Page:
Date: / /

Assignment - 04

- TITLE : Constraint satisfaction Problem

- Problem Statement : Implement crypt-arithmatic problem or n-queens or graph coloring problem (Branch & Bound and Backtracking)

- Objective : To learn and implement constraint satisfaction problem.

- Outcome : Student will able to :
  understand and implement constraint satisfaction problem.

- Software Packages and Hardware Requirements : Operating system : 64 bit open source Linux or windows
  Programming Languages : Python | JAVA
  Python Libraries, Python frameworks

- Theory :
  Constraint Satisfaction Problem :
  A constraint satisfaction problem (CSP) consist of :
  - a set of variables
  - a domain for each variable
  - a set of constraint

  The aim is to choose a value for each variable so that the resulting possible world satisfies the constraint we want a model of the constraint.
  A finit CSP has a finite set of variable and

finite domain for each variable. Many of the methods considered in this chapter only work for finite CSPs, although some are designed for infinite even continuous domains.
The Multidimensional aspects of these problems, where each variable can be seen as a separate dimension, makes them difficult to solve but also provides structure that can be exploited.
Given a CSP there are a number of tasks that can be performed:

- Determine whether or not there is a model
- find a model
- Find all of the models or enumerate the models.
- Count the number of models
- Find the best model given a measure of how good models are
- Determine whether some statement holds in all models.

- Backtracking :

Backtracking is an algorithm technique for solving problems recursively by trying to build a solution incrementally one peice at a time, removing those solution that fail to satisfy the constraints of the problem at any time, removing those solutions that fail to satisfy the constraint of the problem at any time of point.
here are three types of problem in backtracking:
1. Decision problem: In this we search for a desible solution

Page:
Date: / /

2. Optimization Problem : In this, we search for best solution
3. Enumeration Problem : In this we find all feasible solution.

Pseudo code for Backtracking :
1. Recursive backtracking solution :

```
void findSolution (n, other params):
    if (found a solution):
        solutionFound = solutionFound + 1;
        displaySolution ();
        if (solutionFound >= solution target):
            return
    for (val = first to last):
        if (isValid (val, n)):
            applyValue (val, n);
            findSolution (n+1, other params);
```

Graph coloring solution:
Given an undirected graph and number of M, determine if the graph can be colored with at most M colors such that no two adjacent vertices of the graph are colored with the same color. Here coloring of a graph means the algorithm of colors to all vertices.
Input :
1] A 2D array graph [V][V] where V is the number of vertices in graph & graph [V][V] is adjacency matrix representation. A value graph[i][j] is 1 if there is a direct edge from i to j. otherwise graph[i][j] is 0
2] An integer M which is the maximum value of colors

that can be used.

Output : An array color [v] that should have number from 1 to M. color[i] should represent the color assigned to the i^{th} vertex. The code should return false if the graph cannot be colored with M colors.

- Test case :

Description	Input	Output
i] Graph Entered	0 1 1 1 1 0 1 0 1 1 0 1 1 0 1 0 M = 3	Color Matrix : [1, 2, 3, 2]

- Conclusion: Thus we successfully implemented constraint satisfaction problem for Graph coloring.