RA1911026010029
Thejaswin S

Eight Queens Problem

AIM

To write a program for the Eight Queens problem using backtracking and verify the output.

The eight queens puzzle is the problem of placing eight chess queens on an 8x8 chessboard so that no two queens attack each other. Thus, a solution requires that no two queens share the same row, column, or diagonal. The eight queens puzzle is an example of the more general n-queens problem of placing n queens on an NxN chessboard, where solutions exist for all natural numbers N with the exception of 1, 2, and 3.

Algorithm

1) Start in the leftmost column
2) If all queens are placed return true
3) Try all rows in the current column. Do following
   for every tried row.
    a) If the queen can be placed safely in this row
       then mark this [row, column] as part of the
       solution and recursively check if placing
       queen here leads to a solution.
    b) If placing the queen in [row, column] leads to a
       the solution then returns true.
    c) If placing queen doesn't lead to a solution
       then unmark this [row, column] (Backtrack)
       and go to step (a) to try other rows.
3) If all rows have been tried and nothing worked,
   return false to trigger backtracking.

Code

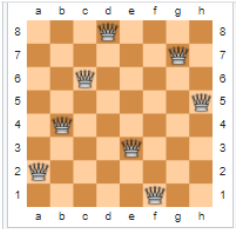
```
def isSafe(mat, r, c):

    # return false if two queens share the same column
    for i in range(r):
        if mat[i][c] == 'Q':
            return False
```

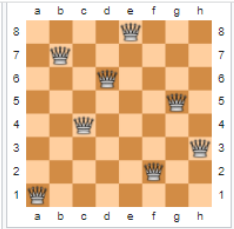
```
    # return false if two queens share the same `` diagonal
    (i, j) = (r, c)
    while i >= 0 and j >= 0:
        if mat[i][j] == 'Q':
            return False
        i = i - 1
        j = j - 1

    # return false if two queens share the same `/` diagonal
    (i, j) = (r, c)
    while i >= 0 and j < len(mat):
        if mat[i][j] == 'Q':
            return False
        i = i - 1
        j = j + 1

    return True


def printSolution(mat):
    for r in mat:
        print(str(r).replace(',', '').replace('\'', ''))
    print()


def nQueen(mat, r):

    # if `N` queens are placed successfully, print the solution
    if r == len(mat):
        printSolution(mat)
        return

    # place queen at every square in the current row `r`
    # and recur for each valid movement
    for i in range(len(mat)):

        # if no two queens threaten each other
        if isSafe(mat, r, i):
            # place queen on the current square
```

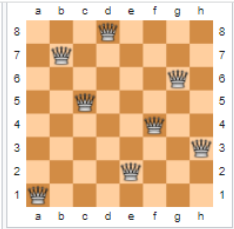
```
        mat[r][i] = 'Q'

        # recur for the next row
        nQueen(mat, r + 1)

        # backtrack and remove the queen from the current square
        mat[r][i] = '*'


if __name__ == '__main__':

    # `N × N` chessboard
    N = 8

    # `mat[][]` keeps track of the position of queens in
    # the current configuration
    mat = [['*' for x in range(N)] for y in range(N)]

    nQueen(mat, 0)
```

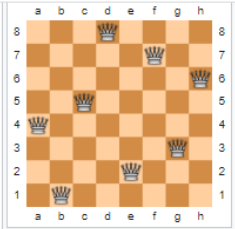
The eight queens puzzle has 92 distinct solutions.
If solutions that differ only by symmetry operations(rotations and reflections) of the board are counted as one the puzzle has 12 unique (or fundamental) solutions.

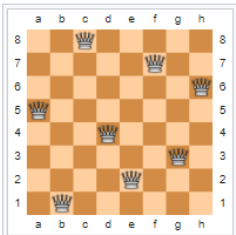
RA1911026010029
Thejaswin S

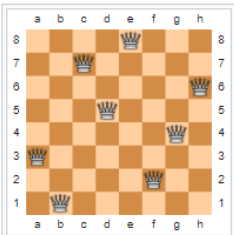

Solution 1

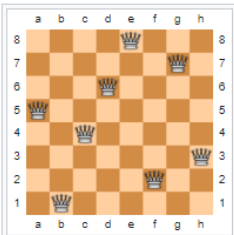

Solution 2

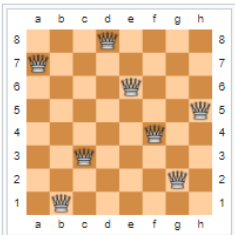

Solution 3

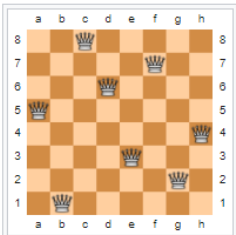

Solution 4

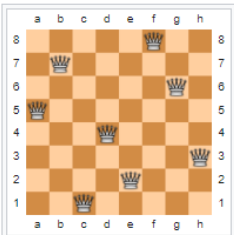

Solution 5

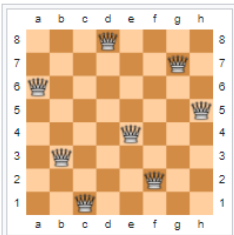

Solution 6

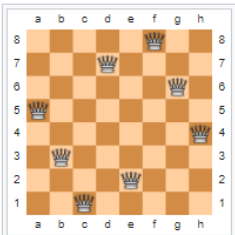

Solution 7


Solution 8









RA1911026010029
Thejaswin S

Run Program in AWS Cloud9 IDE

exp1.py bash - "ip-172-31-2-88"

```python
def isSafe(mat, r, c):

    # return false if two queens share the same column
    for i in range(r):
        if mat[i][c] == 'Q':
            return False

    # return false if two queens share the same `` diagonal
    (i, j) = (r, c)
    while i >= 0 and j >= 0:
        if mat[i][j] == 'Q':
            return False
        i = i - 1
        j = j - 1

    # return false if two queens share the same `/` diagonal
    (i, j) = (r, c)
    while i >= 0 and j < len(mat):
        if mat[i][j] == 'Q':
            return False
        i = i - 1
        j = j + 1

    return True


def printSolution(mat):
    for r in mat:
        print(str(r).replace(',', '').replace('\'', ''))
    print()


def nQueen(mat, r):

    # if `N` queens are placed successfully, print the solution
    if r == len(mat):
        printSolution(mat)
        return

```

To Run the program

```
Mathira:~/environment/RA1911026010029 $ python3 exp1.py
```

RA1911026010029
Thejaswin S

Console OUTPUT

```
[Q * * * * * * *]
[* * * * Q * * *]
[* * * * * * * Q]
[* * * * * Q * *]
[* * Q * * * * *]
[* * * * * * Q *]
[* Q * * * * * *]
[* * * Q * * * *]

[Q * * * * * * *]
[* * * * * Q * *]
[* * * * * * * Q]
[* * Q * * * * *]
[* * * * * * Q *]
[* * * Q * * * *]
[* Q * * * * * *]
[* * * * Q * * *]

[Q * * * * * * *]
[* * * * * * Q *]
[* * * Q * * * *]
[* * * * * Q * *]
[* * * * * * * Q]
[* Q * * * * * *]
[* * * * Q * * *]
[* * Q * * * * *]

[Q * * * * * * *]
[* * * * * * Q *]
[* * * * Q * * *]
[* * * * * * * Q]
[* Q * * * * * *]
[* * * Q * * * *]
[* * * * * Q * *]
```

```
[* * * Q * * * *]
[* * * * * Q * *]
[* * * * * * * Q]
[* * Q * * * * *]
[Q * * * * * * *]
[* * * * * * Q *]
[* * * * Q * * *]

[* Q * * * * * *]
[* * * * Q * * *]
[* * * * * * Q *]
[Q * * * * * * *]
[* * Q * * * * *]
[* * * * * * * Q]
[* * * * * Q * *]
[* * * Q * * * *]

[* Q * * * * * *]
[* * * * Q * * *]
[* * * * * * Q *]
[* * * Q * * * *]
[Q * * * * * * *]
[* * * * * * * Q]
[* * * * * Q * *]
[* * Q * * * * *]

[* Q * * * * * *]
[* * * * * Q * *]
[Q * * * * * * *]
[* * * * * * Q *]
[* * * Q * * * *]
[* * * * * * * Q]
[* * Q * * * * *]
[* * * * Q * * *]
```

```
[* Q * * * * * *]
[* * * * * Q * *]
[* * * * * * * Q]
[* * Q * * * * *]
[Q * * * * * * *]
[* * * Q * * * *]
[* * * * * * Q *]
[* * * * Q * * *]

[* Q * * * * * *]
[* * * * * * Q *]
[* * Q * * * * *]
[* * * * * Q * *]
[* * * * * * * Q]
[* * * * Q * * *]
[Q * * * * * * *]
[* * * Q * * * *]

[* Q * * * * * *]
[* * * * * * Q *]
[* * * * Q * * *]
[* * * * * * * Q]
[Q * * * * * * *]
[* * * Q * * * *]
[* * * * * Q * *]
[* * Q * * * * *]

[* Q * * * * * *]
[* * * * * * * Q]
[* * * * * Q * *]
[Q * * * * * * *]
[* * Q * * * * *]
[* * * * * * Q *]
[* * * Q * * * *]
```

```
[Q * * * * * * *]
[* * * * * * Q *]
[* * * * Q * * *]
[* * * * * * * Q]
[* Q * * * * * *]
[* * * Q * * * *]
[* * * * * Q * *]

[* * Q * * * * *]
[* * * * Q * * *]
[* Q * * * * * *]
[* * * * * * * Q]
[Q * * * * * * *]
[* * * * * * Q *]
[* * * Q * * * *]
[* * * * * Q * *]

[* * Q * * * * *]
[* * * * Q * * *]
[* Q * * * * * *]
[* * * * * * * Q]
[* * * * Q * * *]
[Q * * * * * * *]
[* * * Q * * * *]
[* * * * * * Q *]

[* * Q * * * * *]
[* * * * Q * * *]
[* * * * * * Q *]
[Q * * * * * * *]
[* * * Q * * * *]
[* Q * * * * * *]
[* * * * * * * Q]
[* * * * * Q * *]
```