|  |
| --- |
| Ex. No: 2  10/08/2022 4 – Queens problem  **Iterative Deepening:**  res = []  def totalNQueens(n):  def check(x, y, board):  i = x  j = y  # checking upper left diagonal  while i >= 0 and j >= 0:  if board[i][j] == 1:  return False  i -= 1  j -= 1  i = x  j = y  # checking lower left diagonal  while i < n and j >= 0:  if board[i][j] == 1:  return False  i += 1  j -= 1  i = x  j = y  # checking the column  while j >= 0:  if board[i][j] == 1:  return False  j -= 1  return True  def dfs(col, board, depth):  if col >= n:  res.append([])  for i in range(n):  res[-1].append("")  for j in range(n):  if board[i][j]:  res[-1][-1] += "Q"  else:  res[-1][-1] += "#"  return  if depth <= 0:  return False  for i in range(n):  if check(i, col, board):  board[i][col] = 1  dfs(col+1, board, depth-1)  board[i][col] = 0  board = [  [0]\*n for i in range(n)  ]  depth = int(input())  dfs(0, board, depth)  print(res)  totalNQueens(4)    **Depth Limited:**  res = []  def totalNQueens(n):  def check(x, y, board):  i = x  j = y  # checking upper left diagonal  while i >= 0 and j >= 0:  if board[i][j] == 1:  return False  i -= 1  j -= 1  i = x  j = y  # checking lower left diagonal  while i < n and j >= 0:  if board[i][j] == 1:  return False  i += 1  j -= 1  i = x  j = y  # checking the column  while j >= 0:  if board[i][j] == 1:  return False  j -= 1  return True  def dfs(col, board, maxdepth):  if col >= n:  res.append([])  for i in range(n):  res[-1].append("")  for j in range(n):  if board[i][j]:  res[-1][-1] += "Q"  else:  res[-1][-1] += "#"  return  if maxdepth <= 0:  return False  for i in range(n):  if check(i, col, board):  board[i][col] = 1  dfs(col+1, board, maxdepth-1)  board[i][col] = 0  board = [  [0]\*n for i in range(n)  ]  for i in range(int(input())):  res = []  dfs(0, board, i+1)  print(res)  totalNQueens(4) |
|  |