# 准备工作

在开始本次作业前，请首先阅读压缩包里的`README.pdf`。此小节主要针对`README`的一些不明确之处和容易被遗漏之处做出一些补充说明。

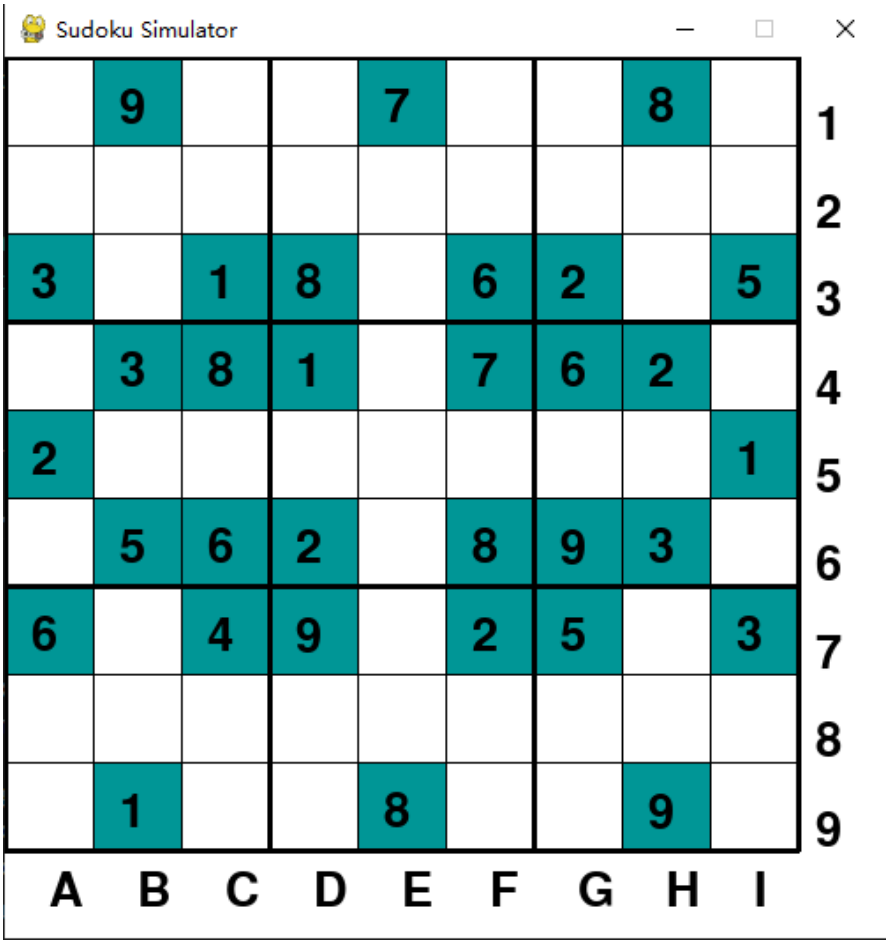
关于环境配置

- 本次作业所需的环境配置仍然很简单，`anaconda`就本次作业而言不是必须的
  - 也就是说：README中的`pip install conda`不需要执行
  - 如果你想要使用`anaconda`，可以参考其官方文档的指引，对其了解更加充分后进行安装：Anaconda Document
- 本次作业需要额外安装的库为`pygame`和`sortedcontainers`
  - 使用`pip install`命令进行安装即可

如README所述，可以使用以下命令运行本次作业的Demo：

```
python main.py --partial_sol part_sol_1 --filtering ac1
```

当上述指令正常执行时，你应该能看到这样一个图形化界面：




并且命令行输出以下内容（pygame、python版本可能会有偏差）：

```
> python main.py --partial_sol part_sol_1 --filtering ac1
pygame 2.6.1 (SDL 2.28.4, Python 3.9.13)
Hello from the pygame community. https://www.pygame.org/contribute.html
```

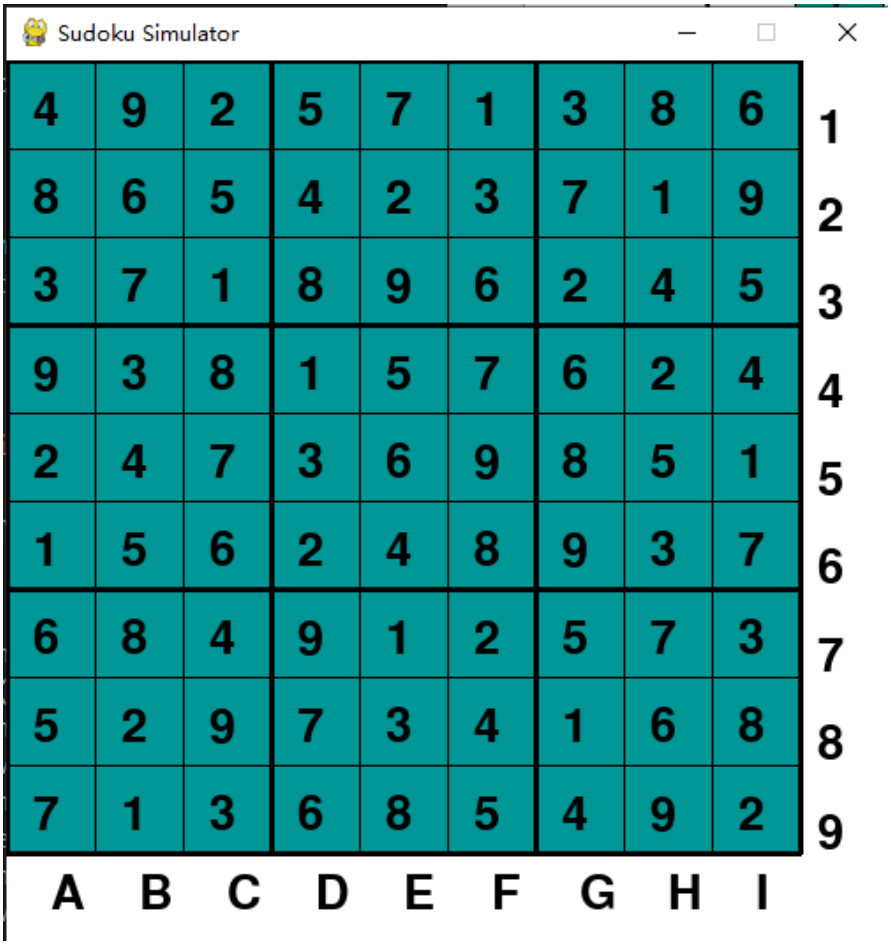
如README所述，在此图形化窗口下按`s`，即可开始执行搜索

- **注意此时使用的输入法需要为英文**

- 如果正确开始，那么你的命令行界面应该会继续输出以下内容：

```
Searching...
Variable Ordering with: Next Unassigned
Value Ordering with: Random
Filtering with: AC1
```

- 等待一段时间（视主机性能，通常在半分钟左右）后，你应该可以看到搜索算法的结果：




此时，本次作业的准备工作就全部完成了。

# 实现指南

实现此次作业的背景知识要求为：

- 理解CSP问题的基本概念
- 掌握AC3算法的实现
- 掌握CSP的Ordering方法：MRV和LCV
  如果对于上述内容仍有疑问，可以参看课程：CS188 的第4、5节课作为复习巩固。

就此次作业而言，你需要**阅读并修改**的部分仅`arc_consistency.py`和`heuristics.py`两个文件，**无需深入回溯搜索算法、CSP类和数独问题的细节实现。**

在此给出一些CSP类提供的接口说明，以帮助更好地理解作业框架：

- `csp.curr_domains`：以字典的形式返回当前各个节点可行的取值域
  - 注：在未初始化的情况下可能是`None`，此时可以用`csp.domains`代替它
  - `csp.choices(node)`提供了一种便捷的检测空情况并替代的方案
- `csp.neighbors`：以字典形式给出当前节点的相邻节点
- `csp.constraint(n1, v1, n2, v2)`：检查节点对`(n1,n2)`的赋值`(v1,v2)`是否符合约束

- `csp.prune`：在当前可行取值域中移除某个值
  - 注：在实现`ac3`时，你只需要在理解的基础上使用`revise`而不需要使用此函数

一些别的杂项说明

- `queue`：AC算法中的弧队列
  - 你或许会发现：初始输入的`queue`并**不包含所有的弧**，这与AC的规约并不完全一致
  - 这在本次作业的Bonus部分可能是一个需要考察的点
- `removals`：CSP接口设计中一个比较失败的耦合累赘，无需关心其含义，按照`ac1`和`revise`的使用方式进行使用即可
- `arc_heuristic`参数：无需关心

在正确实现AC3之后，你可能会发现它的执行（远远）没有AC1快

- 造成此问题的原因部分地在于AC和搜索过程之间的复杂度平衡——由于AC1的实现中并不校验所有的弧，它比看上去要快速很多，实际上更接近不使用AC算法的情况——在本次作业的bonus部分，你可以更加详细地分析这一现象
- 通过继续实现两个启发式排序方法，可以极大地提高AC3的执行效率
- 本次作业的评分中不会考察执行时间，仅仅考察正确性

由于AC4算法并未在课上讲授，我们对它不做严格要求，而是放在了Bonus部分。

# 实现要求和评分规则

本小节对此次作业的要求进行一些**调整和细化**，并按照我们认为合理的顺序排列每一个目标，你可以按照顺序逐个实现它们。

## 基础部分（100）

理解作业框架中的AC1和Revise函数（15分）

- 对作业框架中已有的AC1和Revise函数的实现进行分析说明
- 从理论上分析AC1的实现为什么是低效的，与AC3有什么区别
  - 在此处的分析中，你可以认为`queue`包含了所有的弧

实现AC3方法（45分）

- 正确地实现AC3方法
- 在报告中你需要对你的代码的关键部分进行解释说明，并列出实验结果

实现启发式函数（30分）

- 正确地实现启发式函数MRV（15分）和LCV（15分）
- 在报告中，你需要对你的实现进行解释说明，并给出对比实验的结果

正确、按时地提交你的作业（10分）

- 按照README的要求，提交`arc_consistency.py`，`heuristics.py`，`external_lib.py`（可选）和实验报告，将它们打包成一个压缩包上传canvas（3分）
- 在截止时间之前提交你的作业（7分）

## Bonus部分（20）

实现AC4方法（10分）

- 可以参考A random blog的讲解，或是参考别的教程
- 在报告中，对你的代码实现的关键部分进行解释说明，并列出实验结果

分析搜索算法和AC的复杂度平衡问题（10分）

- 开放性问题，你可以在作业框架上设计各种小实验来验证不同AC算法细节设计和AC算法的使用策略对于整个CSP问题求解效率的影响。你可以加以简单的理论分析来验证你的想法

**不要太卷**

Bonus部分的分数，在期末的最终结算时会在作业分数的上限处**截断**。而作业的基础分数只要你认真实现，基本上都会给满的。
这一部分设计单纯提供大作业的更多探索空间，请不要在这里写大论文。