算法说明：

主要函数：

1. solution_函数：将数字列表形式的解转换为可视化字符串格式

   输入：solution 参数表示皇后在各行的列位置(0-based)，n 表示棋盘大小

   输出：返回一个字符串，用'.'表示空位，'Q'表示皇后位置

2. save_solutions 函数：将所有解保存到指定文件中

   输入：solutions 参数是所有解的列表，n 是棋盘大小，filename 是输出文件名

   输出：无返回值，但会生成包含所有解的文件，每个解都有编号和可视化格式

3. print_compact 函数：以紧凑格式打印所有解

   输入：solutions 参数是所有解的列表

   输出：无返回值，直接在控制台打印每个解(1-based 的列位置)

这个模块通常配合 N 皇后问题的主算法模块使用，用于处理和展示计算结果。

N=4 的解：

```
D:\Python\python.exe C:\Users\ASUS\Desktop\高梦蝶2023141490367人工智能导论\N皇后\nqueen.py
请输入皇后数量N(≥4): 4
输出所有解? (y/n): y

找到2个解，耗时2.038秒
解法1:
.Q..
...Q
Q...
..Q.

解法2:
..Q.
Q...
...Q
.Q..


进程已结束，退出代码为 0
```

N=8 的解：

```
D:\Python\python.exe C:\Users\ASUS\Desktop\高梦蝶2023141490367人工智能导论\N皇后\nqueen.py
请输入皇后数量N(≥4): 8
输出所有解？(y/n): y

找到92个解，耗时2.273秒
解法1:
Q.......
....Q...
.......Q
.....Q..
..Q.....
......Q.
.Q......
...Q....

解法2:
Q.......
.....Q..
.......Q
..Q.....
......Q.
...Q....
.Q......
    n
```

时间增长曲线：