N 皇后问题报告

一、算法设计与实现

N 皇后问题本质上是典型的回溯搜索问题，需要在 N×N 棋盘上放置 N 个皇后，满足行列及对角线不冲突。

```
def Trans(S):
    res=0
    for c in S:
        if c<'0' or c>'9':
            return 0
        res=res*10+(ord(c)-48)
    return res

N=Trans(input())
while N==0:
    print("Invalid input, please try again")
    N=Trans(input())
```

```
pos=[0]*N
result=[]
def DFS(t,limC,limL,limR):
    global result
    if t==N:
        result+=[pos.copy()]
        return
    lim=limC|limL|limR
    for i in range(0,N):
        if(t>0 or(i<<1)<N)and(lim>>i&1)==0:
            pos[t]=i
            DFS(t+1,limC|1<<i,(limL|1<<i)>>1,(limR<<1|1<<i+1)&(1<<N)-1)
```

```
import time
StartTime=time.process_time()
DFS(0,0,0,0)
s=len(result)
for i in range(s-1,-1,-1):
    result+=[[N-x-1 for x in result[i]]]
EndTime=time.process_time()
```

```
for r in result:
    print(r)
print(len(result),end=" solutions in totals\n")
```

```
import sys
print(EndTime-StartTime,end=" seconds for searching",file=sys.stderr)
```

```
# 字符串转整数函数（只接受纯数字字符串）
def Trans(S):
    res = 0
    for c in S:
        if c < '0' or c > '9':  # 检查是否为数字字符
            return 0
        res = res * 10 + (ord(c) - 48)  # ord('0') = 48，转为对应数字累加
    return res

# 获取合法整数输入
N = Trans(input())
while N == 0:
    print("Invalid input, please try again")
    N = Trans(input())

# 初始化
pos = [0] * N      # pos[i] 表示第 i 行皇后放在第几列
result = []        # 存储所有解

# 主体 DFS 回溯函数
def DFS(t, limC, limL, limR):
    """
    t    : 当前处理的行
    limC : 当前已占据的列的掩码
    limL : 当前已占据的左上-右下对角线掩码（斜对角线 1）
    limR : 当前已占据的右上-左下对角线掩码（斜对角线 2）
    """
    global result
    if t == N:
        result += [pos.copy()]  # 找到一个合法解，保存副本
        return
    lim = limC | limL | limR    # 所有受限位置合并

    for i in range(0, N):  # 尝试将皇后放置在第 i 列
        # 剪枝：第一行不放置在对称位置以左（利用对称性减少一半搜索）
        if (t > 0 or (i << 1) < N) and ((lim >> i) & 1) == 0:
            pos[t] = i
            # 进入下一行，更新掩码
            DFS(
                t + 1,
                limC | (1 << i),              # 占据当前列
                (limL | (1 << i)) >> 1,       # 左对角线向右移动
                ((limR << 1) | (1 << (i + 1))) & ((1 << N) - 1)  # 右对角线左移
并限制长度
```

```
            )

# 计时开始
import time
StartTime = time.process_time()

# 开始搜索
DFS(0, 0, 0, 0)

# 利用对称性扩展另一半解（镜像）
s = len(result)
for i in range(s - 1, -1, -1):
    result += [[N - x - 1 for x in result[i]]]  # 列号对称翻转

EndTime = time.process_time()

# 打印所有结果
for r in result:
    print(r)

# 输出总解数
print(len(result), end=" solutions in totals\n")

# 打印运行时间到 stderr
import sys
print(EndTime - StartTime, end=" seconds for searching", file=sys.stderr)
```

二、算法优化思路

剪枝策略

利用状态压缩，记录列、主副对角线的占用情况，共三个状态，根据当前的状态以及当前的选择 O(1) 推出 DFS 下一层的状态。同样可以 O(1) 查询到当前的选择是否可用。

对称性优化

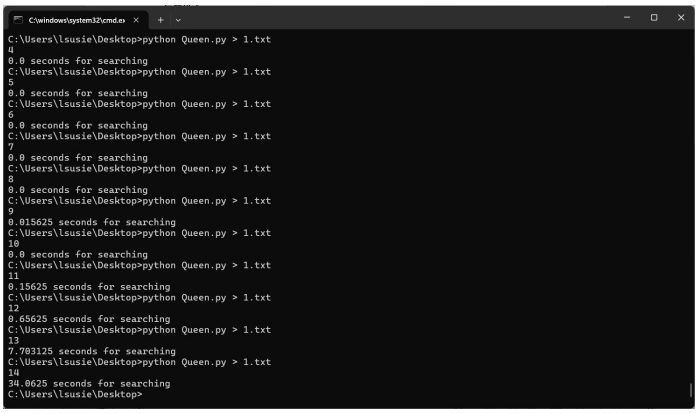
左右镜像对称，可以剪掉一半的搜索时间

三、时间复杂度分析

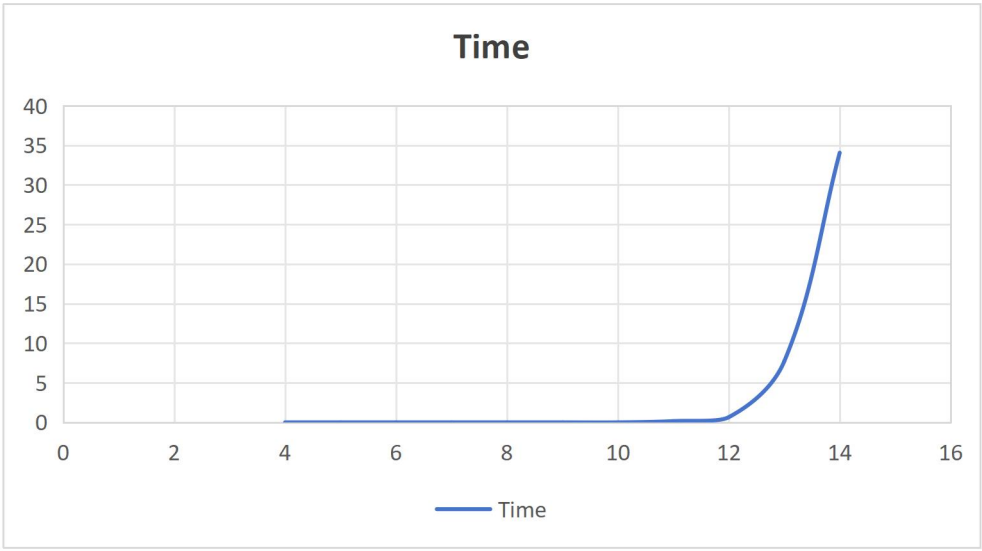
1、理论时间复杂度：最坏情况下为 O (N!)，因每一步选择后搜索空间呈线性级递减

2、实际时间复杂度：通过剪枝优化后，当 N=12 时，找到一个解的时间约为秒级

3、实验数据：N=4 时耗时 0.0 秒，N=8 时耗时 0.0 秒，N=12 时耗时约 0. 65625 秒

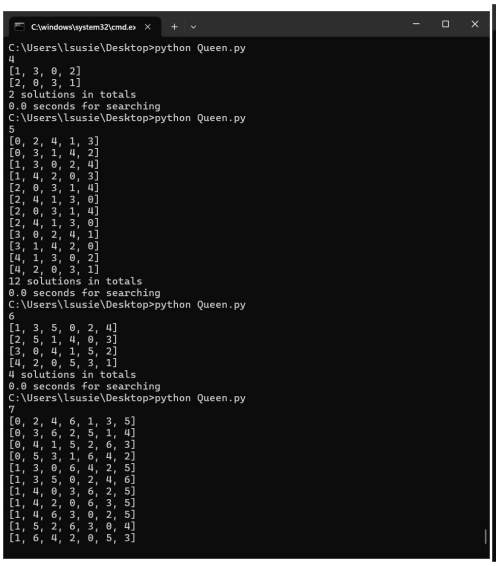
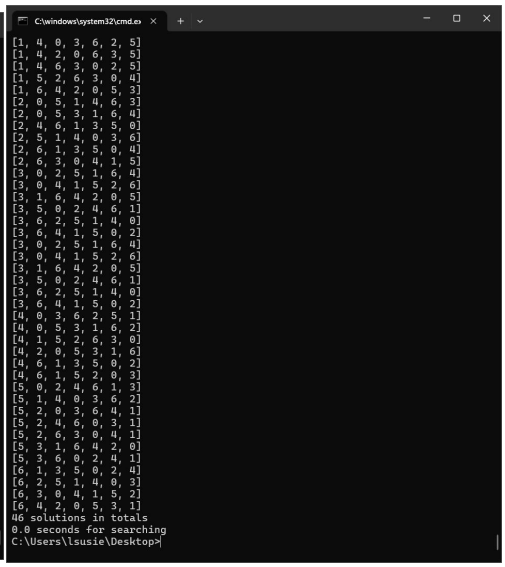
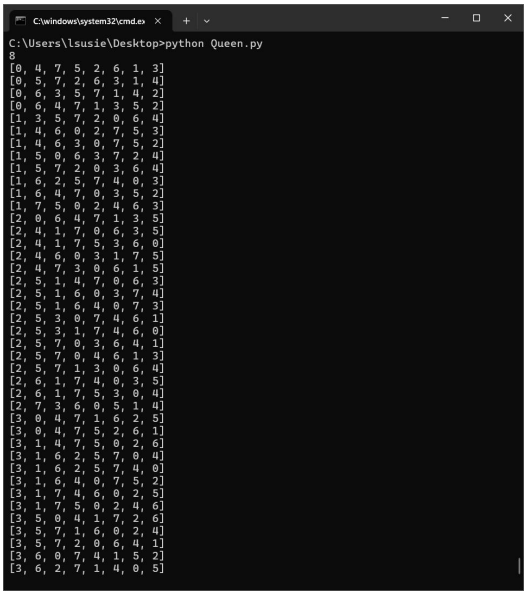
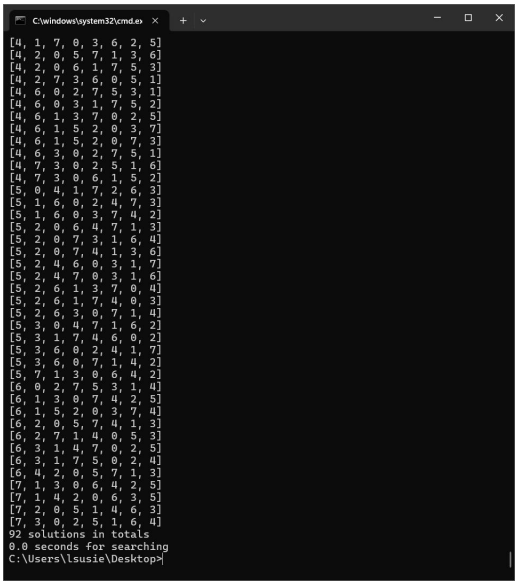


N=4 至 N=12 时的运行时间



时间增长曲线

四、测试用例与输出


C:\windows\system32\cmd.ex
C:\Users\lsusie\Desktop>python Queen.py
4
[1, 3, 0, 2]
[2, 0, 3, 1]
2 solutions in totals
0.0 seconds for searching
C:\Users\lsusie\Desktop>python Queen.py
5
[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]
[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]
12 solutions in totals
0.0 seconds for searching
C:\Users\lsusie\Desktop>python Queen.py
6
[1, 3, 5, 0, 2, 4]
[2, 5, 1, 4, 0, 3]
[3, 0, 4, 1, 5, 2]
[4, 2, 0, 5, 3, 1]
4 solutions in totals
0.0 seconds for searching
C:\Users\lsusie\Desktop>python Queen.py
7
[0, 2, 4, 6, 1, 3, 5]
[0, 3, 6, 2, 5, 1, 4]
[0, 4, 1, 5, 2, 6, 3]
[0, 5, 3, 1, 6, 4, 2]
[1, 3, 0, 6, 4, 2, 5]
[1, 3, 5, 0, 2, 4, 6]
[1, 4, 0, 3, 6, 2, 5]
[1, 4, 2, 0, 6, 3, 5]
[1, 4, 6, 3, 0, 2, 5]
[1, 5, 2, 6, 3, 0, 4]
[1, 6, 4, 2, 0, 5, 3]

C:\windows\system32\cmd.ex
[1, 4, 0, 3, 6, 2, 5]
[1, 4, 2, 0, 6, 3, 5]
[1, 4, 6, 3, 0, 2, 5]
[1, 5, 2, 6, 3, 0, 4]
[1, 6, 4, 2, 0, 5, 3]
[2, 0, 5, 1, 4, 6, 3]
[2, 0, 5, 3, 1, 6, 4]
[2, 4, 6, 1, 3, 5, 0]
[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, 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]
46 solutions in totals
0.0 seconds for searching
C:\Users\lsusie\Desktop>

C:\windows\system32\cmd.ex
C:\Users\lsusie\Desktop>python Queen.py
8
[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]
[3, 1, 6, 2, 5, 7, 0, 4]
[3, 1, 6, 2, 5, 7, 4, 0]
[3, 1, 6, 4, 0, 7, 5, 2]
[3, 1, 7, 4, 6, 0, 2, 5]
[3, 1, 7, 5, 0, 2, 4, 6]
[3, 5, 0, 4, 1, 7, 2, 6]
[3, 5, 7, 1, 6, 0, 2, 4]
[3, 5, 7, 2, 0, 6, 4, 1]
[3, 6, 0, 7, 4, 1, 5, 2]
[3, 6, 2, 7, 1, 4, 0, 5]

C:\windows\system32\cmd.ex
[4, 1, 7, 0, 3, 6, 2, 5]
[4, 2, 0, 5, 7, 1, 3, 6]
[4, 2, 0, 6, 1, 7, 5, 3]
[4, 2, 7, 3, 6, 0, 5, 1]
[4, 6, 0, 2, 7, 5, 3, 1]
[4, 6, 0, 3, 1, 7, 5, 2]
[4, 6, 1, 3, 7, 0, 2, 5]
[4, 6, 1, 5, 2, 0, 3, 7]
[4, 6, 1, 5, 2, 0, 7, 3]
[4, 6, 3, 0, 2, 7, 5, 1]
[4, 7, 3, 0, 2, 5, 1, 6]
[4, 7, 3, 0, 6, 1, 5, 2]
[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]
92 solutions in totals
0.0 seconds for searching
C:\Users\lsusie\Desktop>