4 Queen Problem

Members
HARSHIL AGARWAL(RA1911003010325)
PUNEET SHARMA (RA1911003010331)
PRAKHAR VIJAY (RA1911003010337)
RITESH RAI (RA1911003010333)
BURUGUDDA VIVEK(RA1911003010316)

What is the problem ?

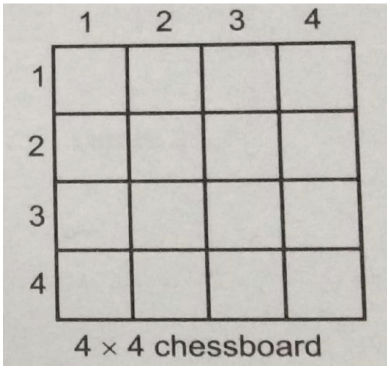
1. Place queens in such a way that no two queens are under attack.
2. So when they are under attack ?
    - When they are in same row, column or in same diagonal
3. So we will have more than one solution which satisfies the above condition.
4. How do we solve this problem ?
    - Dynamic Programming
    - Backtracking

ALGORITHM

- Step 1 − Start from 1st position in the array.
- Step 2 − Place queens on the board and check. Do,
- Step 3 − After placing the queen, mark the position as a part of the solution and then recursively check if this will lead to a solution.
- Step 4 − Now, if placing the queen doesn't lead to a solution and trackback and go to step (a) and place queens to other rows.
- Step 5 − If placing a queen returns a lead to solution return TRUE.
- Step 6 − If all queens are placed return TRUE.
- Step 7 − If all rows are tried and no solution is found, return FALSE.

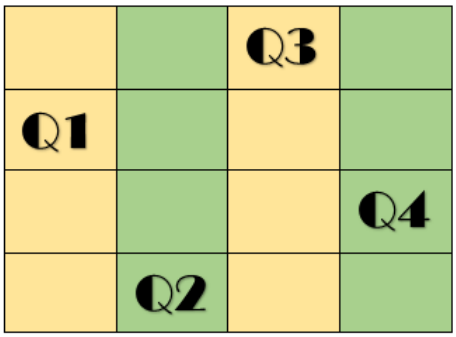
Solution for 4x4 Chess Board

- Given a 4x4 chessboard and number the rows and column of the chessboard are as shown:-

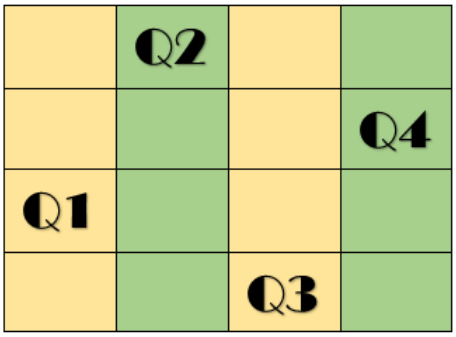


4 × 4 chessboard

- We have to place 4 queens such as q1, q2, q3 and q4 on a chessboard,such that no two queens attack each other
- In such a condition each queen must be placed on a different row, i.e., place queen “i” on row “i”. we place
- queen q1 in the very first acceptable position(1, 1).
- Next,we place queen q2 so that both these queens do not attack each other. We find that if we place q2 in column 1 and 2 then the dead end is encountered.
- The first acceptable position for q2 is column 3 ie, (2, 3)but then no position is left for placing queen q3 safely. So we backtrack one step: Place the queen q2 in (2, 4),the next best possible solution. Then we obtain the position for placing q3 which is (3, 2), But later this position also leads to dead end and no place is found where q4 can be placed safely
- Then we have to backtrack till q1 and place it to (1, 2) and then all the other queens are placed safely by moving q2 to（2, 4), q3 to(3, 1)and q4 to (4, 3), That is,we get the solution（ 2, 4, 1, 3). This is one possible solution for a 4-queens problem.For other possible solutions the whole method is repeated for all partial solutions. The other solution for the 4-queens problem is (3, 1, 4, 2).

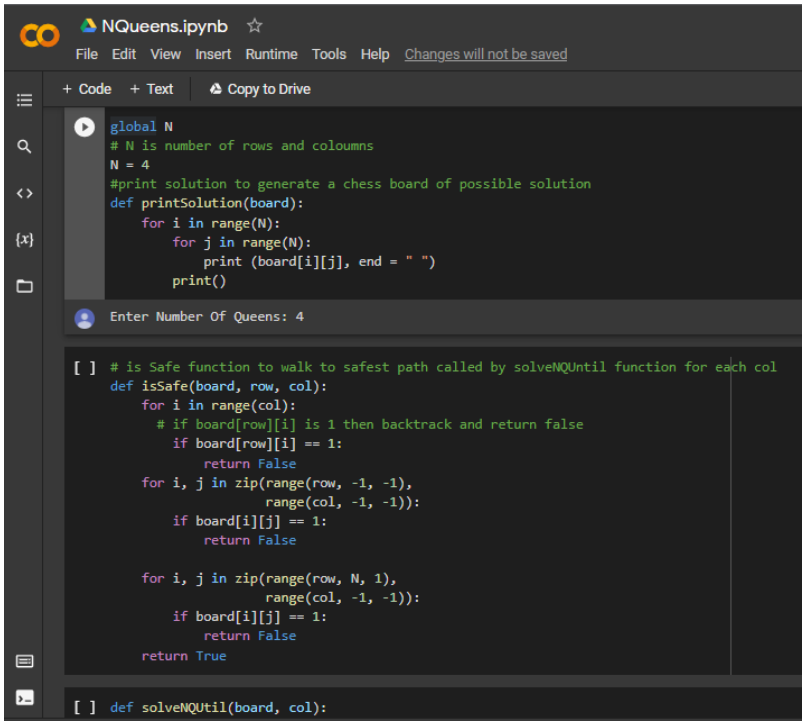


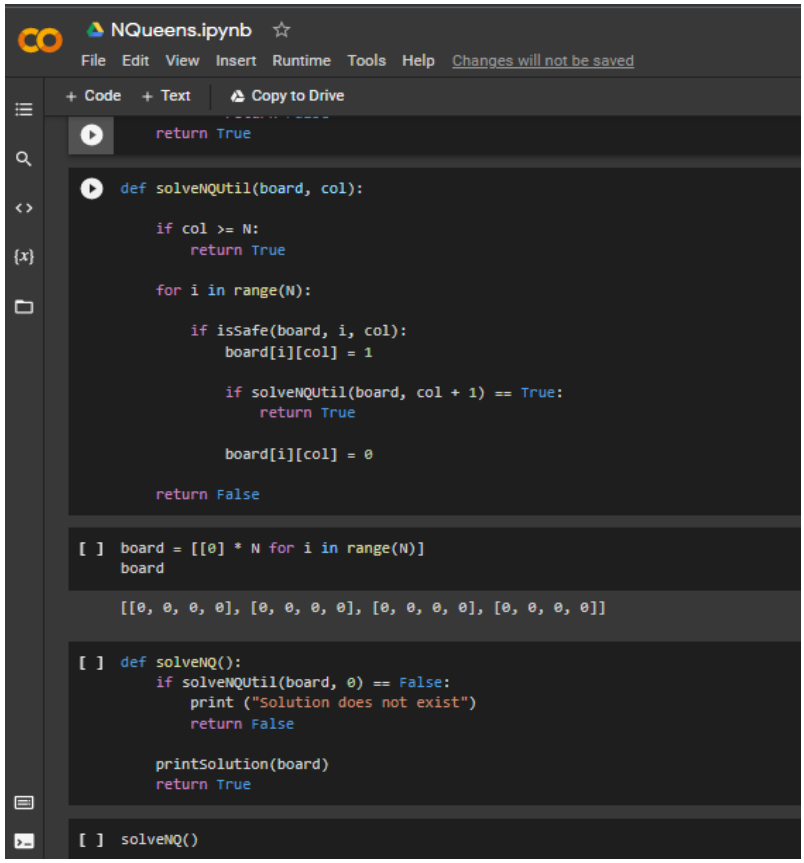
Solution one

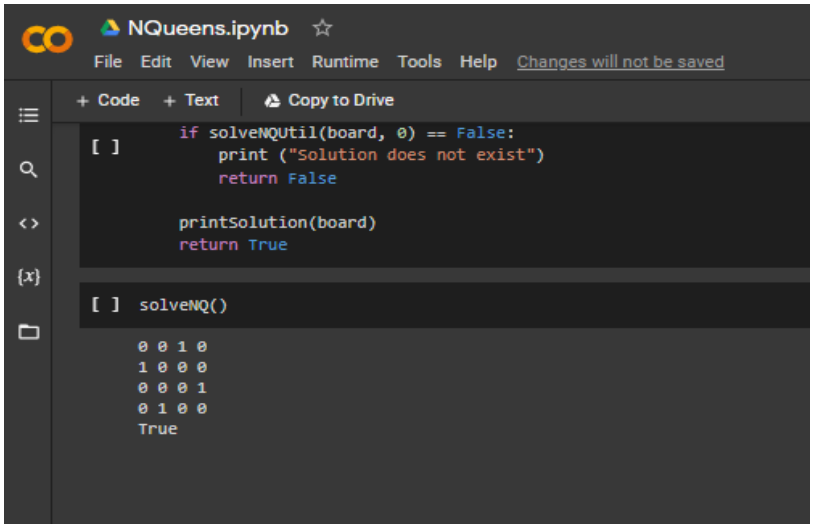


Solution two

Code


NQueens.ipynb
File Edit View Insert Runtime Tools Help Changes will not be saved
+ Code + Text Copy to Drive
global N
# N is number of rows and coloumns
N = 4
#print solution to generate a chess board of possible solution
def printSolution(board):
for i in range(N):
for j in range(N):
print (board[i][j], end = " ")
print()
Enter Number Of Queens: 4
[ ] # is Safe function to walk to safest path called by solveNQUntil function for each col
def isSafe(board, row, col):
for i in range(col):
# if board[row][i] is 1 then backtrack and return false
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, 1),
range(col, -1, -1)):
if board[i][j] == 1:
return False
return True
[ ] def solveNQUtil(board, col):


NQueens.ipynb
File Edit View Insert Runtime Tools Help Changes will not be saved
+ Code + Text Copy to Drive
return True
def solveNQUtil(board, col):
if col >= N:
return True
for i in range(N):
if isSafe(board, i, col):
board[i][col] = 1
if solveNQUtil(board, col + 1) == True:
return True
board[i][col] = 0
return False
[ ] board = [[0] * N for i in range(N)]
board
[[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]]
[ ] def solveNQ():
if solveNQUtil(board, 0) == False:
print ("Solution does not exist")
return False
printSolution(board)
return True
[ ] solveNQ()



Result

We have successfully placed the four queens on a 4x4 chess board in such a way that they don't attack each other.