Suraj Iyer

2021300045

SE Comps A

Batch C

DAA Experiment 7

Aim - The N Queen is the problem of placing N chess queens on an N×N chessboard so that no two queens attack each other.

Details - Initialize an empty chessboard of size NxN.

Algorithm -

* Start with the leftmost column and place a queen in the first row of that column.
* Move to the next column and place a queen in the first row of that column.
* Repeat step 3 until either all N queens have been placed or it is impossible to place a queen in the current column without violating the rules of the problem.
* If all N queens have been placed, print the solution.
* If it is not possible to place a queen in the current column without violating the rules of the problem, backtrack to the previous column.
* Remove the queen from the previous column and move it down one row.
* Repeat steps 4-7 until all possible configurations have been tried.

Code -

| #include <stdbool.h>  #include <stdio.h>  int N;  void printSolution(int board[N][N])  {  for (int i = 0; i < N; i++) {  for (int j = 0; j < N; j++)  printf(" %d ", board[i][j]);  printf("\n");  }  }  bool isSafe(int board[N][N], int row, int col)  {  int i, j;  for (i = 0; i < col; i++)  if (board[row][i])  return false;  for (i = row, j = col; i >= 0 && j >= 0; i--, j--)  if (board[i][j])  return false;  for (i = row, j = col; j >= 0 && i < N; i++, j--)  if (board[i][j])  return false;  return true;  }  bool solveNQUtil(int board[N][N], int col)  {  if (col >= N)  return true;  for (int i = 0; i < N; i++) {  if (isSafe(board, i, col)) {  board[i][col] = 1;  if (solveNQUtil(board, col + 1))  return true;  board[i][col] = 0;  }  }  return false;  }  bool solveNQ()  {  int i,j;  printf("\nEnter the value of N : ");  scanf("%d",&N);  int board[N][N];  for(i=0;i<N;i++)  {  for(j=0;j<N;j++)  {  board[i][j]=0;  }  }  if (solveNQUtil(board, 0) == false) {  printf("Solution does not exist");  return false;  }  printSolution(board);  return true;  }  int main()  {  solveNQ();  return 0;  } |
| --- |

Output -

|  |
| --- |

Conclusion -

The idea is to place queens one by one in different columns, starting from the leftmost column. When we place a queen in a column, we check for clashes with already placed queens. In the current column, if we find a row for which there is no clash, we mark this row and column as part of the solution. If we do not find such a row due to clashes, then we backtrack and return false.