刷新记录

V1.0		2019-3-8
V1.1	修正图27、28示例	2019-3-8
V1.2	修正8.11.4，8.11.5，8.11.6文字表述的车辆号	2019-3-8
V1.3	修正第5章节答案中中文”，”为英文”,” 刷新8.11章节中描述 增加系统调度详细说明 增加循环等待说明 增加程序编译与运行时间限制说明	2019-3-11
V1.4	修正部分不通顺的表述 增加路口多道路的调度顺序举例	2019-3-12
V1.5	增加判题器的伪码描述	2019-3-14

初赛 任务书 V1.5

背景信息

- 道路交通是城市的核心要素之一。
- 随着社会经济的发展，中国城市的车辆保有量已经越来越多，大都市慢慢变成了“堵”市。如何在出行时避免拥堵，是每一个人的目标。
- 日常生活中，很多拥堵是由于车辆行驶路线规划失误，大批车辆集中选择主干道行驶导致通行效率下降。
- 如果车辆都由调度中心统一规划调度路线，拥堵问题将得到大大缓解甚至彻底解决。
- 实际上这一技术已经在工业领域如矿山车辆、无人货仓等得到广泛应用。
- 但道路上的私家车辆尚无法进行统一规划，未来，自动驾驶和物联网技术的结合，使得彻底解决这一难题出现了曙光。
- 请同学们提前出任“首席城市交通规划官”，为未来城市规划好每一辆车的行驶路线。

1. 题目定义

- 在模拟的道路图上为每一辆车规划行驶路线，系统会自动根据规划路线运行。
- 在路线合法的前提下，最终所有车辆按照规划的路线到达目的地。

2. 系统假定

- **路口完全立交**：假定在每一个路口交汇的所有道路都可以完全互连，且车辆通过路口时不考虑在路口的通行时间。
- **无限神奇车库**：我们认为，系统中的每个地点都有一个无限容量的“神奇车库”。车辆在未到既定出发时间前，或者到达目的后，就停放在“神奇车库”中，完全不影响其他车辆通过。但车辆一旦出发，在行驶过程中则不允许进入车库空间。

3. 约束条件

- **不允许超车变道**：即车辆一旦进入某条车道，就必须在此车道内从道路起点驶向道路终点，中途不允许变道，即使前车速度缓慢，也不允许超车。
- **排队先到先行**：在一条道路前排队等待的所有车辆，按照到达时间先后进入道路。若多辆车在同一时间到达，按如下规则进入下一道路：

1. 同一道路牌车道号小（车道的编号）的车辆优先于车道号大的车辆

Figure 1：直行车道优先示例

2. 按现实交通规则，直行车辆有优先通行权，直行车辆优先于转弯车辆

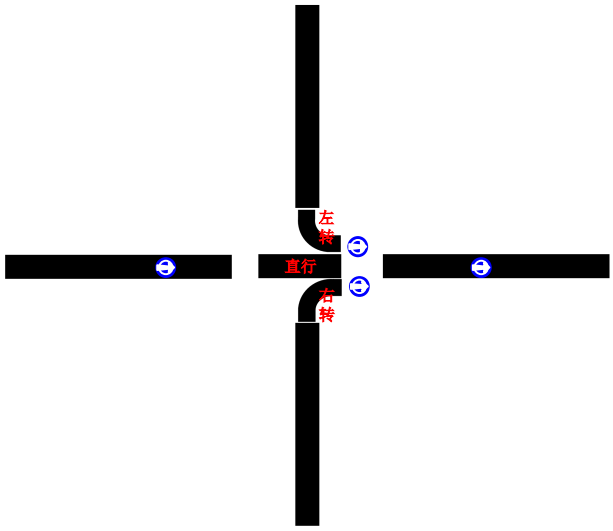

Figure 2：直行优先示例

3. 处于左转进入道路的车辆优先于右转进入道路的车辆

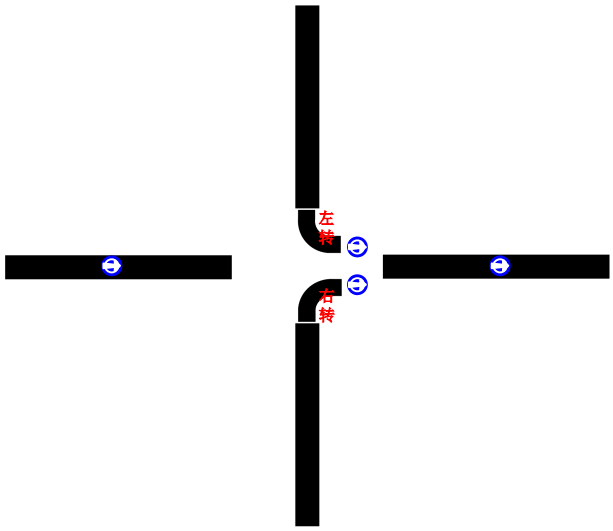

Figure 3：转弯优先示例

- **车道固定进入**：车辆在进入一段道路时按照车道编号从小到大的优先级选择可以进入的车道驶入，与前车的行驶速度无关。

  即就是：车辆优先按车道编号由小到大依次进入，除非车道号小的车道没有空位可进入。

Figure 4：进入车道规则示例

如下Figure 5：车道行驶规则举例所示，左侧道路车辆经一定时间行驶达到右侧道路车辆状态。

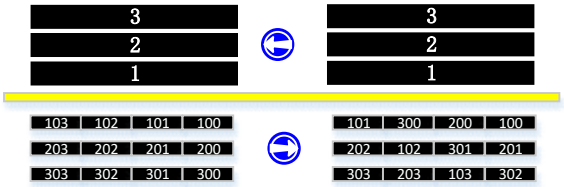

Figure 5：车道行驶规则举例

4. 题目输入

每一个测试用例都分为三部分：

道路

- 道路数据文件“road.txt”文件。

- 每一行数据为一条道路。
- 每条道路数据表示为：**(道路id，道路长度，最高限速，车道数目，起始点id，终点id，是否双向)**格式的向量。例如(502, 10, 6, 5, 2, 3, 1)的向量表示编号为502的道路，连接路口2和路口3的长度为10，限速6的双向5车道路段; (502, 10, 6, 5, 2, 3, 0)的向量表示编号为502的道路，连接路口2和路口3的长度为10，限速6的单向5车道路段。
- 起始点id：路口id（下文中有描述）
- 终止点id：路口id（下文中有描述）
- 是否双向：1：双向；0：单向
- 不管是双向道路还是单向道路，一条道路数据只会有一行数据表示，不会因为双向而多出来一行道路的数据表示。
- “#”开始的数据行为说明性文字，可以理解成为注释。如“#(id,length,speed,channel,form,to,isDuplex)”。
- 对于多车道的道路，相对于行驶方向，车道编号从左至右依次增大。

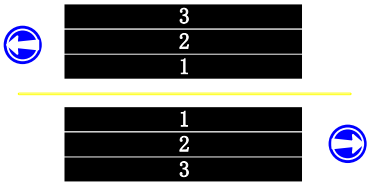

Figure 6：车道编号示例

车辆

- 车辆数据文件“car.txt”。
- 每一行数据为一车辆。
- 每辆车数据表示为：**(车辆id，始发地、目的地、最高速度、计划出发时间)**格式的向量。例如(1001,1,16,6,1)的向量表示一辆编号是1001最高速度为6的车辆要在时间点1从路口1到达路口16。
- 始发地：路口id（下文中有描述）
- 目的地：路口id（下文中有描述）
- “#”开始的数据行为说明性文字，可以理解成为注释。如“#(id,始发地,目的地,最高速度,出发时间)”。

路口

- 路口数据文件“cross.txt”。

- 每一行数据为一个路口。
- 每个路口数据表示为: (**路口id,道路id,道路id,道路id,道路id**)格式的向量。例如(6, 504, 514, 505, 518)表示504，514，505，518这四条路段交汇的编号为6的路口。
- 路口信息数据向量中的道路id，以路口为中心，其所连接的道路id按顺时针方向编排。比如向量(100,21,30,9,55)和(100,21,30,-1,55)分别表示如下：

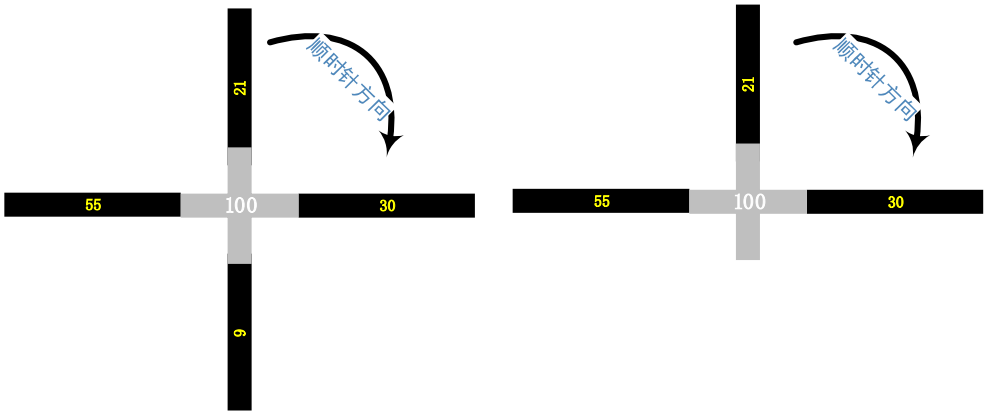

Figure 7：路口数据向量示例

- “#”开始的数据行为说明性文字，可以理解成为注释。如：“#(结点id,道路id,道路id,道路id,道路id)”。

数据样例：

- **Road.txt**

```
#(道路 id，道路长度，最高限速，车道数目，起始点 id，终点 id，是否双向)
(501, 10, 6, 5, 1, 2, 1)
(502, 10, 6, 5, 2, 3, 1)
(503, 10, 6, 5, 3, 4, 1)
(504, 10, 6, 5, 5, 6, 1)
(505, 10, 6, 5, 6, 7, 1)
(506, 10, 6, 5, 7, 8, 1)
(507, 10, 6, 5, 9, 10, 1)
(508, 10, 6, 5, 10, 11, 1)
(509, 10, 6, 5, 11, 12, 1)
(510, 10, 6, 5, 13, 14, 1)
(511, 10, 6, 5, 14, 15, 1)
(512, 10, 6, 5, 15, 16, 1)
(513, 10, 6, 5, 1, 5, 1)
(514, 10, 6, 5, 2, 6, 1)
```

```
(515, 10, 6, 5, 3, 7, 1)
(516, 10, 6, 5, 4, 8, 1)
(517, 10, 6, 5, 5, 9, 1)
(518, 10, 6, 5, 6, 10, 1)
(519, 10, 6, 5, 7, 11, 1)
(520, 10, 6, 5, 8, 12, 1)
(521, 10, 6, 5, 9, 13, 1)
(522, 10, 6, 5, 10, 14, 1)
(523, 10, 6, 5, 11, 15, 1)
(524, 10, 6, 5, 12, 16, 1)
```

- **Car.txt**

```
#(id,始发地,目的地,最高速度,出发时间)
(1001,1,16,6,1)
(1002,1,16,6,1)
(1003,1,16,6,1)
(1004,1,16,6,1)
(1005,1,16,6,1)
(1006,1,16,6,1)
(1007,1,16,6,1)
(1008,1,16,6,1)
```

- **Cross.txt**

```
#(结点 id,道路 id,道路 id,道路 id,道路 id)
(1, 501, 513, -1, -1)
(2, 501, -1, 502, 514)
(3, 502, -1, 503, 515)
(4, 503, -1, -1, 516)
(5, 513, 504, 517, -1)
(6, 504, 514, 505, 518)
(7, 505, 515, 506, 519)
(8, 506, 516, -1, 520)
(9, 517, 507, 521, -1)
(10, 507, 518, 508, 522)
(11, 508, 519, 509, 523)
(12, 509, 520, -1, 524)
(13, 521, 510, -1, -1)
(14, 510, 522, 511, -1)
```

```
(15, 511, 523, 512, -1)
(16, 512, 524, -1, -1)
```

- 图形化示例：

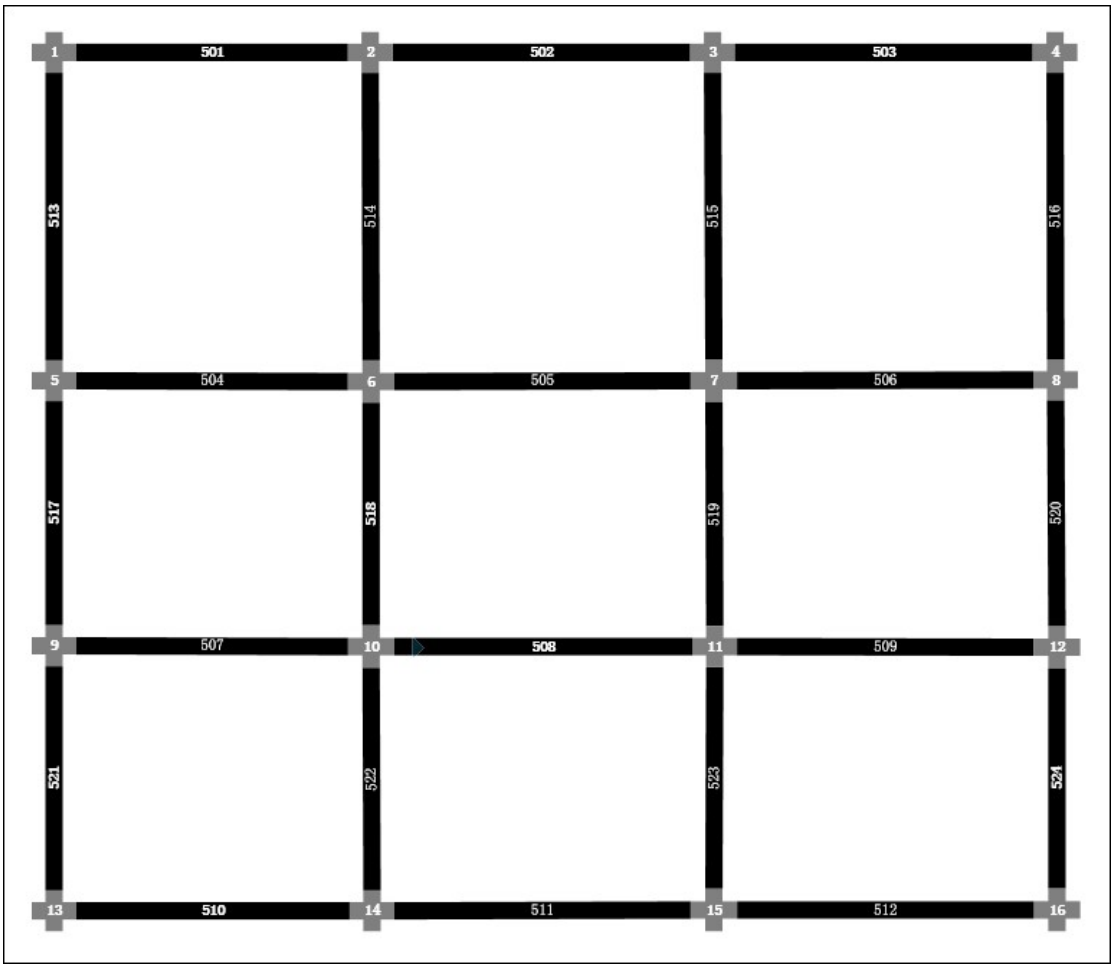

Figure 8：地图示例

5. 答案提交

- 选手应针对题目输入中的所输入的信息为基础，编写完整程序，输出“answer.txt”文件。
- “answer.txt“文件中每行数据表示：每一辆车的行驶路线规划，具体格式为**(车辆id，实际出发时间，行驶路线序列)**格式的向量。例如(1001, 1, 501, 502, 503, 516, 506, 505, 518, 508, 509, 524)即为上述1001号车辆自时间点1开始出发，从道路501、502、503…行驶至道路524的行驶路线。

- 每条答案数据以“(”起始，以“)”终止，路径中各道路以“,”分隔，中间只可以出现0个或多个空格“ ”，不允许出现其他非法字符。
- “#”开始的数据行为说明性文字，可以理解成为注释。系统阅卷会忽略“#”开始的该行数据。如：“#(carId,StartTime,RoadId...)”
- “answer.txt”示例：

```
#(carId,StartTime,RoadId...)
(1001, 1, 501, 502, 503, 516, 506, 505, 518, 508, 509, 524)
(1002, 1, 513, 504, 518, 508, 509, 524)
(1003, 2, 513, 517, 507, 508, 509, 524)
(1004, 1, 501, 502, 515, 519, 509, 524)
(1005, 1, 501, 514, 504, 517, 507, 508, 509, 524)
(1006, 3, 513, 517, 521, 510, 511, 512)
(1007, 2, 513, 504, 518, 507, 521, 510, 511, 512)
(1008, 1, 501, 502, 503, 516, 506, 519, 508, , 522,  511, 512)
```

6. 评分规则

- 系统调度时间短者胜出。系统调度时间指，从系统调度开始时间（非第一辆车实际开始运行的时间），至所有车辆全部到达目的地的时间，两者之差作为系统调度时间。
- 若系统调度时间相同，则所有车辆的行驶用时总时间最少者胜出。一辆车在系统计算中到达目的地的时间点减去其最初计划出发的时间点，称为这辆车的行驶用时。
- 若两队参赛选手的规划路线所有车辆的行驶用时总时间相同，则参赛选手程序计算出所有车辆行驶路线所运行时间最少者胜出。（该时间与前述时间不同，为程序运行时间，单位为ms）
- 若参赛选手程序计算运行时间也相同，则先提交代码的队伍胜出。

7. 概念定义

1. 地图：

- 地图由地点和道路组成，可以理解为数学上的有向连通图。

2. **路口：**

- 地点，是各条道路的交叉口和起始点。每个地点所连接最多道路数量不超过4。
- 地点在系统中有唯一的一个数字id编号。

3. **道路：**

- 一条道路连接两个不同的路口，车辆在道路上行驶。
- 每条道路也有一个全局唯一的数字id编号，道路还拥有其他属性，包括起始点id、终止点id、是否双向、车道数、长度、最高限速。每条道路数据以**(道路id，道路长度，最高限速，车道数目，起始点id，终点id，是否双向)**的向量表示。
- 道路-是否双向：

  道路分为单行道和双向道两种。单行道上只允许从起始点驶向终止点。双向道上既允许起始点驶向终止点，也允许终止点反向驶向起始点。
- 道路-车道数：

  道路上最多允许并排行驶的车辆数目。同时我们认为，对于双向车道而言，两个方向上的车道数是完全一致的，不存在不一样的情况。也即一个3车道的双向道路，是指的每个方向上都有三条车道。
- 道路-长度：

  此段道路的长度。在比赛中，所有道路的长度都是整数，不存在精度达到0.1以下的道路，同时系统中所有的道路长度均不小于6。
- 道路-最高限速：

  在此段道路上行驶的车辆最高允许的速度，系统中不会出现超速。
- 道路-起始点id：

  路段的一端地点id，也就是路口id

  对于单行道而言，只允许车辆从起始点向终止点行驶。
- 道路-终止点id：

  路段的一端地点id，也就是路口id

  对于单行道而言，只允许车辆从起始点向终止点行驶。

4. **车辆：**

- 车辆在道路上从始发点向目的地行驶。
- 车辆的信息包括唯一的车辆id编号，车辆还有其他属性。

- 每个车辆数据以**(车辆id，始发地、目的地、最高速度、计划出发时间)**的向量表示。
- 车辆-车身长度：

  车身长度固定为1 。
- 车辆-最高速度：

  车辆在行驶时的最高速度，取值为整数，不会出现小数。
- 车辆-始发点：

  车辆的出发地点id，也就是路口id。
- 车辆-目的地：

  车辆的目的地点id，也就是路口id。
- 车辆-计划出发时间：

  车辆的计划出发时间。

5. **行车路线：**
   - 行车路线是指一辆车的规划行驶路线，由(车辆id,实际出发时间,地点序列)表示。
   - 行车路线-实际出发时间：

     车辆的实际出发时间。此时间点不得早于车辆的计划出发时间
   - 行车路线-道路序列：

     车辆从始发点到目的地所顺序经过的每一条道路id序列

8. 交通规则补充说明

1. 道路禁止掉头行驶。
2. 每条车道均可以直行、左转、右转，不受车道编号的影响。
3. 每辆车均以可行进的最大车速前进，不可以主动降速行驶。
   - 如车辆最大速度5，道路限速6，则该车辆行驶速度为5，不可以主动降速为5以下的车速行驶。
   - 如车辆的最大速度为5，道路限速为6，因前方车辆的阻碍，该车辆最大可行驶速度为3，则该车辆行驶速度为3，不可以主动降速为3以下的车速行驶。
4. 车辆到达实际出发时间，需要上路行驶。如果存在同时多辆到达出发时间且初始道路相同，则按车辆编号由小到大的顺序上路行驶。

5. 优先运行已经在道路上运行的车辆，再运行等待上路的行驶的车辆。
6. 如果车辆行进过程中不会通过路口，也就是在当前时刻行进后，依然还停留在当前车道，则其只会影响当前车道排在其车辆后的所有车辆的行进。
7. 如果车辆行进过程中会通过路口，则按如下优先级进行行进通过路口，如图Figure 9：通过路口车辆行进顺序中红线顺序所示：
   1) 相同道路车辆，不同车道的车辆，车道序号小的优先通行，且必须通行后，下一车道号的车辆才能通行。即使前面车辆是转弯，后面车辆是直行，也必须等待前面的转弯车辆通行后，后面的直行车辆才可以通行。
   2) 相同道路排在前面的车辆有优先通行权利，不管车辆是否处于相同车道，前面的车辆通行后，排在后面的车辆才能通行。

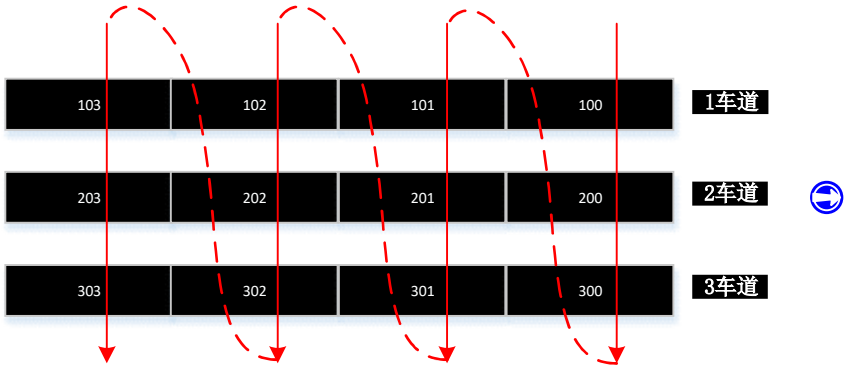

Figure 9：通过路口车辆行进顺序

8. 在通过路口进入下一条道路时，对于不同方向进入道路的车辆，以如下优先级顺序进入：
   1) 直行进入该道路的车辆优先于其他道路转弯进入该道路的车辆；
   2) 左转进入该道路的车辆优先于其他道路右转进入该道路的车辆；
   3) 必须等待其他路口直行或左转进入该道路的车辆行进完毕后，右转进入该道路的车辆才可以进入。
   4) 可以进入该道路的直行车辆、左转车辆、右转车辆的优先级只受直行、左转、右转优先级影响，不受车辆所在位置前后的影响。
   5) 相同道路上车辆出路口的优先级同章节8.7所示。
9. 通过路口时，所有车辆同时通过。虽然车辆速度不同，但是不考虑在同一时刻内速度的细微差异。在同时通过路口时也不允许车速高的超车，只能依照通过路口时车辆优先顺序来决定行进后的位置。

10. 车辆通过路口时，进入下一条道路时，可行驶的速度按如下规则计算：

说明：当前道路和等待进入的道路仅有当前车辆，均无其他车辆阻挡。如若存在其他车辆，按前述交通规则处理。

- 当前道路的最大限速记为R_1，即将进入的下一条道路的最大限速记为R_2。
- 车辆的最高速度记为V。
- 在当前道路最大行驶速度记为$V_1=\min(R_1, V)$，即将进入的下一条道路可行驶的最大速度记为$V_2=\min(R_2, V)$。
- 在当前道路单位时间最大行驶距离记为SV_1（数值与V_1相等），即将进入的下一条道路单位时间可行驶的最大距离记为SV_2（数值与V_2相等）。
- 在当前道路可行驶的距离记为S_1，在下一条道路可行驶的距离记为S_2。

1) 在当前道路的行驶的距离S_1不得超过“在当前道路的最大行驶速度V_1”在单位时间内行驶的距离SV_1（数值与V_1相等）。

2) 在即将进入的下一条道路行驶的距离S_2不得超过“即将进入的下一条道路的最大行驶速度V_2”在单位时间内行驶的距离SV_2（数值与V_2相等）。

3) 在下一条道路的行驶距离S_2不得超过下一条道路的单位时间最大行驶距离SV_2与在当前道路的行驶距离S_1之差，如果此差值小于0，则以0计算。如表格Table 1：车辆通过路口时行驶速度计算中 样例4，5所示。

4) 如果在当前道路的行驶距离S_1已经大于等于下一条道路的单位时间最大行驶距离SV_2，则此车辆不能通过路口，只能行进至当前道路的最前方位置，等待下一时刻通过路口。

5) 速度计算如Table 1：车辆通过路口时行驶速度计算所示：

样例序号	车速 V	当前道路限速 R_1	当前道路最大行驶速度 V_1	道路可行驶的距离 S_1	下一条道路限速 R_2	下一条道路最大可行驶速度 V_2	下一条道路行驶的距离 S_2
1	5	4	4	2	4	4	2
2	5	4	4	2	5	5	3
3	5	4	4	2	3	3	1
4	5	4	4	2	1	1	0
5	5	4	4	2	2	2	0
6	5	2	2	1	5	5	4
7	5	2	2	1	3	3	2

1、当前道路最大行驶距离S_1由车辆在当前道路所处位置决定； 2、下一条道路可行驶的距离S_2由上述规则决定；

Table 1：车辆通过路口时行驶速度计算

11. 如下给出几种情况下车辆行进的示例说明：

1. 假定车辆100、200、300、101、201、301的车速均为5，图中左侧道路与右侧道路限速均为6，图中左侧道路与右侧道路长度均为10，车辆均为直行

   T时刻道路车辆状态如下：

Figure 10：T时刻道路车辆状态

T+1时刻道路车辆状态如下：

Figure 11：T+1时刻道路车辆状态

T+2时刻道路车辆状态如下：

Figure 12：T+2时刻道路车辆状态

2. 假定车辆100、300、101、201、301的车速均为5，车辆200的车速为3，图中左侧道路与右侧道路限速均为6，图中左侧道路与右侧道路长度均为10，车辆均为直行

   T时刻道路车辆状态如下：

Figure 13：T时刻道路车辆状态

T+1时刻道路车辆状态如下：

Figure 14：T+1时刻道路车辆状态

T+2时刻道路车辆状态如下：

Figure 15：T+2时刻道路车辆状态

3. 假定车辆100、201、301的车速均为5，车辆200、300、101的车速为1，图中左侧道路与右侧道路限速均为6，图中左侧道路与右侧道路长度均为10，车辆均为直行

T时刻道路车辆状态如下：

Figure 16：T时刻道路车辆状态

T+1时刻道路车辆状态如下：

Figure 17：T+1时刻道路车辆状态

T+2时刻道路车辆状态如下:

Figure 18：T+2时刻道路车辆状态

4. 假定车辆100、200、300、101、201、301的车速均为5，图中左侧道路限速为4，图中右侧道路限速为1，图中左侧道路与右侧道路长度均为10，车辆均为直行

T时刻道路车辆状态如下：

Figure 19：T时刻道路车辆状态

T+1时刻道路车辆状态如下：

Figure 20：T+1时刻道路车辆状态

T+2时刻道路车辆状态如下:

Figure 21：T+2时刻道路车辆状态

T+3时刻道路车辆状态如下:

Figure 22：T+3时刻道路车辆状态

5. 假定车辆100、200、300、101、201、301的车速均为5，图中左侧道路限速为4，图中右侧道路限速为5，图中左侧道路与右侧道路长度均为10，车辆均为直行

T时刻道路车辆状态如下：

Figure 23：T时刻道路车辆状态

T+1时刻道路车辆状态如下：

Figure 24：T+1时刻道路车辆状态

T+2时刻道路车辆状态如下:

Figure 25：T+2时刻道路车辆状态

6. 假定车辆100、200、300、101、201、301、400、500、600、401、501、601、700、800、900、701、801、901的车速均为5，图中各条道路限速均为5，图中各条道路长度均为10，车辆100、200、300、101、201、301为直行，车辆400、500、600、401、501、601为右转，车辆700、800、900、701、801、901为左转。

T时刻道路车辆状态如下：

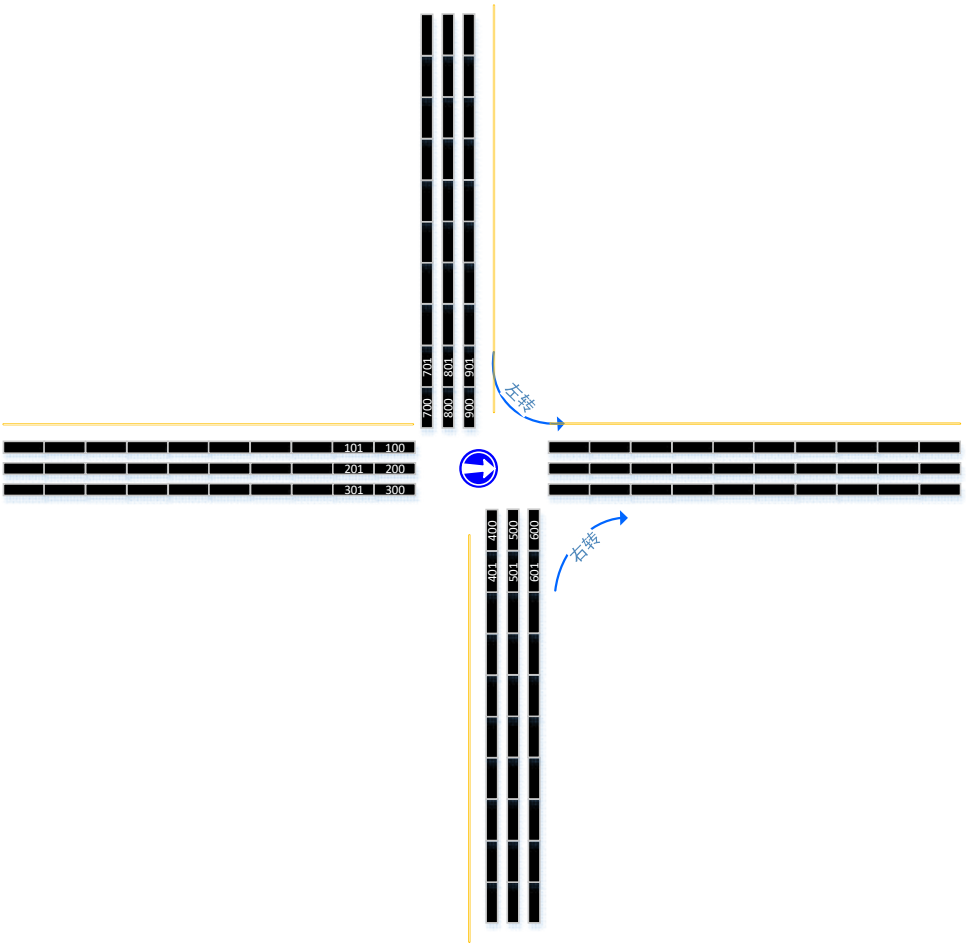

Figure 206:T时刻道路车辆状态

T+1时刻道路车辆状态如下：

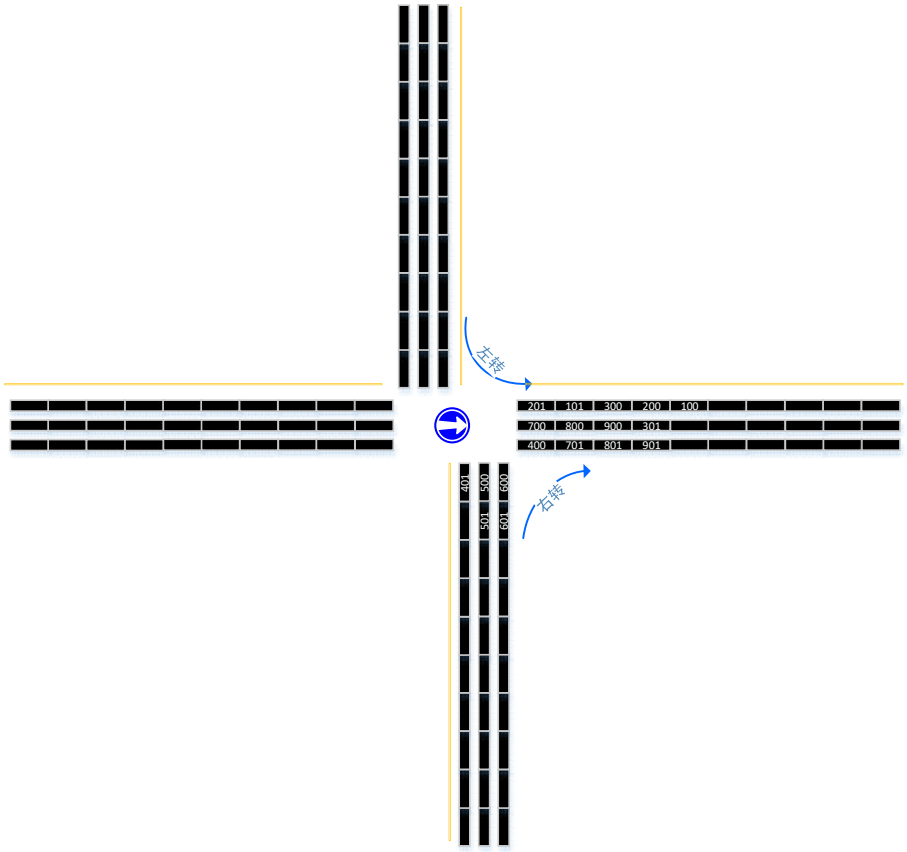

Figure 217:T+1时刻道路车辆状态

T+2时刻道路车辆状态如下：

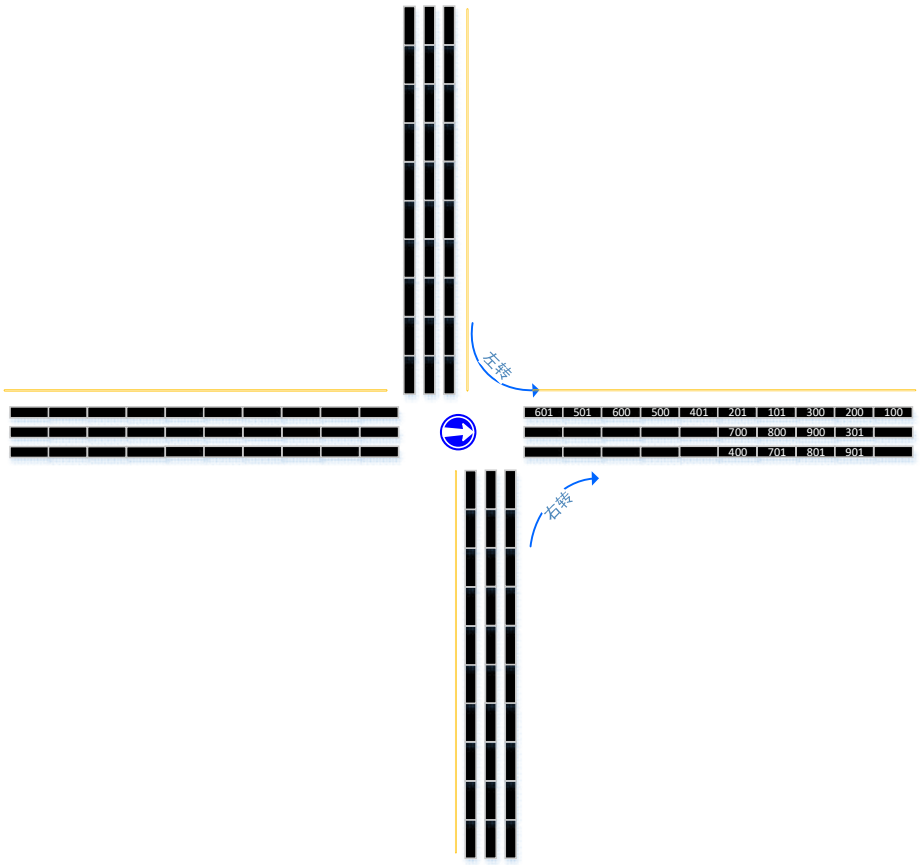

Figure 228:T+2时刻道路车辆状态

9. 系统调度详细说明

1. 系统调度时间从开始调度进行计算，时间计为0时刻，从0时刻开始计算。每个调度计为一个时间片，一个调度驱动所有车辆行驶一个时间单位。系统调度结束所处的时间片数即为系统调度时间。
2. 系统调度最小单位为1个单位，也就是一个时间片，不考虑小数。不会出现1/2时间单位，1/3时间单位，1/4时间单位等小于一个时间单位的情况。也就是说车辆要么走一个时间单位，要么不走，不会出现车辆先走1/2时间单位，再走1/2时间单位。比如一辆车的可以行驶的速度为3，则一次调度一个时间单位行驶距离3，不能以1/3时间单位、1/3时间单位、1/3时间单位调度且相应行驶距离1、距离1、距离1的情况。
3. 车辆最小行驶距离为1，不考虑小数。也就是车辆最小行驶1个单位，不可以先行驶1/2单位，再行驶1/2单位。

4. 系统调度先调度在路上行驶的车辆进行行驶，当道路上所有车辆全部不可再行驶后再调度等待上路行驶的车辆。
5. 调度等待上路行驶的车辆，按等待车辆ID升序进行调度，进入道路车道依然按车道小优先进行进入。
6. 说明一下判题系统的调度处理逻辑：

第一步：

该步骤处理所有道路的车辆的顺序，不影响其他道路上车辆的顺序，因此先调度哪条道路无关紧要。

a) 先处理每条道路上的车辆，将这些车辆进行遍历扫描，如果车在经过行驶速度（前方没有车辆阻挡）可以出路口，将这些车辆标记为等待行驶车辆。

b) 车辆如果行驶过程中，前方没有阻挡并且也不会出路口（v=min(最大车速，道路限速)），则该车辆行驶可行驶的最大车速（v=min(最大车速，道路限速)），此时该车辆在本次调度确定了该时刻的终止位置。该车辆标记为终止状态。

c) 车辆如果行驶过程中，发现前方有车辆阻挡，且阻挡的车辆为等待车辆，则该辆车也被标记为等待行驶车辆。（与阻挡车辆的距离s < v*t)） 其中：v=min(最大车速，道路限速),t=1

d) 车辆如果行驶过程中，发现前方有车辆阻挡，且阻挡的车辆为终止状态车辆，则该辆车也被标记为终止车辆。（与前方阻挡的车辆的距离记为s）则该车辆最大行驶速度为v = min(最高车速，道路限速，s/t) 其中t=1，该车辆最大可行驶距离为s。

e) 遍历道路上车辆由第一排向最后一排进行遍历，确定每辆车的行驶状态。（出道路处为道路第一排，入道路处为是后一排）

第二步：

处理所有路口、道路中处于等待状态的车辆，等待车辆的调度顺序按7、8、9进行调度。

7. 整个系统调度按路口ID升序进行调度各个路口，路口内各道路按道路ID升序进行调度。每个路口遍历道路时，只调度该道路出路口的方向。

如下所示则调度路口5时，只调度道路500从路口6到路口5的方向；调度路口6时，只调度道路500从路口5到路口6的方向。

如：路口5 <-----500------ 路口6

路口5 ------500-----> 路口6

道路500的起始点为路口5，终止点为路口6

8. 道路内部车辆调度按任务书给定的优先顺序进行调度。

a) **在每次调度中，调度到的车辆要么行驶其可行驶的最大车速，要么就会因等待其他车辆行驶而处于等待行驶状态，待所等待行驶的车辆行驶后，再使该车辆行驶其可行驶的最大车速。**

b) **是否发生冲突，只与相关道路的第一优先级车辆的行驶方向进行比较，判定是否发生冲突。**

c) **每条道路如果当前道路第一优先级车辆不能行驶，则当前道路后面的车辆都不能行驶，只有第一优先级的车辆行驶了，后面第二优先级才可以确定是否可以行驶。**

注：

每个道路（道路R）一旦有一辆等待车辆（记为车A，所在车道记为C）通过路口而成为终止状态，则会该道路R的车道C上所有车辆进行一次调度，如第一步所示，仅仅处理该道路该车道上能在该车道内行驶后成为终止状态的车辆（对于调度后依然是等待状态的车辆不进行调度，且依然标记为等待状态）

----尽可能多、尽可能快地将车辆确定为终止状态

（本车道内将可以达到终止状态主要目的是尽快让出空位让其他道路的车辆可以进入该道路，同时会重新识别真正等待出路口的车辆。因为有些车辆是因为前车是等待状态而导致自己也是等待状态，有可能他自己这次根本不会出路口，也就是其不参与出路口的优先级排序---只有出路口的车辆才参与优先级排序）

假定道路有一个车道且有如下车辆，车AB车速为3，车CD车速均为1

A空空B空空CD（路口）

按步骤一后ABCD均为等待状态。

在处理待状态车辆D后，假定D前进到其他道路后，此刻道路状态变化为A空空B空空C空（路口）

接下来因D车辆的前进后，需要对该条道路该车道的所有车辆进行一次调度，只调度经过一次调度会依然在该车道内车辆且状态为终止状态（不出路口）

因此：

C的车速为1，则C车前进1个距离，且为终止状态。A空空B空空空C（路口）

B的车速为3，则B车前进3个距离，且为终止状态。A空空空空空BC（路口）

A的车速为3，则A车前进3个距离，且为终止状态。空空空A空空BC（路口）

调度后道路车辆分布为：空空空A空空BC（路口），且ABC均为终止状态。

9. 一个时间单位内，一次调度时7与8的调度会出现**多次重复调度**，**因每次调度有可能因为等待其他道路车辆的行驶而导致当前车辆无法行驶，因此会循环调度使所有车辆行驶一个时间单位**。
10. 每次车辆调度，该车辆要么不走，要么行驶其可行驶的最大车速。不存在该车辆先走0.5时间单位，再走0.5时间单位的情况。
11. 基于10，**参赛选手需要注意**，一次调度能使所有车辆均到达各车辆的行驶速度行驶，就得保证不能出现各车辆循环等待的情况，否则该次调度就会**锁死**。**循环等待是指比如车辆A等待车辆B，车辆B等待车辆C，车辆C等待车辆D，车辆D等待车辆E，车辆E等待车辆F，车辆F等待车辆A的情况。**
12. 为简化实现，整个系统不存在小数。也就是不存在车辆实际出发时间为小数、调度为小数、行驶距离为小数、车辆速度为小数等。
13. 给出如下图例说明一下路口调度顺序：

    假定各道路均为双向道路，且每条道路长度为20（因便于图中仅标示出来长度5）

    图中双黄线为道路中间线，假定双黄线左侧道路全部是空车道，没有车辆阻挡，能够容纳所有进入的车辆（图中仅标示出来长度为5，假定长度为20）。

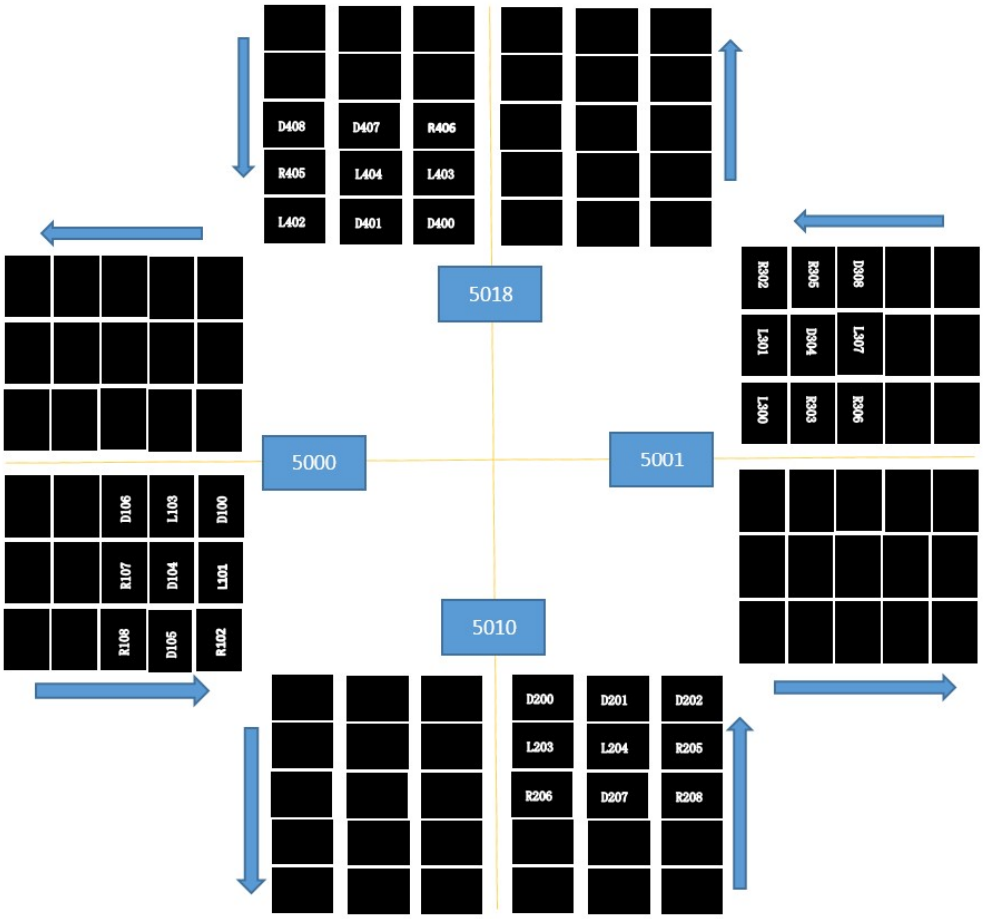

Figure29：路口多道路调度顺序

a) 道路编号如图所示5000、5001、5010、5018

b) 车辆编号前方各字母表示方向，不体现在车辆编号中。D：直行；L：左转；R：右转

c) 车辆编号为字母后纯数字

d) 在本路口调度道路的顺序为5000、5001、5010、5018、5000、5001、5010、5018、5000、5001、5010、5018...5000、5001、5010、5018

e) 假设4条道路上所有车辆均可以通过路口，则车辆调度顺序如下：

D100、D200、D201、D202、L203、L204、R205、R206、D207、R208、D400、D401、L402、L403、L404、R405、R406、D407、D408、L101、L300、L301、R302、R303、D304、R305、R306、L307、D308、R102、L103、D104、D105、D106、R107、R108

- 首先调度5000道路，只有D100可以走，因为D101左转与5010道路的D200冲突，所以必须等待5010车道上没有直行冲突。

- 其次再调度5001道路，因5001道路的L300与道路5018的D400冲突，所以L300必须等待道路5018的D400先行后再调度行驶。
- 再次调度道路5010，D200、D201、D202不与其他道路的车辆冲突，可以直接行驶。L203左转，且道路5001无左转车辆与其冲突，因此L203可以左转。L204也可以左转。L205为右转，与左侧道路5000的车辆L101不冲突（只与道路5000的直行会发生冲突），且与道路5018的D400不发生冲突（只与道路5018的左转车辆发生冲突），因此车辆L205可以右转，依次道路5010上的剩余车辆全部可以通过。
- 接着调度5018道路上的车辆.....
- 再调度道路5000....
- 再调度道路5001...
- 再调度道路5000....

是否发生冲突，只与相关道路的第一优先级车辆的行驶方向进行比较，看是否发生冲突。

每次调度到一条道路，直到该道路无车辆可调度，或该条道路上车辆处于冲突状态。也就是说尽可能多地让该道路行驶，直到没有车辆或者车辆与其他车辆发生冲突不可行驶。

10. 其他说明

1. 若因参赛选手输出的路径导致部分车辆因交通规则限制而无法通行的情况，直接判负。如Figure30：异常堵死情况举例所示：
   a) 红色线路和绿色线路为参赛选手输出的两条车辆规划运行线路。
   b) 在这两条规划的线路上有很多车辆在行驶。
   c) 在路口6的因左转优先，只能红色线路的车辆进行左转。
   d) 在路口12的也是左转优先，只能是绿色线路的车辆进行左转。
   e) 因路口6是左转优先，所以绿色线路的车辆无法右转通行，最终积压导致路口12绿色车辆无法通行。
   f) 因路口12只能绿色线路车辆左转通行，红色线路车辆无法右转，从而导致积压在路口6的红色线路无法左转。

g) 由此产生路口6和路口12出现相互堵死的情况，车辆无法再继续前行。

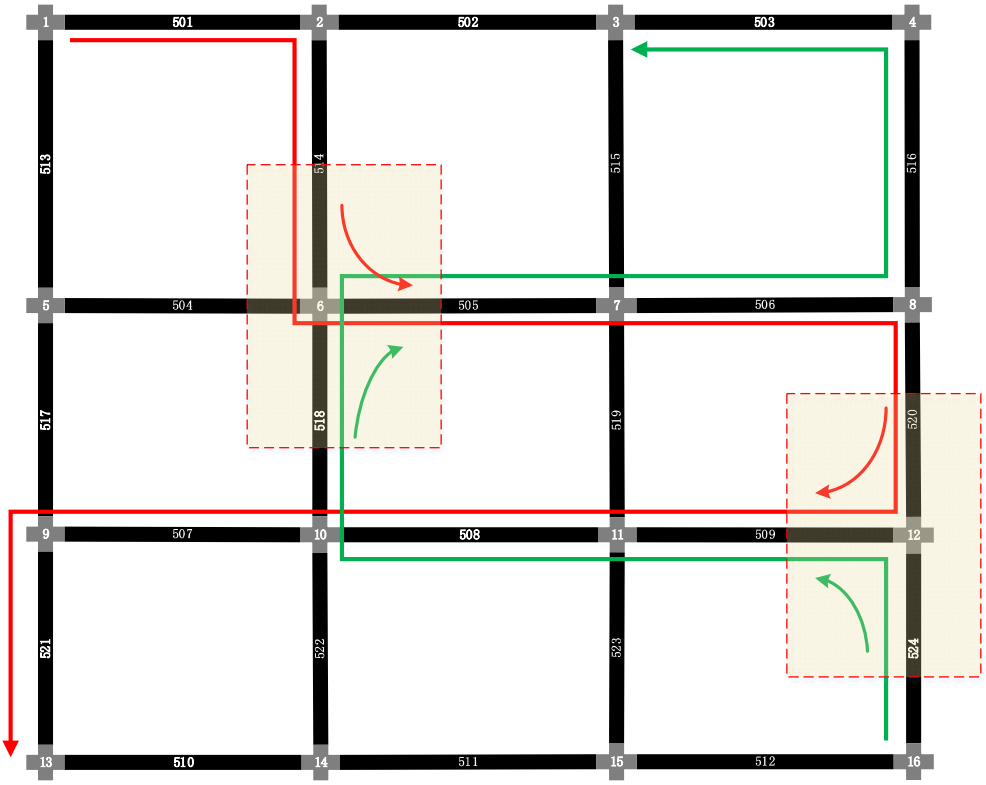

Figure30：异常堵死情况举例

2. 若因参赛选手输出的路径导致部分车辆因调度规则限制而无法通行的情况，直接判负。如Figure30：异常堵死情况举例所示：

a) 假定下图各车辆100、200、300、400、500、600、700、800车速均为6，图中各道路限速均为8，车辆100、300、500、700均为右转。

b) 图中各条道路长度均为10

c) 因车辆100右转，需要等待车辆800的前行而导致车辆100处于等待状态

d) 因车辆100处于等待状态，而导致车辆200也必须处于等待状态

e) 相同的原因车辆700处于等待状态，车辆800也处于等待状态

f) 同理，车辆300、400、500、600均处于等待状态

g) 如下图中车辆100、200、300、400、500、600、700、800均处于等待状态，形成循环等待

h) 如此，下图中各车辆处于相互锁定状态

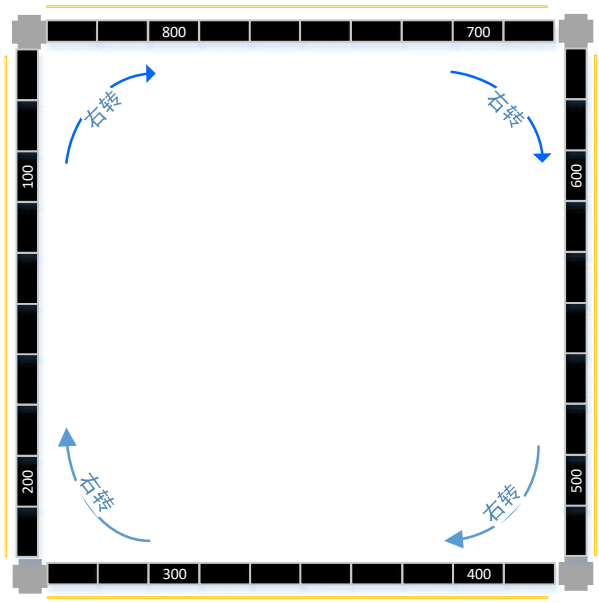

Figure31：循环等待导致死锁样例

3. 整个系统限制参赛选手程序编译时间最大为60s，程序运行生成answer.txt时间为300s。

11. 附判题器伪码

（输入为car.txt,cross.txt,road.txt answer.txt， 输出为调度时间、总调度时间）

```
for(/* 按时间片处理 */) {
    while(/* all car in road run into end state */){
        foreach(roads) {
      /* 调整所有道路上在道路上的车辆，让道路上车辆前进，只要不出路口且可以到达终止状态的车辆
     * 分别标记出来等待的车辆（要出路口的车辆，或者因为要出路口的车辆阻挡而不能前进的车辆）
     * 和终止状态的车辆（在该车道内可以经过这一次调度可以行驶其最大可行驶距离的车辆）*/
            driveAllCarJustOnRoadToEndState(allChannle);/* 对所有车道进行调整 */

            /* driveAllCarJustOnRoadToEndState该处理内的算法与性能自行考虑 */
            }
    }
```

```
    while(/* all car in road run into end state */){
        /* driveAllWaitCar() */
        foreach(crosses){
            foreach(roads){
                Direction dir = getDirection();
                Car car = getCarFromRoad(road, dir);
                if (conflict){
                    break;
                }

                channle = car.getChannel();
                car.moveToNextRoad();

                /* driveAllCarJustOnRoadToEndState该处理内的算法与性能自行考虑 */
                driveAllCarJustOnRoadToEndState(channel);
            }
        }
    }

    /* 车库中的车辆上路行驶 */
    driveCarInGarage();
}
```