Project 6: Weighted Digraphs

Due: Friday April 13, 11:59 pm

1 Assignment Overview

For this project you will be implementing a weighted directed graph (digraph). As a directed graph, connections are not symmetric; an arc from u to v does not imply the existence of an arc from v to u. This graph will have pathfinding capabilities; it is able to find the minimum weight path between two vertices.

2 Assignment Deliverables

You must turn in completed versions of the following files:

- Digraph.py

Be sure to use the specified file name and to submit your files for grading via **D2L Dropbox** before the project deadline.

3 Assignment Specifications

Your task will be to complete the methods listed below:

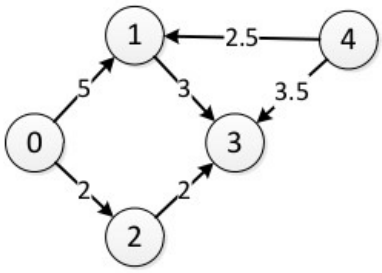
- `insert_arc`: Adds an arc of the given weight between the two vertices.
- `out_degree`: The number of arcs leaving the given vertex.
- `are_connected`: Checks whether an arc exists between two vertices.
- `is_path_valid`: Determines whether an arc exists along each step of the path.
- `arc_weight`: Finds the weight of an arc between two vertices.
- `path_weight`: Finds the sum weight of a directed path.
- `does_path_exist`: Checks whether a directed path exists between two vertices.
- `find_min_weight_path`: Finds a path of minimum sum weight between two vertices.

You may use any of the collections classes included in the default Python distribution that you need to store your digraph. Your digraph will have two member variables: `order`, which is the number of vertices, and `size`, which is the number of arcs. Your `insert_arc` method should update the `size` member. You may add other member variables in your constructor. The memory requirements for your digraph must be $O(|V|+|A|)$. Only the constructor and the `insert_arc` method may modify your graph. You will lose points if any of the other methods modify your graph.

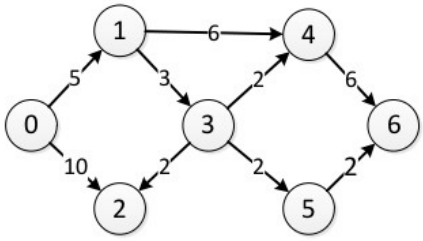
Your `insert_arc`, `out_degree`, `are_connected`, and `arc_weight` functions should all run in constant time. `is_path_valid` and `path_weight` should run in time linear in the length of the input list. Each of

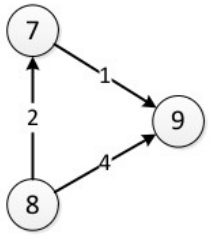
```
5
0 1 5.0
0 2 2.0
2 3 2.0
1 3 3.0
4 1 2.5
4 3 3.5
```

(a) g1.txt



(b) Visual representation of g1





(c) Visual representation of g2

Figure 1: Sample Input

them should also use a fixed amount of extra memory. The `does_path_exist` method should use at most $O(|V|+|A|)$ time and $(|V|)$ extra memory. The `find_min_weight_path` method should use $O(|A|+|V|\log|V|)$ time and $O(|V|+|A|)$ extra memory.

Each of your methods should raise an `IndexError` if one of the vertices is not in the graph. You may assume that the paths input into `is_path_valid` and `path_weight` are non-empty (they contain at least one vertex). You may not assume that all of the arcs in the path exist. Both `arc_weight` and `path_weight` must return infinity if an arc is missing. Your `find_min_weight_path` should raise a `ValueError` if no path exists. No other exceptions should be raised by your program.

You should include comments where appropriate. In particular, you should describe the method used for storing arcs. You must also add documentation for each of the above methods (as well as for any helper functions created) in the same style as for the pre-defined methods.

4 Assignment Notes

- Points will be deducted if your solution has warnings of any type.
- You have been supplied with stubs for the required methods. You must complete these methods to complete the project. You may add more functions than what was provided, but **you may not modify the signatures of the provided methods**.
- You do not have to worry about error checking for valid input. You may assume that the supplied reader function correctly reads the input file.
- You **do** have to worry about accessing elements from an empty map.
- You have been provided with some unit tests that can be used to assess your solution. There are additional test cases that will be kept hidden.
- It is your responsibility to check for potential edge cases. The supplied unit tests do not account for all possible valid inputs
- Several sample graphs have been provided. Visualizations for two of them are shown in Figures 1b and 1c above.
- You may use a wide variety of collections classes. You may want to use `list`, `set`, `dict`, `heapq`, `deque`, or `defaultdict`.