N 皇后实验报告

一、算法说明

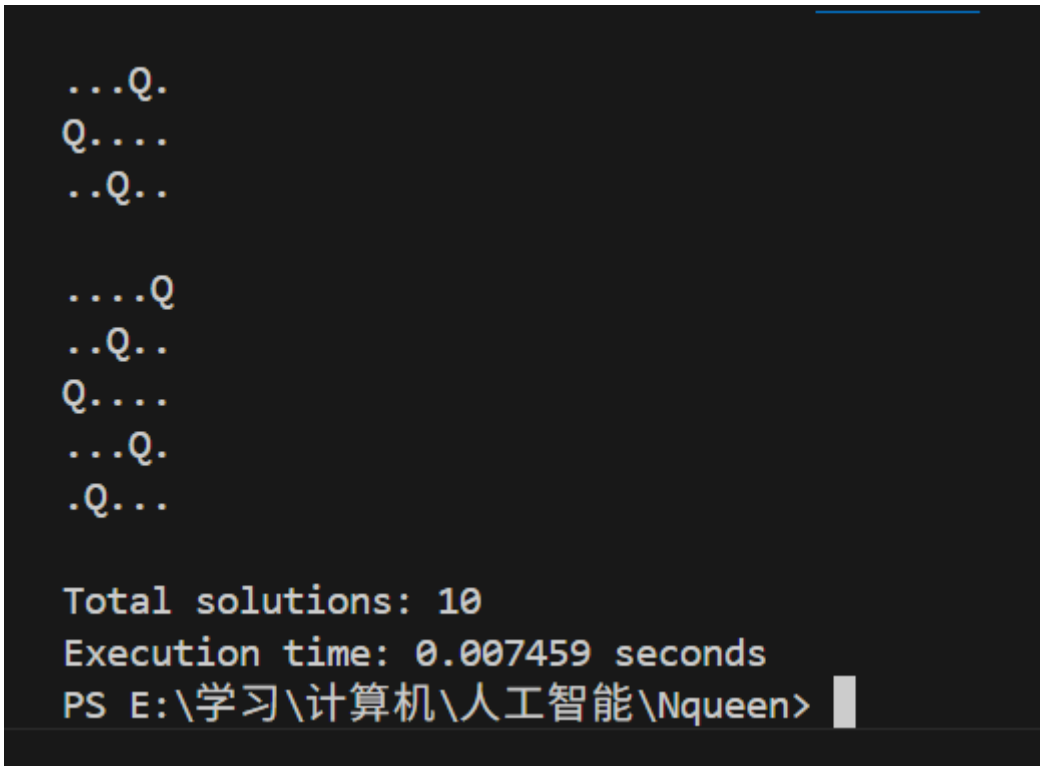
```
import time

n = int(input("input number of queens:"))

g = [['.' for _ in range(n)] for _ in range(n)]

col = [False]*n

dg = [False]*(2*n)

udg = [False]*(2*n)

N = 0




def dfs(u):
    global N  #声明 N 为全局变量
    if u == n:
        #打印当前的解
        for row in g:
            print(''.join(row))
        print()
        N += 1  #在找到一个有效解时增加计数
        return
    for y in range(n):
        #检查是否可以放置皇后
        if not col[y] and not dg[y-u+n] and not udg[y+u]:
            col[y] = True
            dg[y-u+n] = True
            udg[y+u] = True

            g[u][y] = 'Q'

            dfs(u+1)

            #回溯，重置状态
            col[y] = False
            dg[y-u+n] = False
            udg[y+u] = False
```

```
            g[u][y] = '.'


if n < 4:
    print("invalid input")
else:
    start_time = time.time()  # 开始时间
    dfs(0)
    print("Total solutions:", N)  #输出解的总数量
    end_time = time.time()    # 结束时间

    print(f"Execution time: {end_time - start_time:.6f} seconds")
```

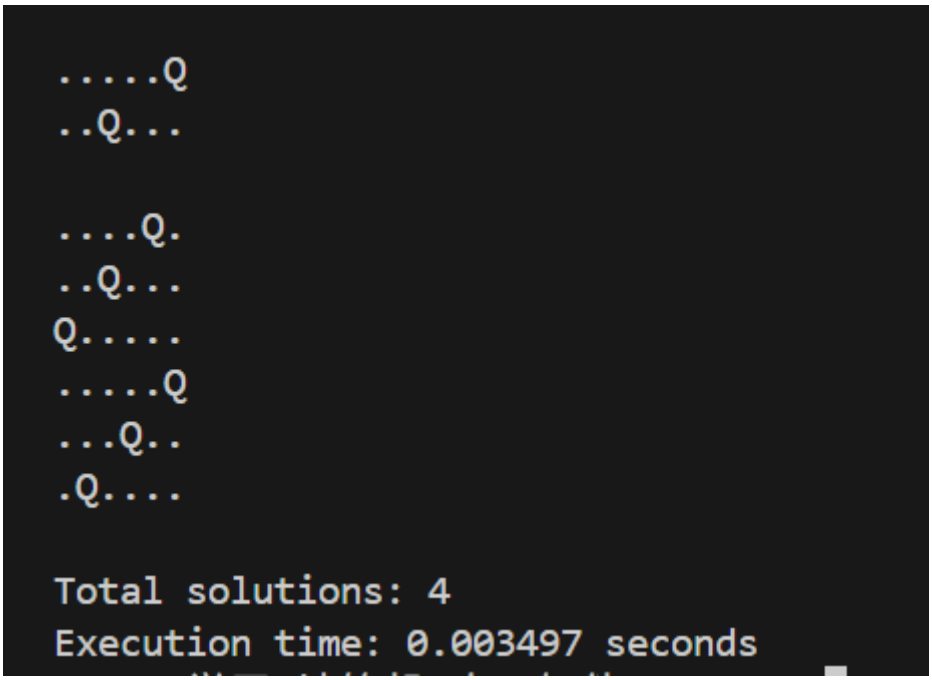
二、实验结果

```
.Q..
...Q
Q...
..Q.

..Q.
Q...
...Q
.Q..

Total solutions: 2
Execution time: 0.002013 seconds
PS E:\学习\计算机\人工智能\Nqueen>
```

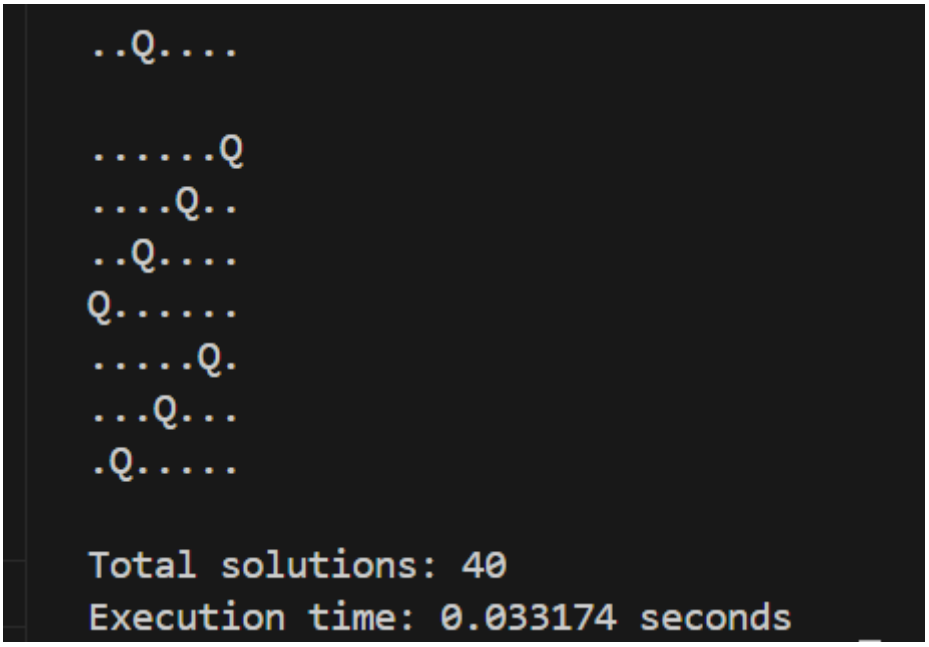
n=4


...Q.
Q....
..Q..
....Q
..Q..
Q....
...Q.
.Q...
Total solutions: 10
Execution time: 0.007459 seconds
PS E:\学习\计算机\人工智能\Nqueen>

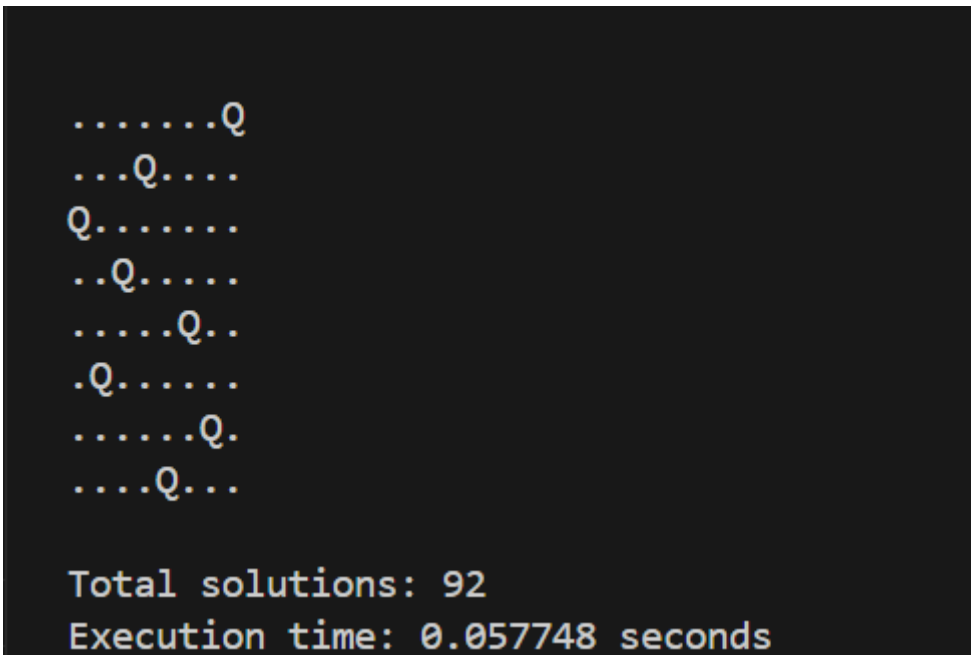
n=5


.....Q
..Q...
....Q.
..Q...
Q.....
.....Q
...Q..
.Q....
Total solutions: 4
Execution time: 0.003497 seconds

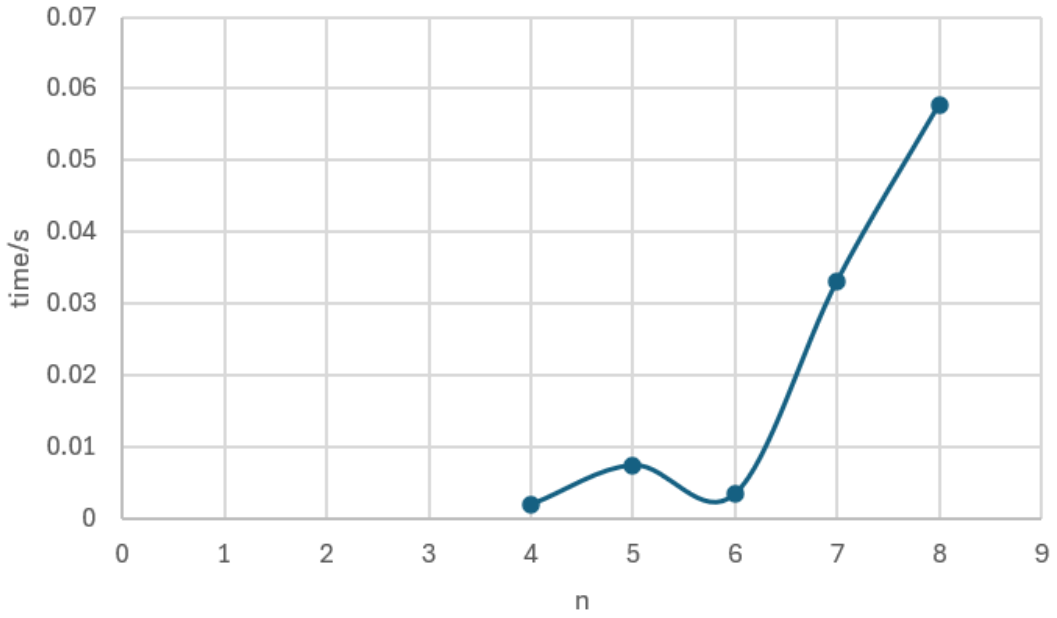
n=6


..Q....
......Q
....Q..
..Q....
Q......
.....Q.
...Q...
.Q.....
Total solutions: 40
Execution time: 0.033174 seconds

n=7


.......Q
...Q....
Q.......
..Q.....
.....Q..
.Q......
......Q.
....Q...
Total solutions: 92
Execution time: 0.057748 seconds

n=8



时间增长曲线

分析：剪枝在很多情况下能大大减少无效的搜索分支，所以实际运行远远快于 O(n!)，尤其在 n 比较小时（如 $n \leq 12$）效率很好。

但是理论复杂度还是不能突破 O(n!) 这个上限，因为在最坏情况下，还是可能遍历所有有效排列。

三、优化思路

1. 减少全局变量使用

   现在用的是 global N，可以改成函数内部变量 count 用 nonlocal 管理，这样代码更整洁、更安全。