人工智能基础课程报告

八皇后问题代码分析

学号：2151094　　姓名：宋正非　　所属学院：电子与信息工程学院　　年级：22

本报告要求阅读原始代码并给出注释。具体写作方法是将原始代码 copy 到文档中， 在每一 行代码的前面增加一行中文注释，根据自己的理解说明这行代码的功能。此外， 对变量定义 应说明每个变量在程序中的功能。

1 程序下载网址:

https://blog.csdn.net/m0_58820574/article/details/127796847?ops_request_misc=%257B%2522request%255Fid%2522%253A%25221712227778168001827285 74%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=171222377816800182728574&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~top_positive~default-2-127796847-null-null.142^v100^pc_search_result_base7&utm_term=%E5%85%AB%E7%9A%87%E5%90%8E%E9%97%AE%E9%A2%98python&spm=1018.2226.3001.4187

自行开发部分注明：

程序参考如上CSDN博客，其主要文件中包含八皇后问题的寻找函数与打印输出函数。

我根据个人编程习惯，基于博客程序进行了修改与优化。将项目分为两个运行程序，功能划分更加清晰明确，main.py文件用于解决八皇后核心问题，添加draw.py文件用于将结果可视化。main.py中的函数改为PLACE_QUEEN函数用于放置皇后，得到所有可行解的集合二维数组Q；添加的draw.py文件，实现了基于所有可行解（8皇后问题解为92个）的结果可视化，将棋盘与皇后放置情况绘制输出，其中每张图含有15种放置情况（3行5列），共得到7张结果图。

2 代码及注释:

#main.py

```
#从 draw 文件引用 draw 函数，用于结果可视化
from draw import draw
#定义全局变量 Q 并将其初始化为空矩阵
global Q
Q = []
#定义 PLACE_QUEEN 放置函数，函数输入的参数为列 row 与行列数 n
def PLACE_QUEEN(row, n):
   #定义全局变量 index，用于计数
    global index
    #进入 for 循环
    for col in range(n):
        #if 判断条件（行标记矩阵中的该行元素为 0 并且该上对角线与下对角线对应的元素值为 0→说明尚未标记可以放置）
        if flag_col[col] == 0 and line_up[row + col] == 0 and line_down[row - col + n - 1] == 0:
            #放置皇后，第 row 行 col 列被占领
            place_q[row] = col
             #占领后该点所在列的标记为 1
            flag_col[col] = 1
```

```
            #该点所在的上对角线被标记为 1
            line_up[row + col] = 1
            #该点所在的下对角线标记为 1
            line_down[row - col + n - 1] = 1
            #if 判断条件：该行不是棋盘的最后一行
            if row < n - 1:
            #继续递归调用，寻找下一行的占领点
                PLACE_QUEEN(row + 1, n)
            #不满足 if 条件（该行已是最后一行）
            else:
            #已经找到了一组可行的八皇后摆放方法，计数加一
                index += 1
                #在 Q 矩阵中添加摆放结果
                Q.append(place_q.copy()) #copy 不能省略，Q 元素全都会变成新的
place_q
            # 标记清空
            flag_col[col] = 0
            line_up[row + col] = 0
            line_down[row - col + n - 1] = 0
    #函数返回矩阵 Q
    return Q
#主函数
if __name__ == '__main__':
    #行列为 8
    n = 8
    #可行解计数的初始值置 0
    index = 0
    #初始化记录第几行的皇后放在第几列上的矩阵 place_q
    place_q = [0 for i in range(n)]
    #初始化表示第 i 列是否被标记的矩阵 flag_col
    flag_col = [0 for i in range(n)]
    #初始化表示上对角线是否被标记的矩阵 line_up（nxn 的棋盘共有 2n-1 条上对角线）
    line_up = [0 for i in range(2 * n - 1)]
    #初始化表示下对角线是否被标记的矩阵 line_up（nxn 的棋盘共有 2n-1 条下对角线）
    line_down = [0 for i in range(2 * n - 1)]
    #存放每种摆放情况的二维数组，调用函数 PLACE_Q
    Q = PLACE_QUEEN(0, n)
    #计算 15 个棋盘排列方式/图，需要输出几次图片结果，用于可视化
    m = int(index/15)+1
    #for 循环，循环 m 次（即产生 m 张图）
    for k in range(m):
        #用于调用 draw 函数，将结果转换为棋盘图像
        draw(Q, k, index)
    #同时也输出每个摆放方式的矩阵表示的结果以及结果总数
    print('Result of Queens:')
    for i in range(index):
        print(Q[i])
    print("总共有{}种结果".format(index))
```

#draw.py

```
#调用 numpy，并将其引用为 np
import numpy as np
#从 matplotlib 中调用 pyplot，并将其引用为 plt
import matplotlib.pyplot as plt
#全局变量 m
global m
```

```
#定义函数 plot_chessboard,用于画棋盘
def plot_chessboard(ax, queens, m):
    #矩阵初始化置零
    board = np.zeros((8, 8))
    #for 循环遍历 8x8 矩阵每个元素
    for i in range(8):
        for j in range(8):
            #画出白灰相间的棋盘效果
            if (i + j) % 2 == 1:
                ax.fill([i, i + 1, i + 1, i], [j, j, j + 1, j + 1], 'lightgrey')
    #for 循环
    for n in range(8):
        #在放置的位置打印皇后
        ax.text(n + 0.5, queens[m][n] + 0.5, '♕', ha='center', va='center',
fontsize=15, color='black')
    #打印每个棋盘的序号
    ax.text(-0.2, 8, f'{m+1}', ha='center', va='center', fontsize=12,
color='black')
#定义函数 draw
def draw(Q, k, index):
    #创建 3 行 5 列子图的网格，并将返回的图像对象存储在 fig 中，将子图存储在变量
axes 中
    fig, axes = plt.subplots(3, 5, figsize=(20, 12))
    #设置数值用于输出各棋盘放置的序号
    m = 15 * k
    #for 循环
    for i in range(3):
        for j in range(5):
            #获取子图对象并赋值给 ax
            ax = axes[i, j]
            if m < index:
                #调整输出效果，不显示刻度值
                ax.set_xticks([])
                ax.set_yticks([])
                #调用 plot_chessboard 函数，用于在 ax 上绘制棋盘图案
                plot_chessboard(ax, Q, m)
                #将子图的纵横比设置为相等，即保持正方形的形状
                ax.set_aspect('equal', adjustable='box')
                #显示坐标轴，即边框
                ax.axis('on')
            else:
                #若棋盘排列已经都画完了，则停止输出，隐藏后续子图的刻度和坐标轴
                ax.set_xticks([])
                ax.set_yticks([])
                ax.axis('off')
                continue
            #用于跟踪绘制的子图数量
            m = m + 1
    #调整子图的布局，确保它们适当填充整个图像区域
    plt.tight_layout()
    #显示绘制的图像
    plt.show()
```

3 程序输出结果拷屏:

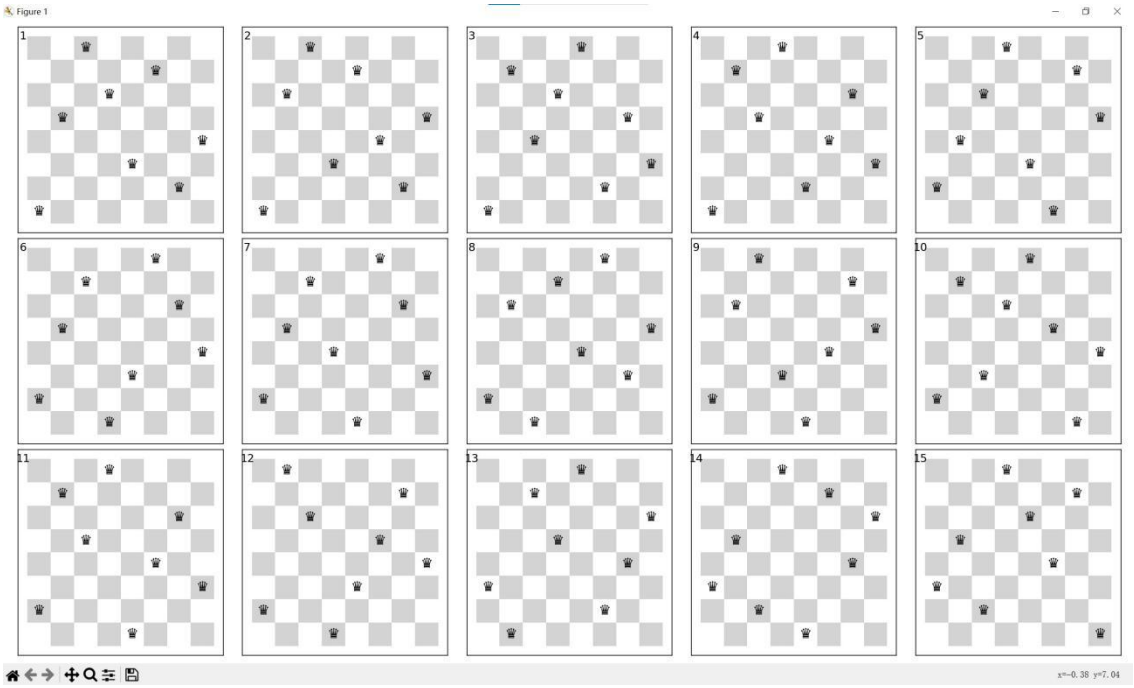

图1

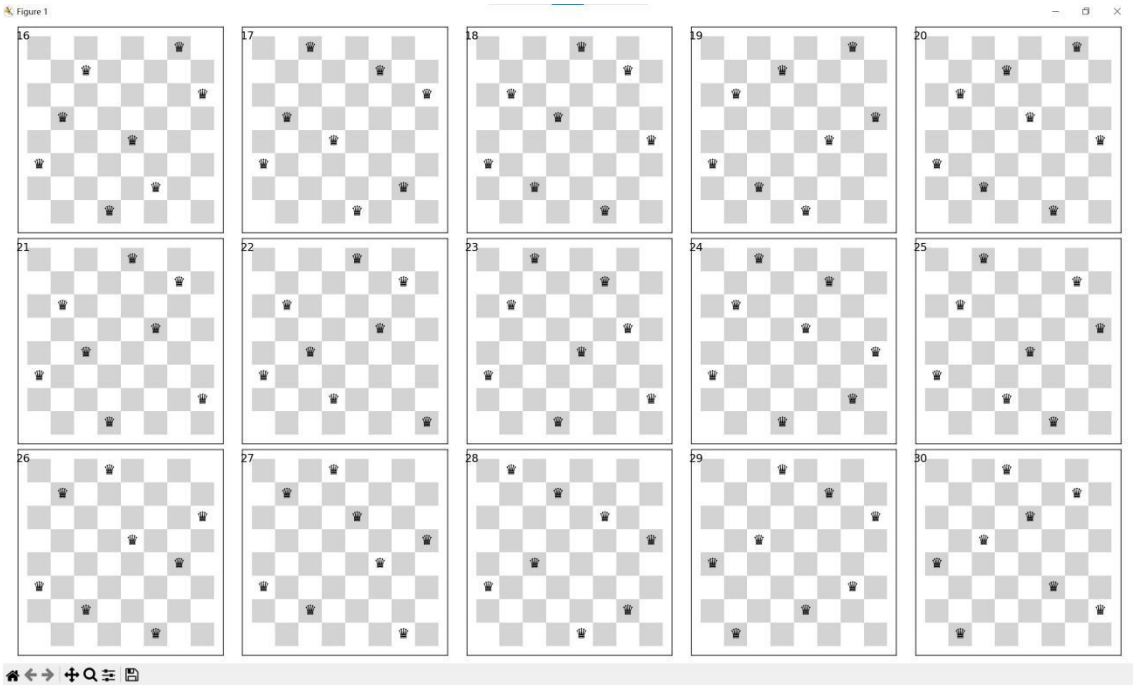

图2

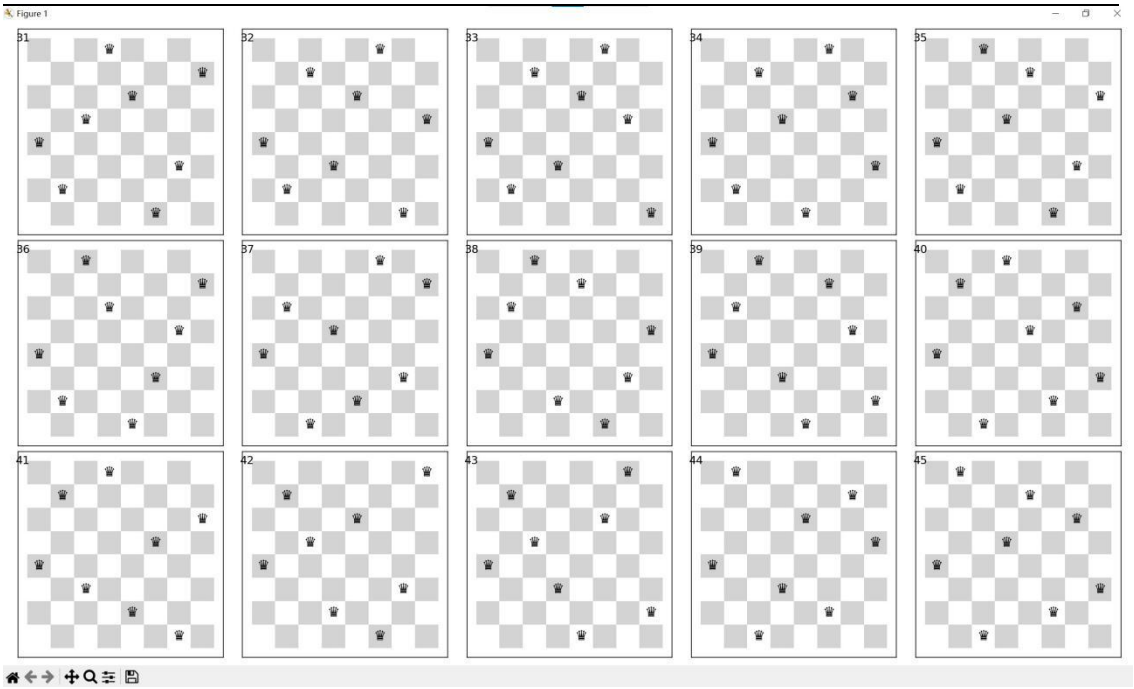
Figure 1
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45

图3

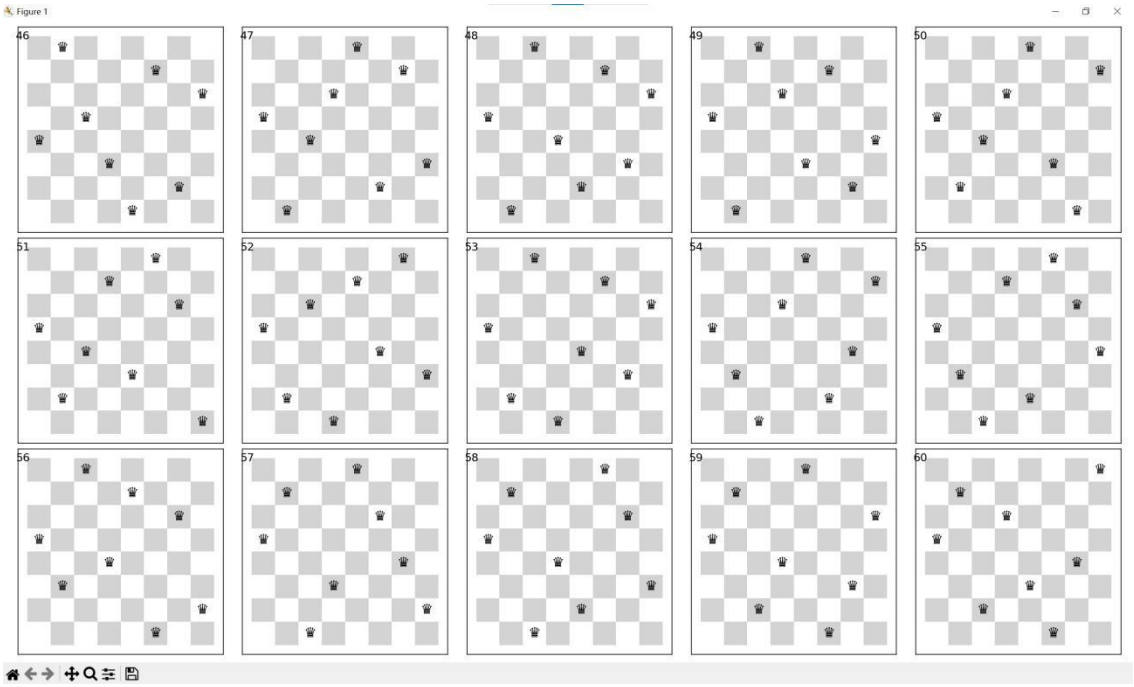
Figure 1
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60

图4

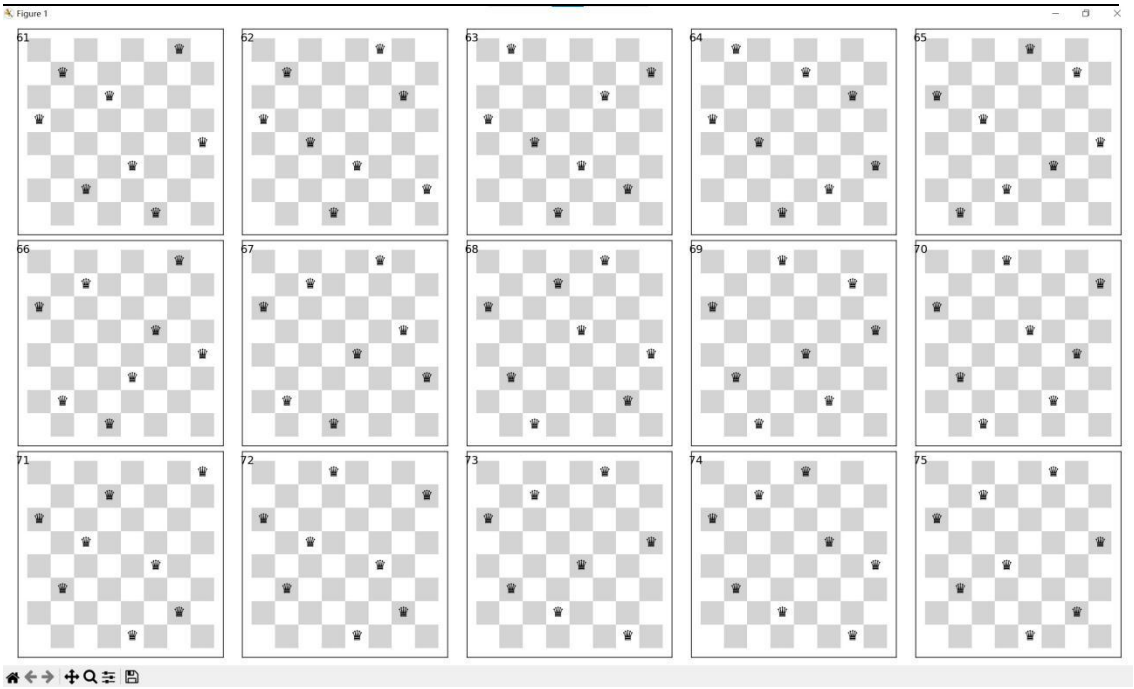
Figure 1
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75

图5

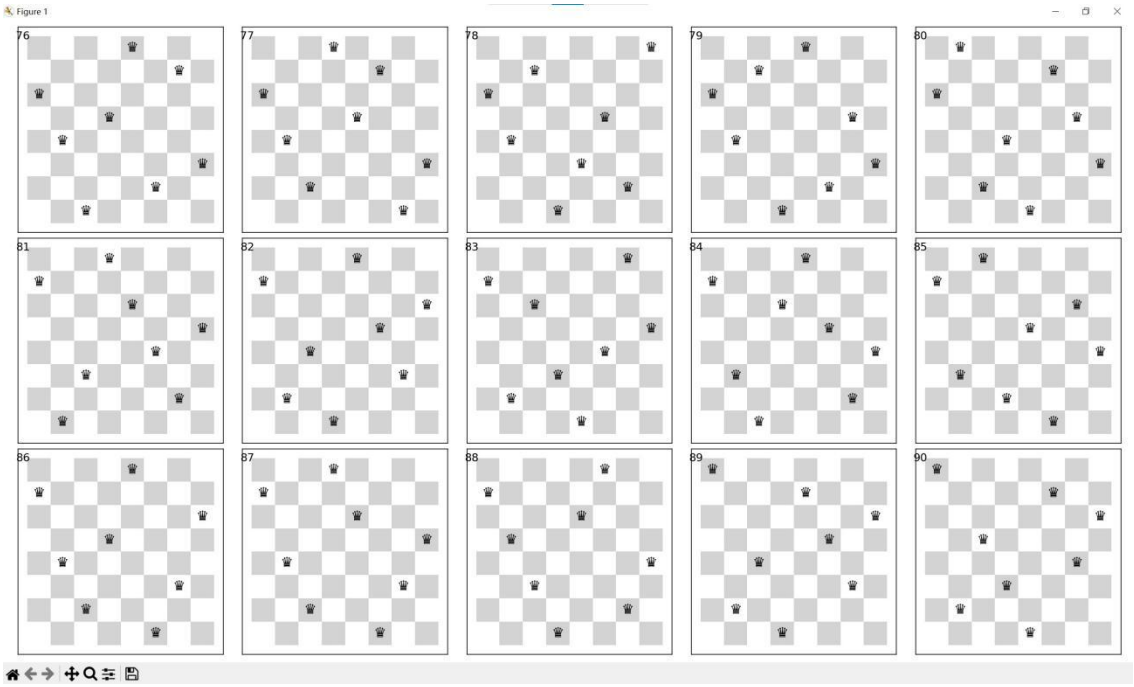
Figure 1
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90

图6

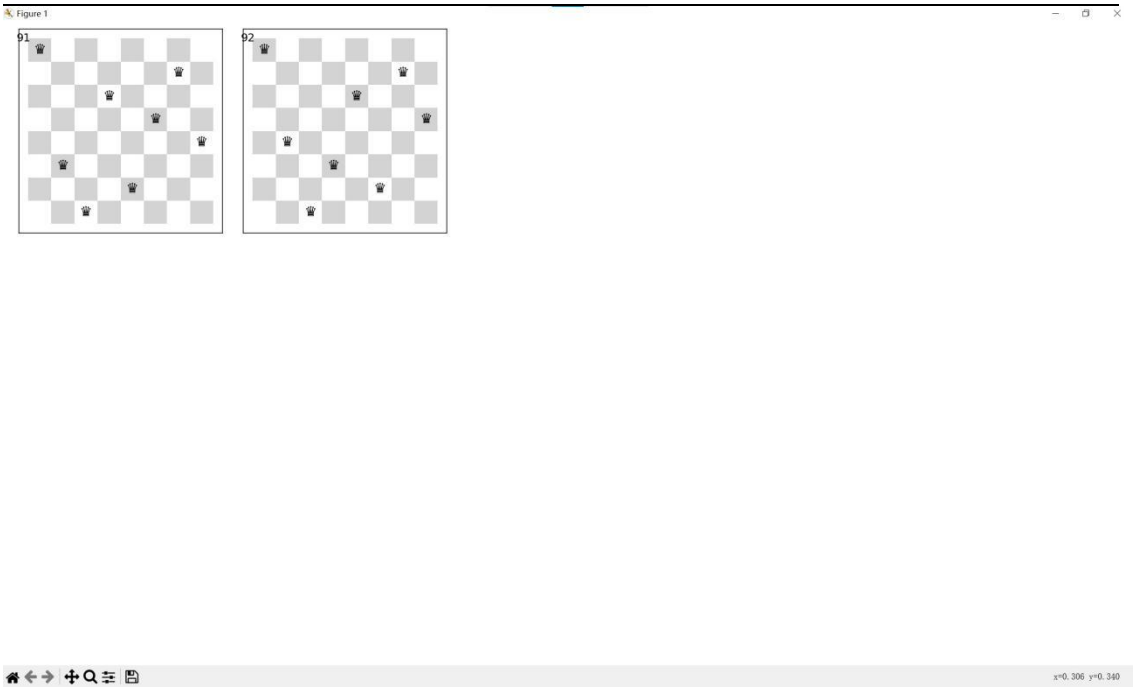

图7

对应如下输出结果：

Result of Queens:
[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]
[3, 6, 4, 1, 5, 0, 2, 7]
[3, 6, 4, 2, 0, 5, 7, 1]
[3, 7, 0, 2, 5, 1, 6, 4]
[3, 7, 0, 4, 6, 1, 5, 2]
[3, 7, 4, 2, 0, 6, 1, 5]
[4, 0, 3, 5, 7, 1, 6, 2]
[4, 0, 7, 3, 1, 6, 2, 5]
[4, 0, 7, 5, 2, 6, 1, 3]
[4, 1, 3, 5, 7, 2, 0, 6]
[4, 1, 3, 6, 2, 7, 5, 0]
[4, 1, 5, 0, 6, 3, 7, 2]
[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]
总共有92种结果