N皇后问题

整体程序模块化设计为

- IO部分（包含输入输出提示）：通过两个独立函数实现用户输入和输出的交互，同时对输入结果进行异常处理
- 主体算法逻辑（回溯法，包含剪枝优化）
- 启发式搜索策略模块：将搜索策略抽象为类，作为解题类的参数，实现搜索策略与解题逻辑的解耦
- 多命令设计（解题，测试与绘图）：使用fire包来实现命令行接口，便于不同功能的调用

额外优化

- 使用uv依赖头，便于直接使用uv来执行脚本
- 使用ruff对代码进行格式化，规范性更好

优化思路

1. 回溯时存储当前列和对角线的占用状态，同时方便判断是否剪枝和回退
2. 对于启发式搜索策略，这里实现了基础搜索策略和最小冲突策略
3. 基础搜索策略即获得没有列冲突的行，然后尝试所有列
4. 最小冲突策略即获得没有列冲突的行，然后尝试所有列，并记录冲突数，选择冲突数最小的行

实验结果

```
uv run -p 3.11 nqueen/nqueen.py solve
```

对于N=4和N=8，结果如下

```
% uv run -p 3.11 nqueen/nqueen.py solve
请输入N的值: a
无效的输入，请输入一个正整数
请输入N的值: 4
求解耗时: 0.0003秒
找到 2 个解
请输入要查看的解[1-2]（输入0退出，输入3显示全部）: 6
请输入要查看的解[1-2]（输入0退出，输入3显示全部）: 3


解法1:

=================
|   | Q |   |   |
=================
|   |   |   | Q |
=================
| Q |   |   |   |
=================
|   |   | Q |   |
=================


解法2:

=================
|   |   | Q |   |
=================
| Q |   |   |   |
=================
|   |   |   | Q |
=================
|   | Q |   |   |
=================
请输入要查看的解[1-2]（输入0退出，输入3显示全部）: 0
```

Figure 1: N=4

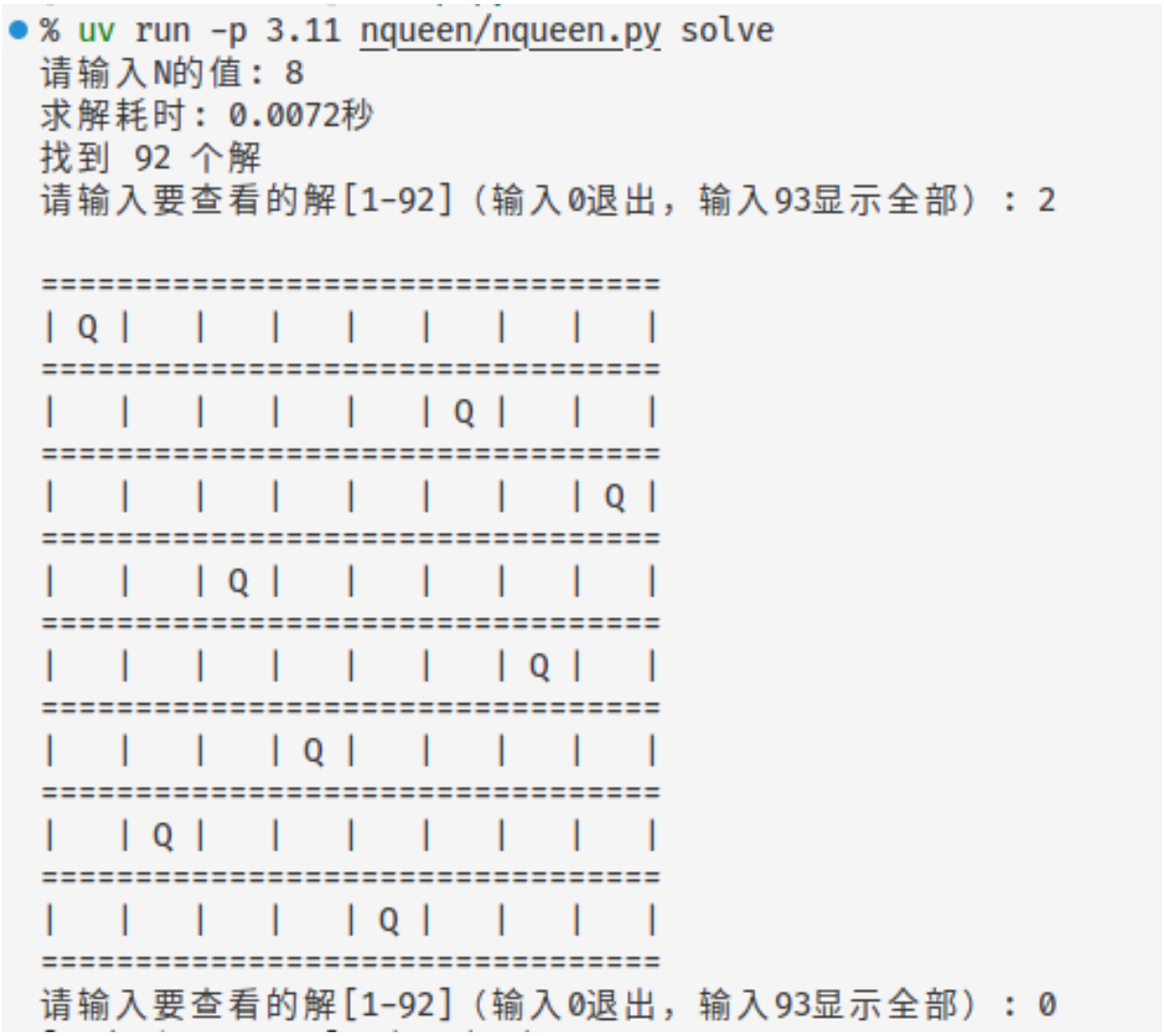


Figure 2: N=8

程序测试结果

N皇后问题的答案从wiki文档中获得，通过对比解的数量来验证结果的正确性

执行下面命令，结果如图

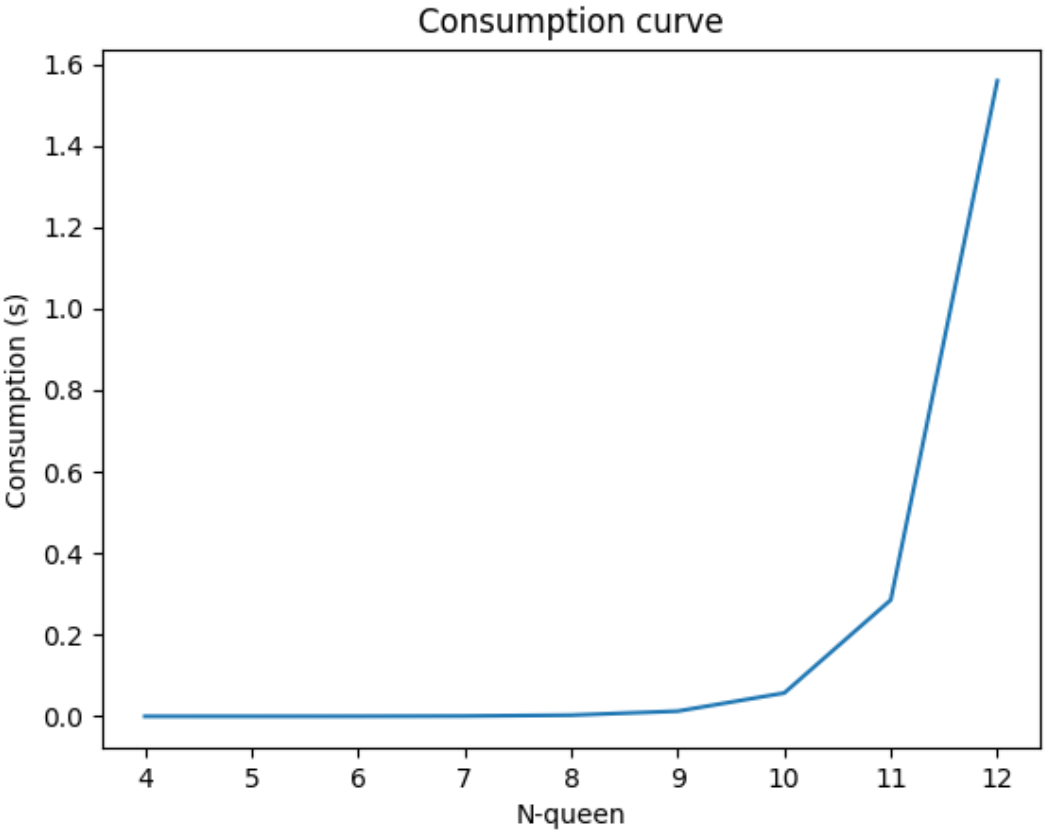
```
uv run -p 3.11 nqueen/nqueen.py test
```

```
% uv run -p 3.11 nqueen/nqueen.py test
[x] N=1 pass
[x] N=2 pass
[x] N=3 pass
[x] N=4 pass
[x] N=5 pass
[x] N=6 pass
[x] N=7 pass
[x] N=8 pass
[x] N=9 pass
[x] N=10 pass
[x] N=11 pass
[x] N=12 pass
测试完成，错误数：0
```

时间增长曲线

```
uv run -p 3.11 nqueen.py curve
```

可以绘制并查看N=4到N=12的运行时间曲线


Consumption curve
Consumption (s)
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
4
5
6
7
8
9
10
11
12
N-queen