 Knapsack Problem
- Job Scheduling Problems
- Assignment Problem

# Branch and Bound - Core Components

| Component | Description |
|---|---|
| State Space Tree | A tree where each node represents a subproblem or partial solution. |
| Branching | Splitting the problem into smaller subproblems (child nodes). |
| Bounding Function | A method to compute a lower bound (min possible cost) or upper bound (max benefit) for a subproblem. |
| Pruning | If the bound of a node is worse than the best known solution, that branch is not explored further. |
| Priority Queue | Often used to explore the node with the best promise (lowest cost or highest value) next. |

# Branch and Bound - Step-by-Step working

1. **Start from the root** of the state space tree (initial solution).
2. **Generate children** (branch) representing all valid options from the current state.
3. For each child:
   - Compute the **bound** (e.g., minimum cost that can be achieved from this point).
4. Insert children into a **priority queue** (sorted by bound).
5. **Pick the most promising node** (lowest bound) to expand next.
6. **Prune** nodes whose bounds exceed the current best solution.
7. Repeat until all nodes are either expanded or pruned.

# Branch and Bound - Bounding Techniques

Bounding is crucial to reduce the search space. Common bounding methods:

- Greedy approximation
- Linear relaxation
- Cost matrix reduction (TSP)
- Manhattan distance (15 Puzzle)
- Fractional knapsack (used as a bound in 0/1 Knapsack)

# Branch and Bound - Pros & Cons

## ✅ Advantages of Branch and Bound

- Guarantees **optimal solution**
- Efficient pruning leads to **faster results** than brute force
- **Flexible design**: You can use problem-specific heuristics for bounding

## ⚠️ Limitations

- Not polynomial-time (still exponential in the worst case)
- Requires good bounding functions for effective pruning
- May consume high memory due to large state space trees

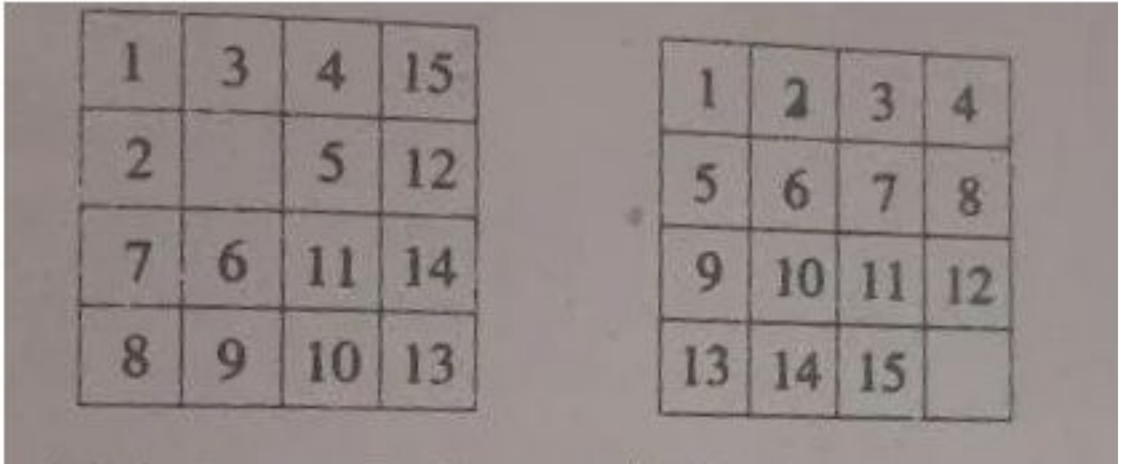
# Topics to be covered

- Backtracking Method
  - N-queen problem
  - Sum of subsets
  - Graph coloring
- Branch and Bound Method
  - 15 Puzzle problem
  - Traveling Salesperson problem

# Branch & Bound - 15 Puzzle



- Sliding puzzle
- **Input** : 4x4 grid with **15 numbered tiles** and **1 empty space**.
- **Goal** : **Move the tiles using the empty space** to achieve the goal configuration.




(a) An arrangement (b) Goal arrangement

# Branch & Bound - 15 Puzzle

**Logic :**

1. Start from the initial state.
2. Generate all possible moves (up, down, left, right).
3. Calculate a cost for each state using: $c'=f(x) + g'(x)$

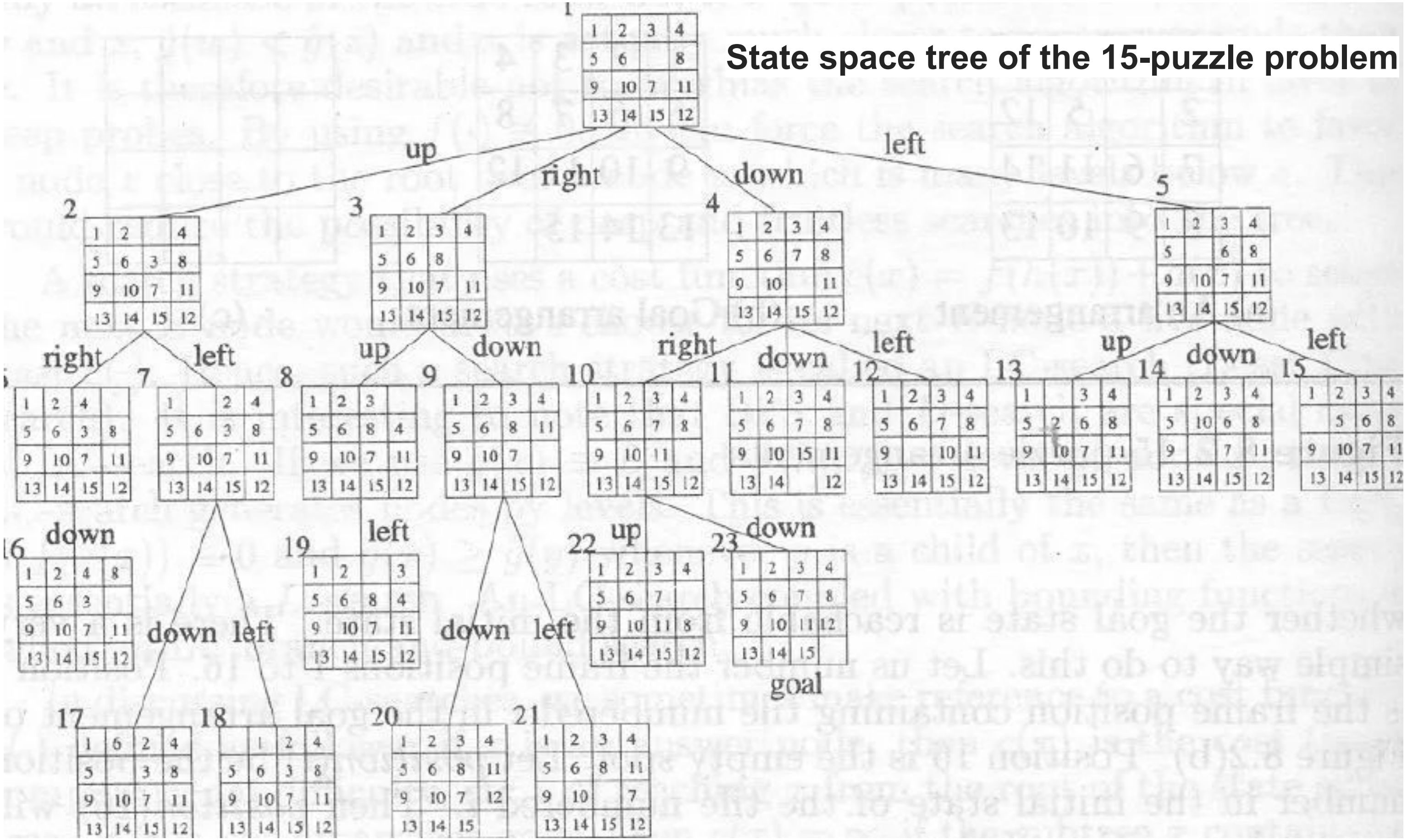
   where $f(x)$ : path length from root to x.

   $g'(x)$ : number of occupied tiles not in the goal position
4. Use a priority queue (min-heap) to explore the state with the lowest cost first.
5. Continue exploring until the goal state is found.

Common Heuristic: **Manhattan Distance =** Sum of distances of each tile from its goal position

$$h(n) = \Sigma\ |current\_row - goal\_row| + |current\_col - goal\_col|$$

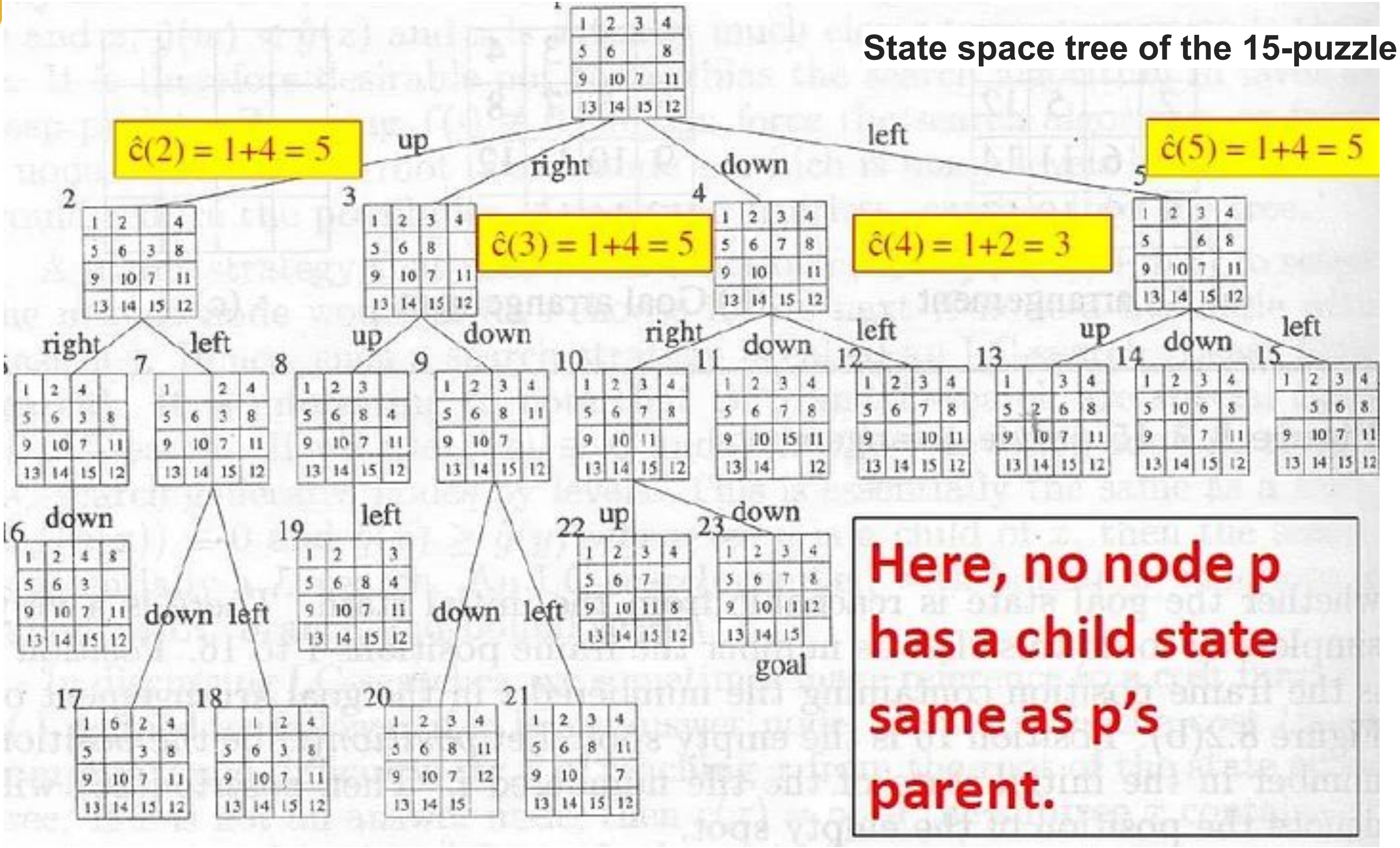
# Branch & Bound - 15 Puzzle




State space tree of the 15-puzzle problem

# Branch & Bound - 15 Puzzle

State space tree of the 15-puzzle problem




c'(2) = 1 + 4 = 5

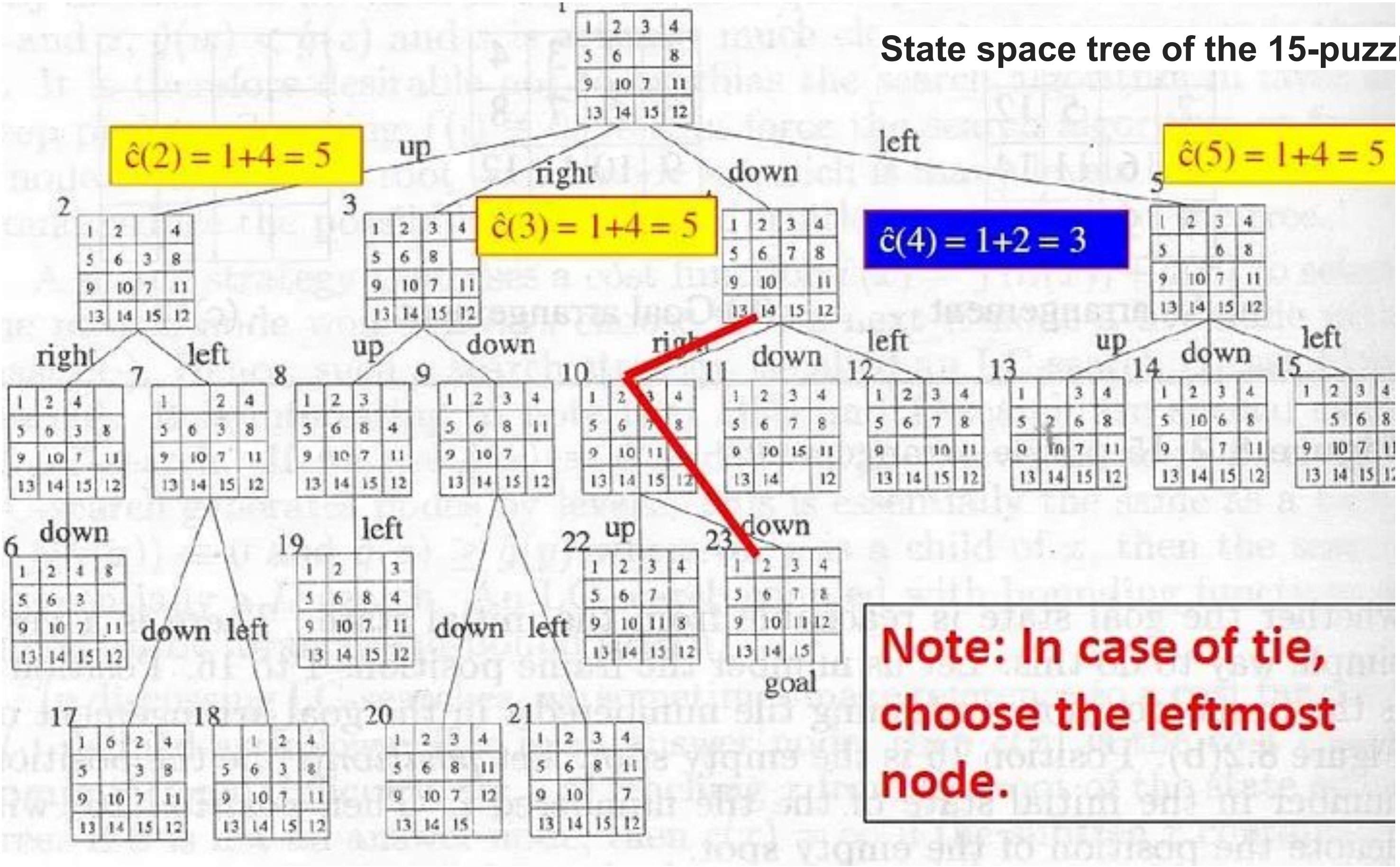
c'(3) = 1 + 4 = 5

**c'(4) = 1 + 2 = 3**

c'(5) = 1 + 4 = 5

# Branch & Bound - 15 Puzzle

State space tree of the 15-puzzle problem




**c’(10)=2+1=3**
c’(11)=2+3=5
c’(12)=2+3=5

# Branch & Bound - 15 Puzzle

## ⏱ Time Complexity

- **Worst-case:** O(N!) → exploring all possible board configurations.
- N = 15 tiles → ~10^13 possible states
- But with **heuristic** + **pruning**, we explore far fewer.

## 💾 Space Complexity

- O(N!) for storing visited states (can be reduced with hashing)
- O(depth) for the recursion/priority queue

# Topics to be covered

- Backtracking Method
  - N-queen problem
  - Sum of subsets
  - Graph coloring
- Branch and Bound Method
  - 15 Puzzle problem
  - Traveling Salesperson problem

# Branch & Bound - Travelling Salesman Problem

**Input** : Given a list of cities and the cost (or distance) between each pair of cities

**Goal** : Visit each city exactly once and

Return to the starting city,

**Minimizing the total travel cost.**

**Real-World Analogy:** A delivery truck that needs to deliver packages to multiple cities.

It must return to the starting depo while minimizing fuel or time.

- TSP is a **NP-hard problem**, so brute force (checking all permutations) is too slow for larger n.
- **Branch and Bound** helps by:
  - Systematically exploring all paths,
  - Pruning those that cannot possibly lead to a better solution,
  - Using lower bound estimates to guide the search.

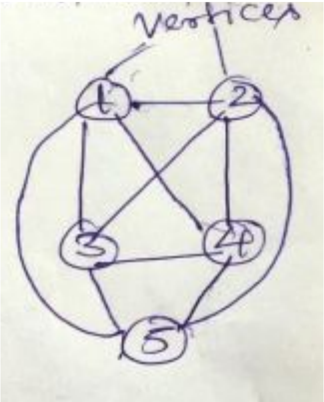
# Branch & Bound - Travelling Salesman Problem

## Logic

1. Start from a city (say, city 0).
2. Generate all possible cities that can be visited next.
3. For each partial tour:
    a. Calculate cost so far (g).
    b. Calculate a **lower bound estimate (h)** on the remaining cost.
    c. **Total cost = g + h.**
4. Use a priority queue (min-heap) to expand the least-cost node first.
5. If the current path visits all cities and returns to the start, check if it's the best solution so far.
6. Prune any path whose cost estimate is worse than the current best.

# Branch & Bound - Travelling Salesman Problem

$$
\begin{array}{c@{\quad}c}
 & \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} &
\begin{bmatrix}
\infty & 20 & 30 & 10 & 11 \\
15 & \infty & 16 & 4 & 2 \\
3 & 5 & \infty & 2 & 4 \\
19 & 6 & 18 & \infty & 3 \\
16 & 4 & 7 & 16 & \infty
\end{bmatrix}
\end{array}
$$




Courtesy : [SJCE]

# Branch & Bound - Travelling Salesman Problem

| | 1 | 2 | 3 | 4 | 5 | Row |
|---|---|---|---|---|---|---|
| 1 | ∞ | 20 | 30 | 10 | 11 | 10 |
| 2 | 15 | ∞ | 16 | 4 | 2 | 2 |
| 3 | 3 | 5 | ∞ | 2 | 4 | 2 |
| 4 | 19 | 6 | 18 | ∞ | 3 | 3 |
| 5 | 16 | 4 | 7 | 16 | ∞ | 4 |
| | | | | | | 21 |

**Row Reduction**

| | 1 | 2 | 3 | 4 | 5 | |
|---|---|---|---|---|---|---|
| 1 | ∞ | 10 | 20 | 0 | 1 | |
| 2 | 13 | ∞ | 14 | 2 | 0 | |
| 3 | 1 | 3 | ∞ | 0 | 2 | |
| 4 | 16 | 3 | 15 | ∞ | 0 | |
| 5 | 12 | 0 | 3 | 12 | ∞ | |
| Col min | 1 | 0 | 3 | 0 | 0 | = 4 |

**Column Reduction.**

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | ∞ | 10 | 17 | 0 | 1 |
| 2 | 12 | ∞ | 11 | 2 | 0 |
| 3 | 0 | 3 | ∞ | 0 | 2 |
| 4 | 15 | 3 | 12 | ∞ | 0 |
| 5 | 11 | 0 | 0 | 12 | ∞ |

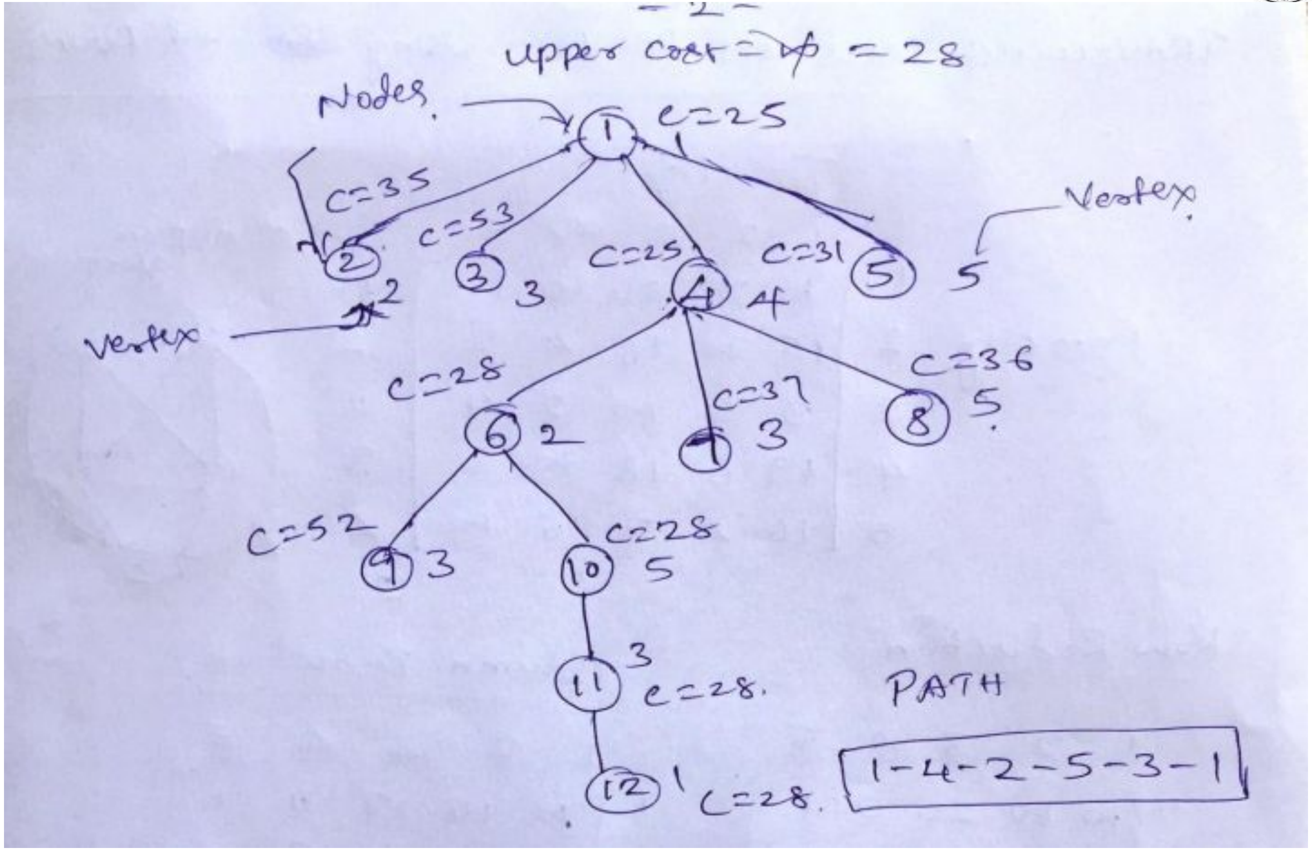
COST MATRIX.

Reduced cost or least travelling cost = 21 + 4 = 25

Courtesy : SJCE

# Branch & Bound - Travelling Salesman Problem

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | ∞ | 20 | 30 | 10 | 11 |
| 2 | 15 | ∞ | 16 | 4 | 2 |
| 3 | 3 | 5 | ∞ | 2 | 4 |
| 4 | 19 | 6 | 18 | ∞ | 3 |
| 5 | 16 | 4 | 7 | 16 | ∞ |




Courtesy : SJCE

# Branch & Bound - Travelling Salesman Problem

$$
\begin{array}{c|c}
 & \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} &
\begin{bmatrix}
\infty & 10 & 17 & 0 & 1 \\
12 & \infty & 11 & 2 & 0 \\
0 & 3 & \infty & 0 & 2 \\
15 & 3 & 12 & \infty & 0 \\
11 & 0 & 0 & 12 & \infty
\end{bmatrix}
\end{array}
$$

COST MATRIX

Cost of Node 2 is given by.

Cost of 1 to 2 in cost matrix + Least/Reduced cost + Current Reduced cost

$C(1,2) + r + \hat{r}$

$10 + 25 + 0 = 35$

To find the cost of node 2 (travel from 1 to 2]
make the first row and second column as ∞
in the cost matrix & also 2 to 1 cost is made ∞

Reduced matrix for Node 2

This ∞ implies that we cannot go back from 2 to 1

$$
\begin{array}{c|c|c}
 & \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} & \text{Row min} \\
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} &
\begin{bmatrix}
\infty & \infty & \infty & \infty & \infty \\
\infty & \infty & 11 & 2 & 0 \\
0 & \infty & \infty & 0 & 2 \\
15 & \infty & 12 & \infty & 0 \\
11 & \infty & 0 & 12 & \infty
\end{bmatrix}
& \begin{array}{c} \\ 0 \\ 0 \\ 0 \\ 0 \end{array} \\
\text{Col min} & \begin{array}{ccccc} 0 & \downarrow & 0 & 0 & 0 \end{array} &
\end{array}
$$

This ∞ implies from one we cannot go anywhere since 1 to 2 is fixed here

This ∞ implies that from 2, 3, 4, or 5 we cannot come back to 2.

Check whether all the rows & columns have zeros. If yes proceed otherwise reduce the matrix.

Courtesy : SJCE

# Branch & Bound - Travelling Salesman Problem

$$
\begin{array}{c|c}
 & \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} &
\left[\begin{array}{ccccc}
\infty & 10 & 17 & 0 & 1 \\
12 & \infty & 11 & 2 & 0 \\
0 & 3 & \infty & 0 & 2 \\
15 & 3 & 12 & \infty & 0 \\
11 & 0 & 0 & 12 & \infty
\end{array}\right]
\end{array}
$$

COST MATRIX

for Node 3 reduced matrix is given by make first row & 3rd column & 3 to 1 ∞.

$$
\begin{array}{c|c|c}
 & \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} & \text{Row min} \\
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} &
\left[\begin{array}{ccccc}
\infty & \infty & \infty & \infty & \infty \\
12 & \infty & \infty & 2 & 0 \\
\infty & 3 & \infty & \infty & 2 \\
15 & 3 & \infty & \infty & 0 \\
11 & 0 & \infty & 12 & \infty
\end{array}\right] &
\begin{array}{c} - \\ 0 \\ 0 \\ 0 \\ 0 \end{array} \\
\text{col min} & \begin{array}{ccccc} 11 & 0 & - & 0 & 0 \end{array} & \boxed{11}
\end{array}
$$

11 ← current Reduced cost

column reduction

So, reduce the matrix by doing

Reduced matrix for node 3

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

$$c(1,3) + r + \hat{r}$$

$$17 + 25 + 11 = 53$$

Courtesy : SJCE

# Branch & Bound - Travelling Salesman Problem

$$
\begin{array}{c|ccccc|}
 & 1 & 2 & 3 & 4 & 5 \\
\hline
1 & \infty & 10 & 17 & 0 & 1 \\
2 & 12 & \infty & 11 & 2 & 0 \\
3 & 0 & 3 & \infty & 0 & 2 \\
4 & 15 & 3 & 12 & \infty & 0 \\
5 & 11 & 0 & 0 & 12 & \infty \\
\hline
\end{array}
$$

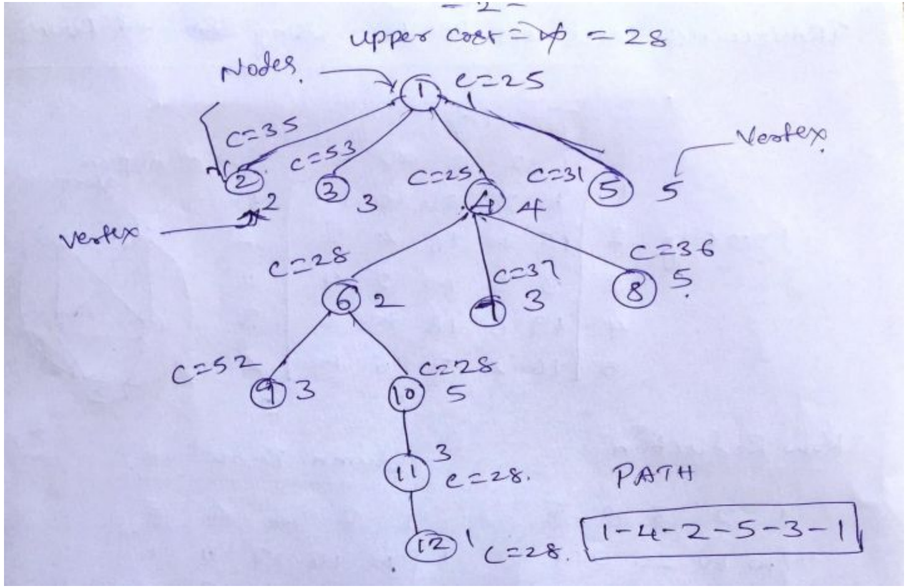
COST MATRIX

Reduced matrix for ~~Node~~ 4.

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

$$C(1,4) + r + \hat{r}$$

$$0 + 25 + 0 = 25$$




Courtesy : SJCE

# Branch & Bound - Travelling Salesman Problem

$$
\begin{array}{c|ccccc}
 & 1 & 2 & 3 & 4 & 5 \\
\hline
1 & \infty & 10 & 17 & 0 & 1 \\
2 & 12 & \infty & 11 & 2 & 0 \\
3 & 0 & 3 & \infty & 0 & 2 \\
4 & 15 & 3 & 12 & \infty & 0 \\
5 & 11 & 0 & 0 & 12 & \infty
\end{array}
$$

COST MATRIX

Reduced matrix for Node 5.

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

After Reduction ⇒

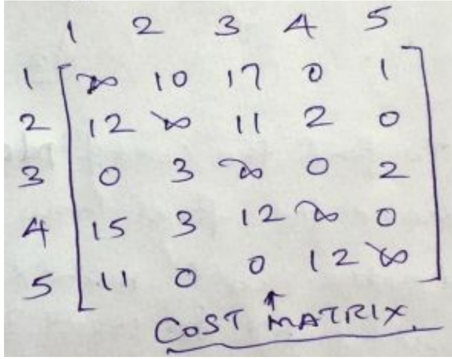
$$
\begin{array}{c|ccccc}
 & 1 & 2 & 3 & 4 & 5 \\
\hline
1 & \infty & \infty & \infty & \infty & \infty \\
2 & 10 & \infty & 9 & 0 & \infty \\
3 & 0 & 3 & \infty & 0 & \infty \\
4 & 12 & 0 & 9 & \infty & \infty \\
5 & \infty & 0 & 0 & 12 & \infty
\end{array}
$$

$c(1,5) + r + \hat{r}$

$1 + 25 + 5 = 31$

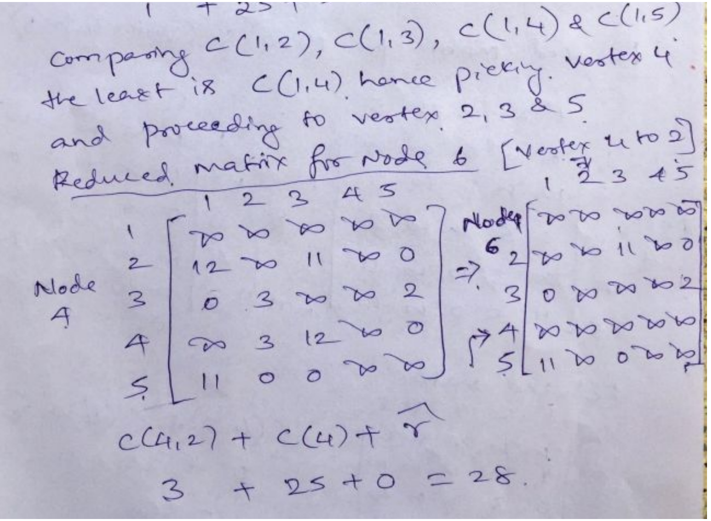
Courtesy : SJCE

# Branch & Bound - Travelling Salesman Problem



| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | ∞ | 10 | 17 | 0 | 1 |
| 2 | 12 | ∞ | 11 | 2 | 0 |
| 3 | 0 | 3 | ∞ | 0 | 2 |
| 4 | 15 | 3 | 12 | ∞ | 0 |
| 5 | 11 | 0 | 0 | 12 | ∞ |

COST MATRIX



Comparing c(1,2), c(1,3), c(1,4) & c(1,5) the least is c(1,4) hence picking vertex 4 and proceeding to vertex 2, 3 & 5.

Reduced matrix for Node 6 [Vertex 4 to 2]

Node 4

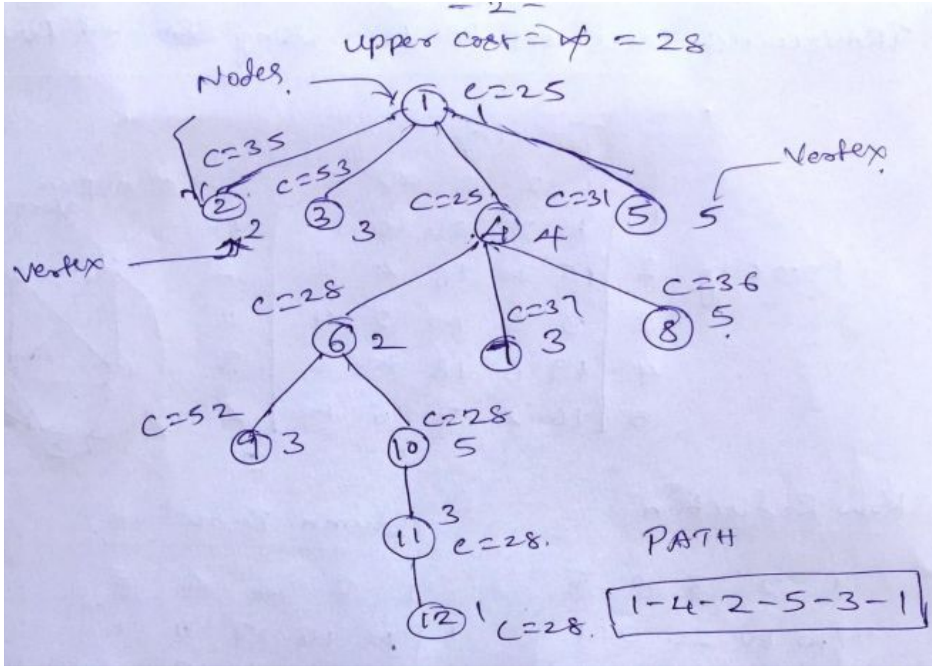
| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | ∞ | ∞ | ∞ | ∞ | ∞ |
| 2 | 12 | ∞ | 11 | ∞ | 0 |
| 3 | 0 | 3 | ∞ | ∞ | 2 |
| 4 | ∞ | 3 | 12 | ∞ | 0 |
| 5 | 11 | 0 | 0 | ∞ | ∞ |

Node 6

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | ∞ | ∞ | ∞ | ∞ | ∞ |
| 2 | ∞ | ∞ | 11 | ∞ | 0 |
| 3 | 0 | ∞ | ∞ | ∞ | 2 |
| 4 | ∞ | ∞ | ∞ | ∞ | ∞ |
| 5 | 11 | ∞ | 0 | ∞ | ∞ |

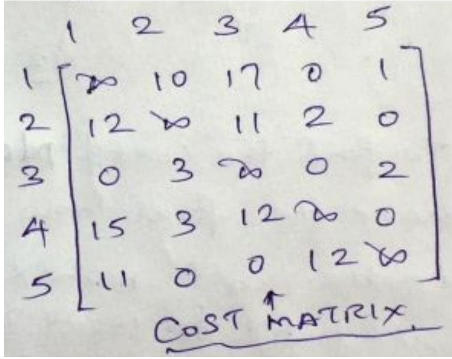
$c(4,2) + c(4) + \hat{r}$

$3 + 25 + 0 = 28.$



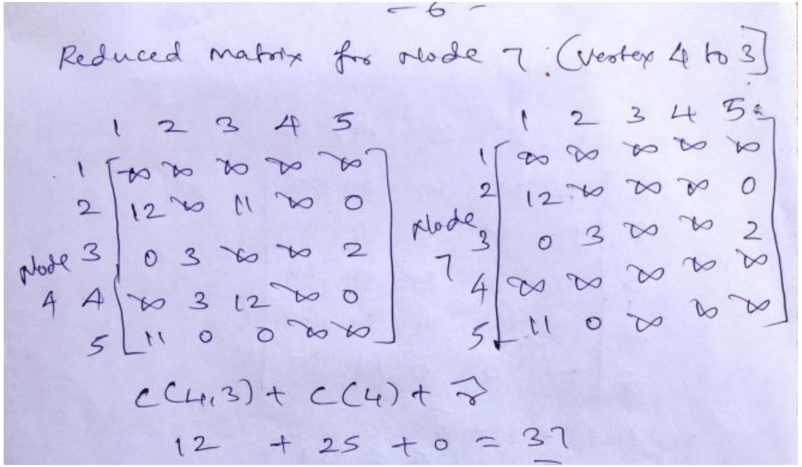

Courtesy : SJCE

# Branch & Bound - Travelling Salesman Problem



$$\begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & \infty & 10 & 17 & 0 & 1 \\ 2 & 12 & \infty & 11 & 2 & 0 \\ 3 & 0 & 3 & \infty & 0 & 2 \\ 4 & 15 & 3 & 12 & \infty & 0 \\ 5 & 11 & 0 & 0 & 12 & \infty \end{array}$$

COST MATRIX



-6-

Reduced matrix for node 7 (vertex 4 to 3)

Node 4

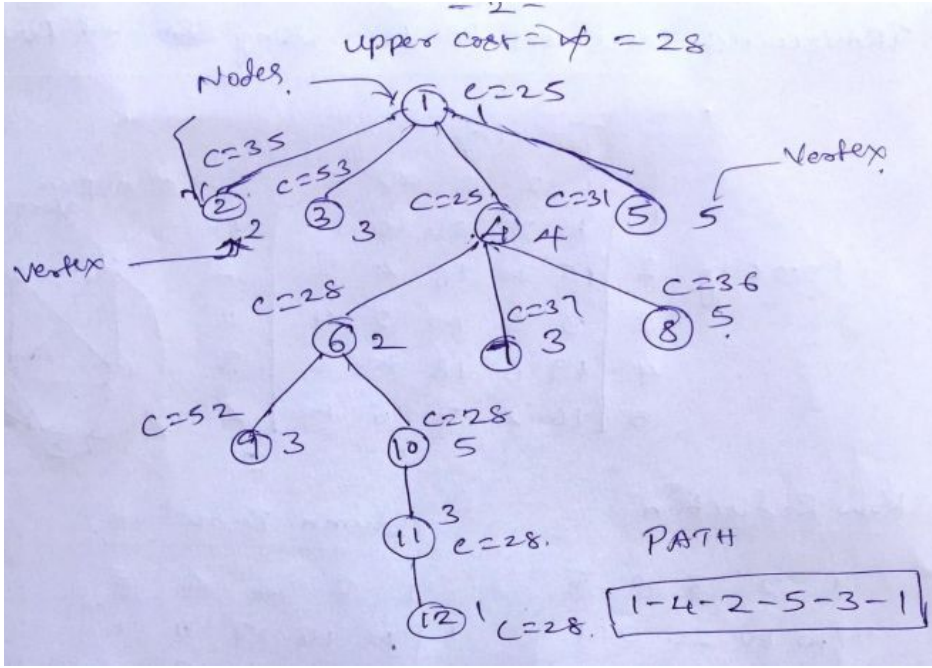
$$\begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & \infty & \infty & \infty & \infty & \infty \\ 2 & 12 & \infty & 11 & \infty & 0 \\ 3 & 0 & 3 & \infty & \infty & 2 \\ 4 & \infty & 3 & 12 & \infty & 0 \\ 5 & 11 & 0 & 0 & \infty & \infty \end{array}$$

Node 7

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

$c(4,3) + c(4) + \hat{r}$

$12 + 25 + 0 = 37$




Courtesy : SJCE

# Branch & Bound - Travelling Salesman Problem

$$\begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & \infty & 10 & 17 & 0 & 1 \\ 2 & 12 & \infty & 11 & 2 & 0 \\ 3 & 0 & 3 & \infty & 0 & 2 \\ 4 & 15 & 3 & 12 & \infty & 0 \\ 5 & 11 & 0 & 0 & 12 & \infty \end{array}$$

COST MATRIX

Reduced matrix for Node 8 (Vertex 4 to 5)

Node 4.

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

Node 8

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

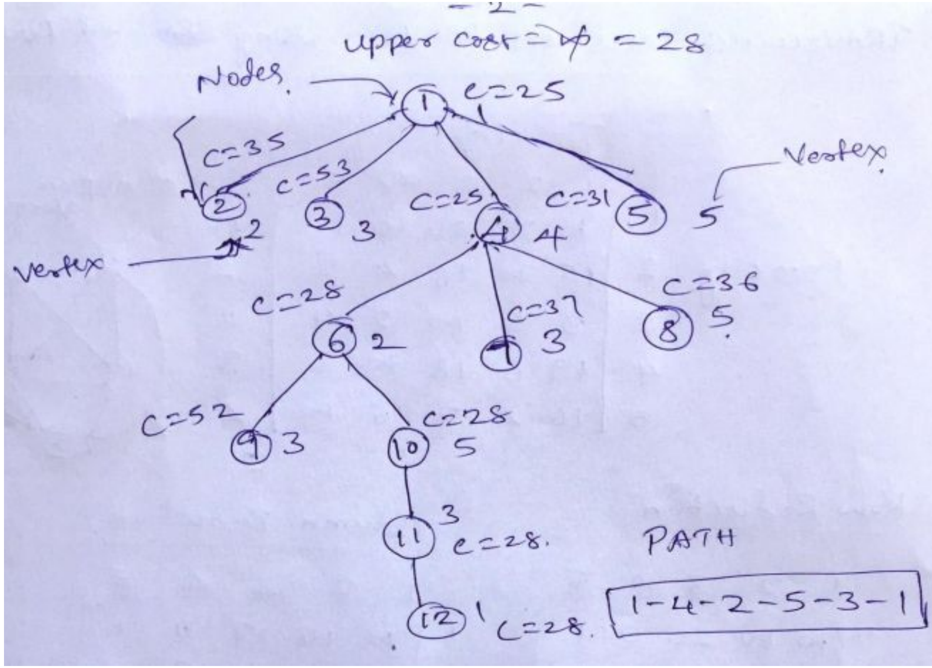
Reduced matrix.

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

$C(4,5) + C(4) + \hat{r}$

$0 + 25 + 11$

$= 36$




Courtesy : SJCE

# Branch & Bound - Travelling Salesman Problem

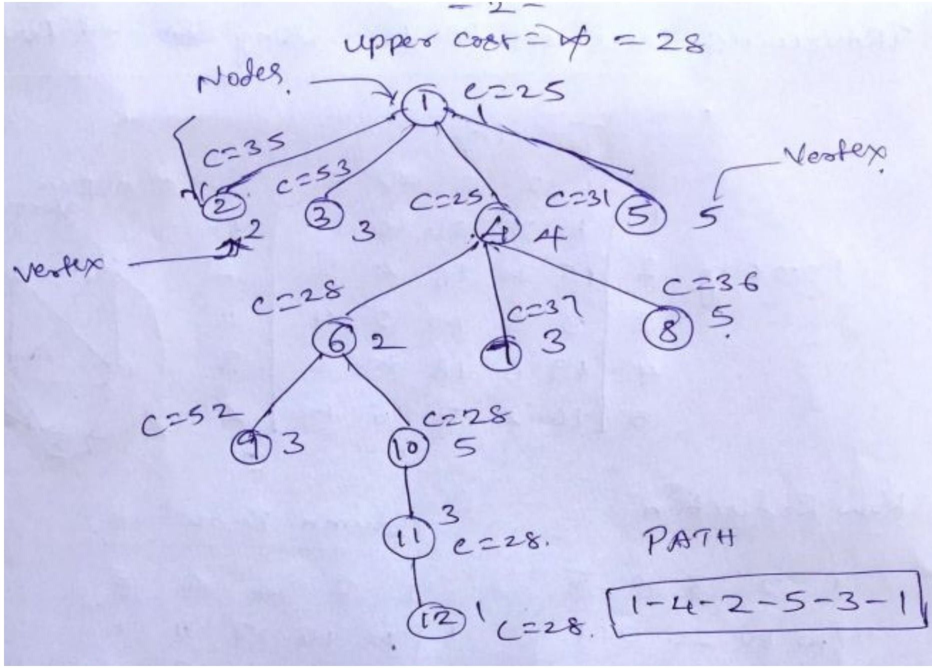
comparing C(1,4,2), C(1,4,3) & C(1,4,5), the least is C(1,4,2). hence picking Vertex 2 and proceeding to vertex 3 & 5.

Reduced matrix for node 9 (Vertex 2 to 3)

$$\text{Node } 6 \quad \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & \infty & \infty & \infty & \infty & \infty \\ 2 & \infty & \infty & 11 & \infty & 0 \\ 3 & 0 & \infty & \infty & \infty & 2 \\ 4 & \infty & \infty & \infty & \infty & \infty \\ 5 & 11 & \infty & 0 & \infty & \infty \end{array} \Rightarrow \text{Node } 9 \quad \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & \infty & \infty & \infty & \infty & \infty \\ 2 & \infty & \infty & \infty & \infty & \infty \\ 3 & \infty & \infty & \infty & \infty & 2 \\ 4 & \infty & \infty & \infty & \infty & \infty \\ 5 & 11 & \infty & \infty & \infty & \infty \end{array}$$

$$C(2,3) + C(6) + \hat{r}$$

$$11 + 28 + 13 = 52$$

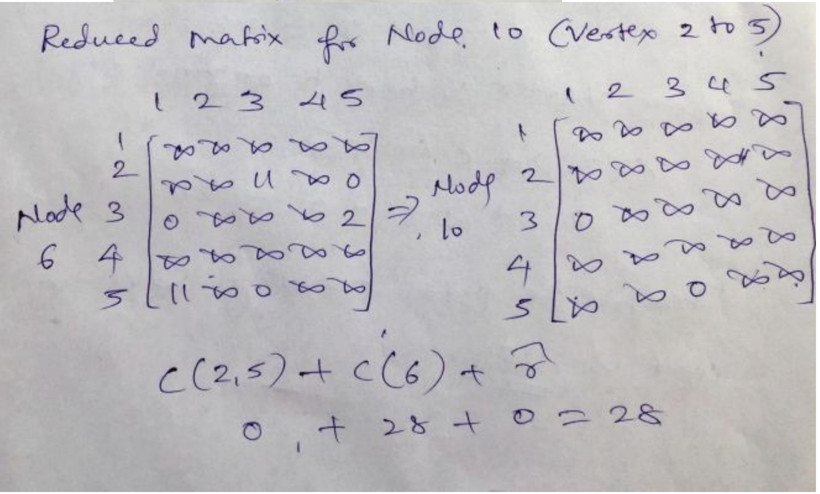



Courtesy : SJCE

# Branch & Bound - Travelling Salesman Problem

$$
\begin{array}{c|ccccc}
 & 1 & 2 & 3 & 4 & 5 \\
\hline
1 & \infty & 10 & 17 & 0 & 1 \\
2 & 12 & \infty & 11 & 2 & 0 \\
3 & 0 & 3 & \infty & 0 & 2 \\
4 & 15 & 3 & 12 & \infty & 0 \\
5 & 11 & 0 & 0 & 12 & \infty
\end{array}
$$

COST MATRIX



Reduced matrix for Node 10 (Vertex 2 to 5)

Node 6:

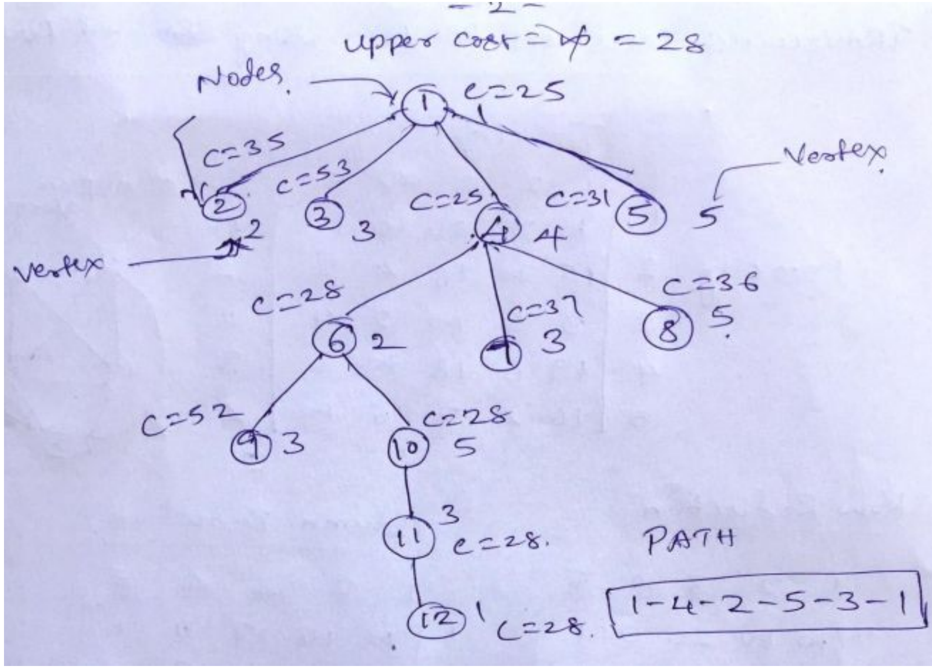
$$
\begin{array}{c|ccccc}
 & 1 & 2 & 3 & 4 & 5 \\
\hline
1 & \infty & \infty & \infty & \infty & \infty \\
2 & \infty & \infty & 11 & \infty & 0 \\
3 & 0 & \infty & \infty & \infty & 2 \\
4 & \infty & \infty & \infty & \infty & \infty \\
5 & 11 & \infty & 0 & \infty & \infty
\end{array}
\Rightarrow
$$

Node 10:

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

$c(2,5) + c(6) + \hat{r}$

$0 + 28 + 0 = 28$




Courtesy : SJCE

# Branch & Bound - Travelling Salesman Problem

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | ∞ | 10 | 17 | 0 | 1 |
| 2 | 12 | ∞ | 11 | 2 | 0 |
| 3 | 0 | 3 | ∞ | 0 | 2 |
| 4 | 15 | 3 | 12 | ∞ | 0 |
| 5 | 11 | 0 | 0 | 12 | ∞ |

COST MATRIX

comparing $c(1,4,2,3)$ & $c(1,4,2,5)$
the least is $c(1,4,2,5)$ hence picking vertex 5
and proceeding to vertex 3

Reduced matrix for Node 11 (vertex 5 to 3)

Node 10

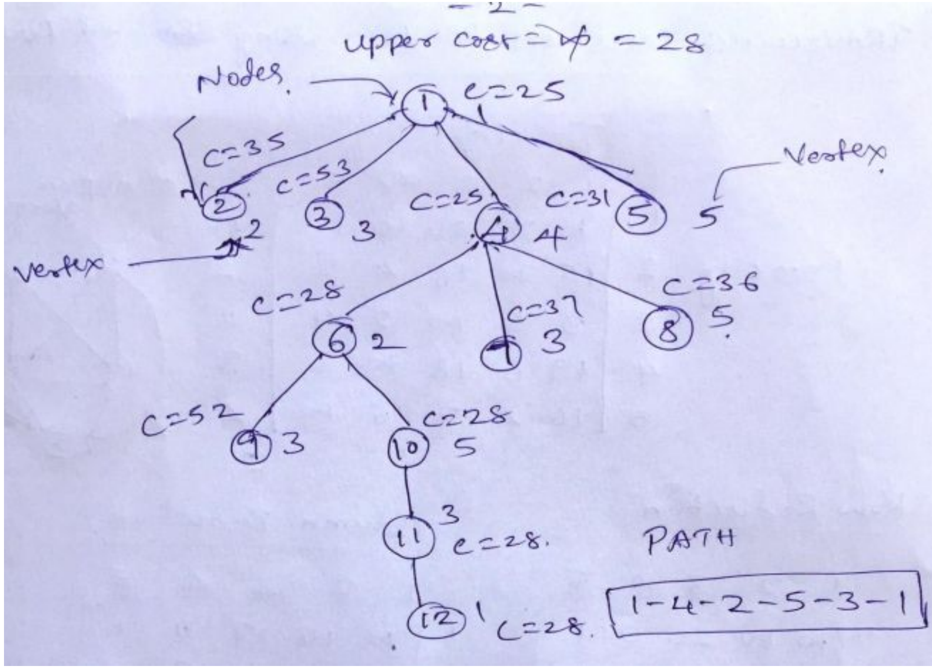
$$\begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & \infty & \infty & \infty & \infty & \infty \\ 2 & \infty & \infty & \infty & \infty & \infty \\ 3 & 0 & \infty & \infty & \infty & \infty \\ 4 & \infty & \infty & \infty & \infty & \infty \\ 5 & \infty & \infty & 0 & \infty & \infty \end{array}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

Node 11

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

$c(5,3) + c(10) + \hat{r}$

$0 + 28 + 0 = 28.$




Courtesy : SJCE

# Branch & Bound - Travelling Salesman Problem

$$
\begin{array}{c|c}
 & \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} &
\begin{bmatrix}
\infty & 10 & 17 & 0 & 1 \\
12 & \infty & 11 & 2 & 0 \\
0 & 3 & \infty & 0 & 2 \\
15 & 3 & 12 & \infty & 0 \\
11 & 0 & 0 & 12 & \infty
\end{bmatrix}
\end{array}
$$

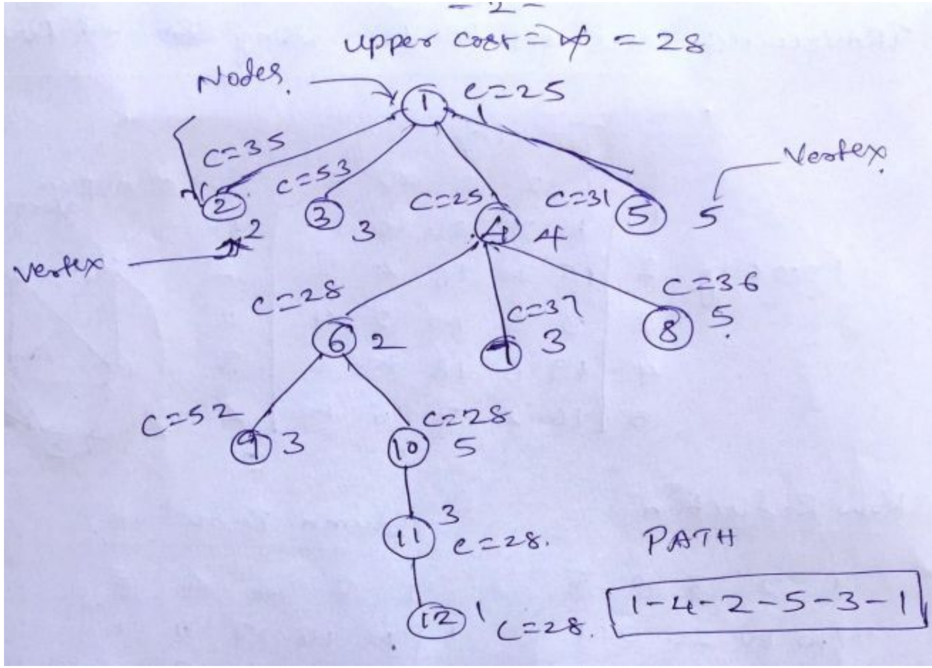
COST MATRIX

from vertex 3 back to original city!

$c(3,1) + c(11) + \hat{r}$

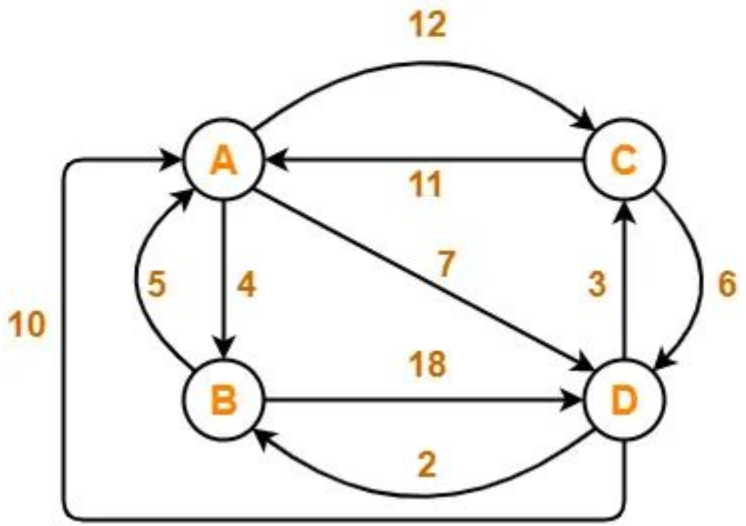
$0 + 28 + 0 = 28.$

Therefore the total travelling Salesman cost is $\underline{28}$.




Courtesy : SJCE

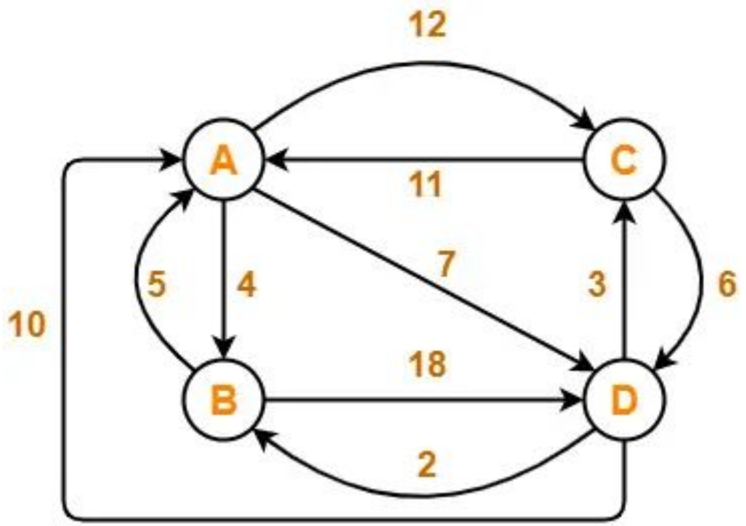
# Branch & Bound - Travelling Salesman Problem




| | A | B | C | D |
|---|---|---|---|---|
| A | ∞ | 10 | 15 | 20 |
| B | 10 | ∞ | 35 | 25 |
| C | 15 | 35 | ∞ | 30 |
| D | 20 | 25 | 30 | ∞ |

Courtesy : Gate Vidyalay

# Branch & Bound - Travelling Salesman Problem




- Optimal path is: **A** → **C** → **D** → **B** → **A**
- Cost of Optimal path = **25 units**

Courtesy : Gate Vidyalay

# Branch & Bound - Travelling Salesman Problem

| | A | B | C | D |
|---|---|---|---|---|
| A | ∞ | 10 | 15 | 20 |
| B | 10 | ∞ | 35 | 25 |
| C | 15 | 35 | ∞ | 30 |
| D | 20 | 25 | 30 | ∞ |

**Optimal Path**: A → B → D → C → A

**Minimum Cost**: 80

Courtesy : Gate Vidyalay

# Branch & Bound - Travelling Salesman Problem

## ⏱ Time Complexity:

- **Worst-case:** $O(n!)$
  (All possible permutations are checked in the worst case)

But with **Branch and Bound pruning**, we typically explore **much fewer nodes.**

## 💾 Space Complexity:

- $O(n^2)$ for cost matrix storage
- $O(n!)$ in worst case for storing all states in the priority queue

# Backtracking Vs Branch & Bound

| Parameter | Backtracking | Branch and Bound |
|---|---|---|
| **1. Problem Type** | Typically used for **decision** and **combinatorial** problems | Used for **optimization** and **combinatorial** problems |
| **2. Solution Strategy** | Explores **all possibilities** and backtracks when constraints are violated | Explores possibilities using **cost-based bounding** to **prune suboptimal branches** |
| **3. Node Evaluation** | Nodes are **accepted/rejected** based on constraint satisfaction | Nodes are **evaluated using a bound (cost estimate)** to decide exploration order |
| **4. Optimality Guarantee** | May not guarantee optimal solution unless all paths are explored | **Always aims for optimal solution** by expanding the **least-cost (best) nodes first** |
| **5. Efficiency** | Can be **inefficient** for large search spaces due to exhaustive exploration | **More efficient** due to bounding and pruning of unpromising paths |