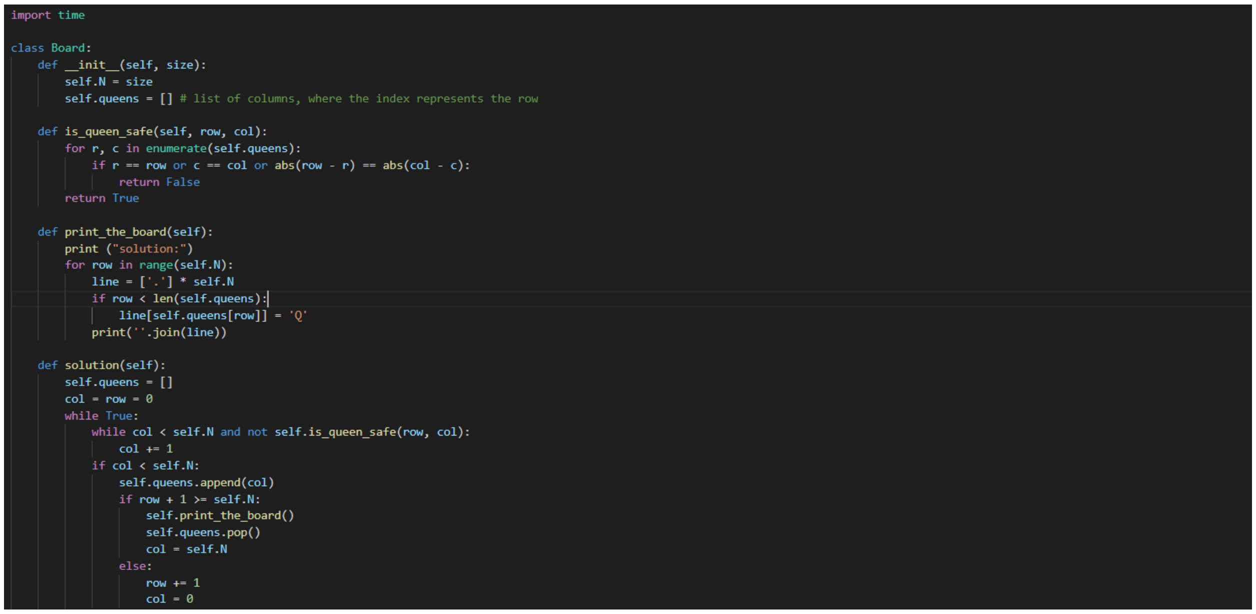
นภัสวรรณ ละมนเทียร 63010495

รายงาน

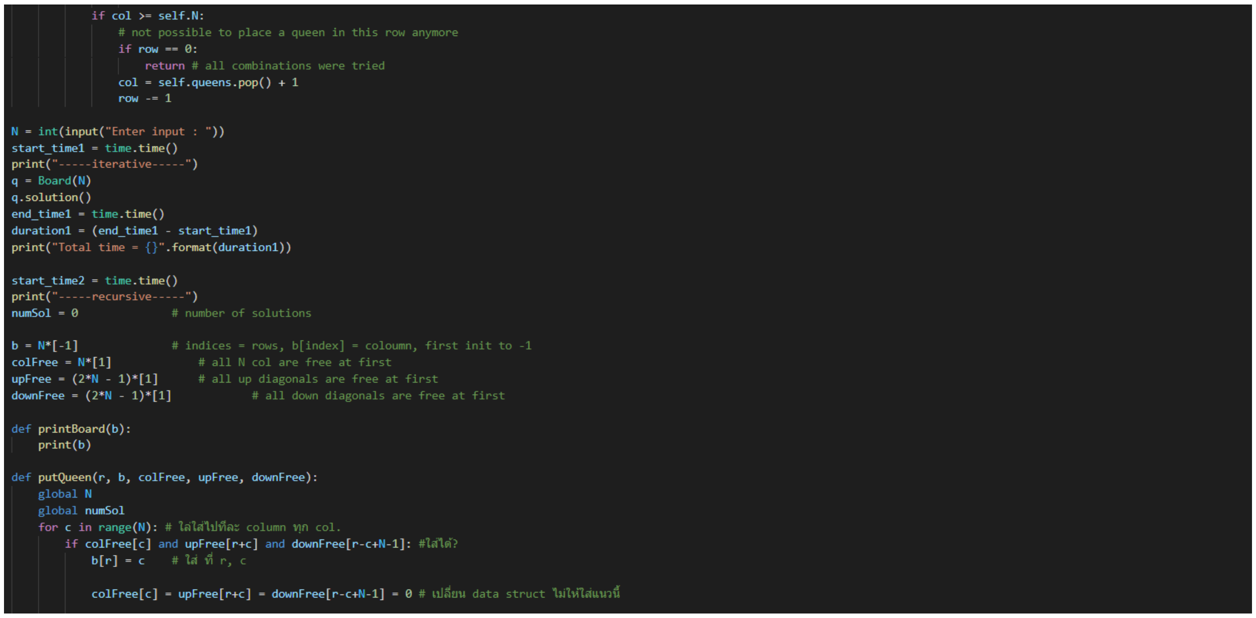
การศึกษาโปรแกรม N-Queen โดยใช้ algorithm แบบ iterative และ recursive

1. Source code

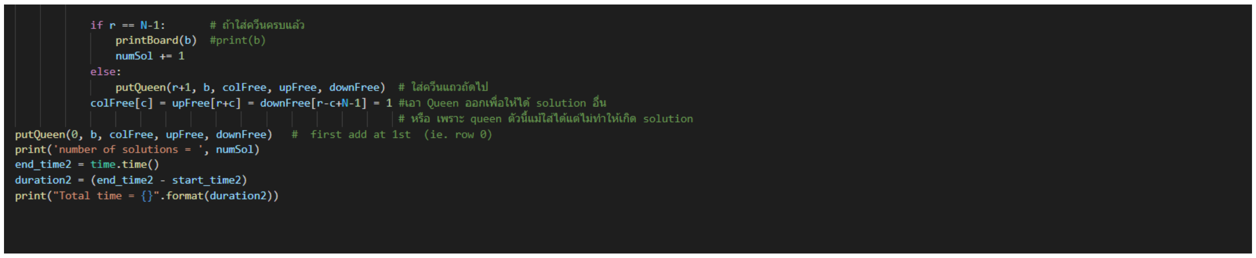
https://colab.research.google.com/drive/1DoRCzxFV4XrKEI5Td_8NMWYvT63zFiaD?usp=sharing



```python
import time

class Board:
    def __init__(self, size):
        self.N = size
        self.queens = [] # list of columns, where the index represents the row

    def is_queen_safe(self, row, col):
        for r, c in enumerate(self.queens):
            if r == row or c == col or abs(row - r) == abs(col - c):
                return False
        return True

    def print_the_board(self):
        print ("solution:")
        for row in range(self.N):
            line = ['.'] * self.N
            if row < len(self.queens):
                line[self.queens[row]] = 'Q'
            print(''.join(line))

    def solution(self):
        self.queens = []
        col = row = 0
        while True:
            while col < self.N and not self.is_queen_safe(row, col):
                col += 1
            if col < self.N:
                self.queens.append(col)
                if row + 1 >= self.N:
                    self.print_the_board()
                    self.queens.pop()
                    col = self.N
                else:
                    row += 1
                    col = 0
```



```python
            if col >= self.N:
                # not possible to place a queen in this row anymore
                if row == 0:
                    return # all combinations were tried
                col = self.queens.pop() + 1
                row -= 1

N = int(input("Enter input : "))
start_time1 = time.time()
print("-----iterative-----")
q = Board(N)
q.solution()
end_time1 = time.time()
duration1 = (end_time1 - start_time1)
print("Total time = {}".format(duration1))

start_time2 = time.time()
print("-----recursive-----")
numSol = 0              # number of solutions

b = N*[-1]              # indices = rows, b[index] = coloumn, first init to -1
colFree = N*[1]             # all N col are free at first
upFree = (2*N - 1)*[1]      # all up diagonals are free at first
downFree = (2*N - 1)*[1]            # all down diagonals are free at first

def printBoard(b):
    print(b)

def putQueen(r, b, colFree, upFree, downFree):
    global N
    global numSol
    for c in range(N): # ใส่ใส่ไปทีละ column ทุก col.
        if colFree[c] and upFree[r+c] and downFree[r-c+N-1]: #ใส่ได้?
            b[r] = c    # ใส่ ที่ r, c

            colFree[c] = upFree[r+c] = downFree[r-c+N-1] = 0 # เปลี่ยน data struct ไม่ให้ใส่แนวนี้
```



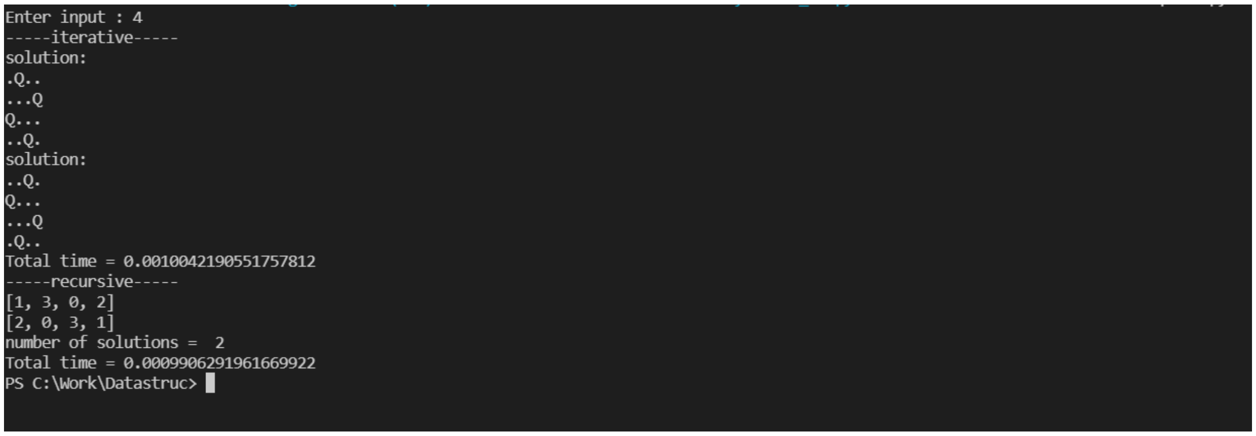
```
            if r == N-1:        # ถ้าใส่ควีนครบแล้ว
                printBoard(b)  #print(b)
                numSol += 1
            else:
                putQueen(r+1, b, colFree, upFree, downFree)  # ใส่ควีนแถวถัดไป
            colFree[c] = upFree[r+c] = downFree[r-c+N-1] = 1 #เอา Queen ออกเพื่อให้ได้ solution อื่น
                                                             # หรือ เพราะ queen ตัวนี้แม้ใส่ได้แต่ไม่ทำให้เกิด solution
putQueen(0, b, colFree, upFree, downFree)   #  first add at 1st  (ie. row 0)
print('number of solutions = ', numSol)
end_time2 = time.time()
duration2 = (end_time2 - start_time2)
print("Total time = {}".format(duration2))
```

2. รายละเอียดเครื่องคอมพิวเตอร์ CPU Memory

   - CPU : Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz   2.60 GHz
   - RAM : 16.0 GB
   - Memory : SSD 458 GB, RAM 16.0 GB

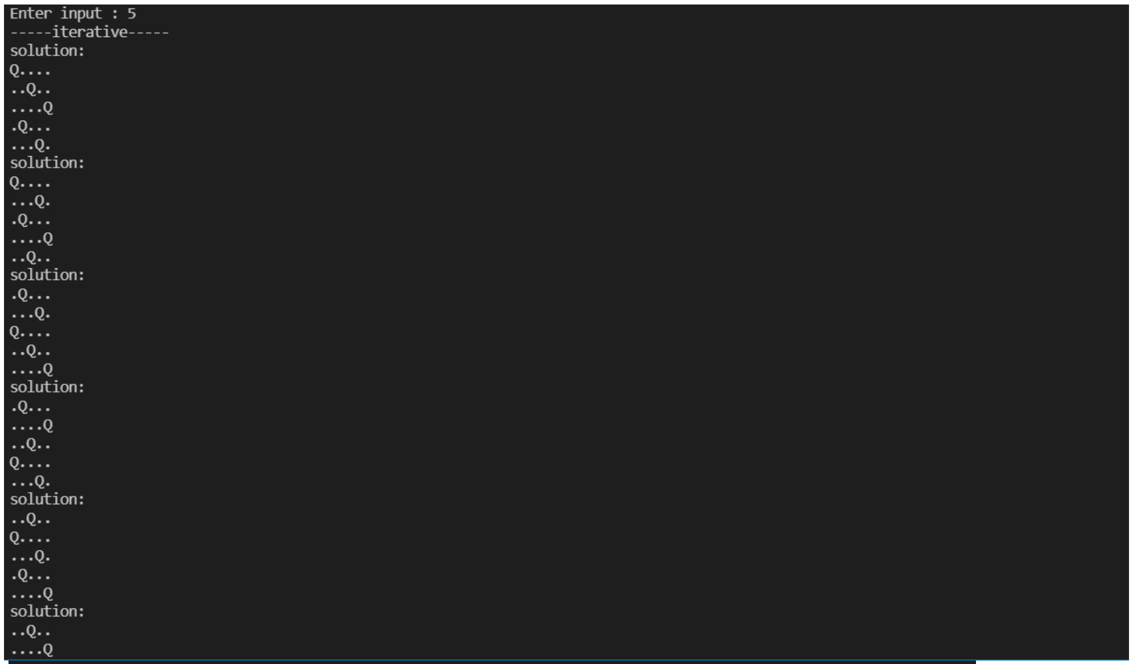
3. Capture การรัน และจับเวลา ของแต่ละอินพุต (อินพุต เริ่มจาก 4 - 12)

   - Input = 4

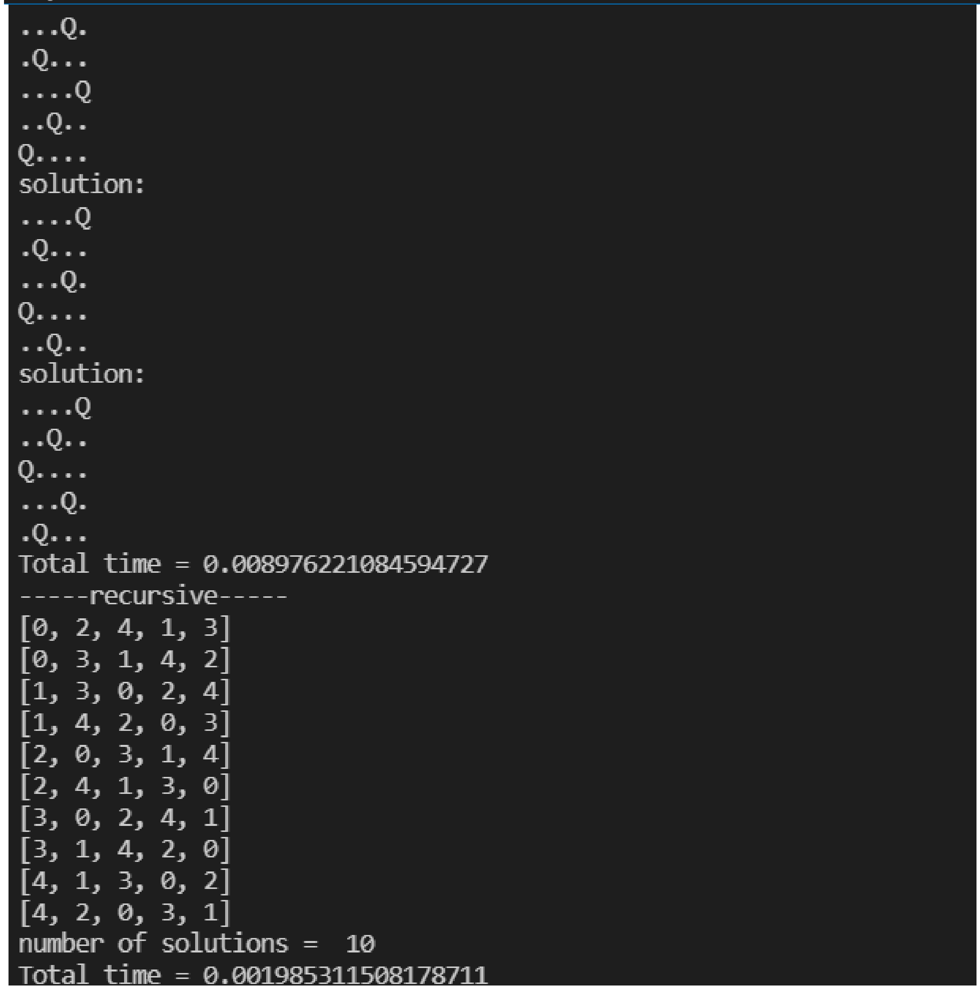


```
Enter input : 4
-----iterative-----
solution:
.Q..
...Q
Q...
..Q.
solution:
..Q.
Q...
...Q
.Q..
Total time = 0.0010042190551757812
-----recursive-----
[1, 3, 0, 2]
[2, 0, 3, 1]
number of solutions =  2
Total time = 0.0009906291961669922
PS C:\Work\Datastruc>
```

- Input = 5

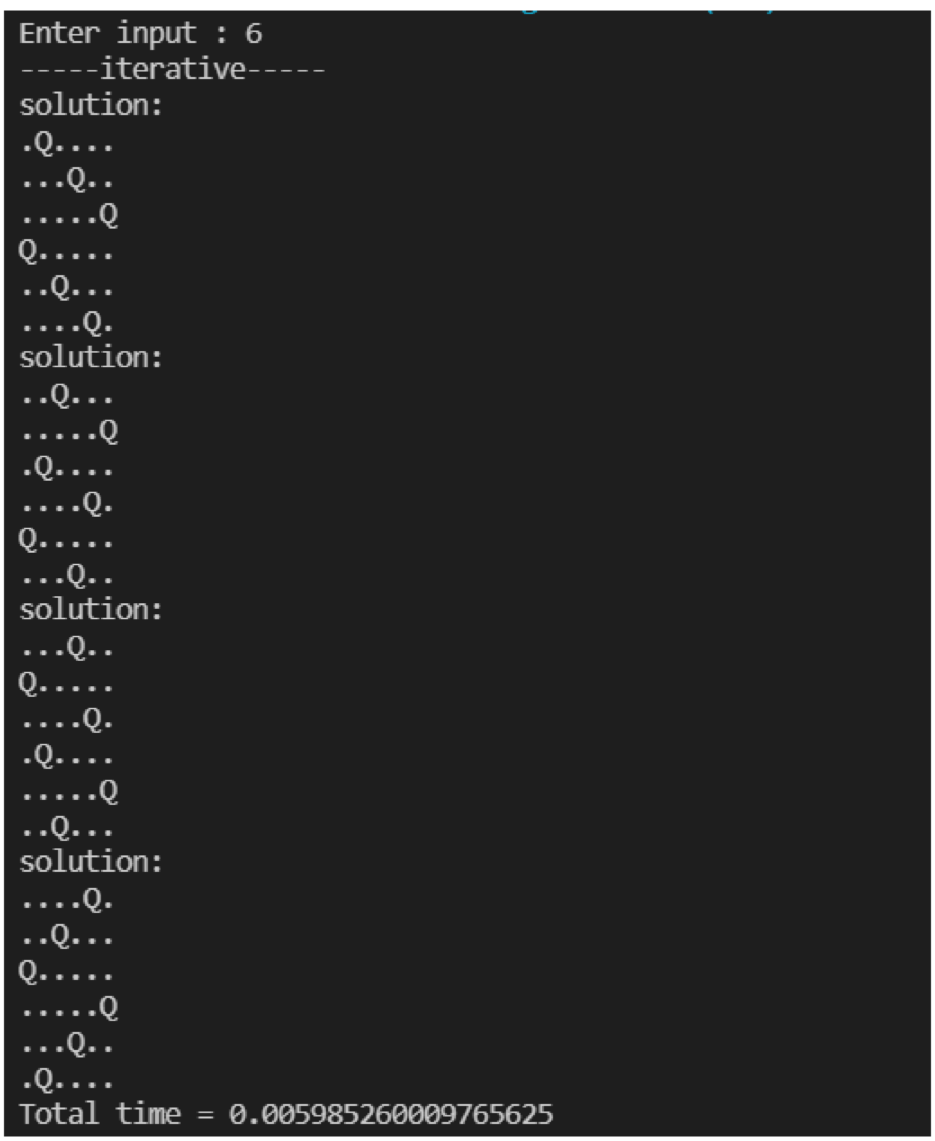


```
Enter input : 5
-----iterative-----
solution:
Q....
..Q..
....Q
.Q...
...Q.
solution:
Q....
...Q.
.Q...
....Q
..Q..
solution:
.Q...
...Q.
Q....
..Q..
....Q
solution:
.Q...
....Q
..Q..
Q....
...Q.
solution:
..Q..
Q....
...Q.
.Q...
....Q
solution:
..Q..
....Q
```

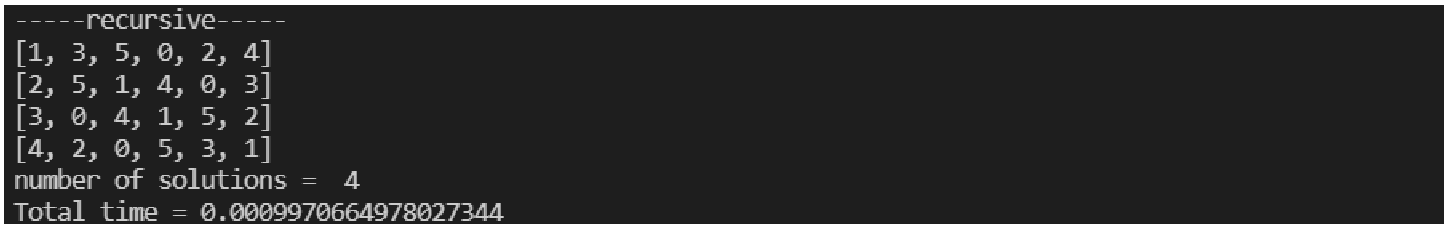


```
...Q.
.Q...
....Q
..Q..
Q....
solution:
....Q
.Q...
...Q.
Q....
..Q..
solution:
....Q
..Q..
Q....
...Q.
.Q...
Total time = 0.008976221084594727
-----recursive-----
[0, 2, 4, 1, 3]
[0, 3, 1, 4, 2]
[1, 3, 0, 2, 4]
[1, 4, 2, 0, 3]
[2, 0, 3, 1, 4]
[2, 4, 1, 3, 0]
[3, 0, 2, 4, 1]
[3, 1, 4, 2, 0]
[4, 1, 3, 0, 2]
[4, 2, 0, 3, 1]
number of solutions =  10
Total time = 0.001985311508178711
```

- Input = 6

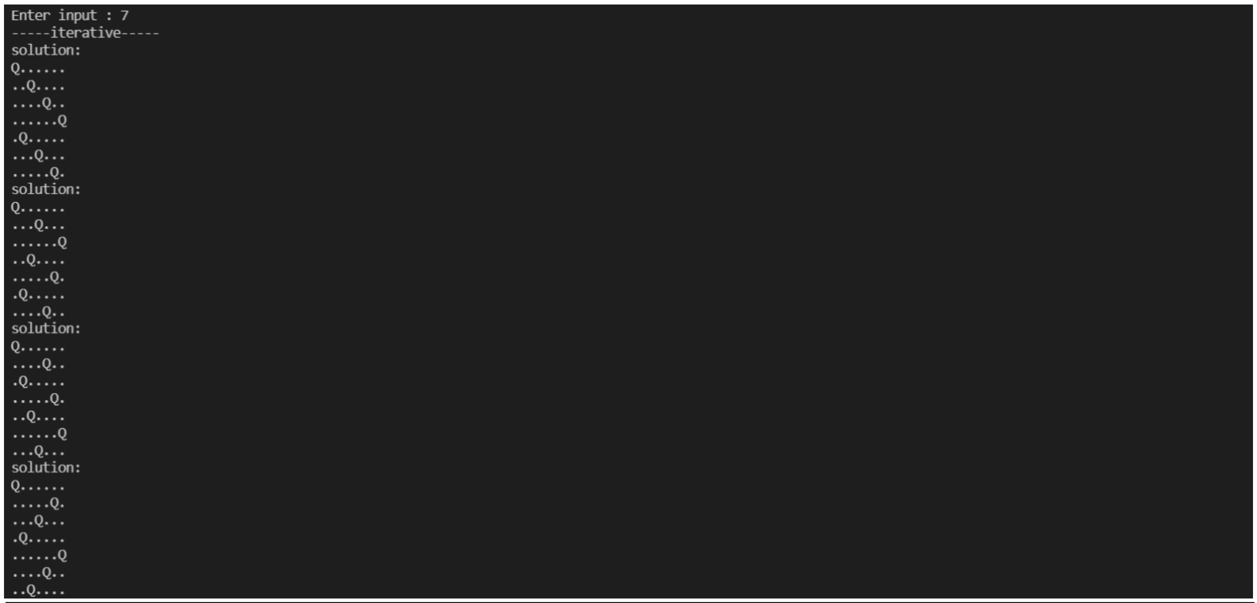


```
Enter input : 6
-----iterative-----
solution:
.Q....
...Q..
.....Q
Q.....
..Q...
....Q.
solution:
..Q...
.....Q
.Q....
....Q.
Q.....
...Q..
solution:
...Q..
Q.....
....Q.
.Q....
.....Q
..Q...
solution:
....Q.
..Q...
Q.....
.....Q
...Q..
.Q....
Total time = 0.005985260009765625
```

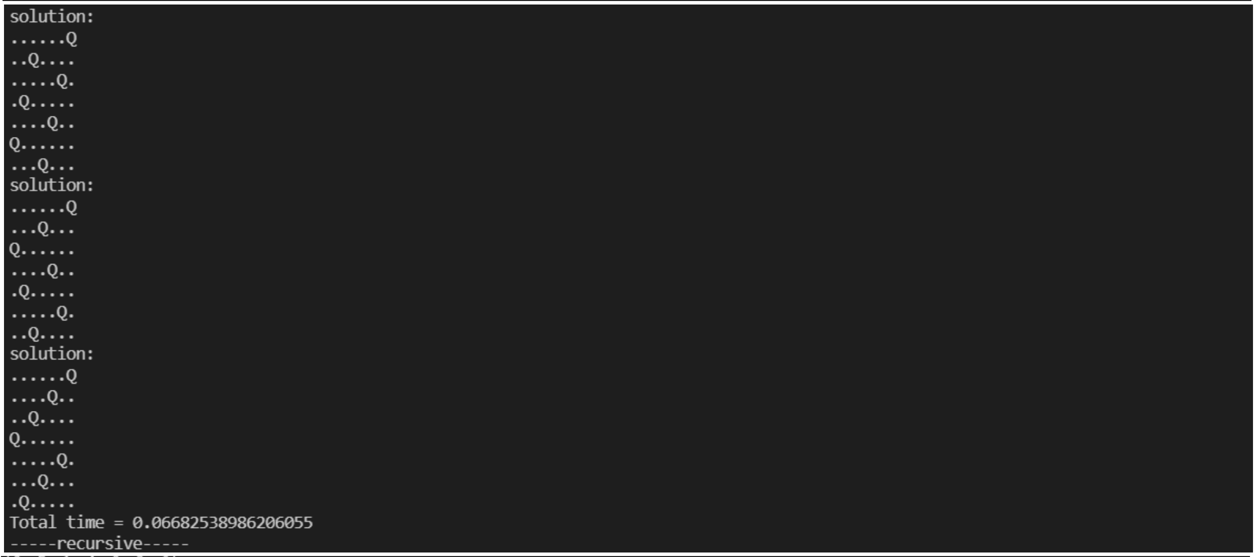


```
-----recursive-----
[1, 3, 5, 0, 2, 4]
[2, 5, 1, 4, 0, 3]
[3, 0, 4, 1, 5, 2]
[4, 2, 0, 5, 3, 1]
number of solutions =  4
Total time = 0.0009970664978027344
```

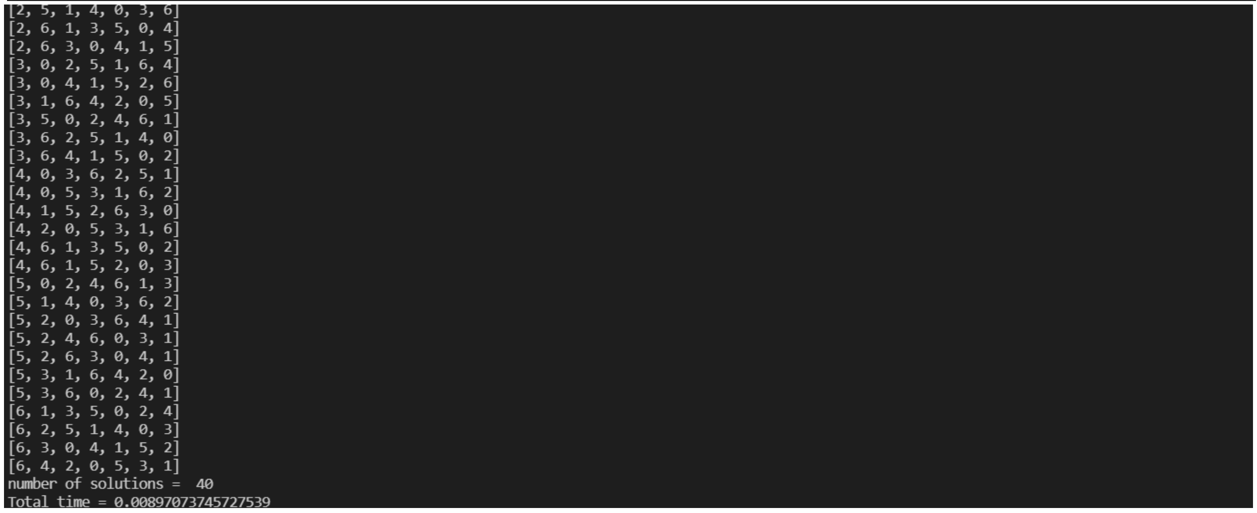
- Input = 7



```
Enter input : 7
-----iterative-----
solution:
Q......
..Q....
....Q..
......Q
.Q.....
...Q...
.....Q.
solution:
Q......
...Q...
......Q
..Q....
.....Q.
.Q.....
....Q..
solution:
Q......
....Q..
.Q.....
.....Q.
..Q....
......Q
...Q...
solution:
Q......
.....Q.
...Q...
.Q.....
......Q
....Q..
..Q....
```

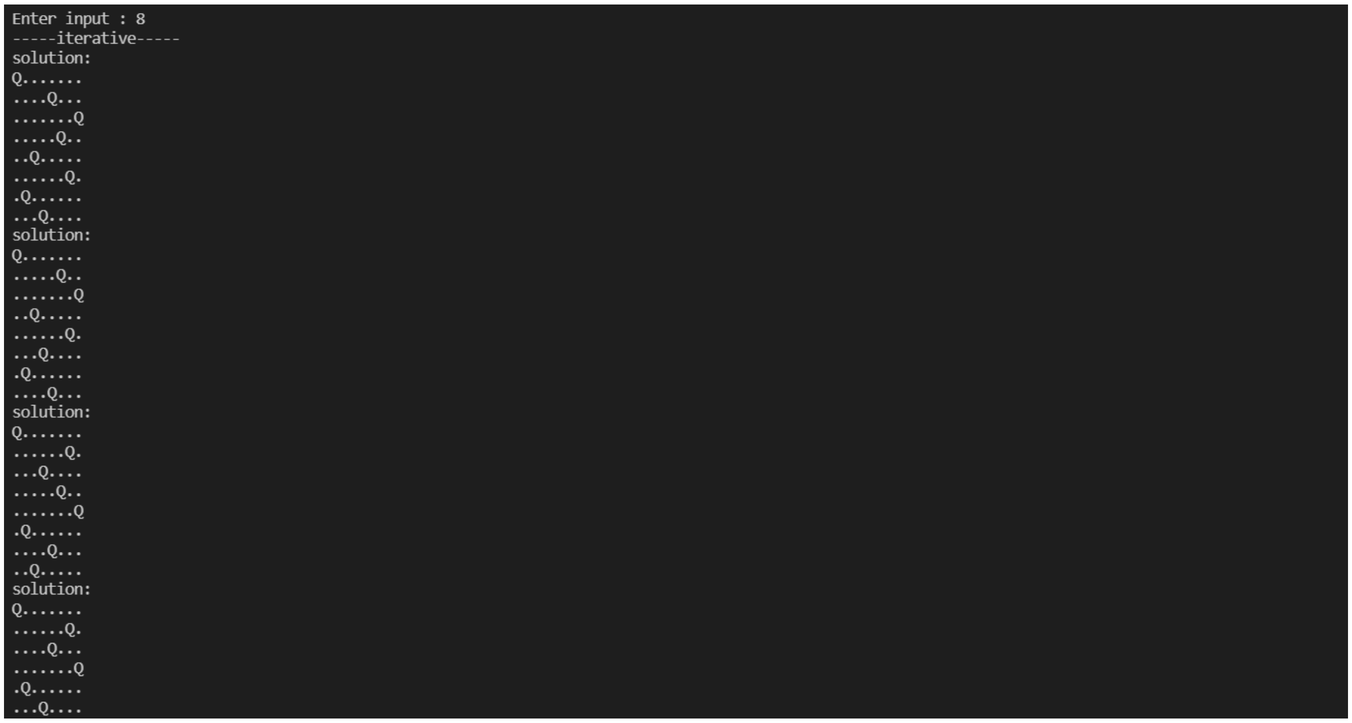


```
solution:
......Q
..Q....
.....Q.
.Q.....
....Q..
Q......
...Q...
solution:
......Q
...Q...
Q......
....Q..
.Q.....
.....Q.
..Q....
solution:
......Q
....Q..
..Q....
Q......
.....Q.
...Q...
.Q.....
Total time = 0.06682538986206055
-----recursive-----
```

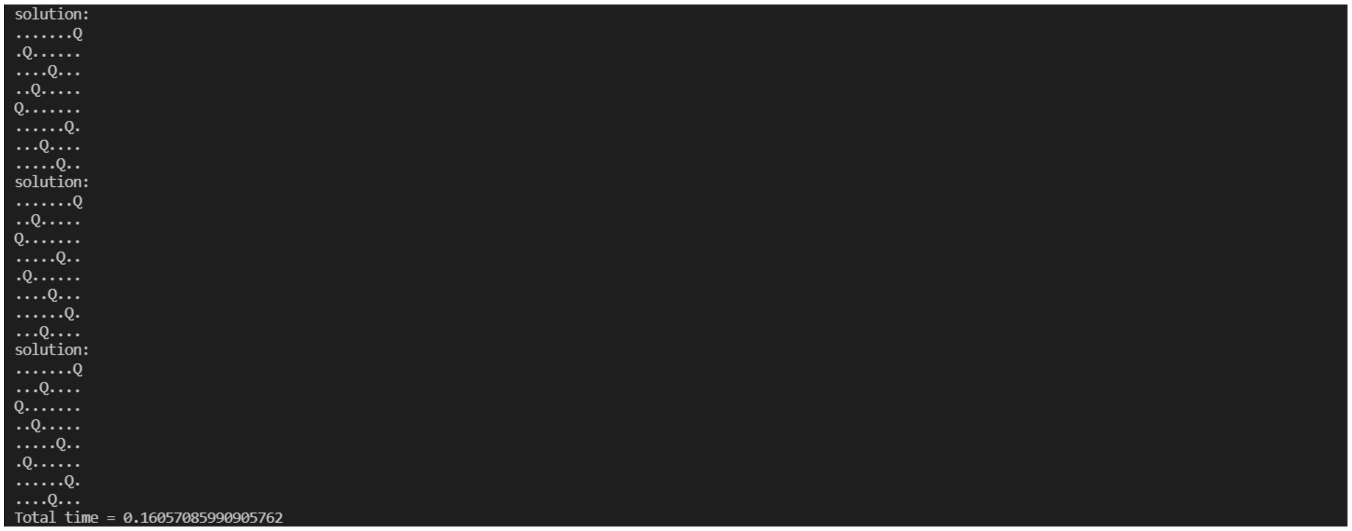


```
[2, 5, 1, 4, 0, 3, 6]
[2, 6, 1, 3, 5, 0, 4]
[2, 6, 3, 0, 4, 1, 5]
[3, 0, 2, 5, 1, 6, 4]
[3, 0, 4, 1, 5, 2, 6]
[3, 1, 6, 4, 2, 0, 5]
[3, 5, 0, 2, 4, 6, 1]
[3, 6, 2, 5, 1, 4, 0]
[3, 6, 4, 1, 5, 0, 2]
[4, 0, 3, 6, 2, 5, 1]
[4, 0, 5, 3, 1, 6, 2]
[4, 1, 5, 2, 6, 3, 0]
[4, 2, 0, 5, 3, 1, 6]
[4, 6, 1, 3, 5, 0, 2]
[4, 6, 1, 5, 2, 0, 3]
[5, 0, 2, 4, 6, 1, 3]
[5, 1, 4, 0, 3, 6, 2]
[5, 2, 0, 3, 6, 4, 1]
[5, 2, 4, 6, 0, 3, 1]
[5, 2, 6, 3, 0, 4, 1]
[5, 3, 1, 6, 4, 2, 0]
[5, 3, 6, 0, 2, 4, 1]
[6, 1, 3, 5, 0, 2, 4]
[6, 2, 5, 1, 4, 0, 3]
[6, 3, 0, 4, 1, 5, 2]
[6, 4, 2, 0, 5, 3, 1]
number of solutions =  40
Total time = 0.008970737457275390
```

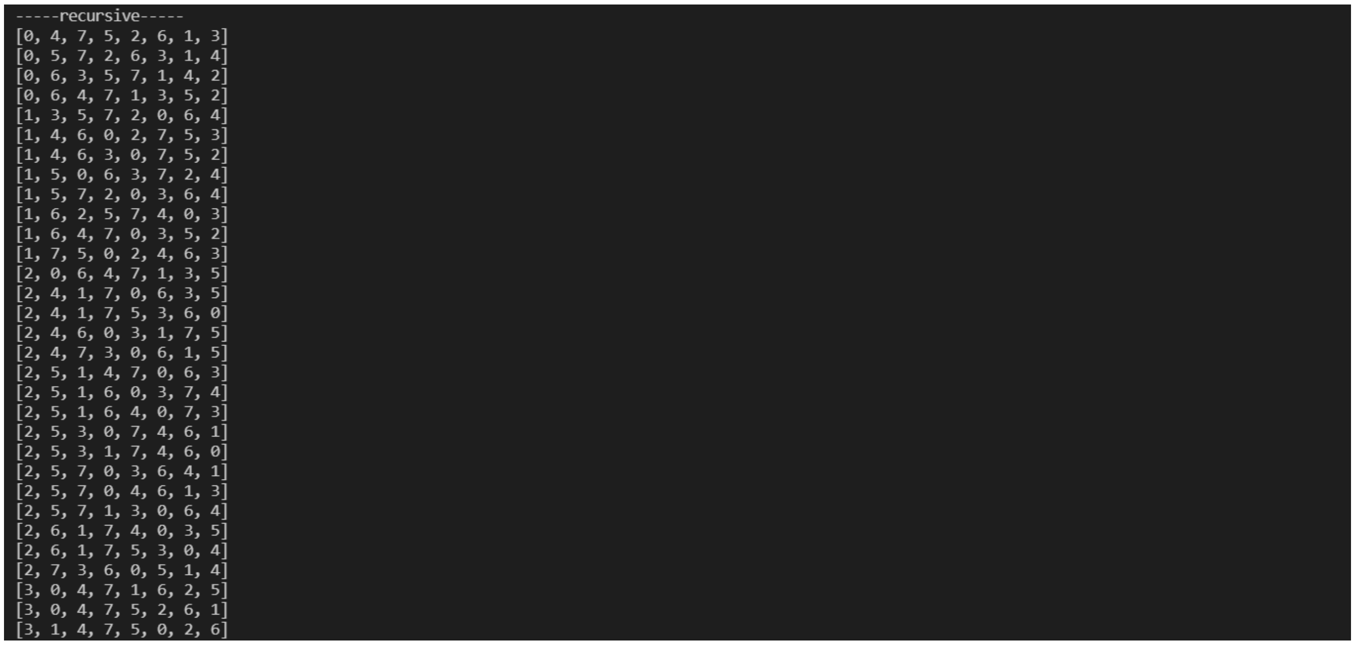
- Input = 8



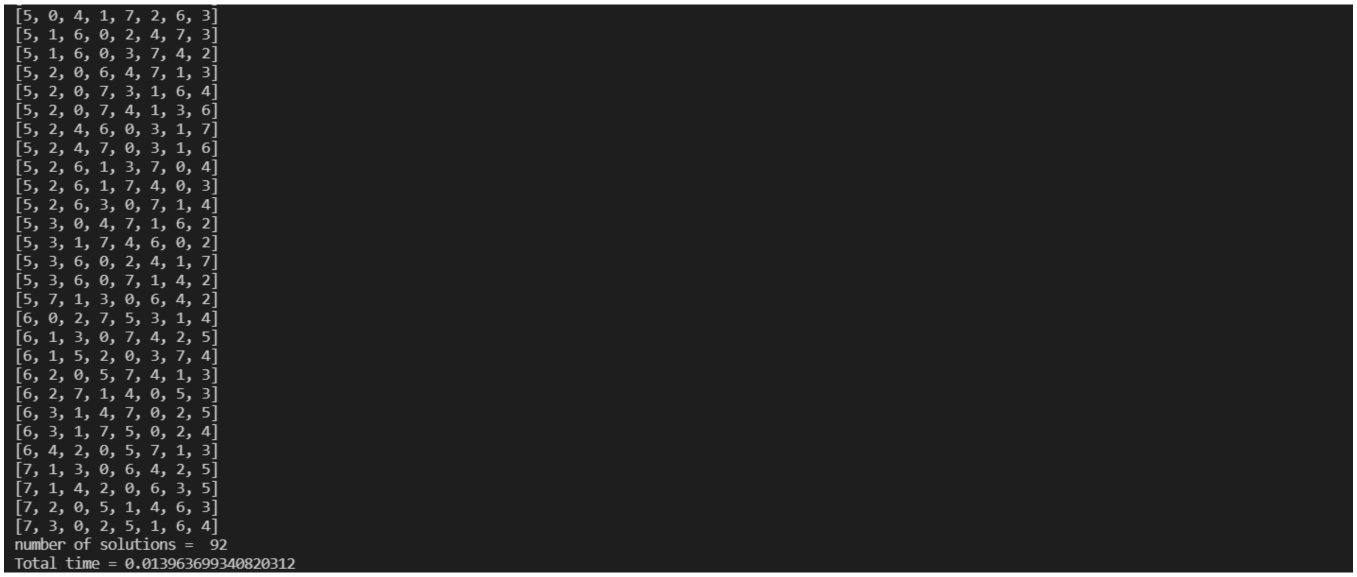
```
Enter input : 8
-----iterative-----
solution:
Q.......
....Q...
.......Q
.....Q..
..Q.....
......Q.
.Q......
...Q....
solution:
Q.......
.....Q..
.......Q
..Q.....
......Q.
...Q....
.Q......
....Q...
solution:
Q.......
......Q.
...Q....
.....Q..
.......Q
.Q......
....Q...
..Q.....
solution:
Q.......
......Q.
....Q...
.......Q
.Q......
...Q....
```



```
solution:
.......Q
.Q......
....Q...
..Q.....
Q.......
......Q.
...Q....
.....Q..
solution:
.......Q
..Q.....
Q.......
.....Q..
.Q......
....Q...
......Q.
...Q....
solution:
.......Q
...Q....
Q.......
..Q.....
.....Q..
.Q......
......Q.
....Q...
Total time = 0.16057085990905762
```

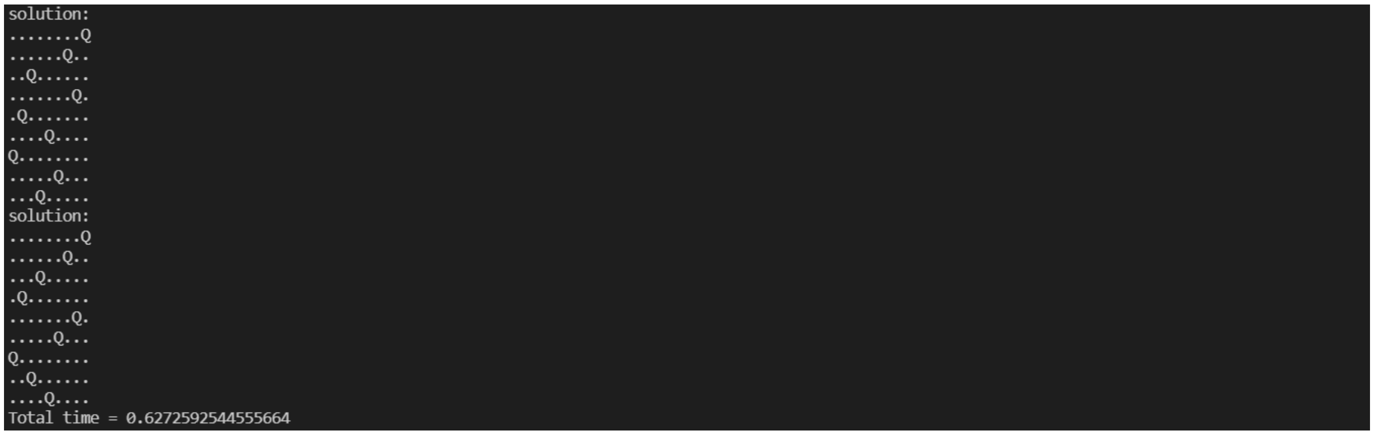


```
-----recursive-----
[0, 4, 7, 5, 2, 6, 1, 3]
[0, 5, 7, 2, 6, 3, 1, 4]
[0, 6, 3, 5, 7, 1, 4, 2]
[0, 6, 4, 7, 1, 3, 5, 2]
[1, 3, 5, 7, 2, 0, 6, 4]
[1, 4, 6, 0, 2, 7, 5, 3]
[1, 4, 6, 3, 0, 7, 5, 2]
[1, 5, 0, 6, 3, 7, 2, 4]
[1, 5, 7, 2, 0, 3, 6, 4]
[1, 6, 2, 5, 7, 4, 0, 3]
[1, 6, 4, 7, 0, 3, 5, 2]
[1, 7, 5, 0, 2, 4, 6, 3]
[2, 0, 6, 4, 7, 1, 3, 5]
[2, 4, 1, 7, 0, 6, 3, 5]
[2, 4, 1, 7, 5, 3, 6, 0]
[2, 4, 6, 0, 3, 1, 7, 5]
[2, 4, 7, 3, 0, 6, 1, 5]
[2, 5, 1, 4, 7, 0, 6, 3]
[2, 5, 1, 6, 0, 3, 7, 4]
[2, 5, 1, 6, 4, 0, 7, 3]
[2, 5, 3, 0, 7, 4, 6, 1]
[2, 5, 3, 1, 7, 4, 6, 0]
[2, 5, 7, 0, 3, 6, 4, 1]
[2, 5, 7, 0, 4, 6, 1, 3]
[2, 5, 7, 1, 3, 0, 6, 4]
[2, 6, 1, 7, 4, 0, 3, 5]
[2, 6, 1, 7, 5, 3, 0, 4]
[2, 7, 3, 6, 0, 5, 1, 4]
[3, 0, 4, 7, 1, 6, 2, 5]
[3, 0, 4, 7, 5, 2, 6, 1]
[3, 1, 4, 7, 5, 0, 2, 6]
```

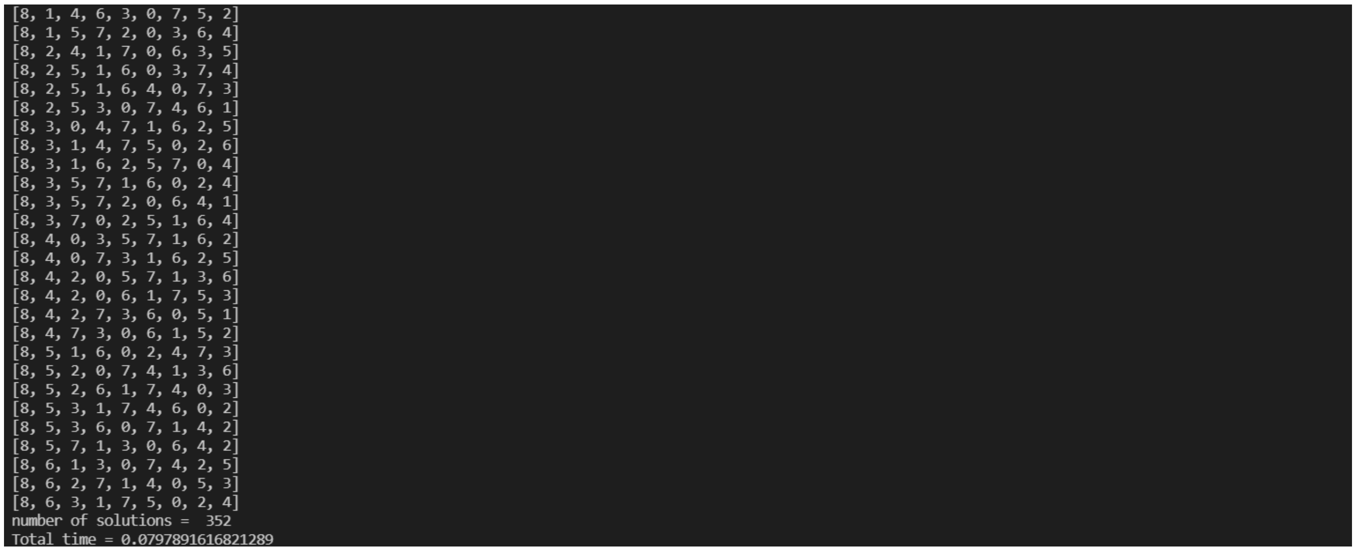


```
[5, 0, 4, 1, 7, 2, 6, 3]
[5, 1, 6, 0, 2, 4, 7, 3]
[5, 1, 6, 0, 3, 7, 4, 2]
[5, 2, 0, 6, 4, 7, 1, 3]
[5, 2, 0, 7, 3, 1, 6, 4]
[5, 2, 0, 7, 4, 1, 3, 6]
[5, 2, 4, 6, 0, 3, 1, 7]
[5, 2, 4, 7, 0, 3, 1, 6]
[5, 2, 6, 1, 3, 7, 0, 4]
[5, 2, 6, 1, 7, 4, 0, 3]
[5, 2, 6, 3, 0, 7, 1, 4]
[5, 3, 0, 4, 7, 1, 6, 2]
[5, 3, 1, 7, 4, 6, 0, 2]
[5, 3, 6, 0, 2, 4, 1, 7]
[5, 3, 6, 0, 7, 1, 4, 2]
[5, 7, 1, 3, 0, 6, 4, 2]
[6, 0, 2, 7, 5, 3, 1, 4]
[6, 1, 3, 0, 7, 4, 2, 5]
[6, 1, 5, 2, 0, 3, 7, 4]
[6, 2, 0, 5, 7, 4, 1, 3]
[6, 2, 7, 1, 4, 0, 5, 3]
[6, 3, 1, 4, 7, 0, 2, 5]
[6, 3, 1, 7, 5, 0, 2, 4]
[6, 4, 2, 0, 5, 7, 1, 3]
[7, 1, 3, 0, 6, 4, 2, 5]
[7, 1, 4, 2, 0, 6, 3, 5]
[7, 2, 0, 5, 1, 4, 6, 3]
[7, 3, 0, 2, 5, 1, 6, 4]
number of solutions =  92
Total time = 0.013963699340820312
```

- Input = 9

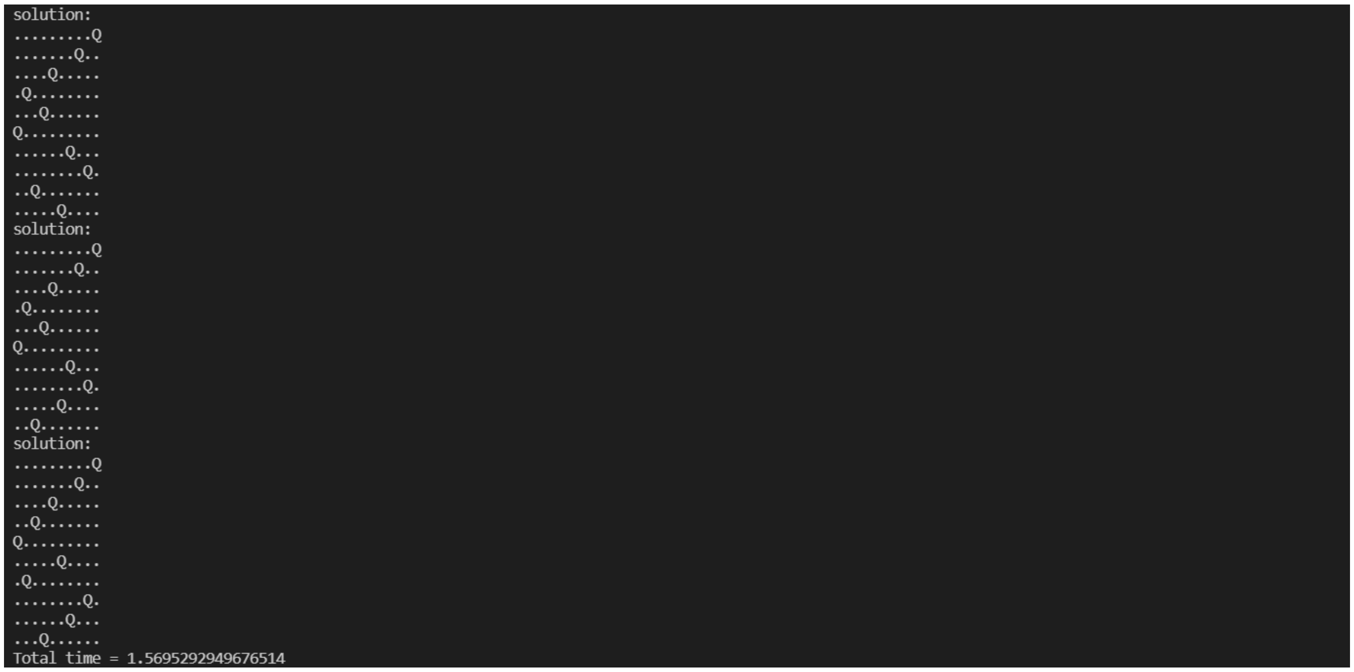


```
solution:
........Q
......Q..
..Q......
.......Q.
.Q.......
....Q....
Q........
.....Q...
...Q.....
solution:
........Q
......Q..
...Q.....
.Q.......
.......Q.
.....Q...
Q........
..Q......
....Q....
Total time = 0.6272592544555664
```

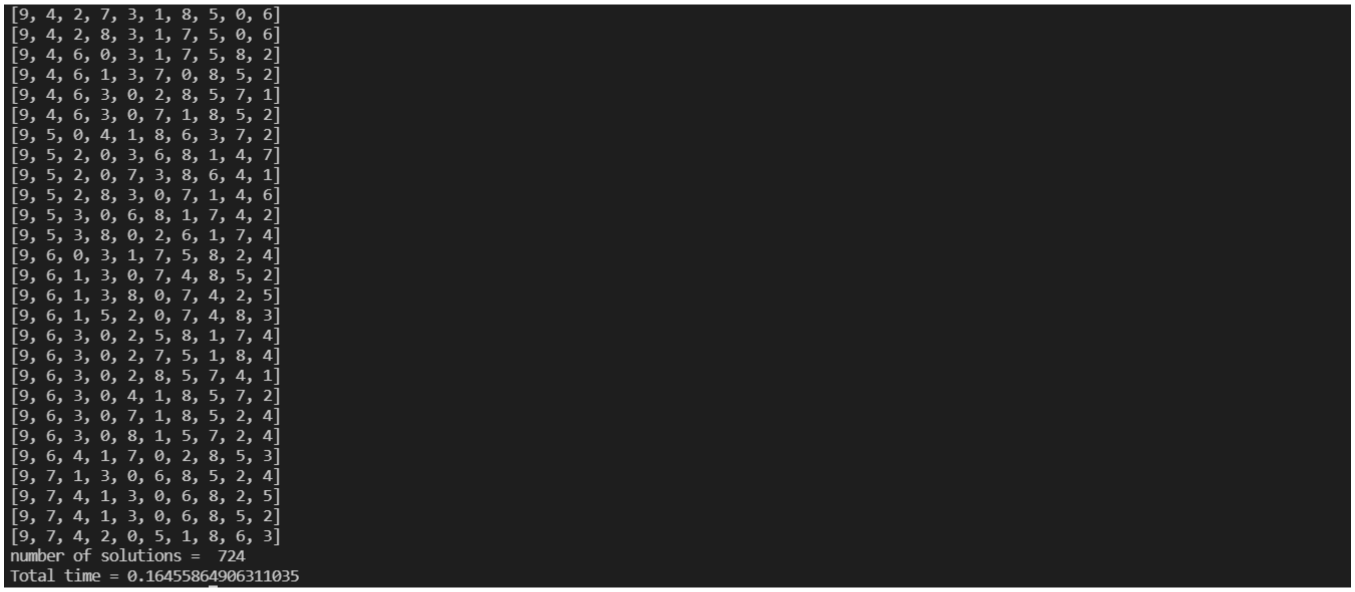


```
[8, 1, 4, 6, 3, 0, 7, 5, 2]
[8, 1, 5, 7, 2, 0, 3, 6, 4]
[8, 2, 4, 1, 7, 0, 6, 3, 5]
[8, 2, 5, 1, 6, 0, 3, 7, 4]
[8, 2, 5, 1, 6, 4, 0, 7, 3]
[8, 2, 5, 3, 0, 7, 4, 6, 1]
[8, 3, 0, 4, 7, 1, 6, 2, 5]
[8, 3, 1, 4, 7, 5, 0, 2, 6]
[8, 3, 1, 6, 2, 5, 7, 0, 4]
[8, 3, 5, 7, 1, 6, 0, 2, 4]
[8, 3, 5, 7, 2, 0, 6, 4, 1]
[8, 3, 7, 0, 2, 5, 1, 6, 4]
[8, 4, 0, 3, 5, 7, 1, 6, 2]
[8, 4, 0, 7, 3, 1, 6, 2, 5]
[8, 4, 2, 0, 5, 7, 1, 3, 6]
[8, 4, 2, 0, 6, 1, 7, 5, 3]
[8, 4, 2, 7, 3, 6, 0, 5, 1]
[8, 4, 7, 3, 0, 6, 1, 5, 2]
[8, 5, 1, 6, 0, 2, 4, 7, 3]
[8, 5, 2, 0, 7, 4, 1, 3, 6]
[8, 5, 2, 6, 1, 7, 4, 0, 3]
[8, 5, 3, 1, 7, 4, 6, 0, 2]
[8, 5, 3, 6, 0, 7, 1, 4, 2]
[8, 5, 7, 1, 3, 0, 6, 4, 2]
[8, 6, 1, 3, 0, 7, 4, 2, 5]
[8, 6, 2, 7, 1, 4, 0, 5, 3]
[8, 6, 3, 1, 7, 5, 0, 2, 4]
number of solutions =  352
Total time = 0.0797891616821289
```

- Input = 10

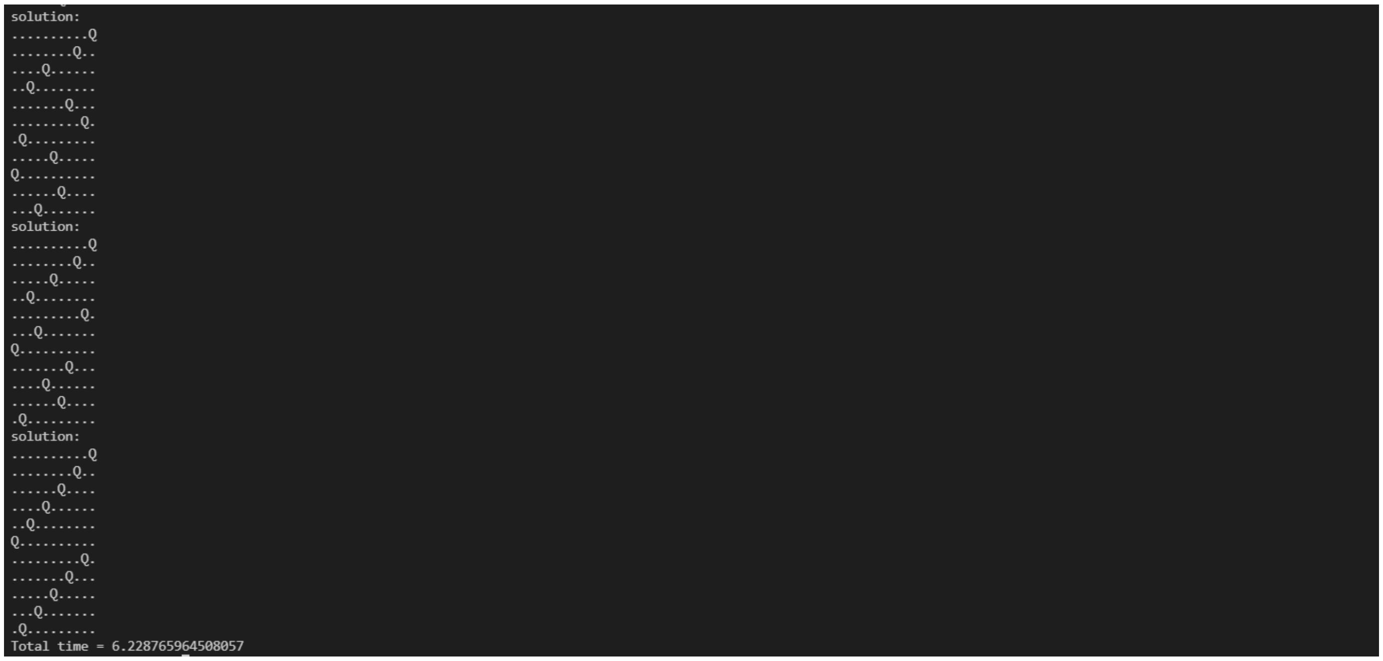


```
solution:
.........Q
.......Q..
....Q.....
.Q........
...Q......
Q.........
......Q...
........Q.
..Q.......
.....Q....
solution:
.........Q
.......Q..
....Q.....
.Q........
...Q......
Q.........
......Q...
........Q.
.....Q....
..Q.......
solution:
.........Q
.......Q..
....Q.....
..Q.......
Q.........
.....Q....
.Q........
........Q.
......Q...
...Q......
Total time = 1.5695292949676514
```

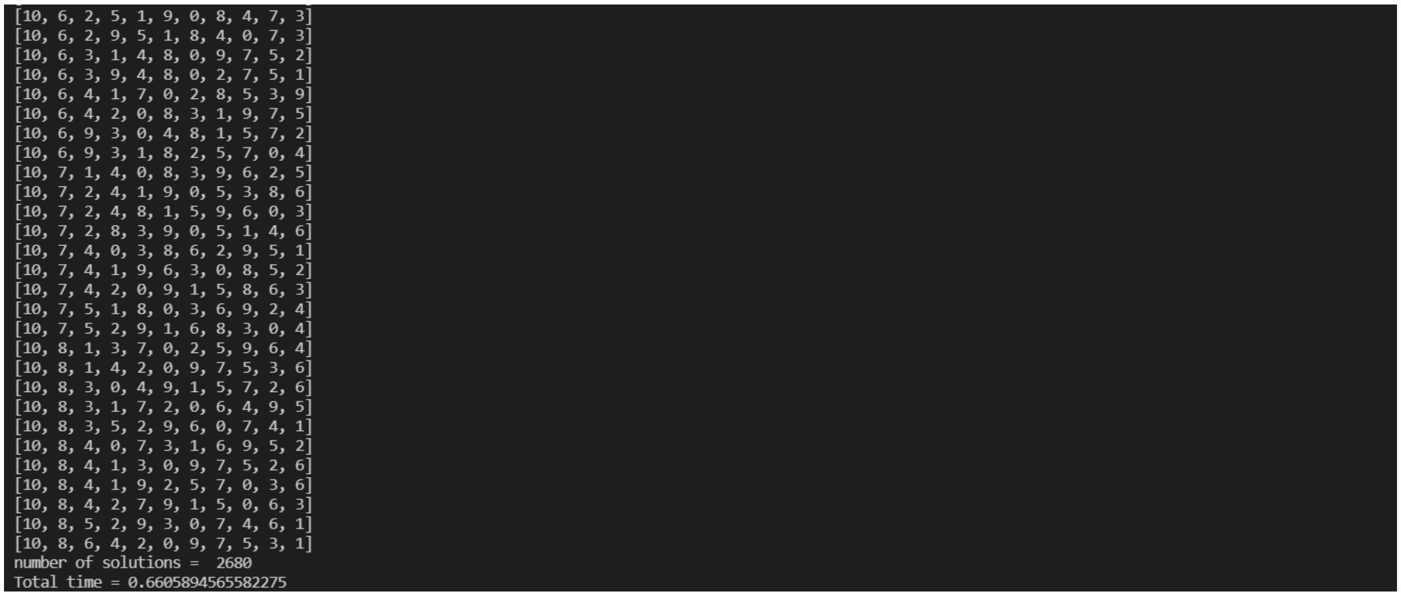


```
[9, 4, 2, 7, 3, 1, 8, 5, 0, 6]
[9, 4, 2, 8, 3, 1, 7, 5, 0, 6]
[9, 4, 6, 0, 3, 1, 7, 5, 8, 2]
[9, 4, 6, 1, 3, 7, 0, 8, 5, 2]
[9, 4, 6, 3, 0, 2, 8, 5, 7, 1]
[9, 4, 6, 3, 0, 7, 1, 8, 5, 2]
[9, 5, 0, 4, 1, 8, 6, 3, 7, 2]
[9, 5, 2, 0, 3, 6, 8, 1, 4, 7]
[9, 5, 2, 0, 7, 3, 8, 6, 4, 1]
[9, 5, 2, 8, 3, 0, 7, 1, 4, 6]
[9, 5, 3, 0, 6, 8, 1, 7, 4, 2]
[9, 5, 3, 8, 0, 2, 6, 1, 7, 4]
[9, 6, 0, 3, 1, 7, 5, 8, 2, 4]
[9, 6, 1, 3, 0, 7, 4, 8, 5, 2]
[9, 6, 1, 3, 8, 0, 7, 4, 2, 5]
[9, 6, 1, 5, 2, 0, 7, 4, 8, 3]
[9, 6, 3, 0, 2, 5, 8, 1, 7, 4]
[9, 6, 3, 0, 2, 7, 5, 1, 8, 4]
[9, 6, 3, 0, 2, 8, 5, 7, 4, 1]
[9, 6, 3, 0, 4, 1, 8, 5, 7, 2]
[9, 6, 3, 0, 7, 1, 8, 5, 2, 4]
[9, 6, 3, 0, 8, 1, 5, 7, 2, 4]
[9, 6, 4, 1, 7, 0, 2, 8, 5, 3]
[9, 7, 1, 3, 0, 6, 8, 5, 2, 4]
[9, 7, 4, 1, 3, 0, 6, 8, 2, 5]
[9, 7, 4, 1, 3, 0, 6, 8, 5, 2]
[9, 7, 4, 2, 0, 5, 1, 8, 6, 3]
number of solutions =  724
Total time = 0.16455864906311035
```

- Input = 11

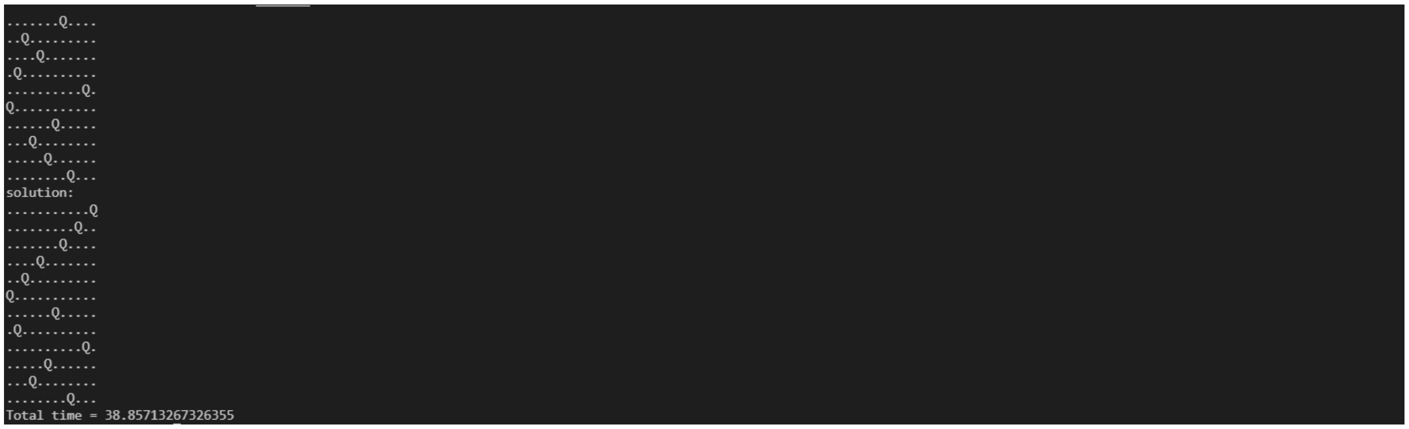


```
solution:
..........Q
........Q..
....Q......
..Q........
.......Q...
.........Q.
.Q.........
.....Q.....
Q..........
......Q....
...Q.......
solution:
..........Q
........Q..
.....Q.....
..Q........
.........Q.
...Q.......
Q..........
.......Q...
....Q......
......Q....
.Q.........
solution:
..........Q
........Q..
......Q....
....Q......
..Q........
Q..........
.........Q.
.......Q...
.....Q.....
...Q.......
.Q.........
Total time = 6.228765964508057
```

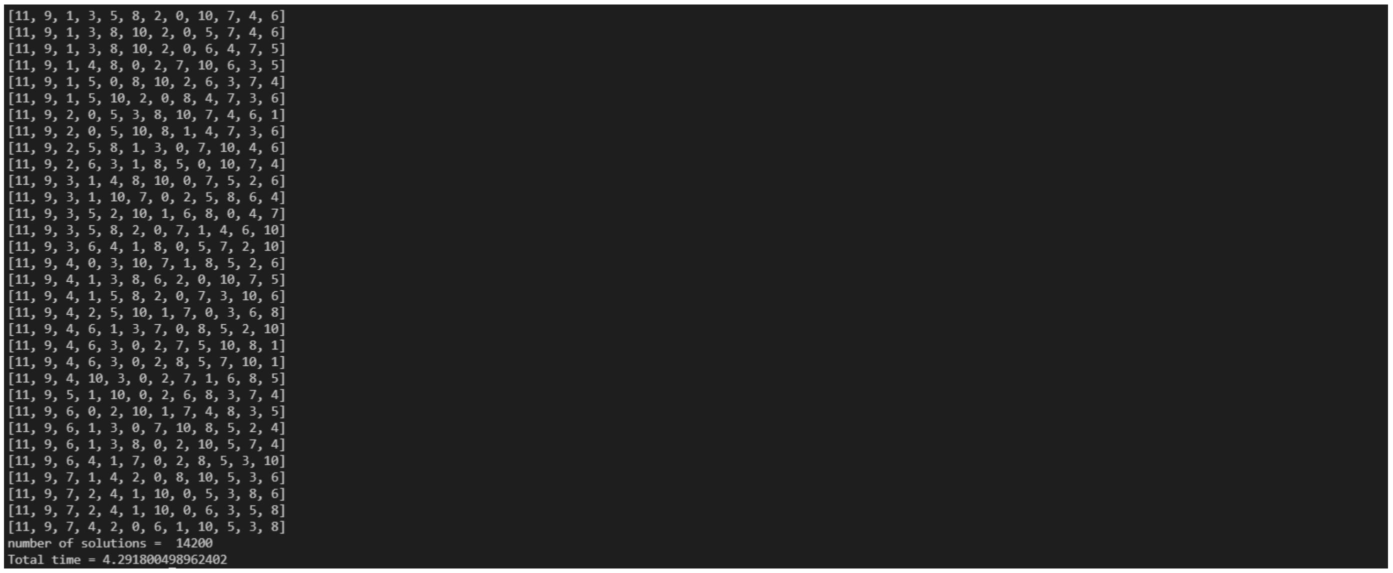


```
[10, 6, 2, 5, 1, 9, 0, 8, 4, 7, 3]
[10, 6, 2, 9, 5, 1, 8, 4, 0, 7, 3]
[10, 6, 3, 1, 4, 8, 0, 9, 7, 5, 2]
[10, 6, 3, 9, 4, 8, 0, 2, 7, 5, 1]
[10, 6, 4, 1, 7, 0, 2, 8, 5, 3, 9]
[10, 6, 4, 2, 0, 8, 3, 1, 9, 7, 5]
[10, 6, 9, 3, 0, 4, 8, 1, 5, 7, 2]
[10, 6, 9, 3, 1, 8, 2, 5, 7, 0, 4]
[10, 7, 1, 4, 0, 8, 3, 9, 6, 2, 5]
[10, 7, 2, 4, 1, 9, 0, 5, 3, 8, 6]
[10, 7, 2, 4, 8, 1, 5, 9, 6, 0, 3]
[10, 7, 2, 8, 3, 9, 0, 5, 1, 4, 6]
[10, 7, 4, 0, 3, 8, 6, 2, 9, 5, 1]
[10, 7, 4, 1, 9, 6, 3, 0, 8, 5, 2]
[10, 7, 4, 2, 0, 9, 1, 5, 8, 6, 3]
[10, 7, 5, 1, 8, 0, 3, 6, 9, 2, 4]
[10, 7, 5, 2, 9, 1, 6, 8, 3, 0, 4]
[10, 8, 1, 3, 7, 0, 2, 5, 9, 6, 4]
[10, 8, 1, 4, 2, 0, 9, 7, 5, 3, 6]
[10, 8, 3, 0, 4, 9, 1, 5, 7, 2, 6]
[10, 8, 3, 1, 7, 2, 0, 6, 4, 9, 5]
[10, 8, 3, 5, 2, 9, 6, 0, 7, 4, 1]
[10, 8, 4, 0, 7, 3, 1, 6, 9, 5, 2]
[10, 8, 4, 1, 3, 0, 9, 7, 5, 2, 6]
[10, 8, 4, 1, 9, 2, 5, 7, 0, 3, 6]
[10, 8, 4, 2, 7, 9, 1, 5, 0, 6, 3]
[10, 8, 5, 2, 9, 3, 0, 7, 4, 6, 1]
[10, 8, 6, 4, 2, 0, 9, 7, 5, 3, 1]
number of solutions =  2680
Total time = 0.6605894565582275
```

- Input = 12



```
.......Q....
..Q.........
....Q.......
.Q..........
..........Q.
Q...........
......Q.....
...Q........
.....Q......
........Q...
solution:
...........Q
.........Q..
.......Q....
....Q.......
..Q.........
Q...........
......Q.....
.Q..........
..........Q.
.....Q......
...Q........
........Q...
Total time = 38.85713267326355
```



```
[11, 9, 1, 3, 5, 8, 2, 0, 10, 7, 4, 6]
[11, 9, 1, 3, 8, 10, 2, 0, 5, 7, 4, 6]
[11, 9, 1, 3, 8, 10, 2, 0, 6, 4, 7, 5]
[11, 9, 1, 4, 8, 0, 2, 7, 10, 6, 3, 5]
[11, 9, 1, 5, 0, 8, 10, 2, 6, 3, 7, 4]
[11, 9, 1, 5, 10, 2, 0, 8, 4, 7, 3, 6]
[11, 9, 2, 0, 5, 3, 8, 10, 7, 4, 6, 1]
[11, 9, 2, 0, 5, 10, 8, 1, 4, 7, 3, 6]
[11, 9, 2, 5, 8, 1, 3, 0, 7, 10, 4, 6]
[11, 9, 2, 6, 3, 1, 8, 5, 0, 10, 7, 4]
[11, 9, 3, 1, 4, 8, 10, 0, 7, 5, 2, 6]
[11, 9, 3, 1, 10, 7, 0, 2, 5, 8, 6, 4]
[11, 9, 3, 5, 2, 10, 1, 6, 8, 0, 4, 7]
[11, 9, 3, 5, 8, 2, 0, 7, 1, 4, 6, 10]
[11, 9, 3, 6, 4, 1, 8, 0, 5, 7, 2, 10]
[11, 9, 4, 0, 3, 10, 7, 1, 8, 5, 2, 6]
[11, 9, 4, 1, 3, 8, 6, 2, 0, 10, 7, 5]
[11, 9, 4, 1, 5, 8, 2, 0, 7, 3, 10, 6]
[11, 9, 4, 2, 5, 10, 1, 7, 0, 3, 6, 8]
[11, 9, 4, 6, 1, 3, 7, 0, 8, 5, 2, 10]
[11, 9, 4, 6, 3, 0, 2, 7, 5, 10, 8, 1]
[11, 9, 4, 6, 3, 0, 2, 8, 5, 7, 10, 1]
[11, 9, 4, 10, 3, 0, 2, 7, 1, 6, 8, 5]
[11, 9, 5, 1, 10, 0, 2, 6, 8, 3, 7, 4]
[11, 9, 6, 0, 2, 10, 1, 7, 4, 8, 3, 5]
[11, 9, 6, 1, 3, 0, 7, 10, 8, 5, 2, 4]
[11, 9, 6, 1, 3, 8, 0, 2, 10, 5, 7, 4]
[11, 9, 6, 4, 1, 7, 0, 2, 8, 5, 3, 10]
[11, 9, 7, 1, 4, 2, 0, 8, 10, 5, 3, 6]
[11, 9, 7, 2, 4, 1, 10, 0, 5, 3, 8, 6]
[11, 9, 7, 2, 4, 1, 10, 0, 6, 3, 5, 8]
[11, 9, 7, 4, 2, 0, 6, 1, 10, 5, 3, 8]
number of solutions =  14200
Total time = 4.291800498962402
```

4. แหล่งอ้างอิง บรรณานุกรม

- https://colab.research.google.com/drive/1nhVvTij1LuF-nB1okf9MHtyTdpmARzdG
- https://stackoverflow.com/questions/42318343/avoid-duplicates-in-n-queen-iterative-solutions-no-recursion-allowed

5. ตารางบันทึกผล

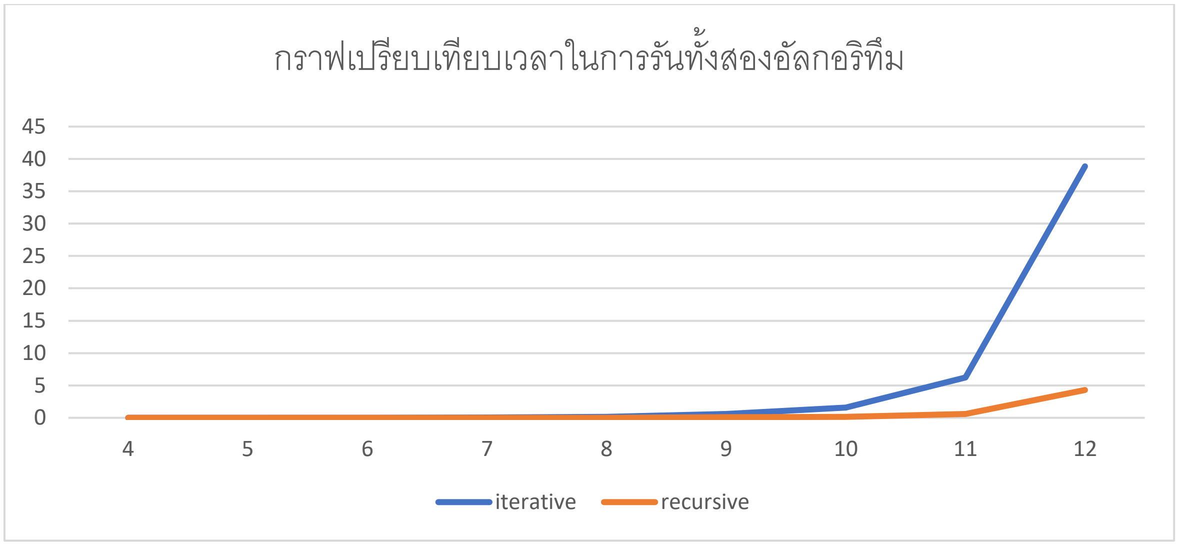
- โปรแกรม N-Queen โดยใช้ algorithm แบบ iterative

Input	ผลลัพธ์ที่ได้	เวลาที่ใช้ในการรัน(วินาที)
4	2	0.0010042190551757812
5	10	0.008976221084594727
6	4	0.005985260009765625
7	40	0.06682538986206055
8	92	0.16057085990905762
9	352	0.5926032066345215
10	724	1.5695292949676514
11	2680	6.228765964508057
12	14200	38.85713267326355

- โปรแกรม N-Queen โดยใช้ algorithm แบบ recursive

Input	ผลลัพธ์ที่ได้	เวลาที่ใช้ในการรัน(วินาที)
4	2	0.0009906291961669922
5	10	0.001985311508178711
6	4	0.0009970664978027344
7	40	0.00897073745727539
8	92	0.013963699340820312
9	352	0.07143235206604004
10	724	0.16455864906311035
11	2680	0.6024734973907471
12	14200	4.291800498962402

6. กราฟเปรียบเทียบเวลาในการรัน ทั้ง สอง อัลกอริทึม



7. การวิเคราะห์ผลลัพธ์ที่ได้

ผลลัพธ์ที่ได้จากการรันโปรแกรม N-queen โดยใช้อัลกอริทึมแบบ iterative และ recursive คือ การรันโปรแกรมโดยใช้อัลกอริทึมแบบ iterative กับ recursive จะมีจำนวนผลลัพธ์เท่ากัน ส่วนเวลาที่ใช้ในการรันโปรแกรมแบบ iterative จะใช้เวลามากกว่าแบบ recursive