Course Scheduler

Credit for the inspiration of this project goes to Sergey Karayev from Gradescope

Description

It's hard coming up with a course schedule for your full-tenure at Make School! You've got all these requirements to satisfy before graduating, and you have to take certain courses in a particular order due to prerequisites. Doing this manually could be incredibly frustrating, and also doesn't scale (what if MS adds another track? Or another 3?).

In this project, you will solve this manual headache by creating a system that can do this for any track for any number of courses! To do this, there are 3 parts of this project to complete:

1. **Create a graph data structure**. First you must build a graph data structure that supports nodes and directed edges. Nodes will have data associated with them, and can have 0-infinite edges. Edges will be directed, meaning they can only be traveled in one direction
2. **Create a directed graph based on a JSON file:** given a JSON file of course names and prerequisites, generates a graph where prerequisites are connected by directed edges.
3. **Write a function that given a graph and a course name, finds how many prerequisites it has**. This will be done by traversing the graph!
4. **Stretch Challenge (Optional)**: given a JSON file of course names and prerequisites,generate a schedule by printing the names of courses in a prerequisite-satisfying order.

Learning Outcomes

By the end of this project, you will be able to:

1. Build a graph data structure from scratch
2. Convert a JSON file into a graph
3. Implement graph traversal

Requirements

1. Graph data structure must have the following:
    a. A creation method
    b. Nodes
        i. Must contain data and edges properties
    c. Directed Edges
        i. Must contain the nodes it connects as properties

2. Write a function `jsonToGraph` that takes a JSON file as input, and generates a directed graph of courses and their prerequisites
3. Must write a function `numPrereqs` that takes a graph and a course name as input, and returns the number of prerequisites that course has
4. (Optional) write a function called `scheduler` that takes a JSON file as input, and prints out a list of courses in the order that they should be taken such that prerequisites will be satisfied
5. (Optional) `scheduler` can detect if there are prerequisites that are cyclical (i.e. CS 1.0 requires CS 1.1 as a prerequisite, but CS 1.1 also requires CS 1.0 as a prerequisite) and will return an error if one or more are found.

Examples

JSON

Below are examples of what the JSON files should look like:

```
[
   {
      "name": "CS 1.0",
      "prerequisites": []
   },
   {
      "name": "WEB 1.0",
      "prerequisites": []
   },
   {
      "name": "WEB 1.1",
      "prerequisites": ["WEB 1.0"]
   },
   {
      "name": "FEW 1.2",
      "prerequisites": ["WEB 1.0"]
   },
   {
      "name": "FEW 2.3",
      "prerequisites": ["FEW 1.2", “WEB 1.1”]
   }
]
```

`numPrereqs` Example Output

If given the JSON in the above example, the `numPrereqs` function would take the graph-equivalent of the JSON file as input, and “FEW 2.3” as input, it would output “FEW 1.2, WEB 1.1”

Stretch Challenge Example Output

If given the JSON in the above example, the `scheduler` function would print out the following:

CS 1.0
WEB 1.0
FEW 1.2
WEB 1.1
FEW 2.3

Rubric

Rubric Item	**Points**
Node data structure with working properties	10
Edges work as expected	10
Graph data structure works as expected	30
`jsonToGraph` works as expected	20
`numPrereqs` works as expected	30
	Total Points: 100

Commit Requirements

This project should have ~10 commits

Resources

- Folder of JSON files to use for testing
- Khan's Algorithm (for stretch challenges)