Python Lab 7

Program Assignment: Shortest path algorithm

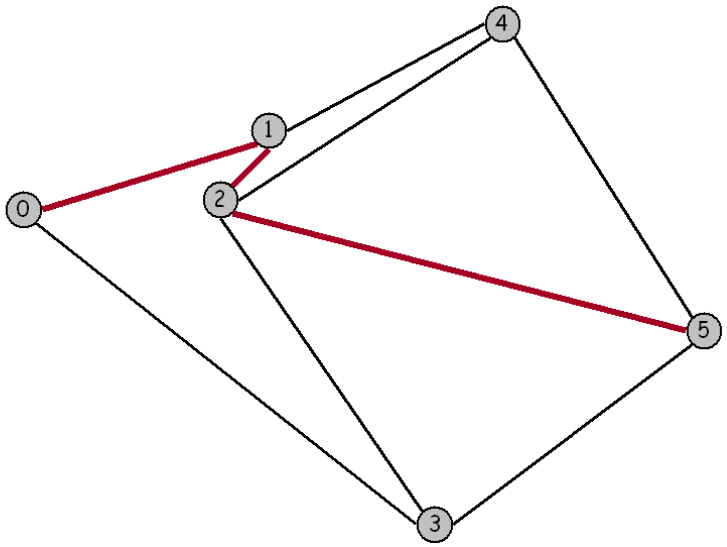
Implement both of ***Dijkstra's Shortest path*** and ***Bellman-Ford algorithms***.

The figure lists the number of vertices and edges, then list the vertices (index followed by its x and y coordinates), then list the edges (pairs of vertices), and finally the source and sink vertices.

For this assignment, we will be working with *maps*, or ***graphs*** whose vertices are points in the plane and are connected by edges whose weights are Euclidean distances.

```
6 9
0 1000 2400
1 2800 3000
2 2400 2500
3 4000    0
4 4500 3800
5 6000 1500

0 1
0 3
1 2
1 4
2 4
2 3
2 5
3 5
4 5
0 5
```



Your program will generate a executable program: FindPath, and it will take three command line arguments. The first argument (D or B) represents either Dijkstra or Bell-Ford algorithm.

Then the remaining two arguments represent the source node and destination node respectively. For example, to find the shortest path from node 0 to node 5 using Dijkstra algorithm:

% FindPath D 0 5

Your program will print the following message:

> **The distance from node 0 to node 5 using Dijkstra algorithm is $$$**
>
> **The execution time takes $ seconds.**
>
> **The shortest path from node 0 to node 5 is: 0, 1, 2, 5.**

Your goal is to **reduce the amount of work** involved per shortest path computation, without using too much additional space.

Tip: Fibonacci heap

Testing: The file usa.txt contains 87,575 intersections and 121,961 roads in the continental United States. The graph is very sparse - the average degree is 2.8. Your main goal should be to answer shortest path queries ***quickly*** for pairs of vertices on this network. Your algorithm will likely perform differently depending on whether the two vertices are nearby or far apart. We provide input files that test both cases. You may assume that all of the x and y coordinates are integers between 0 and 10,000.

Grades: Your works will be tested in the same environment with various inputs. Grades is evaluated by ***correctness*** and the ***efficiency*** ***(the amount of time)*** of your program.

Top 20% : 95
20% ~ 40% : 90
40% ~ 60% : 85
60% ~ : 80