交通规划原理期末项目

项目组人员

- 小组人员：**王森，刘治学，田景颢，刘杰**
- 指导老师：**熊耀华**

注意事项

- 项目在`Github`上开源，地址：https://github.com/DavidingPlus/TP_Project
- 项目依赖`python3`环境，依赖`PyQt5`包，`windows`系统和`Linux`系统均兼容，本测试在`windows`上运行
    - 安装`PyQT5`：在终端使用如下命令

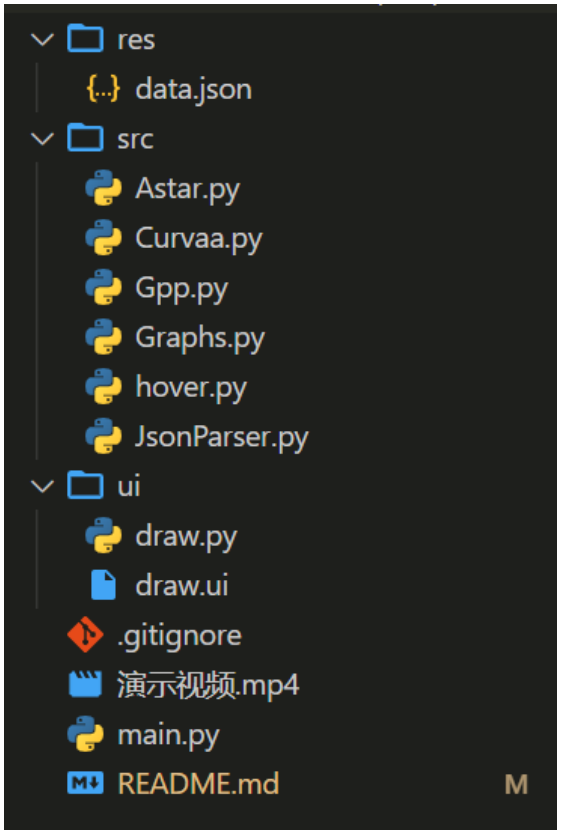
```
pip install -i https://pypi.tuna.tsinghua.edu.cn/simple pyqt5 pyqt5-tools
```

- 项目根目录中的`项目文档`是本`README.md`的`pdf`版本
- 项目根目录中的`演示视频`是该项目的测试视频

代码分析

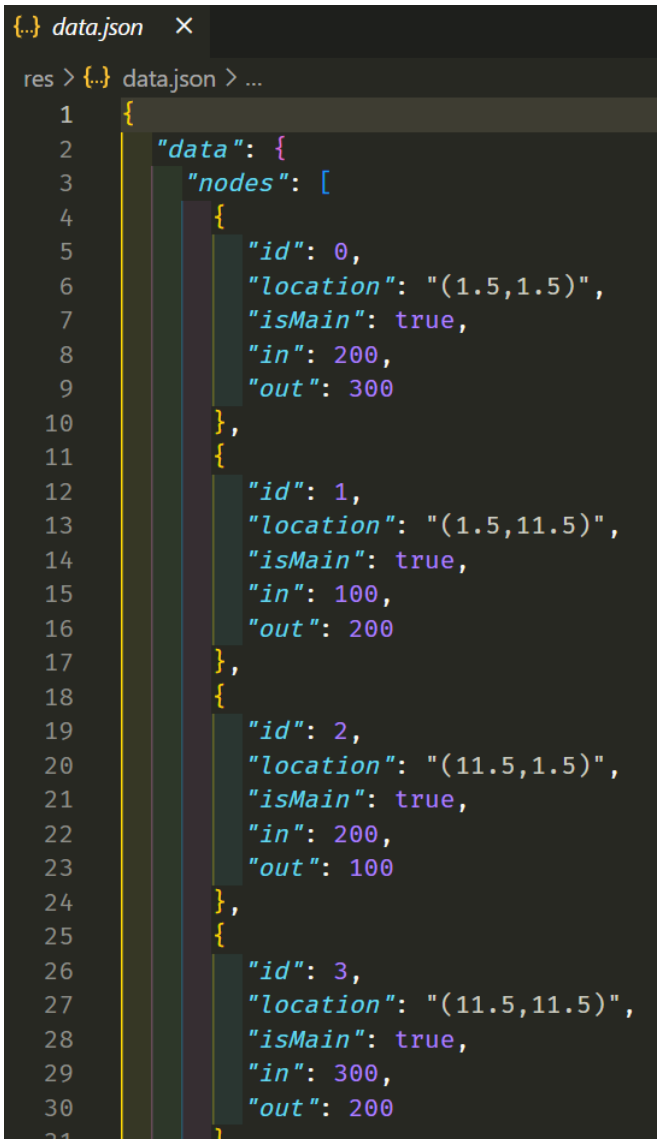
整体框架

- 代码的框架如下：



res目录

- `res`目录：存放交通网络结点和边的数据，是一个`json`文件



src目录

- `src`目录：存放代码执行需要的依赖文件，类似于库或者包
  - `Astar.py`
    - `Astars`是一个用于进行路径计算的算法库，在该算法中，我们可以进行`Astars`作为算法的起始，我们选用曼哈顿距离作为启发函数

Astar.py

```python
"""
Astars
========

Astars 是一个用于进行路径计算的算法库，在该算法中，我们可以进行Astars作为算法的起始，我们选用
曼哈顿距离作为启发函数
"""

import collections
import heapq


class Astars:
    def __init__(self, nodes):
        self.graph = collections.defaultdict(list)
        self.nodes = nodes

    def add_edge(self, start, end, distance, resistiance):
        self.graph[start].append((end, distance, resistiance))

    def calculate_shortest_path(self, G, source, target, weight='distance'):
        visited = set()
        previous_nodes = {node: None for node in G.graph}
        min_distance = {node: float('inf') for node in G.graph}
        min_distance[source] = 0
        priority_queue = [(0, source)]

        while priority_queue:
            current_distance, current_node = heapq.heappop(priority_queue)
```

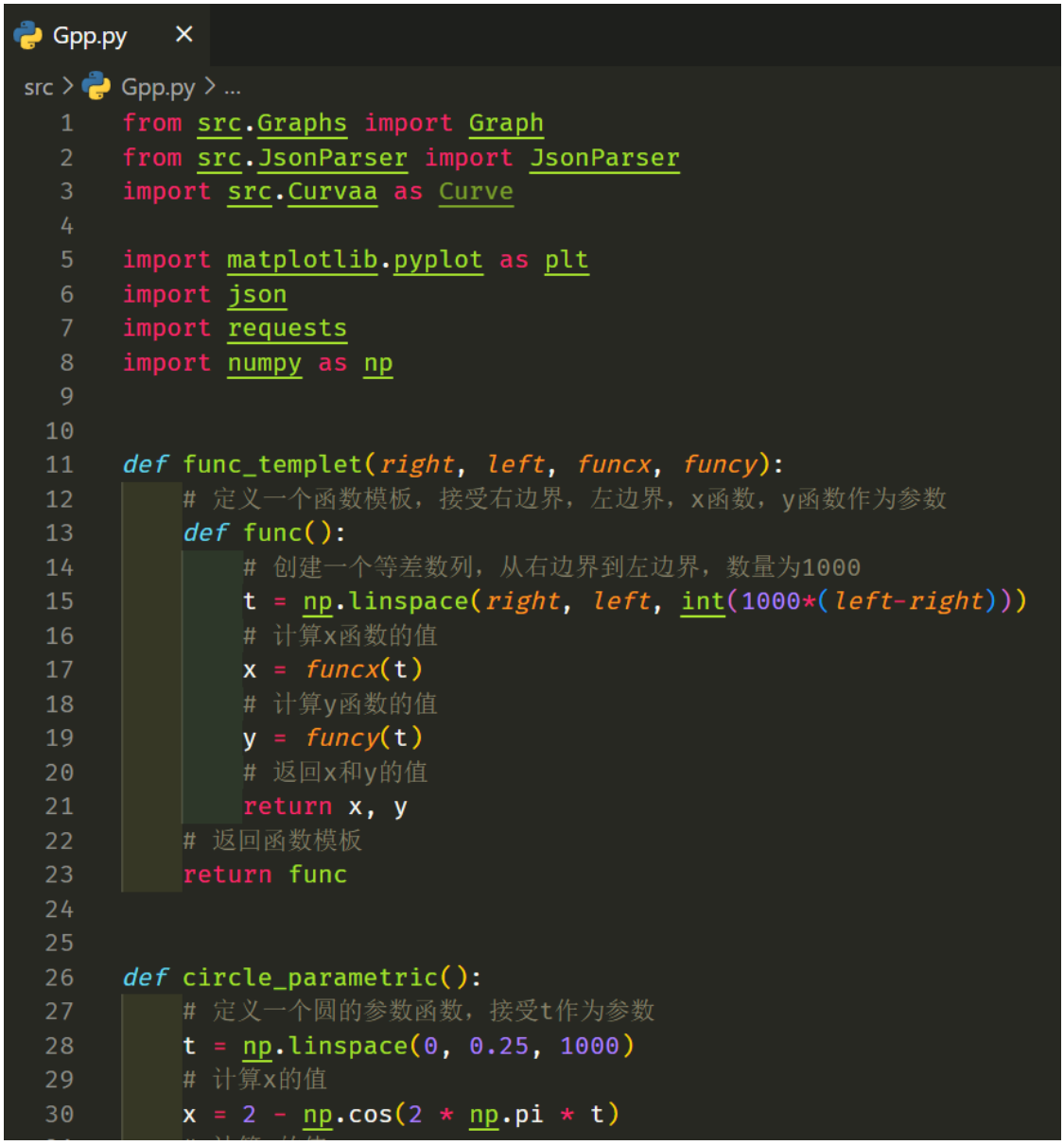
- `Curvva.py`
  - `Curvaa`是一个用于进行路径计算、绘制的库，在该算法中，我们可以进行计算绘制，画出参数方程形式的路径

Curvaa.py

```python
"""
Curvaa
========

Curvaa 是一个用于进行路径计算、绘制的库，在该算法中，我们可以进行
计算绘制，画出参数方程形式的路径。
"""

import math
from matplotlib import patches
from matplotlib.colors import LogNorm
import numpy as np
import matplotlib.pyplot as plt


class Curve:
    def __init__(self, left, right, func) -> None:
        self.func = func
        self.left = left
        self.right = right

    def get_curve(self):
        x, y = self.func()
        length = self.curve_length(x, y)
        # print("Curve Length:", length)
        return length

    def draw_function(self, cmaps, resistiance, type="Graph", vmax=100, vmin=0):
        x, y = self.func()
        print(resistiance)
```

- `Gpp.py`

- `Gpp.py` 是用来进行图形的基础绘制的文件，里面通过拿取 `data.json` 的数据，将数据转化为图展示出来，在其中还包括了发送 `get` 请求从远端服务器获得数据的操作



Gpp.py

src > Gpp.py > ...

```python
from src.Graphs import Graph
from src.JsonParser import JsonParser
import src.Curvaa as Curve

import matplotlib.pyplot as plt
import json
import requests
import numpy as np


def func_templet(right, left, funcx, funcy):
    # 定义一个函数模板，接受右边界，左边界，x函数，y函数作为参数
    def func():
        # 创建一个等差数列，从右边界到左边界，数量为1000
        t = np.linspace(right, left, int(1000*(left-right)))
        # 计算x函数的值
        x = funcx(t)
        # 计算y函数的值
        y = funcy(t)
        # 返回x和y的值
        return x, y
    # 返回函数模板
    return func


def circle_parametric():
    # 定义一个圆的参数函数，接受t作为参数
    t = np.linspace(0, 0.25, 1000)
    # 计算x的值
    x = 2 - np.cos(2 * np.pi * t)
```

- `Graphs.py`
  - `Graphs` 是基于 `plt` 和 `networkx` 的简单的计算规划展示交通流的类
  - `Graphs` 主要使用了 `Astar` 算法进行计算，并通过该算法得到所需的结果进一步获取

Graphs.py

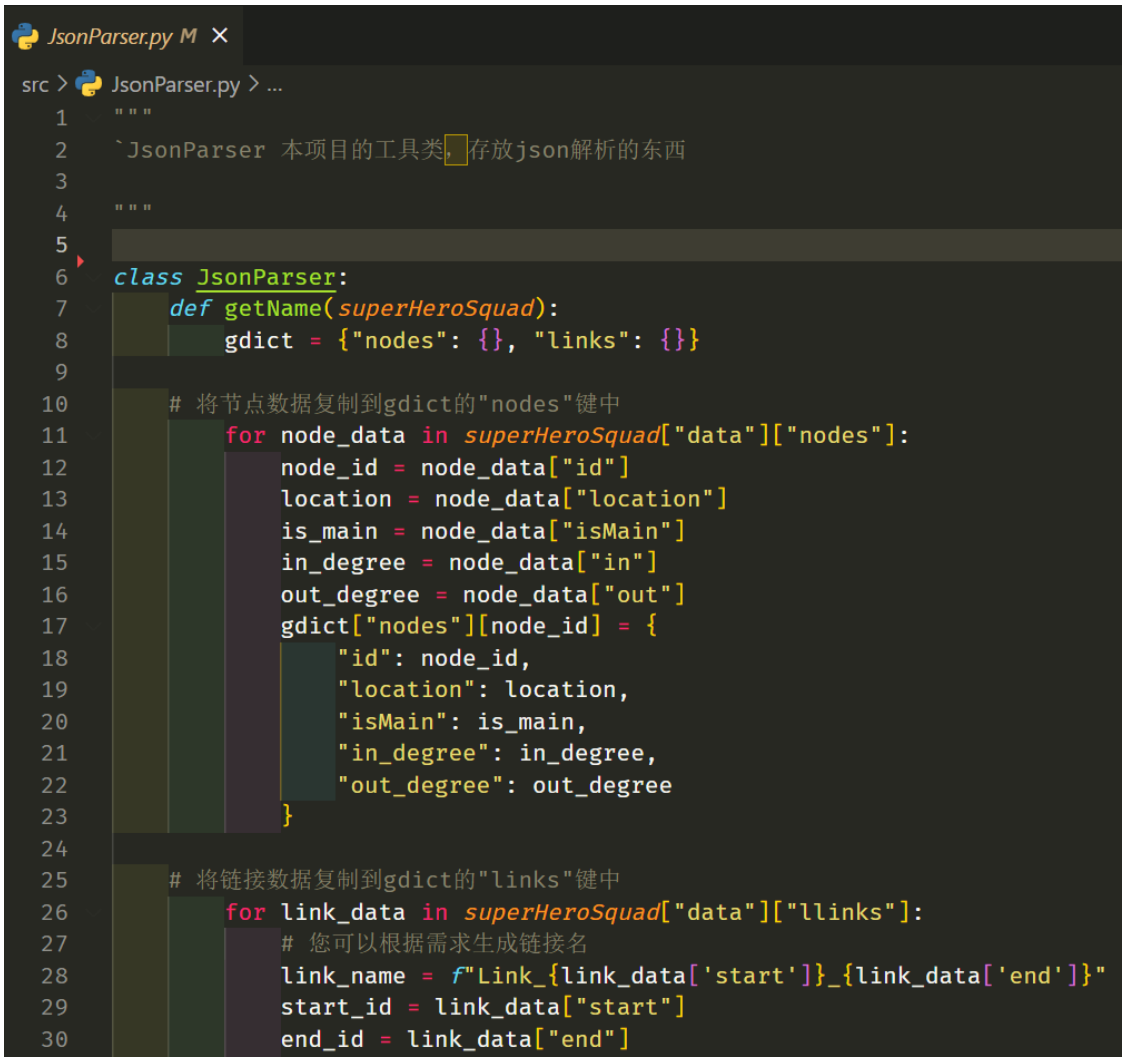
```python
"""
`Graphs` 是基于plt和netWorkX的简单的计算规划展示交通流的类

Graphs主要使用了Astar算法进行计算，并通过该算法得到所需的结果进一步获取。

"""

import src.Curvaa as Curve
from src.Astar import Astars

from matplotlib.cm import ScalarMappable
from matplotlib.colors import LinearSegmentedColormap, LogNorm
import matplotlib.pyplot as plt
import networkx as nx
import math
from matplotlib.backend_bases import PickEvent
import matplotlib.pyplot as plt


class Graph:
    """Extensible Graph traffic net slove class.
    """

    def __init__(self, gdict=None):
        if gdict is None:
            gdict = {"nodes": {}, "links": {}}
        self.gdict = gdict
        self.curves = {}
        self.ODMartix = {}
        self.node_positions = {}  # 用于存储节点位置的字典
```

- `hover.py`
  - 是一个颜色的工具库，最后的成品有三种不同的颜色主题进行展示，相关在这里进行定义

hover.py

```python
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap

# 创建一个包含几种颜色的Colormap
original_cmap = plt.cm.get_cmap('RdYlGn_r')

# 添加您希望的颜色（这里以蓝色为例）
custom_colors = ['#FF0000', '#00FF00', '#0000FF']  # 添加红、绿、蓝三种颜色
new_colors = original_cmap.colors + custom_colors

# 创建一个新的Colormap，包含添加的颜色
custom_cmap = ListedColormap(new_colors, name='custom_cmap')

# 使用新的Colormap绘制图形
data = [0, 1, 2, 3, 4]
plt.scatter(data, data, c=data, cmap=custom_cmap, s=100)
plt.colorbar(label='Data')

plt.show()
```

- `JsonPraser.py`
  - `JsonParser`本项目的工具类，存放`json`解析的东西



ui目录

- `ui`目录：存放成品展示的界面的`python`文件和`ui`文件
  - `draw.py`
    - 运用`pythonQt`库，在画出一个主界面加上三个按钮并且对应不同的事件

draw.py

ui > draw.py > ...

```python
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'draw.ui'
#
# Created by: PyQt5 UI code generator 5.15.9
#
# WARNING: Any manual changes made to this file will be lost when pyuic5 is
# run again.  Do not edit this file unless you know what you are doing.


from PyQt5 import QtCore, QtGui, QtWidgets


class Ui_Form(object):
    def setupUi(self, Form):
        Form.setObjectName("Form")
        Form.setEnabled(True)
        Form.resize(600, 600)
        sizePolicy = QtWidgets.QSizePolicy(
            QtWidgets.QSizePolicy.Preferred, QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(Form.sizePolicy().hasHeightForWidth())
        Form.setSizePolicy(sizePolicy)
        Form.setMinimumSize(QtCore.QSize(600, 600))
        Form.setMaximumSize(QtCore.QSize(600, 600))
        self.label = QtWidgets.QLabel(Form)
        self.label.setGeometry(QtCore.QRect(80, 110, 440, 190))
        font = QtGui.QFont()
        font.setFamily("楷体")
```

- `draw.ui`
  - 界面的`ui`文件

draw.ui

ui > draw.ui

```xml
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>Form</class>
 <widget class="QWidget" name="Form">
  <property name="enabled">
   <bool>true</bool>
  </property>
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>600</width>
    <height>600</height>
   </rect>
  </property>
  <property name="sizePolicy">
   <sizepolicy hsizetype="Preferred" vsizetype="Preferred">
    <horstretch>0</horstretch>
    <verstretch>0</verstretch>
   </sizepolicy>
  </property>
  <property name="minimumSize">
   <size>
    <width>600</width>
    <height>600</height>
   </size>
  </property>
  <property name="maximumSize">
   <size>
    <width>600</width>
```

主程序

- `main.py`：程序的主入口，创建`MainWindow`对象，并且运行之后展示出来，然后进行后续发送`get`请求，并处理数据画图的逻辑

main.py

```python
# Python Qt 图形化的主程序
import src.Gpp as Gpp
import ui.draw as draw

from PyQt5.QtWidgets import QApplication, QMainWindow
import sys

# json的数据文件路径
path = "./res/data.json"


# 自己封装的一个MainWindow类，我们工作逻辑的代码就放在Work中
class MainWindow:
    def __init__(self) -> None:
        # 主程序对象成员
        self.mainWindow = QMainWindow()

        # 创建ui，引用draw文件中的Ui_Form类
        self.ui = draw.Ui_Form()

        # 调用Ui_MainWindow类的setupUi，创建初始组件
        self.ui.setupUi(self.mainWindow)

        # 调用工作函数
        self.work()

    # 工作函数都在这里面
    def work(self) -> None:
        self.ui.DownGraphBtn.clicked.connect(self.DownGraph_Slot)
        self.ui.EasyGraphBtn.clicked.connect(self.EasyGraph_Slot)
```

具体分工

- 王森
  - 完成工具算法库`Astar`，`Curvaa`，`Graphs`的编写
  - 完成图形绘制和数据请求整合文件`Gpp.py`的编写
- 刘治学
  - 完成`PyQt5`的界面编写，包括`draw.py`和`draw.ui`
  - 完成主程序入口`main.py`的编写
  - 完成项目整体架构的设计和代码的汇总，以及文档的编写
- 田景颢
  - 完成`data.json`数据格式的设计和数据的生成
  - 完成`get`请求服务端的配置，能通过浏览器获取到数据的页面以及通过代码发送`get`请求获得数据
- 刘杰

- 完成颜色工具库 `hover` 和 `Json` 数据处理工具库 `JsonParser` 的编写
- 完成演示视频的录制和剪辑