# Dr. D. Y. Patil College of Engineering and Innovation Varale, Talegaon

#### **Department of Computer Engineering**

Practical: Laboratory Practice-II

Class: TE Computer A.Y. 24-25 Sem-II

#### **Assignment No: 4**

## 1. Title of Assignment:

Implement a solution for a Constraint Satisfaction Problem using Branch and Bound /

Backtracking for: A] n-queens problem B] Graph Coloring Problem

#### 2. Prerequisite:

Basic knowledge of CSP, Backtracking

#### 3. Objective:

In this experiment, we will be able to do the following:

- Study how to place N queens on board with non-attacking mode using backtracking.
- What is CSP Problem.
- 4. **Outcome:** Successfully able to place N queens on board with non-attacking mode using backtracking.

#### 5. Software and Hardware Requirement:

Programming Languages: Python/Java

Operating System: Windows/Ubuntu

#### 6. Relevant Theory / Literature Survey:

#### **Constraint Satisfaction Problem**

## CSP means solving a problem under certain constraints or rules

CSP depends on three components X: It

is a set of variables

D: It is a set of domains

There is a specific domain for each variable.

Domain is possible values each variable can take.

C: It is a set of constraints which are followed by the set of variables. Rules that restrict how variables can be assigned.

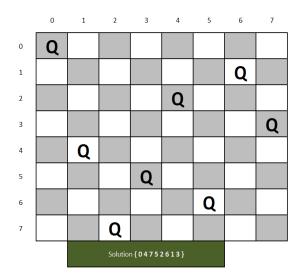
#### **Backtracking Algorithm**

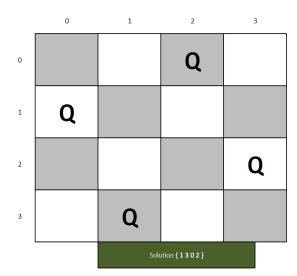
Backtracking is an algorithmic-technique for solving problems recursively by trying to build a solution incrementally, one piece at a time, removing those solutions that fail to satisfy the constraints of the problem at any point of time (by time, here, is referred to the time elapsed till reaching any level of the search tree).

## A] n-Queens problem

The n-Queens problem is a puzzle of placing exactly N queens on an NxN chessboard, such that no two queens can attack each other in that configuration. Thus, no two queens can lie in the same row, column or diagonal.

The idea is to place queens one by one in different columns, starting from the leftmost column. When we place a queen in a column, we check for clashes with already placed queens. In the current column, if we find a row for which there is no clash, we mark this row and column as part of the solution. If we do not find such a row due to clashes then we backtrack and return false.





# **N-Queens problem Algorithm**

- 1) Start in the leftmost column
- 2) If all queens are placed return true
- 3) Try all rows in the current column. Do the following for every tried row.
  - a) If the queen can be placed safely in this row then mark this [row, column] as part of the solution and recursively check if placing the queen here leads to a solution.
  - b) If placing the queen in [row, column] leads to a solution then return true.
  - c) If placing queen doesn't lead to a solution then unmark this [row, column] (Backtrack) and go to step (a) to try other rows.
- 4) If all rows have been tried and nothing worked, return false to trigger backtracking.

#### **B**| Graph-Coloring Problem

Graph Colouring is a classic Constraint Satisfaction Problem (CSP) where the goal is to assign colours to vertices of a graph such that:

- 1. No two adjacent vertices share the same colour (edge constraint).
- 2. Minimize the number of colours used (optional optimization).

Formulating Graph Coloring as a CSP

- 1. Variables (V): Each vertex of the graph is a variable.
- 2. Domain (D): A set of available colors (e.g., {Red, Green, Blue}).
- 3. Constraints (C): If two vertices are connected by an edge, they must have different colors.

Example: Colouring a Graph

Consider the following graph:

#### **Step 1: Define the CSP Components**

- 1. Variables (V): A, B, C, D
- 2. Domain (D): {Red, Green, Blue}
- 3. Constraints (C):
  - o  $A \neq B$
  - o  $A \neq C$
  - o  $B \neq C$
  - o  $B \neq D$
  - o  $C \neq D$

#### **Step 2: Solve the Graph Coloring CSP**

We will use the Backtracking algorithm to assign colors:

- 1. Assign A = Red.
- 2. For B, valid options: {Green, Blue}. Choose B = Green.
- 3. For C, valid options: {Green, Blue}. Choose C = Blue.
- 4. For D, valid options:  $\{\text{Red}, \text{Blue}\}\ (\text{since B} = \text{Green}).$  Choose D = Red.

Solution:

$$A = Red, B = Green, C = Blue, D = Red.$$

## Step 3: Techniques for Solving Graph Colouring CSP

- 1. Backtracking: Explore all possibilities and backtrack if a conflict occurs.
- 2. Forward Checking: After assigning a colour, eliminate invalid options for adjacent vertices.
- 3. Minimum Remaining Values (MRV): Choose the vertex with the fewest legal colour option

## **Branch and Bound**

Branch and bound is an algorithm design paradigm which is generally used for solving combinatorial optimization problems. These problems are typically exponential in terms of time complexity and may require exploring all possible permutations in the worst case. The Branch and Bound Algorithm technique solves these problems relatively quickly.

Parameter	Backtracking	Branch and Bound
	Backtracking is used to find all	Branch-and-Bound is used to solve
Approach	possible solutions available to a	optimization problems. When it realizes
	problem. When it realizes that it has	that it already has a better optimal
	made a bad choice, it undoes the last	solution that the pre-solution leads to, it
	choice by backing it up. It searches the	abandons that pre-solution. It
	state space tree until it has found a	completely searches the state space tree
	solution for the problem.	to get an optimal solution.
Traversal	Backtracking traverses the state	Branch-and-Bound traverse the tree in
	space tree by DFS (Depth First	any manner, DFS or BFS.
	Search) manner.	
	Backtracking involves a feasibility	Branch-and-Bound involves a
Function	function.	bounding function.
	Backtracking is used for solving	Branch-and-Bound is used for
Problems	Decision Problems.	solving the optimization Problem.
Searching	In backtracking, the state space tree	In Branch-and-Bound as the
	is searched until the solution is	optimum solution may be present
	obtained.	anywhere in the state space tree,
		so the tree needs to be searched
		completely.
	Backtracking is more efficient.	Branch-and-Bound is less efficient.
Efficiency		
	Useful in solving N-Queen Problem,	Useful in solving Knapsack
Applications	Sum of subset.	Problem, Traveling Salesman
		Problem.

# 7. Questions:

- **Q. 1:** Which are the constraints required to solve the N- Queens Problem?
- Q. 2: Explain Graph Coloring problem with Example
- Q. 3: Compare Backtracking and branch and bound methods.
- **Q. 4**: What do you mean by Constraint satisfaction problem?
- **8. Conclusion:** In This way we have studied how to solve the CSP problem and how to place N queens on board with non-attacking mode using backtracking. Also studied how to solve Graph Coloring problem