def is\_safe(board, row, col, n):

# چک کردن آیا می‌توان وزیری را در سلول (row, col) قرار داد یا خیر

# چک کردن ردیف افقی (سمت چپ)

for i in range(col):

if board[row][i] == 1:

return False

# چک کردن قطر بالا به چپ

for i, j in zip(range(row, -1, -1), range(col, -1, -1)):

if board[i][j] == 1:

return False

# چک کردن قطر پایین به چپ

for i, j in zip(range(row, n, 1), range(col, -1, -1)):

if board[i][j] == 1:

return False

return True

def solve\_n\_queens\_util(board, col, n):

# حالت پایه: اگر تمام وزیرها قرار گرفته باشند

if col >= n:

return True

# برای هر سلول در ستون فعلی

for i in range(n):

# چک کردن آیا می‌توان وزیر را در این سلول قرار داد

if is\_safe(board, i, col, n):

# قرار دادن وزیر در این سلول

board[i][col] = 1

# ادامه به جستجوی ستون بعدی

if solve\_n\_queens\_util(board, col + 1, n):

return True

# اگر قرار گرفتن وزیر در این سلول به حل مسئله منجر نشود، آن را از صفحه حذف می‌کنیم

board[i][col] = 0

# اگر هیچ یک از سلول‌ها منجر به حل مسئله نشود

return False

def solve\_n\_queens(n):

# ایجاد صفحه شطرنج خالی

board = [[0 for \_ in range(n)] for \_ in range(n)]

# حل مسئله با فراخوانی اولیه از ستون اول

if not solve\_n\_queens\_util(board, 0, n):

print("هیچ راه حلی وجود ندارد.")

return False

# نمایش جواب

for i in range(n):

for j in range(n):

print(board[i][j], end=" ")

print()

return True

# تابع را فراخوانی می‌کنیم با n=8 برای حل مسئله 8 وزیر

solve\_n\_queens(8)