Ex No: 1

Date:

N-Queens Problem

Aim:

To solve the N-Queens Problem using the backtr algorithm in Python, where the goal is to place queens on an N×N chess board such that no queens threaten each other. The program will N as input from the user and provide all P solution for placing the queens.

Algorithm:

1. Initialize the Borad
   - Create an N×N board initialized with z Each cell represent an empty spot where a might be placed.
2. Define the Backtracking function:
   - Create a recursive function solveNQueen (board, col) that attempts to place queen board column by column.
   - If the column index equals N, it means all are successfully placed, and we have found Solution.
3. Check Safety:
   - Create a helper function is Safe (boa to check whether placing a queen at [row][col] is safe. Ensure no other quee on the same row, column or diagonal.

19/11/2024 17:52

4. Place the Queen
- For each row in the current column, check if it's safe to place a queen. If safe, place the queen and make a recursive call to place queen in the next column.

5. Backtrack
- If placing a queen in any row of the current column doesn't lead to a solution, backtrack by removing the queen and trying the next row.

6. Print the Solution:
- Once all queens are placed successfully, add the board configuration to the list of solutions.

7. Handle Multiple Solutions:
- Continue searching for solutions by backtracking to find all possible ways to place N queens.

8. User Input:
- Take input N from the user to determine the size of the board and the number of queens.

Program:

```
def isSafe(board, row, col, N):
    for i in range(col):
        if board[row][i] == 1:
            return False
    for i,j in zip(range(row, -1, -1), range(col, -1, -1)):
        if board[i][j] == 1:
            return False
```

```
        for i, j in zip(range(row, N), range(col, -1, -1)):
            if board[i][j] == 1:
                return False
    return True
def solveNQueens(board, col, N):
    if col >= N:
        return True
    for i in range(N):
        if is_safe(board, i, col, N):
            board[i][col] = 1
            if solveNQueens(board, col+1, N):
                return True
            board[i][col] = 0
    return False
def printSolution(board, N):
    for i in range(N):
        for j in range(N):
            if board[i][j] == 1:
                print("Q", end=" ")
            else:
                print(".", end=" ")
        print()
    print("\n")
def solveNQueensProblem(N):
    board = [[0 for _ in range(N)] for _ in range
    if not solveNQueens(board, 0, N):
        print("Solution does not exist")
```

19/11/2024 17:52

return False

Print Solution (board, N)

return True

if __name__ == "__main__":

N = int(input("Enter the value of N (4, 6, 8, etc.)

SolveNQueens Problem(N)

Output:

Input = N = 4

Q . . .

. . Q .

. Q . .

. . . Q

or

. . . Q .

Q . . .

. . . Q

. Q . .

Result: