Freie Universität Berlin
FB Mathematik und Informatik
WS 2021

Prof. Dr. Ralf Borndörfer
Ricardo Euler

Optimization I

Programming Exercise 1

Submission: Sunday 14th November, 2021, 23:59 CET via Whiteboard

Write Python programs `dijkstra.py` and `bellman_ford.py` where you implement Dijkstra's algorithm and the Bellman-Ford algorithm, respectively, for directed and arc-weighted graphs. The idea is not to use any external packages[1]. In particular, you need to write a DiGraph class for basic graph attributes and methods on directed graphs.

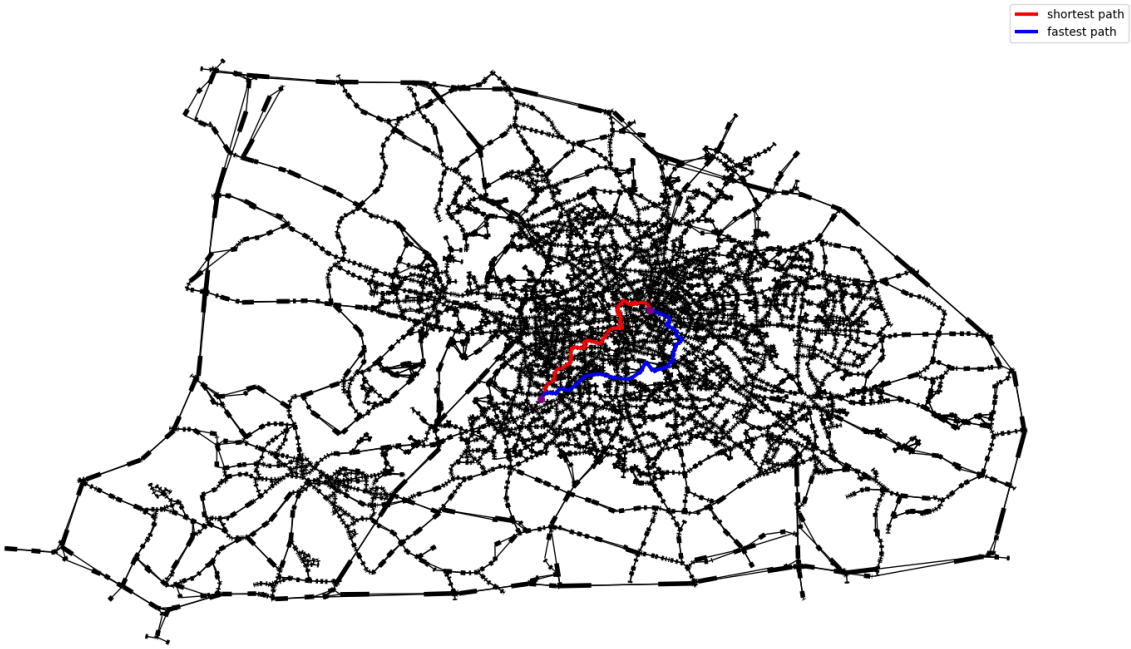
Also, you should write a test script `test_shortest_path.py` that builds the graph, runs both algorithms, and measures the respective running times.

Data:
Graph data files are provided in the Whiteboard (KVV). The lines specify only the arcs of the graph. A list of nodes is not given separately, but we assume instead that they are given implicitly as head and tail nodes of the arcs.

While the first three instances are small exemplary networks, `graph4.txt` contains all main roads of Berlin[2] as shown in the figure below.

instance	#nodes	#arcs	s	t
graph1.txt	8	12	1	8
graph2.txt	8	12	1	8
graph3.txt	8	12	1	8
graph4.txt	12981	28376	7743	5983

[1] but feel free to use everything from the Python Standard Library
[2] simplified version of the Berlin-Center instance from https://github.com/bstabler/TransportationNetworks



Input/Output:
Your program should be able to accept calls of the form

```
python test_shortest_path.py graph1.txt 1 8
```

This call should start Dijkstra's algorithm on the specified digraph and compute a shortest path from node 1 to node 8. The resulting output of this call should be like

```
Dijkstra:
A shortest 1-8-path is
1, 5, 6, 2, 7, 8
with a total length of 11.0.
The computation took 0.0 seconds.

Bellman-Ford:
A shortest 1-8-path is
1, 5, 6, 3, 4, 8
with a total length of 11.0.
The computation took 0.0 seconds.
```

Presentation of your programs:
A single group member should be prepared to present the key components of your code in a 15 minute code review. You should bring your own laptop and demonstrate that your code runs correctly for all given instances.

The code presentations will take place in the week 15. – 19. Nov. We will propose time slots for the groups in advance.

Bonus puzzle for those looking for a challenge:

Come up[3] with an algorithm to compute the second shortest s-t-path in a digraph with non-negative arc weights and implement your algorithm as well. Maybe you can use your Dijkstra implementation as a subroutine.

[3] i.e., do not google it as this takes away all the fun