Branch & Bound, Backtracking.

Que- Explain Traveling Sales person Problem using Branch & Bound with example.

⇒ The Traveling salesperson Problem (TSP) is a classic optimization Problem in which a salesperson must visit a set of cities exactly once, returning to the starting city, with the goal of minimizing the total travel cost.
Branch and Bound (B&B) is an efficient algorithm to solve this problem by systematically exploring possible solutions while pruning unpromising ones.

Steps of the Branch and Bound Algorithm :-

1) Define the Problem :- Represent the cities and distances in a cost matrix.
2) Lower Bound Calculation :- Calculate a bound for the minimum cost of any solution derived from the current node.
3) Branching :- Explore paths by selecting one city to visit next.
4) Pruning :- Discard branches with a cost higher than the best-known solution.
5) Continue :- Repeat until all nodes are explored or pruned.

Example :-
Given :-
- 4 cities (A, B, C, D) with the following distance matrix :-

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

(Here, ∞ represents no direct path from a city to itself).

Steps in B&B for TSP

1) Initial Lower Bound (Root Node):-

- Row Reduction:- subtract the smallest value in each row from all elements in that row.

Row 1 → 10
Row 2 → 10
Row 3 → 15
Row 4 → 20

	A	B	C	D
A	∞	0	5	10
B	0	∞	25	15
C	0	20	∞	15
D	0	5	10	∞

- Column Reduction :- subtract the smallest value in each column from all elements in that column

Column 1 → 0
Column 2 → 0
Column 3 → 5
Column 4 → 10

	A	B	C	D
A	∞	0	0	0
B	0	∞	20	5
C	0	20	∞	5
D	0	5	5	∞

Row Reduction cost = 10 + 10 + 15 + 20 = 55
Column Reduction cost = 0 + 0 + 5 + 10 = 15

Lower Bound = Row Reduction cost + Column Reduction cost
= 55 + 15
= 70

∴ Branch 1 : A → B
step1 :- Fix the Path A → B
- set row A and column B to ∞ (to prevent revisiting).
- Eliminate the reverse path B → A by setting matrix [B][A] = ∞

	A	B	C	D
A	∞	∞	∞	∞
B	∞	∞	20	5
C	0	∞	∞	5
D	0	∞	5	∞

Step2 :- Row Reduction
(1) Row B :- Smallest value = 5
20 - 5 = 15 , 5 - 5 = 0
(2) Row C :- Smallest value = 5
5 - 5 = 0
(3) Row D :- Smallest value = 0
- No change

	A	B	C	D
A	∞	∞	∞	∞
B	∞	∞	15	0
C	0	∞	∞	0
D	0	∞	5	∞

Row Reduction cost :- 5 + 5 + 0 = 10

Step 3 :- Column Reduction :-

1) column C :- Smallest value = 5
   - 15 - 5 = 10, ∞ - 5 = ∞, 5 - 5 = 0

2) column D :- Smallest value = 0
   • No change.

	A	B	C	D
A	∞	∞	∞	∞
B	∞	∞	10	0
C	0	∞	∞	0
D	0	∞	0	∞

Column Reduction cost :- 5 + 0 = 5

Step 4 :- Calculate New Bound

New Bound = Previous Bound + cost of A → B
+ Row Reduction cost + column Reduction' cost
= 70 + 10 + 10 + 5
= 95

∴ The bound for branch A → B is 95.

Similarly,

A → C is 100

A → D is 100

Advantages :-
1) Optimal solution Guarantee
2) Efficient pruning
3) Flexibility
4) Partial Solutions Insight.

Disadvantages :-
1) High computational complexity
2) Memory usage
3) Depends on Good Heuristics
4) computational overhead.

Que- Explain 15 puzzle Problem using branch & bound with example.

⇒

-The 15-Puzzle problem is a sliding puzzle consisting of 15 numbered tiles (1 through 15) placed in a 4×4 grid with one empty space.
-The objective is to move the tiles around and reach a specified goal. Configuration, typically with the tiles arranged in ascending order (1 through 15) and the empty space at the bottom-right corner.

Steps :-
1) Initial state
2) Possible moves
3) Heuristic Evaluation
4) Branch selection
5) Repeat

Estimated cost: $\hat{C}(x) = F(x) + g(x)$

where,

$F(x)$ = Length of the path from the root to node x

$g(x)$ = Estimated shortest path length from x to goal node

= Number of non-blank tiles not in the goal Position.

Example :-

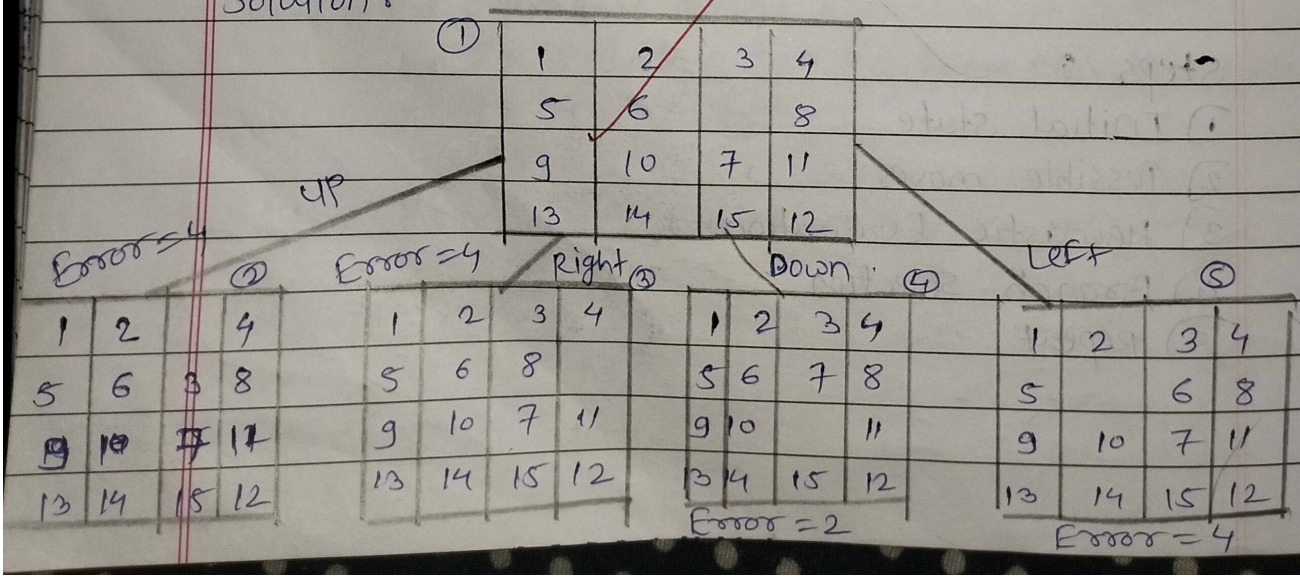
Initial state :-

1	2	3	4
5	6		8
9	10	7	11
13	14	15	12

Goal state :-

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

Solution :-



Step 1 :- Initial state

1	2	3	4
5	6	–	8
9	10	7	11
13	14	15	12

- Calculate g(x) and h(x) for this initial state.
  - g(x) = 0 (no moves yet).
  - h(x) = (manhattan distance for misplaced tiles) :-
    - Tile 7 :- move 1 step (down)
    - Tile 11 :- move 1 step (down)
    - Tile 12 :- move 1 step (left)
    - h(x) = 3
- f(x) = g(x) + h(x) = 0 + 3 = 3

Step 2 :- Generate child states

Now, we generate all possible states by sliding the blank space in four directions (up, down, left, right).

1) Move up :-

1	2	–	4
5	6	3	8
9	10	7	11
13	14	15	12

- g(x) = 1 (one move)
- Calculate h(x) :-
  - Tile 3 :- move 1 step (down)
  - Tile 7 :- move 1 step (down)
  - Tile 11 :- move 1 step (down)
  - Tile 12 :- move 1 step (left)
  - h(x) = 4
- f(x) = g(x) + h(x) = 1 + 4 = 5

2] Move left :-

1	2	3	4
5	–	6	8
9	10	7	11
13	14	15	12

- $g(x) = 1$ (one move)
- Calculate $h(x)$:-
  - Tile 7 :- Move 1 step (up)
  - Tile 11 :- move 1 step (down)
  - Tile 12 :- Move 1 step (left)
  - $h(x) = 3$
- $f(x) = g(x) + h(x) = 1 + 3 = 4$

3] Move right :-

1	2	3	4
5	6	8	–
9	10	7	11
13	14	15	12

- $g(x) = 1$ (one move)
- Calculate $h(x)$:-
- Tile 7 :- Move 1 step (up)
- Tile 11 :- Move 1 step (down)
- Tile 8 :- Move 1 step (left)
- Tile 12 :- Move 1 step (left)
- $h(x) = 4$
- $f(x) = g(x) + h(x) = 1 + 4 = 5$

Step 3 :- Process the Node with the minimum f(x).
- The nodes are added to the priority queue. we will process the node with the smallest F(x).
- states to explore :-
  - $f(x) = 5$ (move up).
  - $f(x) = 4$ (move left)

- f(x) = 5 (move right)
- The state with f(x) = 4 (move left) is processed first.

step 4 :- Expand the state with f(x) = 4

1	2	3	4
5	–	6	8
9	10	7	11
13	14	15	12

- Generate Possible moves (up, down, left, right) :-
- move up :-

1	2	3	4
5	6	–	8
9	10	7	11
13	14	15	12

- g(x) = 2 (2 moves)
- h(x) = 3
- f(x) = 2 + 3 = 5
- move down :-

1	2	3	4
5	~~6~~	~~7~~	8
9	10	–	11
13	14	15	12

- This state may be worse and can be skipped based on pruning.

Step 5 :-
- Repeat the above steps, selecting the state with the smallest f(x) from the priority queue and generating new states.
- Once the goal state is reached, the path from the the initial state to the goal state is the solution.

- Advantages :-
1) optimal solution 2) systematic Exploration 3) pruning Inefficient paths.

Disadvantages:- 1] High memory consumption
2] Time complexity
3] Lack of Parallelization
4] Difficulty with Large puzzles

Que ~~Explain 8-queen Problem using Backtraking with example.~~

Que Explain branch & bound with example.

⇒

-Branch and Bound is an algorithmic technique to solve optimization problems and decision problems, especially when the problem involves exploring all possible solutions systematically.
-It efficiently reduces the search space by "branching" into subproblems and "bounding" suboptimal solutions to discard them early.

Key steps in Branch and Bound:-
1] Branching :- Divide the problem into smaller subproblems (branches) to explore all possibilities.
2] Bounding :- Use a bound to eliminate branches that cannot lead to optimal solutions.
3] Pruning :- Stop exploring branch if it exceeds the bound or cannot lead to a better solution.

Example :- 8-Puzzle Problem

Problem :-
Initial state :-

1	2	3
4	-	5
7	8	6

Goal state :-

1	2	3
4	5	6
7	8	-

- The goal is to move the tiles into the correct positions with the least number of moves.

Steps :-

Step 1 :- Define a cost Function

$f(n) = g(n) + h(n)$

where,

$g(n)$ = number of moves made so far.

$h(n)$ = heuristic value.

Step 2 :- start at the Initial state.

- The initial state is

1	2	3
4	_	5
7	8	6

- $g(n) = 0$ (no moves made yet)
- $h(n) = 2$ (Manhattan Distance for tiles 5 and 6).
- $f(n) = g(n) + h(n) = 0 + 2 = 2$

Step 3 :- Generate All Possible Moves (Branching).

1) Move up :-

1	_	3
4	2	5
7	8	6

- $g(n) = 1$
- $h(n) = 3$ (tiles 2, 5, 6 are misplaced)
- $f(n) = 1 + 3 = 4$

2) Move Down :-

1	2	3
4	8	5
7	_	6

- $g(n) = 1$
- $h(n) = 3$ (tiles 5, 6, 8 are misplaced)
- $f(n) = 1 + 3 = 4$

3) Move Left :-

1 2 3
- 4 5
7 8 6

- g(n) = 1
- h(n) = 2 (tiles 5, 6 are misplaced)
- F(n) = 1+2 = 3

4) Move Right :-

1 2 3
4 5 -
7 8 6

- g(n) = 1
- h(n) = 1 (tile 6 is misplaced)
- F(n) = 1+1 = 2

Step 5 :- Choose the Best Move (Bounding)
The move with the lowest F(n) value is chosen.

1 2 3
4 5 -
7 8 6

F(n) = 2

Step 6 :- Repeat until the Goal state is Reached.
- Continue generating moves, calculating F(n) and selecting the move with lowest cost. The process continues until the goal state is reached.

Goal State :-

1 2 3
4 5 6
7 8 -

Advantages:-
1] optimal solutions
2] Pruning saves Time
3] Flexible Heuristics

Disadvantages:-
1] Memory Intensive
2] Time complexity
3] Heuristic Dependence.

Que- Explain 8-queen problem using Backtracking with example.

=>

-The 8-queen problem is a classic problem.
-The objective is to place 8 queens on an 8x8 chessboard such that ino two queens threaten each other.
-A queen can attak another queen if they are on the same row, column or diagonal.

Problem Statement:-
- Place 8 queens on a chessboard (8x8) grid.
- No two queens can be in the same row, column, or diagonal.

Solution Approach:-
-The backtracking algorithm is used to solve this problem. Backtracking is a methodical way of trying all possible configurations and "backing out" when a configuration fails to meet the constraints.

Backtracking steps :-

1) Start with the first row and place a queen in one of the columns.
2) Move to the next row and attempt to place a queen in a column such that no two queens threaten each other.
3) If a valid position is found, move to the next row.
4) If a valid position cannot be found, backtrack to the previous row and move the queen to the next possible column.
5) Repeat until all 8 queens are placed on the board or all possible configurations are tried.

Key constraints :-

- No two queens can be in the same column.
- No two queens can be on the same diagonal.
  - Diagonals can be identified by the difference between the row and column indexes :-
  row - col for the main diagonal and row + col for the anti-diagonal.

Example :-

Step 1 :- Start with the First Row

- place a queen in the first row, First column

Q - - - - - - -
- - - - - - - -
- - - - - - - -
- - - - - - - -
- - - - - - - -
- - - - - - - -
- - - - - - - -
- - - - - - - -

step2:- Move to the second row
- Try to place a queen in the second row; avoiding the same column and the diagonals.

Q - - - - - - -
- - - Q - - - -
- - - - - - - -
- - - - - - - -
- - - - - - - -
- - - - - - - -
- - - - - - - -
- - - - - - - -

step3:- Move to the Third Row
- Try placing a queen in the third row, avoiding columns 1 and 4 and checking the diagonals.

Q - - - - - - -
- - - Q - - - -
- - - - - - Q -
- - - - - - - -
- - - - - - - -
- - - - - - - -
- - - - - - - -
- - - - - - - -

step4:- Move to the fourth row
- Try placing a queen in the fourth row. Avoid columns 1, 4, and 7.

Q - - - - - - -
- - - Q - - - -
- - - - - Q - -
- Q - - - - - -
- - - - - - - -
- - - - - - - -
- - - - - - - -
- - - - - - - -

Steps: Continue with Further Rows

- Continue applying the same method for the remaining rows.

- If at any point you reach a row where no valid positions for the queen are possible (due to column or diagonal conflicts), backtrack to the previous row and try placing the queen in a different column.

- After trying and backtracking as needed,

Q - - - - - - - -
- - - Q - - - - -
- - - - - - - Q -
- Q - - - - - - -
- - - - - Q - - -
- - Q - - - - - -
- - - - - - Q - -
- - - - - - - - Q

- This is one possible solution where all 8 queens are placed in such a way that no two queens threaten each other.

Advantages:-

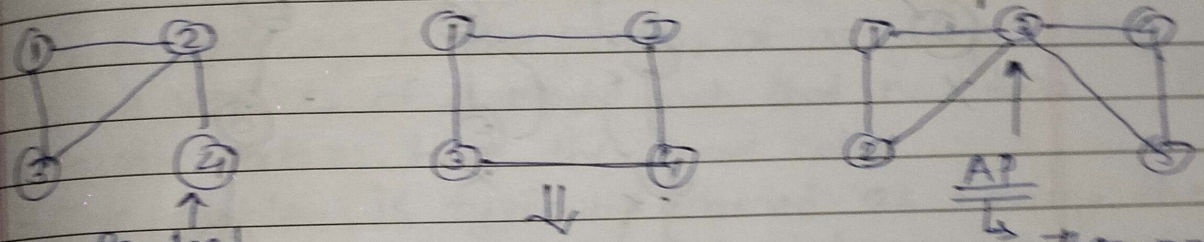
1) Systematic search
2) Memory efficiency
3) optimality

Disadvantages:-

1) Time complexity
2) No guarantee of immediate solution

Explain Hamiltonian cycle using Backtracking with Example.

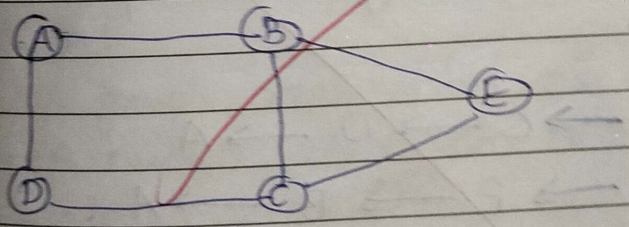
- A Hamiltonian cycle is a path in a graph that visits each vertex exactly once and ends at the starting vertex.
- using backtracking, we systematically explore all possible paths to find such a cycle.



2 → 4 → 3 → 1 → 2

2 → 1 → 3 → 4 → 2
(Hamiltonian cycle)

Example.

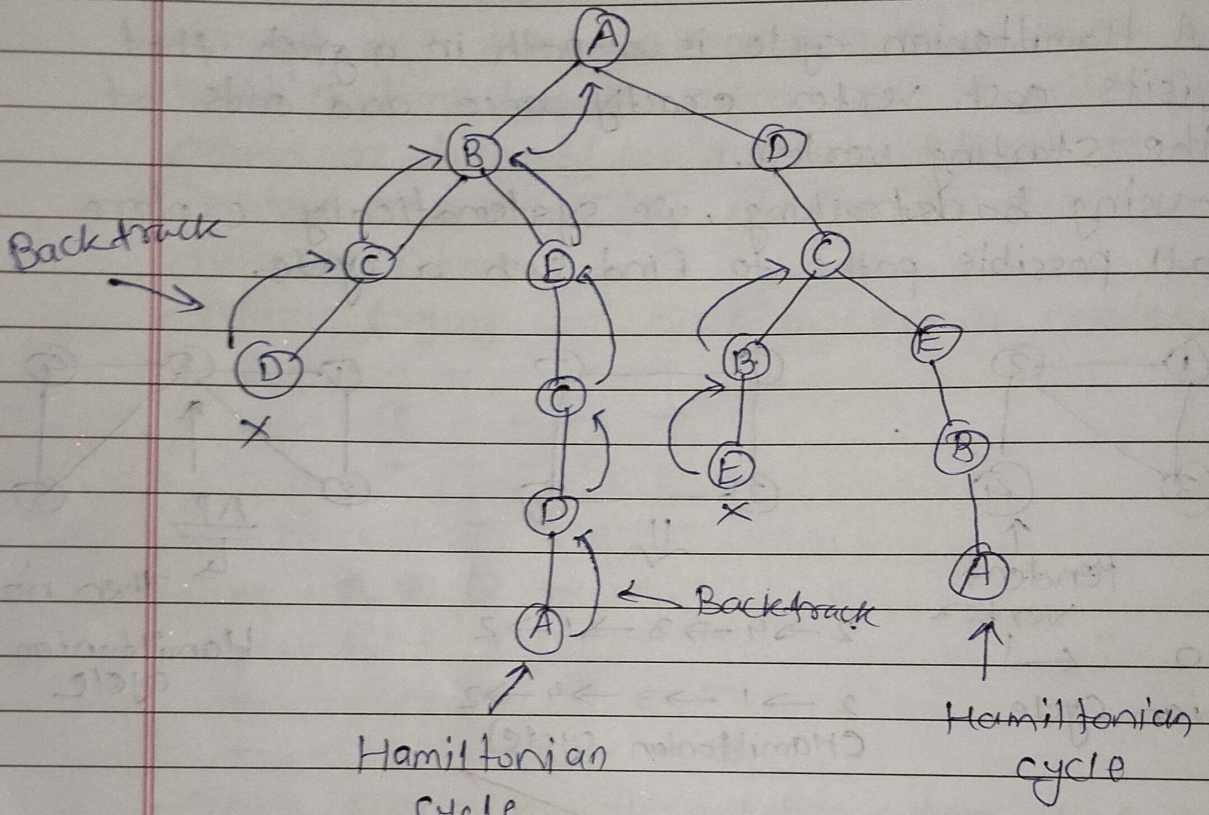


step-1 :- Adjacency matrix

graph =

	A	B	C	D	E
A	0	1	0	1	0
B	1	0	1	0	1
C	0	1	0	1	1
D	1	0	1	0	0
E	0	1	1	0	0

Step 2:- Generate space tree if adjacent node present go front otherwise Backtrack



Now, The hamiltonian cycle is present in these Example.

① A → B → E → C → D → A

② A → D → C → E → B → A

Advantages:

1. Systematic Exploration
2. simplicity
3. Versatility
4. optimal For small Graphs.

Disadvantages:-
1) High Time complexity
2) Inefficient for Large graphs
3) Memory usage
4) No Guarantee of optimization.

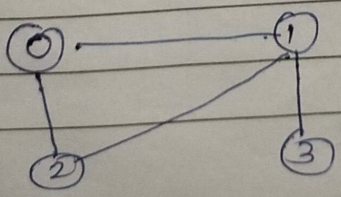
Que- Explain Graph coloring using Backtracing with example.

⇒

-Graph coloring is the process of assigning colors to the vertices of a graph such that no two adjacent vertices have the same color.
- The problem is typically framed as finding the minimum number of colors (chromatic number) needed to color the graph.
-Backtracking can be used to systematically try all possible color assignments until a valid solution is found.

Problem statement:-
Given:-
-A graph represented as an adjacency matrix
-A number m, which is the maximum number of colors available.
Goal :- Assign a color to each vertex such that:
1) No two adjacent vertices share the same color.
2) use at most m colors.

Example:- consider the following graph with 4 vertices:



Step 1 :- Adjacency matrix

graph =

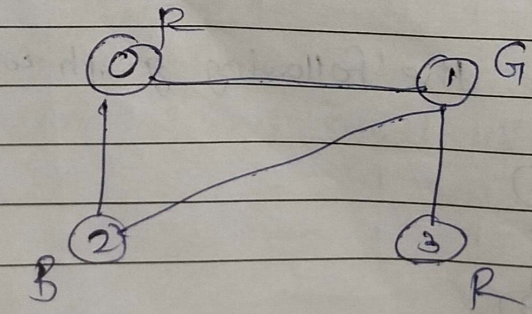
	0	1	2	3
0	0	1	1	0
1	1	0	1	1
2	1	1	0	0
3	0	1	0	0

Step 2 :- use 3 colors (m = 3) (R, G, B)

Step 3 :- color Assignment

- vertex 0 :- Assign color Red (R)
  color = [R, 0, 0, 0]
- vertex 1 :- Assign color Green (G) (adjacent to vertex 0)
  color = [R, G, 0, 0]
- vertex 2 :- Assign color Blue (B) (adjacent to vertex 0 and 1)
  Color = [R, G, B, 0]
- vertex 3 :- Assign color Red (R) (adjacent to vertex 1)
  color = [R, G, B, R]

Final graph :-



Advantages :-
1) Systematic Exploration
2) Flexibility
3) simplicity
4) Complete and Exact

Disadvantages :-
1) Exponential Time Complexity
2) Memory usage
3) Dependent on m
4) Redundant Exploration

M
28/5/24.