ENGSCI 233

Data Lab

Date: May 3, 2021

Lab Objective

Often we'll store data or the output of a computer model across a series of different files, e.g., for different days, or different simulations. These might also be contained in different directories. Ideally, these files will be organised according to some sensible directory hierarchy or naming convention. Therefore, it is useful to be able to *navigate directory structures and access files* from within a computer program.

In this lab, your overall goal is to read data from a series of files and directories containing information about the NZ electricity network. This information will then be passed to a plotting function that will display a map of the network.

In order to complete this objective, you have been provided with the following files:

- an example dataset in `example_file.txt`
- partially completed classes and methods in `datalab_functions.py`
- an example network structure in `network.txt`
- data summarising the NZ electricity network in `nz_network.zip`
- some practice exercises in `datalab_practice.py`
- an implementation file to generate the NZ electricity network in `datalab_NZnetwork.py`

Your **main tasks** will be to complete the methods `add_node`, `join_nodes`, and `read_network` defined as part of the `Network` class. Although these methods initially appear in the non-assessed practice exercises, they must eventually be completed as part of the assessed exercise. You will also be completing the `read_network` method defined as part of the `NZNetwork` class. These methods can be located in `datalab_functions.py`.

Practice exercises

In your IDE, open `datalab_practice.py` and read the commented instructions. Complete the exercises before moving onto the assessed task in `datalab_NZnetwork.py`.

Reading data quickly using the `genfromtxt` command

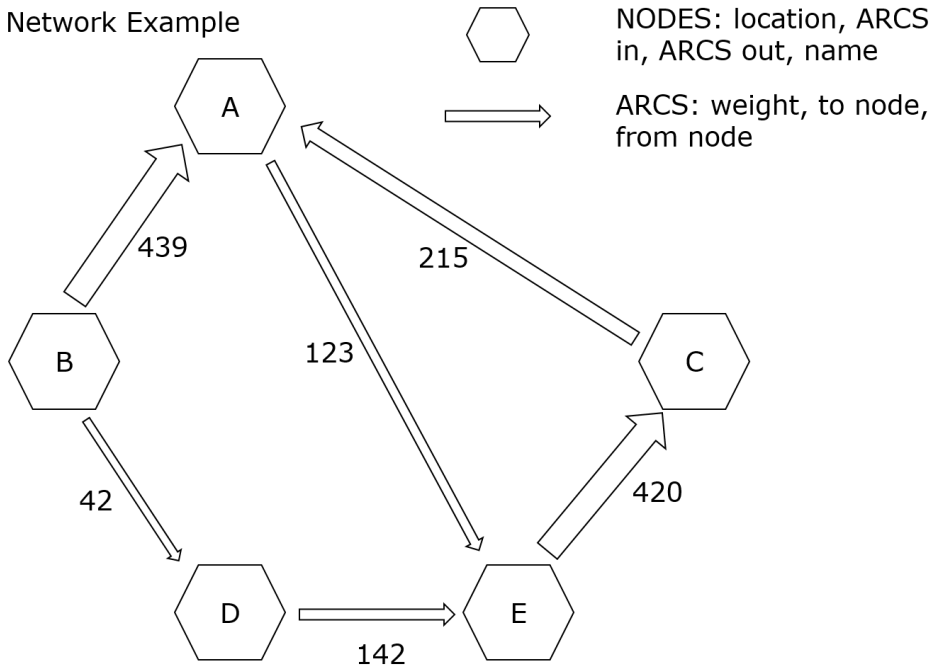
In Section 1 of the `data.ipynb` notebook, we saw how a text file could be opened, read, and its contents interpreted in terms of data and metadata.

RESEARCH how to use the function `np.genfromtxt` to read data from `example_file.txt`, returning vectors `xs` and `Ts`. You may want to investigate further the role of the `delimiter` and `skip_header` arguments passed to `np.genfromtxt`.

Networks

We can think of a network as a collection of *nodes* linked together by *arcs*:

- Each arc has associated with it a *weight*, which could be, say, the capacity of that network connection. An arc points *from* one node *to* another, i.e., these arcs are *singly directed*.
- Each node has associated with it a *name* that identifies it, a *value*, a list of arcs *entering* the node, and a list of arcs *leaving* the node.



Representing networks in a computer program

We want to express the concepts above - nodes, arcs, weights - using computational structures like objects, attributes and methods. To do this, you have been provided pre-prepared scaffolding - a set of partially completed classes and methods - in the file `datalab_functions.py`.

Open `datalab_functions.py` and inspect the method `add_node` defined in the `Network` class. Complete the `add_node` method.

This will require you to **CREATE** an empty node object and then **ASSIGN** values to its attributes. There is a space to write pseudocode for these steps if you wish, or simply complete the relevant commands. Note the hints provided if you are having difficulty.

When you have completed `add_node`, *test* its functionality by running `datalab_practice.py` and checking that you pass the two Part 1 `assert` commands. If your code fails the asserts, use the debugger to investigate the error.

Complete the `join_nodes` method.

The code your write here will be similar to the previous method. You will need to **CREATE** an empty arc object and **ASSIGN** values to its attributes. In addition, you will need to ensure that the arc object is **SAVED** to the Network object (i.e. `self`). You will also need to modify the input node arguments so that they are **ASSOCIATED** with the arc you have created.

It may help to draw a picture to understand how and with which attribute each item relates to the others. The Network object **OWNS** the arcs and the nodes. Arcs **KNOW** about the nodes they link (to and from). Nodes **KNOW** about the arcs that link with them (into and out of).

When you have completed `join_nodes`, test its functionality by checking that you pass the Part 2 `assert` commands. If your code fails the asserts, use the debugger to investigate the error.

Complete the `read_network` method for the `Network` class.

Considerably more of this method has been written for you, including some pseudocode and a `while` loop. You will need to:

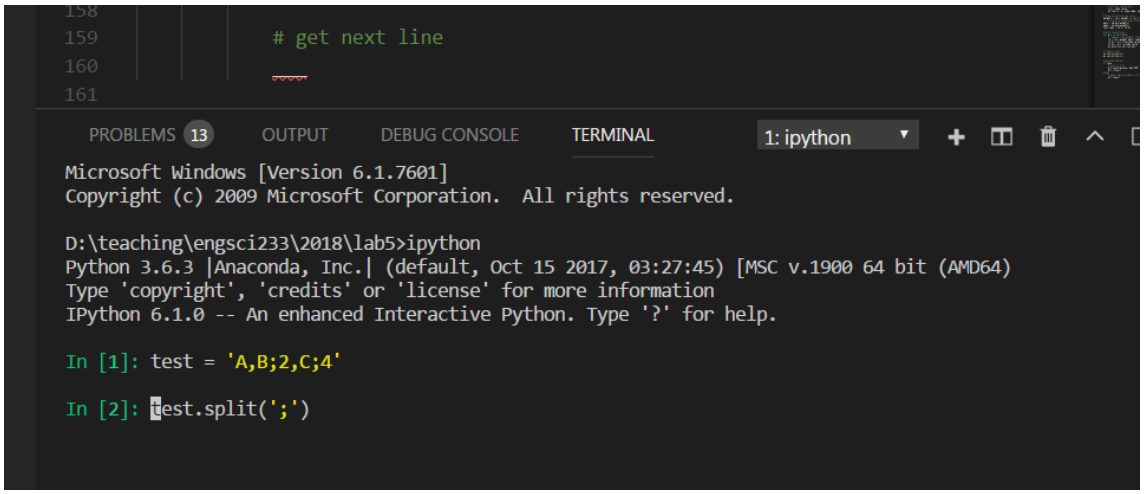
- Split individual strings, corresponding to each line of the network file, to get

their information.

- Add new nodes (this one already done).
- Get a node (an object) from a node name (a string).
- Join nodes to create arcs.
- Get the next line in the network file.

If you're unfamiliar, then it can be difficult to know how a particular command will work. It is often useful to have some test environment in which to experiment.

In VS Code, click on the *TERMINAL* tab, just to the right of *DEBUG CONSOLE*. This is like a Windows Command Prompt, and we can **open a Python Interpreter** by running the command `ipython`. A Python interpreter works in a similar way to the MATLAB workspace. You can execute individual commands and inspect variables.



Try running the commands

```
test = 'A,B;2,C;4'
test.split(';')
test.split(',')
test.split(',')[0]
```

To exit the interpreter, type `exit`.

Once completed, you can test your implementation of the `read_network` method by running the Part 3 commands. In particular, the `network.display()` command should print the screen output below

```
network has 6 nodes: A, B, C, D, E, F,
A --> B with weight 2
A --> C with weight 4
B --> C with weight 1
B --> D with weight 4
C --> D with weight 2
C --> E with weight 1
D --> E with weight 2
D --> F with weight 2
E --> F with weight 3
```

Assessed Exercise

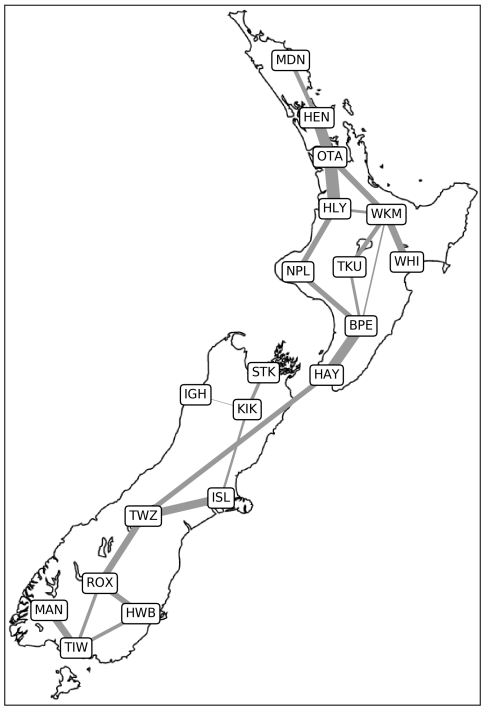
For this task, you will need to read data from a series of files and directories containing information about the NZ electricity network. The nodes of the network represent locations in NZ where electricity is generated or consumed. The arcs are high-voltage power lines that link the nodes, with the arc weights representing the amount of current usually carried by that line.

You will be working on the `read_network` method of the `NZNetwork` object. This object is a **derived class** of the `Network` object, which means it keeps all the same methods of `Network` (including `add_node` and `join_nodes` that you defined earlier) **EXCEPT** for the ones that you choose to overwrite. In this case, you will be overwriting or "overloading" the `read_network` method.

Because ***`add_node`*** *and* ***`join_nodes`*** *will be required when implementing* ***`read_network`****, all three methods form part of the assessment.*

Extract the contents of `nz_network.zip` and inspect the files and directory structure. Make a plan (pseudocode) for how you are going to read this information into the `NZNetwork` object. Finally, write code to implement your `read_network` method.

If you're getting stuck, check the suggestions at the bottom of `datalab_NZnetwork.py`. You will know that your method is *working correctly* if you are able to generate the file `datalab_network.png` looking the same as the plot below.



Submission Instructions and Rubric

For this lab, you should upload the following files to Canvas:

- `datalab_functions.py`
- `datalab_network.png`

DO NOT modify the name of these files. **DO NOT** submit any other files (e.g. you do not need to submit `datalab_NZnetwork.py` or `datalab_practice.py`). **DO NOT** put your submission in a zip archive.

Lab assignments are due by the time advertised on Canvas.

The marking rubric is available on the Canvas assignment page.

Finally, all submissions are compared against each other and those from previous years for similarity. Copying someone else's code and changing the variable names constitutes academic misconduct by both parties.