B. K. DHARANEESH
CSE – B II
230701072

DAA

ASSIGNMENT – 5

Backtracking

(i) N-Queen's problem

The problem of trying to n queens on a nxn chessboard, such that no queen can attack another queen

→ Generally queens can attack
(i) Horizontally
(ii) Vertically
(iii) Diagonally.

This means that no 2 queens can share the same row, column or diagonal

Backtracking Approach:

- It involve placing queens one by one in different columns and rows.
- If placing a queen in a specific position leads to a conflict, we backtrack by removing the queen & trying the next possible position.

Steps to solve:

(i) Initialize the board: start with an empty nxn board.

(ii) place the queen: Begin by placing the 1st queen in the first column of the 1st row.

(iii) check for conflicts: Ensure no 2 queens threaten each other. If a conflict occurs backtrack to the previous step.

(iv) Move to the next row: place the next queen in the next row. Continue this process for all rows.

(v) Backtrack as needed: If you cannot place a queen in a row without a conflict; remove the last placed queen and try the next side position in the previous row.

vi) Repeat: continue this process until all queens are placed or the board or all possibilities are exhausted.

(Standard chess board 8x8)

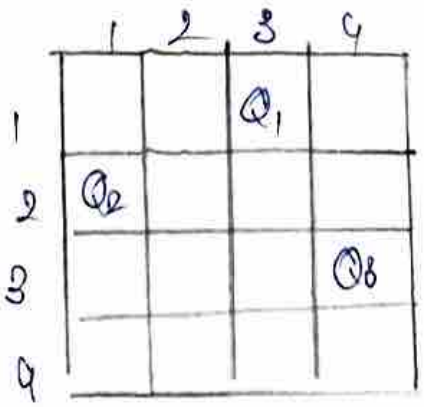
16_{C4}

State Space Tree

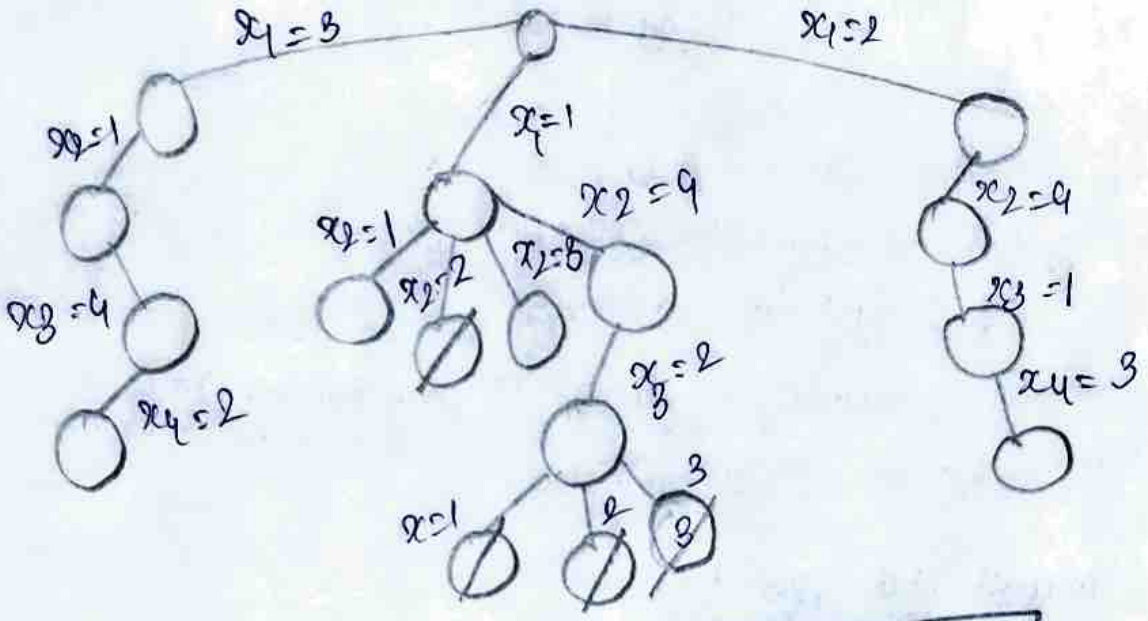
$$1 + \sum_{i=0}^{3} \left[\frac{1}{A}(4-j) \right]$$

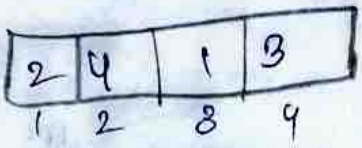
ALGORITHM:

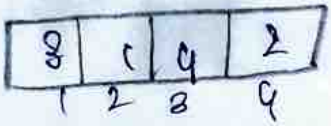
- Initialize a NxN board with all positions set to 0.
- Start from the 1st row (i=0)
- Try to place the queen in each column (j=0 to N-1) of the current row.
- Check if the queen can be placed at the position (i, j) without any conflicts check rows - ensure no queen exist in the same column check diagonals - ensure no queen is also placed in the same diagonal
- If the queen can be placed set board[i][j]=1 recursively call the algorithm for i+1.
- Recursive call returns true, then all queens are placed so, return true.
- if the r.c returns false Backtrack set board[i][j]=0 Try the next column j+1
- If all columns are tried & no solution is found return false

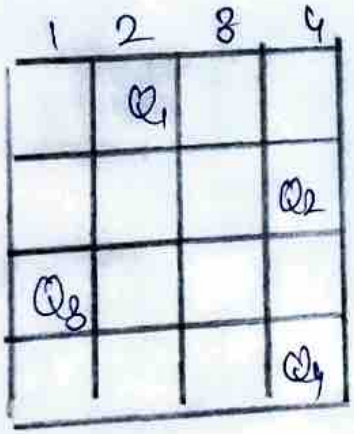


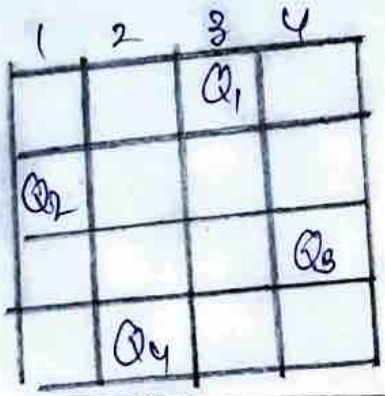
State Space Tree











→ Graph colouring

- 'G' be a graph and 'm' be an integer.
- The m- colorability decision problem is finding whether the nodes of 'G' can be coloured using 'm' colours in such a way that no two adjacent nodes have the same colour
- The m-colorability optimitization problem is finding the smallest integer 'm' for which the graph can be coloured.
- The integer m is referred to as the chromatic number.

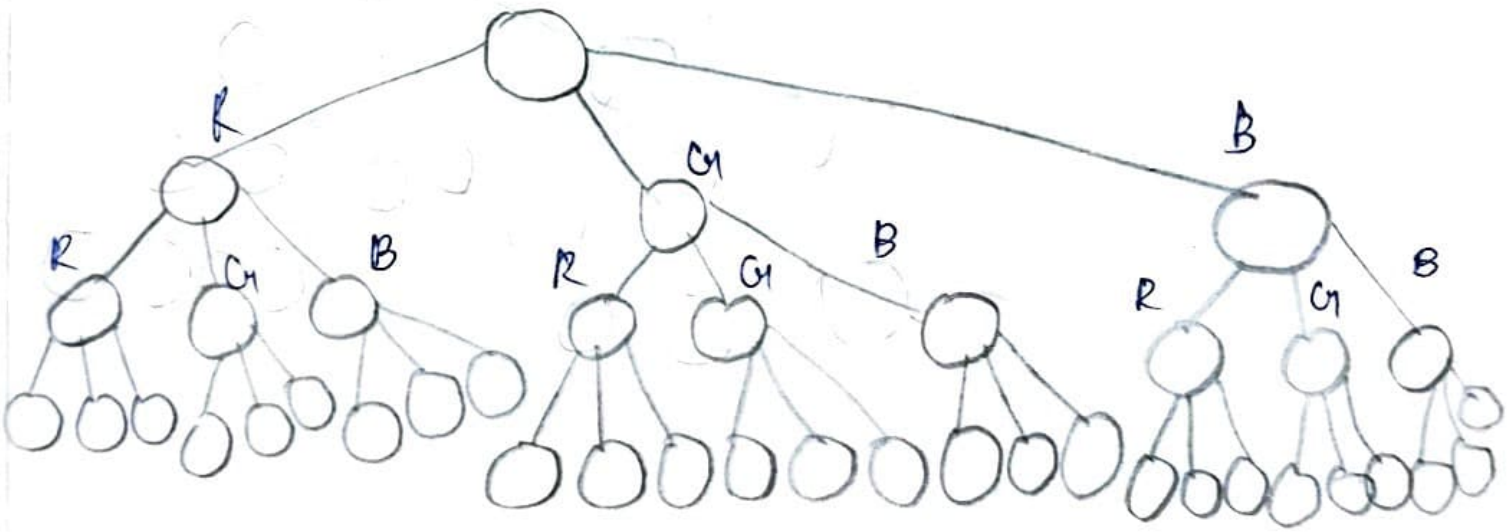
→ Steps are followed to perform valid colouring for the Graph G=(V,E)

- Order the nodes arbitrarily.
- Assign the 1st node with a color.
- Give the partial assignments of colors $(c_1, c_2, c_3 \cdots c_{i-1})$ to the i-1 nodes, try to find the colour for the ith node.
- If there are no possible color for the ith node then backtrack & change the option for the previous node

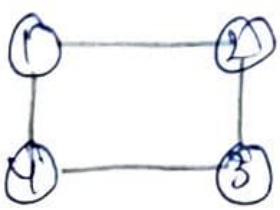
Normal State Space tree

$m = \{R, G, B\}$

3^{n+1}

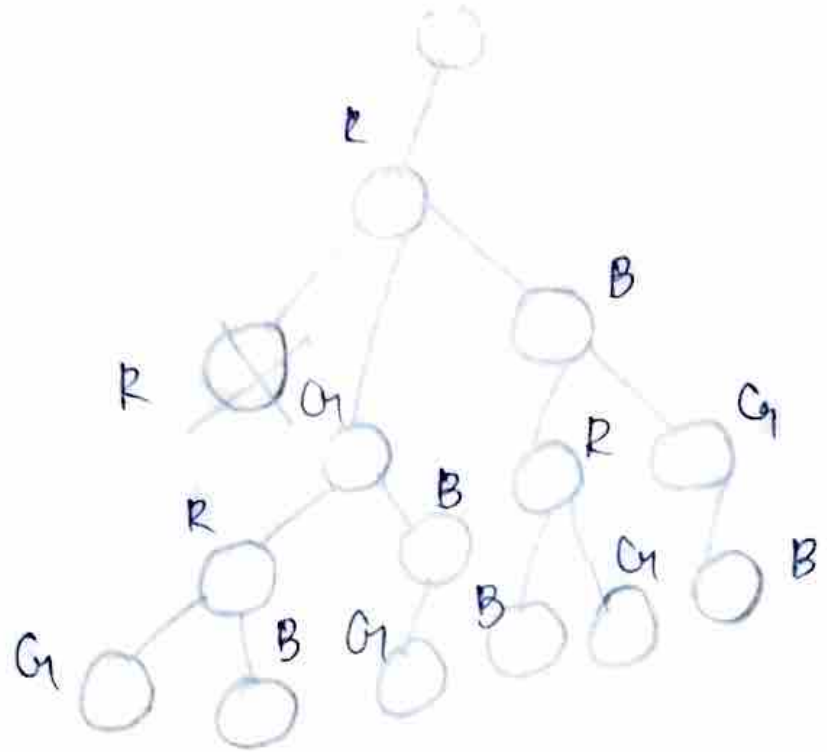


Now using Back tracking



$m = \{R, G, B\}$

State Space Tree


B
R
G
R
B
R
G
B
G
B
G
B
G
B