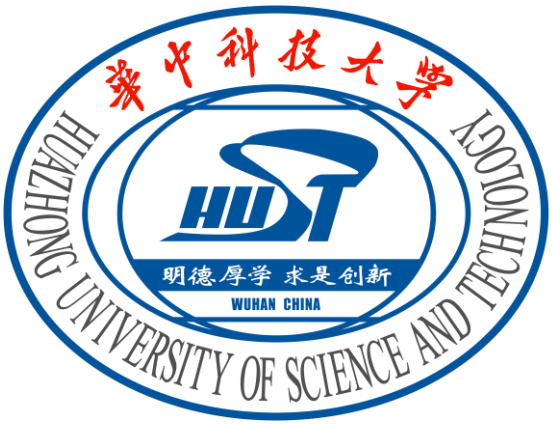

华中科技大学

HUAZHONG UNIVERSITY SCIENCE AND TECHNOLOGY

课 程 实 验 报 告

（实验二）

课程名称 ： 算法分析与设计

院　　系 ： 新闻与信息传播学院

专业班级 ： 传播学 2201 班

学　　号 ： U202217034

姓　　名 ： 余易昕

指导教师 ： 王多强

2025 年 5 月 23 日

实验二：N 皇后问题

https://github.com/sakaaanaYu/CS_Experiments/blob/main/algorithm_learning/code/exp_2.ipynb

1. 需求分析

在国际象棋棋盘上，皇后可以攻击同行、同列、同一斜线上的棋子。在 n×n 的棋盘上放置彼此不受攻击的 n 个皇后，该如何放置？

解题思路如下：

1. 先放行：使用一维数组 row 对每一行皇后的存放位置进行保存；
2. 再排除列：用一维数组 col 保存每一列皇后的摆放情况，col[i] = true 表示第 i 列已然放置皇后，col[i] = false 表示第 i 列暂未放置皇后。
3. 不能处于同一对角线：两皇后坐标分别为(i, j)和(l, k)，有 abs(i-l) != abs(j-k)。

2. 源码及说明

```
# 算法核心函数
def dfs(n, row, res, results):
    if row == n: # 已放置完毕，退出循环
        print(res)
        results.append(res.copy())
        return

    for col in range(n): # 对每一行来说，都需要从头开始遍历列，确保放置的内容不与任何列重合
        if judge_col_and_diagonal(res, row, col):
            res[row] = col # 通过检测，放置结果进 res
            dfs(n, row + 1, res, results) # 上一个放置成功了就需要从 n = 0 开始遍历下一行，否则接着遍历就行
            res[row] = 0 # 有放置未成功，则初始化该行结果，继续大循环

def judge_col_and_diagonal(res, row, col):
    for i in range(row):
        if res[i] == col or abs(res[i] - col) == abs(i - row):
            return False
    return True

# 输出实现
def draw_chessboard(n, res):
    for i in range(n): # 绘制每一行
        for j in range(n): # 绘制行内每一列
            if res[i] == j:
                print('1', end = ' ')
```

```
            else:
                print('0', end = ' ')
        print()
```

3. 代码测试

```
n = int(input("请输入皇后数量："))

if __name__ == "__main__":
    # 初始化 res 和 row
    res = [0] * n
    row = 0

    results = []
    dfs(n, row, res, results)

    for res in results:
        draw_chessboard(n, res)
        print('\n')
```

1. **N = 4：**

```
[1, 3, 0, 2]
[2, 0, 3, 1]
0 1 0 0
0 0 0 1
1 0 0 0
0 0 1 0



0 0 1 0
1 0 0 0
0 0 0 1
0 1 0 0
```

2. **N = 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]
[3, 0, 2, 4, 1]
[3, 1, 4, 2, 0]
[4, 1, 3, 0, 2]
[4, 2, 0, 3, 1]
1 0 0 0 0
0 0 1 0 0
0 0 0 0 1
0 1 0 0 0
0 0 0 1 0


1 0 0 0 0
0 0 0 1 0
0 1 0 0 0
0 0 0 0 1
0 0 1 0 0


0 1 0 0 0
0 0 0 1 0
1 0 0 0 0
0 0 1 0 0
0 0 0 0 1
```

```
0 1 0 0 0
0 0 0 0 1
0 0 1 0 0
1 0 0 0 0
0 0 0 1 0


0 0 1 0 0
1 0 0 0 0
0 0 0 1 0
0 1 0 0 0
0 0 0 0 1


0 0 1 0 0
0 0 0 0 1
0 1 0 0 0
0 0 0 1 0
1 0 0 0 0


0 0 0 1 0
1 0 0 0 0
0 0 1 0 0
0 0 0 0 1
0 1 0 0 0
```

```
0 0 0 1 0
0 1 0 0 0
0 0 0 0 1
0 0 1 0 0
1 0 0 0 0


0 0 0 0 1
0 1 0 0 0
0 0 0 1 0
1 0 0 0 0
0 0 1 0 0


0 0 0 0 1
0 0 1 0 0
1 0 0 0 0
0 0 0 1 0
0 1 0 0 0
```