DPU
Dr. D. Y. Patil Institute of Technology

Assignment - 5

- Title :- CSP using DFS for n-queens problem.

- Problem Statement :- Implement a solution for constraint satisfaction problem using branch & bound and backtracking for n-queens problem.

- Objective :-

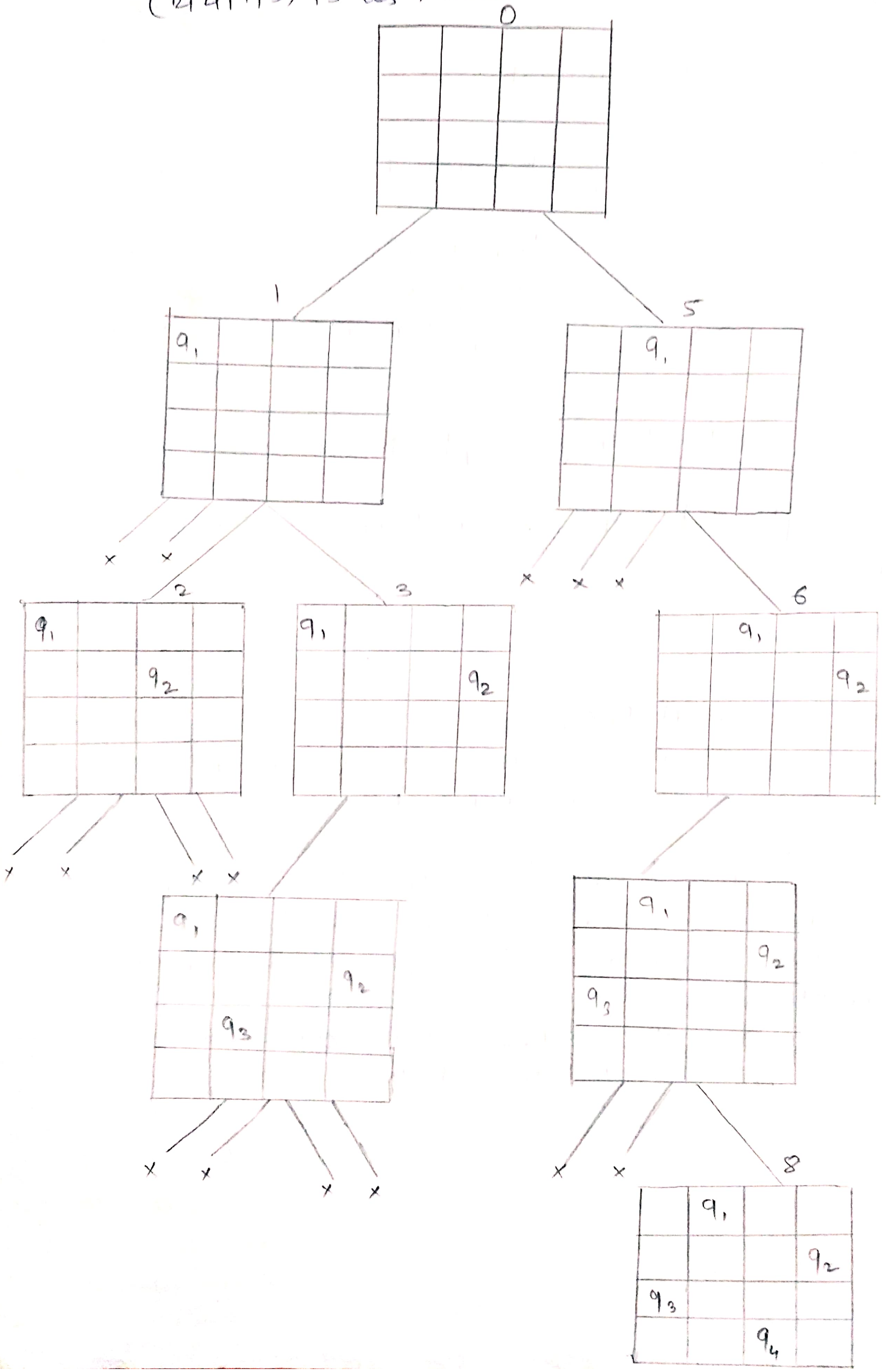
> To understand the concept of constraint satisfaction problem.
> To implement DFS search techniques for N-queens problem.

- Theory :-

N-queens :- N-Queens problem is to place n-queens in such a manner on an n x n chessboard that no queens attack each other by being in the same row, column or diagonal. It can be seen that for n = 1, the problem has a trivial solution & no solution exists for n = 2 & n = 3. So first we will consider the 4 queens problem & then generate it to n-queens problem.

Given a 4x4 chessboard & number the rows & column of the chessboard 1 through 4.

Example: The implicit tree for 4-queen problem for a solution (2,4,1,3) is as follows:



Since, we have to place 4 queens such as q_1 q_2 q_3 & q_4 on the chessboard, such that no two queens attack each queen must be placed on a different row i.e. we put queen "i" on row "i".
Now, we place queen q_1 in the very first acceptable position (1, 1). Next, we put queen q_2 so that both these queens do not attack each other. We find that if we place q_2 in column 1 & 2, then the dead end is encountered. Thus the first acceptable position for q_2 in column 3, i.e. (2, 3) but then no position is left for for placing queen 'q_3' safely. So we backtrack one step & place the queen 'q_2' in (2, 4), the next best possible solution.

Then we obtain the position for placing 'q_3' which is (3,2). But later this position also leads to a dead end, & no place is found where 'q_4' can be placed safely. Then we have to backtrack till 'q_1' & place it to (1,2) & then all other queens are placed safely by moving q_2 to (2,4), q_3 to (3,1) & q_4 to (4,3). That is we get the solution (2,4,1,3). This is one possible solution for the 4-queens problem. For another possible solution, the whole method is repeated for all partial solutions. The other solutions for 4-queens problems is (3,1,4,2) i.e.

DPU
Dr. D. Y. Patil Institute of Technology

Psuedocode for N-queens :

If two queens are placed at position (i,j) & (k,l).
Then they are on same diagonal only if (i-j) = k-l or
the first equation implies that j-l = i-k.
The second equation implies that j-l = k-i.
Therefore, two queens lie on the duplicate diagonal if
& only if |j-l| = |i-k|.

```
Place (k, i)
{
  For j ← 1 to k-1
  do if (x[j] = i)
  or (Abs x [j] - i) = (Abs (j-k))
  then return false;
  return true;
}
 N-Queens (k,n)
{
   For i ← 1 to n
      do if Place (k, i) then
   {
    x[k] ← i;
    if (k==n) then
    write (x[1...n]);
    else
    N-Queens (k+1, n);
   }
}
```

DPU

Dr. D. Y. Patil Institute of Technology

- Algorithm :-

1. Start in the leftmost column
2. If all queens are placed return true.
3. Try all rows in the current column. Do following for every tried row.
   a] If the queen can be replaced safely then mark this [row, column] as part of the solution & recursively check if placing queen here leads to a solⁿ
   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) & go to step (a) to try other rows.
4. If all rows have been tried & nothing worked, return false to trigger backtracking.

- Complexity Analysis:-

> The time complexity is $O(n^2)$ because we are selectin if we can put or not put a Queen at that place.

> The space is the board size times the result.

- Conclusion:

Thus, we have implemented DFS for n-queens problem.