《 智能系统 》实验报告

学号	姓名	承担任务	贡献度	得分
20215279	李宽宇	共同完成环境配置、实验设计、代码实现、文档	25	
20215027	禹天尹	共同完成环境配置、实验设计、代码实现、文档	25	
20215481	张志维	共同完成环境配置、实验设计、代码实现、文档	25	
20214261	朱齐高	共同完成环境配置、实验设计、代码实现、文档	25	
			共 100	

实验题目	数据库与知识库设计		
实验时间	2024.3.31	实验地点	DS1401
实验成绩		实验性质	☐验证性 ☐设计性 ☑综合性

教师评价：

☐算法/实验过程正确　　☐源程序/实验内容提交

☐程序结构/实验步骤合理　　☐实验结果正确

☐语法、语义正确　　☐报告规范

其他：

评价教师签名：

一、实验目的

为实现十字路口红绿灯智能控制，本次实验的目的是：

（1） 练习知识库的设计，测试与维护

（2） 大型知识库的实现设计。

（3） 可信度知识库的设计与实现（可选）

报告创建时间：

二、实验项目内容

1、基于 pyknow，设计十字路口红绿灯智能控制的知识库

（1） 基于 pyknow 语法，给出十字路口红绿灯智能控制的完整知识库。

（2） 通过模拟运行，测试并修改知识库，保证十字路口红绿灯智能控制的正确性。

2、基于关系数据库管理系统，（如 mysql），设计并实现大型知识库的存储和管理。

（1） 为什么需要用关系数据库来存储大型知识库？

（2） 对 1 中的知识库，设计出表结构（给出 DDL 程序）

（3） 将 1 中的知识库存入关系数据库（给出 DML 程序）

（4） 举例给出知识库的使用和维护过程（给出 DML 程序）

3、可信度知识库设计（可选）

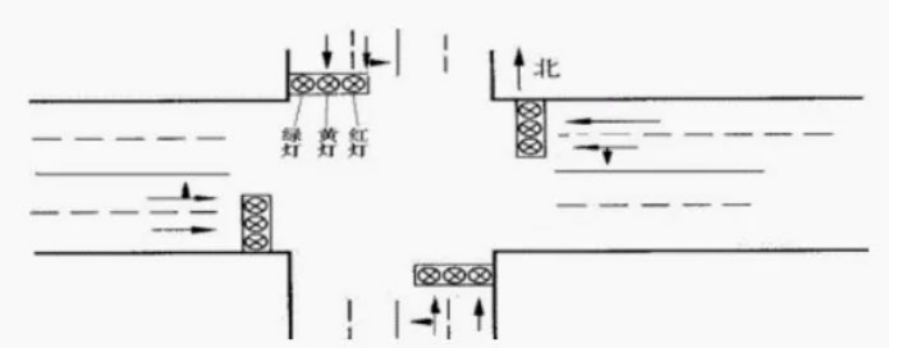
三、实验过程或算法（代码）

1. 基于pyknow，设计十字路口红绿灯智能控制的知识库

注：实验框架的pyknow 包已经不在pypi 中，降低python版本至3.4，运行实验包的setup.py即可安装pyknow

（1）十字路口红绿灯智能控制的完整知识库

分析：



①红灯：禁止直行或左转，车辆允许右转。

②绿灯：车辆可以按照指示行驶。

③当东西方向红灯亮起时，南北方向绿灯亮起；

④当南北方向红灯亮起，东西方向绿灯亮起。

⑤红绿灯的时间分配也应当考虑某方向的车辆占总车辆的比值来决定当前方向分配到的绿灯时间占总时间的比值。

故构建知识库如下

事实：

```
yield Fact(Ticks=0)
yield Fact(NSsign='RED')
yield Fact(WEsign='GREEN')
yield Fact(switch=False)
yield Fact(switchTime=5)
yield Fact(NeedChange=False)
yield Fact(NeedShow=False)
```

Ticks: 当前时间 int

NSsign: 南北信号灯 Str

WEsign: 东西信号灯 Str

Switch: 修改信号灯颜色 bool

switchTime: 举例下一次转换的时间 int

NeedChange: 修改红绿灯时间长度比例 bool

NeedShow: 显示当前路口灯的情况 bool

```
PERIOD = 10
# 每个周期需要通过 NS, WE 的车辆数目 模拟列表，第0个元素代表第0个周期
CARS = [[random.randint( a: 0,  b: 20), random.randint( a: 0,  b: 20)]
        for i in range(100)]
CARS.insert( _index: 0,  _object: [0, 0])
CARS.insert( _index: 1,  _object: [3, 7])
```

PERIOD: 单个周期长度

CARS: 每个周期需要通过 NS, WE 的车辆数目 模拟列表，第0个元素代表第0个周期

规则：

①计时，每秒都会输出时间

```
    """
    @Rule(AS.oldFact << Fact(Ticks=MATCH.times))
    def ticks(self, times, oldFact):
        time.sleep(1)
        self.retract(oldFact)
        self.declare(Fact(Ticks=times + 1))
        print("====== 启动时间{}秒 ======".format(times))
```

②计时到半周期，需要交换红绿灯并显示

```
@Rule(
    *args: Fact(Ticks=MATCH.times),
    Fact(switchTime=MATCH.switchTime),
    TEST(lambda switchTime, times: times % 10 == switchTime),
    salience=3
)
def needSwitch(self):
    self.declare(Fact(switch=True))
    self.declare(Fact(NeedShow=True))
```

③一个周期后修改比例

```
@Rule(
    *args: Fact(Ticks=MATCH.times),
    TEST(lambda switchTime, times: times % PERIOD == 0),
    salience=2
)
def needChange(self):
    self.declare(Fact(NeedChange=True))
```

④换灯，南北RED->GREEN,东西Green->RED，

```
@Rule(
    *args: AS.oldSwtich << Fact(switch=True),
    AS.oldNS << Fact(NSsign='RED'),
    AS.oldWE << Fact(WEsign='GREEN'),
    salience=2
)
def switch1(self, oldSwtich, oldNS, oldWE):
    self.declare(Fact(NSsign='GREEN'))
    self.declare(Fact(WEsign='RED'))
    self.retract(oldSwtich)
    self.retract(oldWE)
    self.retract(oldNS)
```

⑤换灯，南北Green->RED,东西RED->GREEN

```
@Rule(
    *args: AS.oldSwtich << Fact(switch=True),
    AS.oldNS << Fact(NSsign='GREEN'),
    AS.oldWE << Fact(WEsign='RED'),
    salience=2
)
def switch2(self, oldSwtich, oldNS, oldWE):
    self.declare(Fact(NSsign='RED'))
    self.declare(Fact(WEsign='GREEN'))
    self.retract(oldSwtich)
    self.retract(oldWE)
    self.retract(oldNS)
```

⑥显示当前的指示灯情况

```
@Rule(
    *args: Fact(NSsign=MATCH.NScolor),
    Fact(WEsign=MATCH.WEcolor),
    Fact(switchTime=MATCH.switchTime),
    AS.oldFact << Fact(NeedShow=True),
    salience=4
)
def show(self, NScolor, switchTime, oldFact):
    self.retract(oldFact)
    print('当前路况')
    if NScolor == "RED":
        print(Fore.WHITE + '==========' + Fore.RED + "剩余时间: {} s".format(switchTime) + Fore.WHITE + '==========')
        print(Fore.GREEN + "剩余时间: {} s".format(
            switchTime) + Fore.WHITE + '========' + Fore.GREEN + "剩余时间: {} s".format(switchTime))
        print(Fore.WHITE + '==========' + Fore.RED + "剩余时间: {} s".format(switchTime) + Fore.WHITE + '==========')
    if NScolor == "GREEN":
        print(
            Fore.WHITE + '==========' + Fore.GREEN + "剩余时间: {} s".format(
                PERIOD - switchTime) + Fore.WHITE + '==========')
        print(Fore.RED + "剩余时间: {} s".format(
            PERIOD - switchTime) + Fore.WHITE + '========' + Fore.RED + "剩余时间: {} s".format(PERIOD - switchTime))
        print(
            Fore.WHITE + '==========' + Fore.GREEN + "剩余时间: {} s".format(
                PERIOD - switchTime) + Fore.WHITE + '==========')
```

⑦修改红绿灯时长比例，红绿灯的时间分配也应当考虑某方向的车辆占总车辆的比值来决定当前方向分配到的绿灯时间占总时间的比值。

```
@Rule(
    *args: Fact(Ticks=MATCH.times),
    AS.oldFact1 << Fact(switchTime=MATCH.switchTime),
    AS.oldFact2 << Fact(NeedChange=True),
    salience=2
)
def changeSwitchTime(self, oldFact1, oldFact2, times, switchTime):
    print("修改时间")
    self.retract(oldFact1)
    self.retract(oldFact2)
    cur_epoch = times // PERIOD
    newSwitchTime = switchTime
    # 修正红绿灯
    print("第 {} 个周期，南北方向通过{}辆车，东西方向通过{}辆车".format( *args: cur_epoch, CARS[cur_epoch][0],
                                                                    CARS[cur_epoch][1]))
    if cur_epoch != 0:
        newSwitchTime = math.floor(10 / (CARS[cur_epoch][0] / (CARS[cur_epoch][1] + 0.00001) + 1))
    self.declare(Fact(switchTime=newSwitchTime))
    self.declare(Fact(switch=True))
    self.declare(Fact(NeedShow=True))
```

其中，修正红绿灯的公式为

$$南北方向红灯时长_{下个周期} = \left\lfloor 周期长度 - \frac{周期长度}{\frac{南北方向通过的车辆数目}{东西方向通过的车辆数目 + 0.0001} + 1} \right\rfloor$$

（2）通过模拟运行，测试并修改知识库，保证十字路口红绿灯智能控制的正确性。

通过修改，模拟车流的随机数生成，测试十字路口红绿灯智能

控制的正确性。

实验：南北假设是在10-20的均匀分布，东西是在0-10的均匀分布

```
CARS = [[random.randint( a: 10,  b: 20), random.randint( a: 0,  b: 10)]
        for i in range(100)]
```

分析与结论见四(3)

2、基于关系数据库管理系统，（如 mysql），设计并实现大型知识库的存储和管理。

（1） 为什么需要用关系数据库来存储大型知识库？

关系数据库在存储大型知识库时具有以下优势：

①**结构化数据存储**：它使用表格形式组织数据，提供高效的数据管理。

②**灵活的查询语言**：使用 SQL 进行复杂查询，对大型知识库的数据查询和分析至关重要。

③**数据完整性和一致性**：支持事务处理，确保数据插入、更新或删除时的数据完整性和一致性。

④**扩展性和性能优化**：通过水平和垂直扩展应对大型知识库的存储需求，并提供性能优化技术，如索引、查询优化等。

⑤**数据安全和权限控制**：提供安全性和权限控制机制，确保只有授权的用户能访问和修改知识库中的数据，保护敏感信息的安全。

（2） 对 1 中的知识库，设计出表结构（给出 DDL 程序）

事实：

Ticks: 当前时间 int，作为主键

NSsign: 南北信号灯 Str

WEsign: 东西信号灯 Str

Switch: 修改信号灯颜色 bool

switchTime: 举例下一次转换的时间 int

NeedChange: 修改红绿灯时间长度比例 bool

NeedShow: 显示当前路口灯的情况 bool

PERIOD: 单个周期长度

所有的字段都是关于一个十字路口的属性，因此可以合理地将它们放在一张表中。

```
CREATE TABLE TrafficSignal (
    Ticks INT,
    NSsign VARCHAR(255),
    WEsign VARCHAR(255),
    Switch BOOLEAN,
    switchTime INT,
    NeedChange BOOLEAN,
    NeedShow BOOLEAN,
    PERIOD INT,
    PRIMARY KEY (Ticks)
);
```

CARS: 每个周期需要通过 NS, WE 的车辆数目模拟列表。

```
CREATE TABLE Cars (
    CarID INT AUTO_INCREMENT,
    RandomValue1 INT,
    RandomValue2 INT,
    PRIMARY KEY (CarID)
);
```

（3）将1中的知识库存入关系数据库（给出DML程序）

代码如下：

```
INSERT INTO TrafficSignal (Ticks, NSsign, WEsign, Switch, switchTime, NeedChange, NeedShow)
VALUES (0, 'RED', 'GREEN', FALSE, 5, FALSE, FALSE);
```

（4）举例给出知识库的使用和维护过程（给出DML程序）

使用：SQL语句获取数据，以时间为过滤条件经行查询

```
20 •    SELECT *
21      FROM TrafficSignal
22      WHERE Ticks = 100;
```

维护：在数据更新后，例如下1s时，所有数据都会更新，用如下语句

```
INSERT INTO TrafficSignal (Ticks, NSsign, WEsign, Switch, switchTime, NeedChange, NeedShow)
VALUES (0, 'RED', 'GREEN', FALSE, 5, FALSE, FALSE);
```

四、实验结果及分析

（1）启动智能系统，显示当前路况，初始化为 5s 切换红绿灯。

```
修改时间
第 0 个周期，南北方向通过0辆车，东西方向通过0辆车
当前路况
=========剩余时间: 5 s=========
剩余时间: 5 s=======剩余时间: 5 s
=========剩余时间: 5 s=========
====== 启动时间0秒 ======
====== 启动时间1秒 ======
====== 启动时间2秒 ======
====== 启动时间3秒 ======
====== 启动时间4秒 ======
当前路况
=========剩余时间: 5 s=========
剩余时间: 5 s=======剩余时间: 5 s
=========剩余时间: 5 s=========
====== 启动时间5秒 ======
====== 启动时间6秒 ======
====== 启动时间7秒 ======
====== 启动时间8秒 ======
====== 启动时间9秒 ======
```

（2）运行一段时间后，根据车流量计算红绿灯的比例

```
第 2 个周期，南北方向通过20辆车，东西方向通过10辆车
当前路况
=========剩余时间: 6 s=========
剩余时间: 6 s========剩余时间: 6 s
=========剩余时间: 6 s=========
====== 启动时间20秒 ======
====== 启动时间21秒 ======
====== 启动时间22秒 ======
====== 启动时间23秒 ======
====== 启动时间24秒 ======
====== 启动时间25秒 ======
当前路况
=========剩余时间: 4 s=========
剩余时间: 4 s========剩余时间: 4 s
=========剩余时间: 4 s=========
====== 启动时间26秒 ======
====== 启动时间27秒 ======
====== 启动时间28秒 ======
====== 启动时间29秒 ======
```

（3）分析：南北假设是在10-20的均匀分布，东西是在0-10的均匀分布

发现：南北向绿灯时间：东西向红灯时间>1，符合随机数的分布

结论：经过多组数据的测试，认为在同一时间段内，车流量稳定的假设下，能保证十字路口红绿灯智能控制的正确性。