greedy-algorithm

Team 1

We implement Breath-first search, Depth-first search and then our modified DFS with timsort.

Runnig

This program requires Phython 3.6.0+

If you are in the of the program directory you can just do `python main.py`. It outputs the path taken by each algorithm for each of the three graph that are read form the `./test` directory.

Input

We represent a graph using an adjacency list, i.e a pair of vertices represents an edge and a cost for the path.

Below is a simple graph:

```
a,b,.5
b,c,.6
c,f,.7
a,d,.1
d,e,.2
e,f,.4
```

The program takes a graph as a file. Few examples can be see in the `./test` directory which it currently reads form.

Depending on which file you want to use, you call `handle_graph_file("test/simple_one.graph")`. This loads the contents of the file `test/simple_one.graph` as a graph

Then you call the function you want in the main method like `depth_first_search(Vertex("a"), Vertex("f"))`. Here depth_first_search refers to our implementation of the DFS algorithm and prints out a path from "a" to "b" based on the graph we loaded in.

As an example all three algorithms are being called and there outputs printed when you run `main.py`

Output

Below is the result of running all three algorithms on the graph in the **Input** section:

```
+   breadth_first_search   +
BFS Path: ['a', 'b', 'd', 'c', 'e', 'f']
Cost of BFS: 2.0999999999999996
+   depth_first_search   +
DFS Path ['a', 'b', 'c', 'f']
Cost of DFS: 1.8
+   depth_first_search_with_timsort   +
Greedy DFS Path: ['a', 'd', 'e', 'f']
Cost of Greedy DFS: 0.7000000000000001
```

Examples

simple_one.graph

Test File

simple_one.graph

```
`This is a simple graph:
`Each line represents a edge, e.g. the first line represents the edge going from Vertex a being
`connected to Vertex b and that edge have a cost of .5`
a,b,.5
b,c,.6
c,f,.7
a,d,.1
d,e,.2
e,f,.4
```

We also allow comments in the test file using back-ticks(`) as demonstrated above.

Result

```
●■■■■■ simple_one.graph ■■■■■■●
+   breadth_first_search   +
BFS Path: ['a', 'b', 'd', 'c', 'e', 'f']
Cost of BFS: 2.0999999999999996
+   depth_first_search   +
DFS Path ['a', 'b', 'c', 'f']
Cost of DFS: 1.8
+   depth_first_search_with_timsort   +
Greedy DFS Path: ['a', 'd', 'e', 'f']
Cost of Greedy DFS: 0.7000000000000001
```

---

simple_two.graph

Test File

simple_two.graph

```
a,b,.3
a,c,.2
b,d,.1
b,e,.5
c,f,.2
c,g,.1
e,h,.6
```

Result

```
●■■■■■ simple_two.graph ■■■■■■●
++++++breadth_first_search++++++
BFS Path: ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h']
Cost of BFS: 2.0
+++++depth_first_search++++++
DFS Path ['a', 'b', 'd', 'e', 'h']
Cost of DFS: 1.4
+   depth_first_search_with_timsort   +
Greedy DFS Path: ['a', 'c', 'g', 'f', 'b', 'd', 'e', 'h']
Cost of Greedy DFS: 1.4
```

---

simple_three.graph

Test File

simple_three.graph

```
a,b,.5
a,c,.1
b,d,.6
c,e,.2
d,f,.7
e,f,.4
```

Result

```
●■■■■ simple_three.graph ■■■■■●
++++++breadth_first_search++++++
BFS Path: ['a', 'b', 'c', 'd', 'e', 'f']
Cost of BFS: 2.0999999999999996
+++++depth_first_search+++++
DFS Path ['a', 'b', 'd', 'f']
Cost of DFS: 1.8
+   depth_first_search_with_timsort   +
Greedy DFS Path: ['a', 'c', 'e', 'f']
Cost of Greedy DFS: 0.7000000000000001
```