实验1.1

一、背景

注意看，你从黑暗中醒来，发现自己深处一个狭小的地牢。你的身边只有一台破烂的笔记本电脑和一扇门，门被一个二进制迷锁所锁定。你需要解开它才能逃出地牢。。。

二、问题描述

这个二进制迷锁具有一个大小为 $N \times N$ 的拨轮锁盘, 每一个格点上具有一个可转动的拨轮，上面刻着数字 0 （表示非锁定）和 1（锁定）。

由于拨轮之间相互链接的关系，拨轮切换锁定的规则如下：

- 只能同时转动相邻呈“L”字形（四个方向的朝向均可）的三个拨轮，将它们同时由各自的锁定切换为非锁定状态，或从非锁定切换为锁定状态。

以一个 3×3 的锁盘为例，即拨动中心为 `(1,1)` 四种的情况为：

1. 第一种，同时转动 `(1,1)`, `(1,2)`, `(0,1)`:

```
0 1 0       0 0 0
0 1 1 -->>  0 0 0 (成功解锁)
0 0 0       0 0 0
```

1. 第二种，同时转动 `(1,1)`, `(0,1)`, `(1,0)`:

```
0 1 0       0 0 0
0 1 1 -->>  1 0 1
0 0 0       0 0 0
```

3. 第三种，同时转动 `(1,1)`, `(1,0)`, `(2,1)`:

```
0 1 0       0 1 0
0 1 1 -->>  1 0 1
0 0 0       0 1 0
```

4. 第四种，同时转动 `(1,1)`, `(2,1)`, `(1,2)`:

```
0 1 0       0 1 0
0 1 1 -->>  0 0 0
0 0 0       0 1 0
```

锁盘目前是打乱的状态，你的目标是

- 为这个问题设计一个合适的启发式函数，并证明它是 admissible 的，并论证其是否满足 consistent 性质。
- 根据上述启发式函数，开发对应的 A* 算法找到一个解法，将它恢复为全 0 状态以解开这个迷锁。
- 设置启发式函数为 0，此时 A* 退化为 Dijkstra 算法，比较并分析使用 A* 方法带来的优化效果。

输入

详见 `./astar/input/` 共有 10 个文件，形式如下：

```
Line 1: 第一行包含一个正整数：N，表示锁盘的大小为 NxN
Line 2 to N+1: 接下来的 N 行，每行包含 N 个整数，表示锁盘拨轮状态。拨轮状态取 0 或 1 表示非锁定或锁定。
```

举例：

```
3
1 1 1
1 1 0
0 0 0
```

输出

- 如果不能解锁，请输出 “No valid solution.”
- 否则，输出解锁步骤如下：

```
Line 1: 第一行包含一个正整数：T，表示解锁所需的步骤数
Line 2 to T+1: 每行包含三个整数 i、j、s，用 ',' 隔开，表示该步切换的中间拨轮位置为第 i 行，第 j 列，朝向为第 s 种。
```

其中，四种朝向和数字的映射关系为：

```
s == 1: (i,j), ( i ,j+1), (i-1, j );
s == 2: (i,j), (i-1, j ), ( i ,j-1);
s == 3: (i,j), ( i ,j-1), (i+1, j );
s == 4: (i,j), (i+1, j ), ( i ,j+1).
```

举例，上述输入的解锁步骤为：

```
3
0,1,3
0,1,4
1,1,2
```

分析：

```
1 1 1      0 0 1      0 1 0      0 0 0
1 1 0 -->> 1 0 0 -->> 1 1 0 -->> 0 0 0 (解锁成功)
0 0 0      0 0 0      0 0 0      0 0 0
```

三、作业要求

1、编程语言限制为 C/C++, 写清核心代码注释。

2、报告中需要描述：

1. 描述你的启发式函数，证明它是 admissible 的，并论证其是否满足 consistent 性质。
2. 算法的主要思路；
3. 与 Dijkstra 算法进行比较，并分析使用 A* 方法带来的优化效果。

3、严禁抄袭。

实验1.2

一、背景

学校新招募了一批宿管阿姨，不巧的是负责排班的管理人员生病请假了。
你的任务是开发一个 CSP 算法，为学校的这批宿管阿姨安排一个值班表，以满足给定的约束条件，并尽可能满足阿姨们的轮班请求。

二、问题描述

你将获得以下信息：

- 宿管阿姨数量（staff_num, N）
- 值班天数（days_num, D）
- 每日轮班次数（shifts_num, S）
- 轮班请求 $\text{Requests} \subset \{0,1\}^{N \times D \times S}$

课程表必须满足以下约束条件：

1. 每天分为轮班次数个值班班次；
2. 每个班次都分给一个宿管阿姨，同一个宿管阿姨不能工作连续两个班次；
3. 公平起见，每个宿管阿姨在整个排班周期中，应至少被分配到 $\left\lfloor \frac{D \cdot S}{N} \right\rfloor$ 次值班。

你的目标是：

- 构造一个排班表 $\text{Shifts} \subset \{0,1\}^{N \times D \times S}$
- 在满足上述约束的条件下，尽可能最大化满足的请求数，即 $\max_{\text{Shifts}} \sum_{n \in N} \sum_{d \in D} \sum_{s \in S} \text{Requests}_{n,d,s} \times \text{Shifts}_{n,d,s}$
- 请尽量最大化即可，但最终得分将考虑满足的请求数量

输入

详见 `./csp/input/` 共有 10 个文件。形式如下：

```
Line 1: 第一行包含三个正整数：N、D、S。
Line 2 to N*D+1: 接下来的 N*D 行，每行包含 S 个整数，表示轮班请求（Requests）。轮班请求的每个值取 0 或 1。若第 n 位宿管阿姨请求值第 d 天的第 s 轮班，则第 2+n*D+d 行的第 s 个数字为 1；否则为 0。
```

举例：

```
3,7,3
1,0,1
1,1,1
0,0,1
1,1,1
0,0,1
1,0,1
1,1,0
... ...
```

- 第一行表示总共有 3 个阿姨，需要值班 7 天，每天共有 3 班；

- 第二行表示第 1 个阿姨在第 1 天中，请求值第 1 或第 3 班；
- 第三行表示第 1 个阿姨在第 2 天中，请求值第 1、第 2 或第 3 班；以此类推。

输出

- 如果没有有效排班表，请输出 “No valid schedule found.”。
- 否则，输出排班表 Shifts 如下：

```
Line 1 to D: 每行包含 S 个整数，每个整数为宿管阿姨的编号 (1 ... N+1)，以 ',' 分割。
Line D+1: 第 D+1 行输出 1 个整数，为满足的轮班请求数量。
```

举例：

```
1,2,3
2,1,3
... ...
20
```

- 第一行第 1 天的三班分别由第 1，2，3号阿姨值班；
- 第二行第 2 天的三班分别由第 2，1，3号阿姨值班；以此类推。
- 最后一行表示该分配方式满足了 20 个请求。

你的算法应该利用最小剩余值（Minimum Remaining Values，MRV）启发式、前向检查（Forward Checking）或约束传播（Constraint Propagation）等优化技术，以快速解决该问题，并最大化满足的请求数。

三、作业要求

1. 编程语言限制为 C/C++, 写清核心注释。
2. 报告中需要描述：
    1. 描述实验中的变量集合、值域集合以及约束集合；
    2. 算法的主要思路；
    3. 使用的优化方法，并分析使用该优化方法带来的优化效果。
    4. 特别的，请在报告中给出 `input0.txt` 中你的安排方式。
3. 严禁抄袭。

实验提交

1. 提交方式：bb 系统。
2. 截止日期：6 月 1 日晚 11:59。
3. 提交目录树结构如下：

```
./PB12345678_XXX_exp1/
├── PB12345678_XXX_report.pdf
├── astar
│   ├── output
│   │   ├── output0.txt
    │   ├── output1.txt
    │   ├── ...
    │   └── output9.txt
│   └── src
│       └── astar.cpp
```

```
└── csp
    ├── output
    │   ├── output0.txt
    │   ├── output1.txt
    │   ├── ...
    │   └── output9.txt
    └── src
        └── csp.cpp
```

4. 最后将文件压缩成 zip 格式进行提交，注意，请大家务必按照目录树结构组织文件，否则可能导致检查脚本读取结果失败。s