



Experiment No. : 7

Title: N Queen problem using Backtracking



Batch: B-4

Roll No.: 16010422234

Name: Chandana Ramesh Galgali

Experiment No.: 07

Aim: To Implement N Queen problem using Backtracking & Virtual Lab on N Queen algorithm using Backtracking

Algorithm of N Queen problem:

The N-Queen problem involves placing N queens on an $N \times N$ chessboard so that no two queens threaten each other. The backtracking algorithm is a common approach to solve this problem. Here's a high-level description of the algorithm:

1. Start in the first row:
Place the queen in the first available column.
 2. Move to the next row:
Try to place another queen in a safe column (a column not under attack from any previously placed queen).
 3. Check for threats:
Ensure no two queens are in the same row, column, or diagonal.
 4. Backtrack if needed:
If no column is safe in the current row, backtrack to the previous row and move the queen to the next safe column. Then proceed again.
 5. Repeat:
Continue this process for each row until all queens are placed safely or all possibilities are exhausted.
 6. Record the solution:
Once all queens are placed, store the board configuration as a solution.
 7. Find all solutions:
After finding one solution, backtrack to find the next possible configuration.
-

Working of N Queen problem:

The backtracking process involves recursively placing queens on the board, moving row by row, and checking for clashes with already placed queens. When placing each queen, the algorithm verifies if the current column and the diagonals are free from other queens.

Virtual Lab of N Queen problem:

Simulate Manual steps of N Queen Problem using Backtracking for 5x5 Matrix using following Link and display the Output.

<https://vsit.edu.in/vlab/AI/main%20ckt/simulator/4QueenSolution.htm>

Simulation of 4 QUEENS/ N QUEENS

The problem here is to place queens on the empty chessboard in such a way that no queen attacks any other queen. Select Play mode as manual and place the queen on the board to get the solution. Or select Play mode as Simulation to get the solution directly.

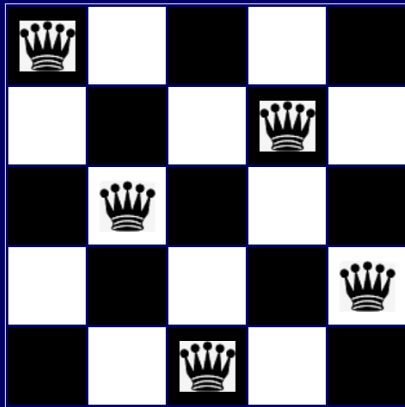
Please select the play mode: ☒ Manual ☐ Simulation

Please select Simulation Speed: ☒ Slow ☐ Moderate ☐ Fast

Please select Boardsize N x N: 5 x 5

Number of solutions found: 1

[13524]



Simulation of 4 QUEENS/ N QUEENS

The problem here is to place queens on the empty chessboard in such a way that no queen attacks any other queen. Select Play mode as manual and place the queen on the board to get the solution. Or select Play mode as Simulation to get the solution directly.

Please select the play mode: ☒ Manual ☐ Simulation

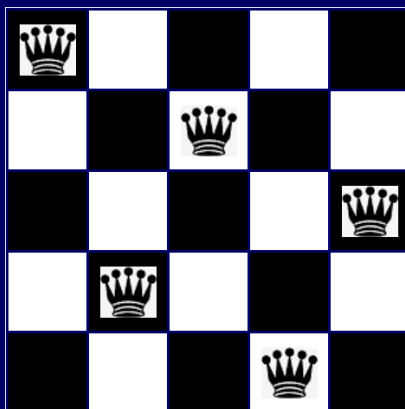
Please select Simulation Speed: ☒ Slow ☐ Moderate ☐ Fast

Please select Boardsize N x N: 5 x 5

Number of solutions found: 2

[13524]

[14253]



Simulation of 4 QUEENS/ N QUEENS

The problem here is to place queens on the empty chessboard in such a way that no queen attacks any other queen. Select Play mode as manual and place the queen on the board to get the solution. Or select Play mode as Simulation to get the solution directly.

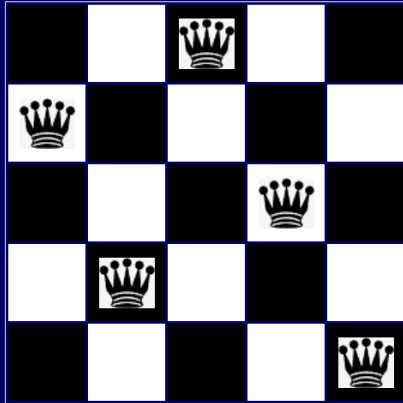
Please select the play mode: ☒ Manual ☐ Simulation

Please select Simulation Speed: ☒ Slow ☐ Moderate ☐ Fast

Please select Boardsize N x N: 5 x 5

Number of solutions found: 3

[13524]
[14253]
[24135]



Simulation of 4 QUEENS/ N QUEENS

The problem here is to place queens on the empty chessboard in such a way that no queen attacks any other queen. Select Play mode as manual and place the queen on the board to get the solution. Or select Play mode as Simulation to get the solution directly.

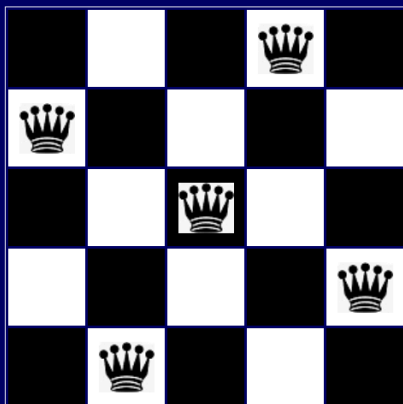
Please select the play mode: ☒ Manual ☐ Simulation

Please select Simulation Speed: ☒ Slow ☐ Moderate ☐ Fast

Please select Boardsize N x N: 5 x 5

Number of solutions found: 4

[13524]
[14253]
[24135]
[25314]



Simulation of 4 QUEENS/ N QUEENS

The problem here is to place queens on the empty chessboard in such a way that no queen attacks any other queen. Select Play mode as manual and place the queen on the board to get the solution. Or select Play mode as Simulation to get the solution directly.

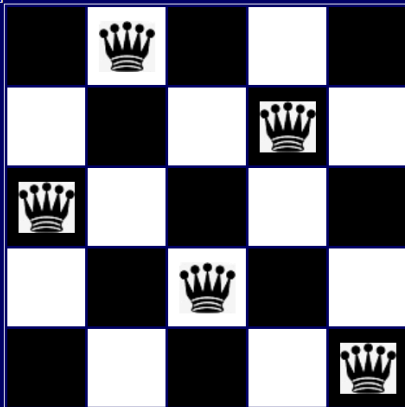
Please select the play mode: ☒ Manual ☐ Simulation

Please select Simulation Speed: ☒ Slow ☐ Moderate ☐ Fast

Please select Boardsize N x N: 5 x 5

Number of solutions found: 5

[13524]
[14253]
[24135]
[25314]
[31425]



Simulation of 4 QUEENS/ N QUEENS

The problem here is to place queens on the empty chessboard in such a way that no queen attacks any other queen. Select Play mode as manual and place the queen on the board to get the solution. Or select Play mode as Simulation to get the solution directly.

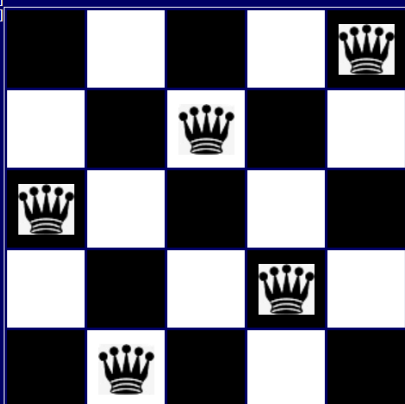
Please select the play mode: ☒ Manual ☐ Simulation

Please select Simulation Speed: ☒ Slow ☐ Moderate ☐ Fast

Please select Boardsize N x N: 5 x 5

Number of solutions found: 6

[13524]
[14253]
[24135]
[25314]
[31425]
[35241]



Simulation of 4 QUEENS/ N QUEENS

The problem here is to place queens on the empty chessboard in such a way that no queen attacks any other queen. Select Play mode as manual and place the queen on the board to get the solution. Or select Play mode as Simulation to get the solution directly.

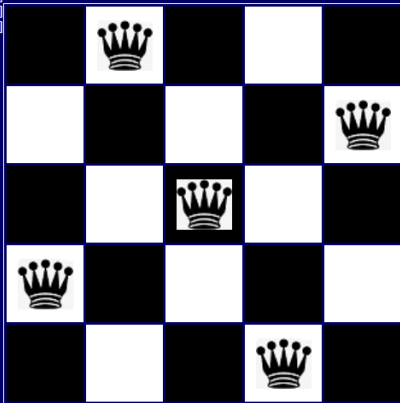
Please select the play mode: ☒ Manual ☐ Simulation

Please select Simulation Speed: ☒ Slow ☐ Moderate ☐ Fast

Please select Boardsize N x N: 5 x 5

Number of solutions found: 7

[13524]
[14253]
[24135]
[25314]
[31425]
[35241]
[41352]



Simulation of 4 QUEENS/ N QUEENS

The problem here is to place queens on the empty chessboard in such a way that no queen attacks any other queen. Select Play mode as manual and place the queen on the board to get the solution. Or select Play mode as Simulation to get the solution directly.

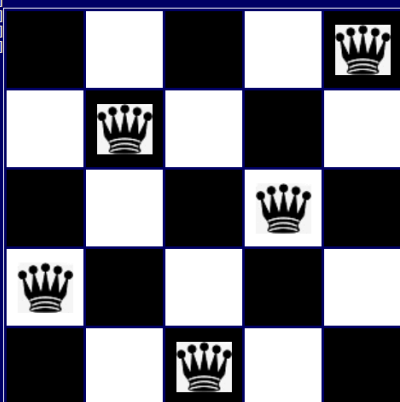
Please select the play mode: ☒ Manual ☐ Simulation

Please select Simulation Speed: ☒ Slow ☐ Moderate ☐ Fast

Please select Boardsize N x N: 5 x 5

Number of solutions found: 8

[13524]
[14253]
[24135]
[25314]
[31425]
[35241]
[41352]
[42531]



Simulation of 4 QUEENS/ N QUEENS

The problem here is to place queens on the empty chessboard in such a way that no queen attacks any other queen. Select Play mode as manual and place the queen on the board to get the solution. Or select Play mode as Simulation to get the solution directly.

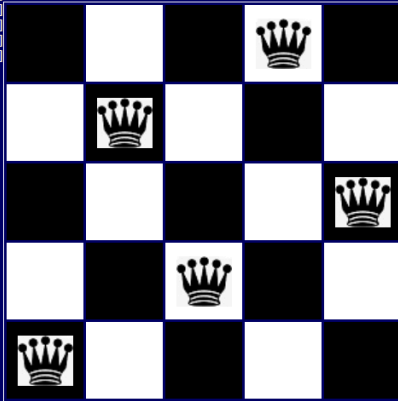
Please select the play mode: ☒ Manual ☐ Simulation

Please select Simulation Speed: ☒ Slow ☐ Moderate ☐ Fast

Please select Boardsize N x N: 5 x 5

Number of solutions found: 9

[13524]
[14253]
[24135]
[25314]
[31425]
[35241]
[41352]
[42531]
[52413]



Simulation of 4 QUEENS/ N QUEENS

The problem here is to place queens on the empty chessboard in such a way that no queen attacks any other queen. Select Play mode as manual and place the queen on the board to get the solution. Or select Play mode as Simulation to get the solution directly.

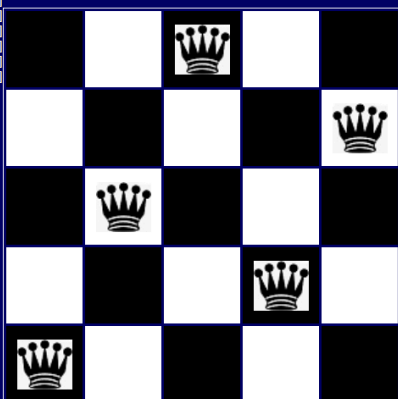
Please select the play mode: ☒ Manual ☐ Simulation

Please select Simulation Speed: ☒ Slow ☐ Moderate ☐ Fast

Please select Boardsize N x N: 5 x 5

Number of solutions found: 10

[13524]
[14253]
[24135]
[25314]
[31425]
[35241]
[41352]
[42531]
[52413]
[53142]



Time Complexity Derivation of N Queen problem:

The time complexity of the N-Queen problem using backtracking cannot be easily expressed in terms of N because it depends on how many configurations are valid as the algorithm progresses. The worst-case scenario can be $O(N!)$, assuming that for every row, we have N possibilities to explore, although the actual number of states explored is less due to pruning by the backtracking.

Program(s) of N Queen problem:

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>

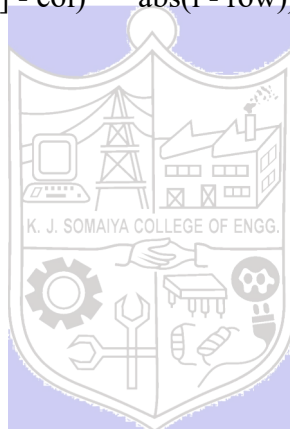
int *board;
int solutions = 0;

bool isSafe(int row, int col, int n) {
    for (int i = 0; i < row; i++) {
        if (board[i] == col || abs(board[i] - col) == abs(i - row)) {
            return false;
        }
    }
    return true;
}

void placeQueen(int row, int n) {
    if (row == n) {
        solutions++;
        for (int i = 0; i < n; i++) {
            printf("%d ", board[i] + 1);
        }
        printf("\n");
        return;
    }

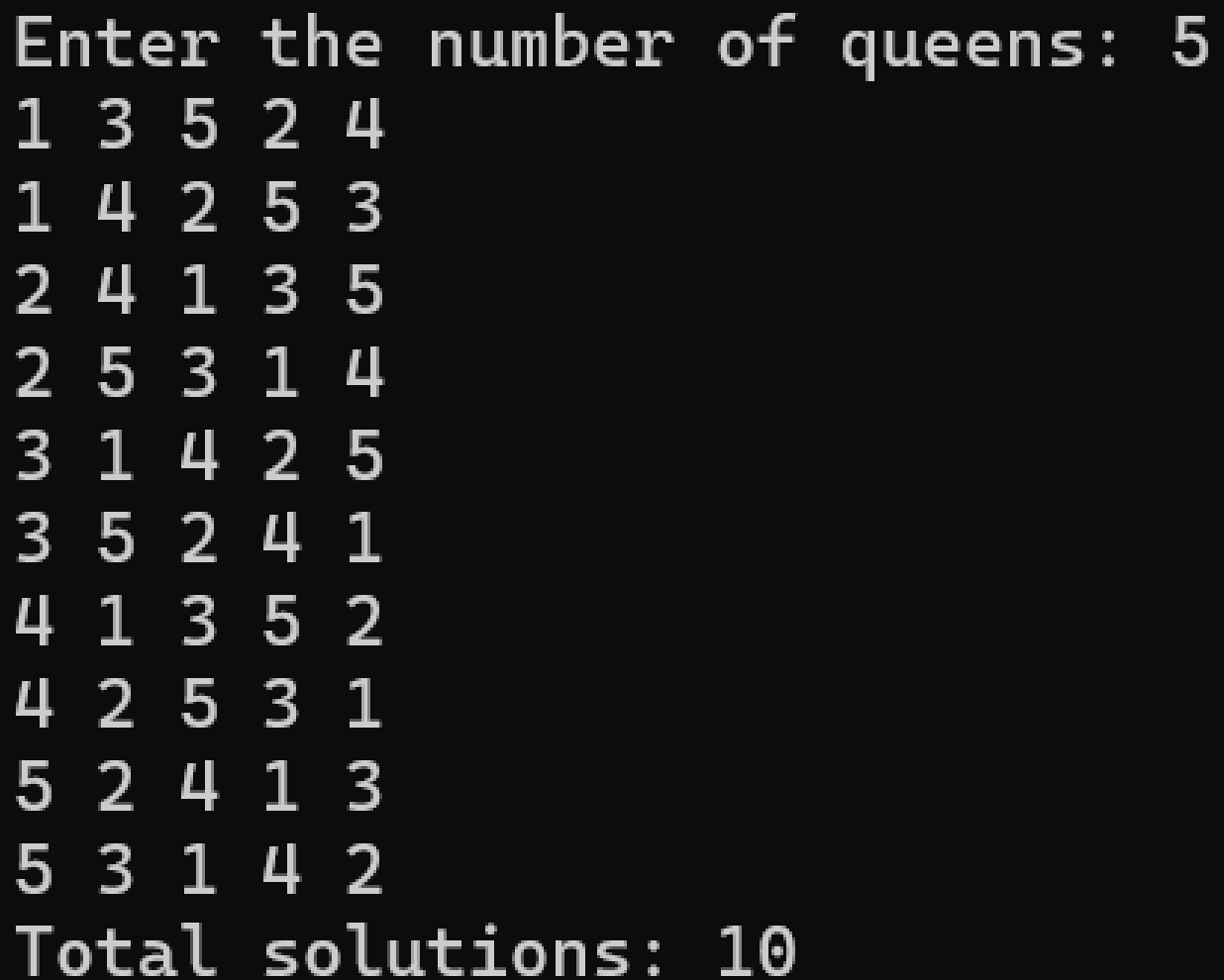
    for (int i = 0; i < n; i++) {
        if (isSafe(row, i, n)) {
            board[row] = i;
            placeQueen(row + 1, n);
        }
    }
}

int main() {
```




```
int n;  
printf("Enter the number of queens: ");  
scanf("%d", &n);  
board = malloc(n * sizeof(int));  
  
placeQueen(0, n);  
printf("Total solutions: %d\n", solutions);  
free(board);  
return 0;  
}
```

Output(o) of N Queen problem:



```
Enter the number of queens: 5  
1 3 5 2 4  
1 4 2 5 3  
2 4 1 3 5  
2 5 3 1 4  
3 1 4 2 5  
3 5 2 4 1  
4 1 3 5 2  
4 2 5 3 1  
5 2 4 1 3  
5 3 1 4 2  
Total solutions: 10
```

Post Lab Questions:-

Explain the process of backtracking?

Ans: Backtracking is a recursive, systematic way of trying out various sequences of decisions until a solution is found. When a current decision leads to an impossibility, backtracking allows reverting back to the previous decision and trying the next possible path.

What are the advantages of backtracking as against brute force algorithms?

Ans: Backtracking significantly reduces the search space by eliminating paths that lead to conflicts, unlike brute force algorithms which test every possible configuration regardless of feasibility. This makes backtracking more efficient in many cases.

What do you mean by P and NP Problem?

Ans: P problems are those that can be solved in polynomial time. NP problems are those for which every solution can be verified in polynomial time. While all P problems are in NP (since verifying a solution is at most as hard as finding it), whether all NP problems are P is a major open question in computer science.

Conclusion: (Based on the observations)

Based on the observations from running the backtracking algorithm for the N-Queen problem, it's evident that backtracking is an effective way to explore and find solutions to combinatorial problems efficiently. The ability to prune non-viable paths and explore feasible solutions reduces computation time compared to exhaustive search methods.

Outcome: Implement Backtracking and Branch-and-bound algorithms

References:

1. Richard E. Neapolitan, " Foundation of Algorithms ", 5th Edition 2016, Jones & Bartlett Students Edition
2. Harsh Bhasin , " Algorithms : Design & Analysis", 1st Edition 2013, Oxford Higher education, India
3. T.H. Cormen ,C.E. Leiserson,R.L. Rivest, and C. Stein, " Introduction to algorithms", 3rd Edition 2009, Prentice Hall India Publication
4. Jon Kleinberg, Eva Tardos, " Algorithm Design", 10th Edition 2013, Pearson India Education Services Pvt. Ltd.