Green University of Bangladesh

Department of Computer Science and Engineering (CSE)
Faculty of Sciences and Engineering
Spring 2024, B.Sc. in CSE (DAY)

Lab Report NO # 06

Course Title: Artificial Intelligence Lab
Course Code: CSE 316 Section: CSE 213 – D1

Problems / Tasks / Domains:

Lab Exercise (Submit as a report)

- The N-queens puzzle is the problem of placing N queens on a (N×N) chessboard such that no two queens can attack each other. Find all distinct solution to the N-queen problem.
  - Hint: For N = 4 there are two possible solutions -

0	0	1	0
1	0	0	0
0	0	0	1
0	1	0	0

Solution 1

0	1	0	0
0	0	0	1
1	0	0	0
0	0	1	0

Solution 2

Student Details

Name	ID
Md. Shahidul Islam Prodhan	**213902017**

Lab Assigned Date: 7th May 2024

Submission Date: 7th May 2024

Course Teacher's Name: Fairuz Shaiara, Lecturer

[For Teacher's use only: Don't write anything inside this box]

Lab Report Status

Marks:	**Signature:**
Comments:	**Date:**

1. TITLE OF THE LAB REPORT EXPERIMENT

Solve N-Queen Problem Using Backtracking Algorithm.

2. OBJECTIVES/AIM

- To understand how backtrack algorithm performs to solve constraint satisfaction problem
- To understand how N-queen problem is solvable by using backtrack method.

3. PROCEDURE / ANALYSIS / DESIGN

Procedure:
The N-Queens problem-solving approach implemented in this report follows a backtracking algorithm. The procedure involves recursively exploring all possible placements of queens on the chessboard, ensuring that no two queens attack each other.

Analysis:

- **Initialization:**
  1. The program starts by taking user input for the number of queens to be placed on the chessboard.
  2. An instance of the NQueen class is created with the input value.

- **Backtracking Algorithm:**
  1. The program utilizes a backtracking algorithm to explore all possible configurations of queen placements.
  2. The solve_nq_util method recursively tries to place queens on the board, column by column, while ensuring that no two queens attack each other.
  3. If a solution is found, it is appended to the list of solutions.

- **Printing Solutions:**
  1. Once all solutions are found, the program prints the total number of distinct solutions and then prints each solution's configuration.

Design:

- **Class Structure:**
  1. The program is structured around the NQueen class, which encapsulates the methods and attributes necessary for solving the N-Queens problem.
  2. Methods such as print_solution, is_safe, solve_nq_util, and solve_nq are defined within the class to facilitate the solution process.

- **Data Storage:**

Solutions are stored in a list within the NQueen class, allowing for easy retrieval and printing.

- **User Interaction:**

The program interacts with the user through the console, prompting for the number of queens to be placed on the chessboard.

4. IMPLEMENTATION

N-Queen_distinct_LR6.py

D: > Spring 2024 [7th Semester] > CSE316_Artificial Intelligence Lab > lab report 6 > N-Queen_

```python
class NQueen:
    def __init__(self, a):
        self.N = a
        self.solutions = []

    def print_solution(self, board):
        for i in range(self.N):
            for j in range(self.N):
                print(" " + str(board[i][j]) + " ", end="")
            print()
        print()
                                        #shahidul_213902017
    def is_safe(self, grid, row, col):
        for i in range(col):
            if grid[row][i] == 1:
                return False

        for i, j in zip(range(row, -1, -1), range(col, -1, -1)):
            if grid[i][j] == 1:
                return False

        for i, j in zip(range(row, self.N), range(col, -1, -1)):
            if grid[i][j] == 1:
                return False

        return True
```

```python

    def solve_nq_util(self, grid, col):
        if col >= self.N:
            self.solutions.append([row[:] for row in grid])
            return

        for i in range(self.N):
            if self.is_safe(grid, i, col):
                grid[i][col] = 1
                self.solve_nq_util(grid, col + 1)
                grid[i][col] = 0

    def solve_nq(self):
        grid = [[0] * self.N for _ in range(self.N)]
        self.solve_nq_util(grid, 0)
        return self.solutions


def main():
    n = int(input("Number of queens to place: "))
    queen = NQueen(n)
    solutions = queen.solve_nq()
    print("Total distinct solutions:", len(solutions))
    for idx, solution in enumerate(solutions, start=1):
        print("Solution", idx)
        queen.print_solution(solution)


if __name__ == "__main__":
    main()

```

5. TEST RESULT / OUTPUT

Test Cases:

Num of n (queen)	Num of Total Distinct Solution
1	1
2	0
3	0
4	2
5	10
6	4

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\theSh> & C:/Users/theSh/AppData/Local/Micr
Semester]/CSE316_Artificial Intelligence Lab/lab repor
Number of queens to place: 1
Total distinct solutions: 1
Solution 1
 1

PS C:\Users\theSh> & C:/Users/theSh/AppData/Local/Micr
Semester]/CSE316_Artificial Intelligence Lab/lab repor
Number of queens to place: 2
Total distinct solutions: 0
PS C:\Users\theSh> & C:/Users/theSh/AppData/Local/Micr
Semester]/CSE316_Artificial Intelligence Lab/lab repor
Number of queens to place: 3
Total distinct solutions: 0
PS C:\Users\theSh> & C:/Users/theSh/AppData/Local/Micr
Semester]/CSE316_Artificial Intelligence Lab/lab repor
```

```
Semester]/CSE316_Artificial Intelligence Lab/lab repor
Number of queens to place: 4
Total distinct solutions: 2
Solution 1
 0  0  1  0
 1  0  0  0
 0  0  0  1
 0  1  0  0

Solution 2
 0  1  0  0
 0  0  0  1
 1  0  0  0
 0  0  1  0
```

```
PS C:\Users\theSh> & C:/Users/theSh/
Semester]/CSE316_Artificial Intellig
Number of queens to place: 5
Total distinct solutions: 10
Solution 1
 1  0  0  0  0
 0  0  0  1  0
 0  1  0  0  0
 0  0  0  0  1
 0  0  1  0  0

Solution 2
 1  0  0  0  0
 0  0  1  0  0
 0  0  0  0  1
 0  1  0  0  0
 0  0  0  1  0

Solution 3
 0  0  1  0  0
 1  0  0  0  0
 0  0  0  1  0
 0  1  0  0  0
 0  0  0  0  1

Solution 4
 0  0  0  1  0
 1  0  0  0  0
 0  0  1  0  0
 0  0  0  0  1
 0  1  0  0  0

Solution 5
 0  1  0  0  0
 0  0  0  1  0
 1  0  0  0  0
 0  0  1  0  0
 0  0  0  0  1
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
Solution 5
 0  1  0  0  0
 0  0  0  1  0
 1  0  0  0  0
 0  0  1  0  0
 0  0  0  0  1

Solution 6
 0  0  0  0  1
 0  0  1  0  0
 1  0  0  0  0
 0  0  0  1  0
 0  1  0  0  0

Solution 7
 0  1  0  0  0
 0  0  0  0  1
 0  0  1  0  0
 1  0  0  0  0
 0  0  0  1  0

Solution 8
 0  0  0  0  1
 0  1  0  0  0
 0  0  0  1  0
 1  0  0  0  0
 0  0  1  0  0

Solution 9
 0  0  0  1  0
 0  1  0  0  0
 0  0  0  0  1
 0  0  1  0  0
 1  0  0  0  0

Solution 10
 0  0  1  0  0
 0  0  0  0  1
 0  1  0  0  0
 0  0  0  1  0
 1  0  0  0  0

PS C:\Users\theSh> & C:/Users/theSh/AppD
Semester]/CSE316_Artificial Intelligence
```

PROBLEMS OUTPUT DEBUG CONSOLE TE

```
PS C:\Users\theSh> & C:/Users/theSh
Semester]/CSE316_Artificial Intelli
Number of queens to place: 6
Total distinct solutions: 4
Solution 1
 0  0  0  1  0  0
 1  0  0  0  0  0
 0  0  0  0  1  0
 0  1  0  0  0  0
 0  0  0  0  0  1
 0  0  1  0  0  0

Solution 2
 0  0  0  0  1  0
 0  0  1  0  0  0
 1  0  0  0  0  0
 0  0  0  0  0  1
 0  0  0  1  0  0
 0  1  0  0  0  0

Solution 3
 0  1  0  0  0  0
 0  0  0  1  0  0
 0  0  0  0  0  1
 1  0  0  0  0  0
 0  0  1  0  0  0
 0  0  0  0  1  0

Solution 4
 0  0  1  0  0  0
 0  0  0  0  0  1
 0  1  0  0  0  0
 0  0  0  0  1  0
 1  0  0  0  0  0
 0  0  0  1  0  0

PS C:\Users\theSh>
```

6. ANALYSIS AND DISCUSSION

The implemented N-Queens solver exhibits robustness in finding valid configurations while adhering to the problem's constraints. It effectively utilizes a backtracking algorithm to explore all possible placements, ensuring correctness in the solutions. However, its efficiency is contingent on the board size, with larger sizes leading to exponential growth in computation time. Memory usage could also become a concern, especially for larger boards with numerous solutions. The program's user interface is intuitive, simplifying user interaction and solution presentation. Nonetheless, addressing symmetries to eliminate duplicate solutions remains a key area for improvement, enhancing solution accuracy. Overall, while the solver effectively tackles the N-Queens problem, enhancements in efficiency, memory management, and symmetry handling could further optimize its performance.

7. SUMMARY

In summary, the implemented solution effectively solves the N-Queens problem using a backtracking algorithm. It provides a robust framework for finding and printing all distinct solutions for a given board size. While the solution correctly identifies solutions and interacts with the user, further enhancements could be made to improve efficiency and handle symmetries to ensure that only unique solutions are counted.