IT 427, Design and Analysis of Algorithms

Programming Assignment 3: Minimum Arborescence

Due date: Oct. 7, 2023, Monday, 11:55 PM 60 points (40 on programs, 20 on report)

For this assignment, you are asked to implement Edmond's algorithm to find an arborescence of a directed weighted graph with respect to a designated vertex as the root.

Program requirement: The name of the program should be `Edmond.py`. Code all needed classes and methods in one program file, i.e., do not create any external class for this project. I will compile and run your program on our Linux server as follows, where `dwg1.txt` is the input file. If your program fails to compile, you will get 0 point. I may test your program on a different graph file.

```
python3 Edmond.py dwg1.txt
```

Input: The format of the input files are similar to the input file for the previous assignment except the graphs are **directed**. There are 7 test files in the public directory. I will run your program on all of them for testing. See the following example (`dwg1.txt`) where a directed edges from vertex x to vertex y with weight w is represented by (x, y, w). Also, note that the edges are not symmetric. e,g. $(0, 1, 0.79261) \in E$ but there is no edge from 1 to 0.

```
10 directed weighted graphs; all are reachable from vertex 0.

** G1:  |V| = 5
(u, v, w) E = {
     ( 0, 1, 0.79261)
     ( 0, 2, 0.48497)
     ( 0, 4, 0.10921)
     ( 1, 2, 0.63677)
     ( 1, 3, 0.72796)
     ( 1, 4, 0.06738)
     ( 2, 1, 0.50480)
     ( 3, 2, 0.01000)
     ( 3, 4, 0.12665)
     ( 4, 1, 0.00789)
     ( 4, 2, 0.43075)
     ( 4, 3, 0.90364) }
---------------

** G2:  |V| = 5
(u, v, w) E = {
     ( 0, 1, 0.62333)
     ( 0, 2, 0.16064)
     ......
     ......
```

Output: The program should list the minimum arborescence rooted at vertex 0 for every graphs in the input file according to the ascending order of x (leaving vertices), and if x are the same, use y (ending

vertices) as the secondary key. See the following sample output, which are the minimum arborescent trees of G_1 and G_2 in `dwg1.txt`, respectively.

```
Minimum Arborescences in dwg1.txt

G1:  |V|=5 ----------------------
   Arborescence --
       1:  (0, 4, 0.10921)
       2:  (1, 3, 0.72796)
       3:  (3, 2, 0.01000)
       4:  (4, 1, 0.00789)
   Total weight:  0.85506 (0 ms)

G2:  |V|=5 ----------------------
   Arborescence --
       1:  (0, 2, 0.16064)
       2:  (2, 1, 0.19761)
       3:  (2, 4, 0.21258)
       4:  (4, 3, 0.27132)
   Total weight:  0.84215 (0 ms)

       ......
       ......

   By Chung-Chih Li
```

One edge per line started by `n:` where `n` is simply the counter to count how many edges have been listed. An arborescence always has $|V| - 1$ edges. After each arborescence is listed, the total weight of the arborescence should be printed as shown. `(0 ms)` is optional, which is the milliseconds used to find the arborescence. At the end of the output, print your name.

Submission: As in the previous programming assignment, prepare your programs on our Linux Server.

- Make a directory `asg3` under your `IT427`, i.e., `~/IT427/asg3/`. All needed files for this assignment should be saved under your `~/IT427/asg3` before run `submit427.sh`.
- Check the contents of my `/home/ad.ilstu.edu/cli2/Public/IT427/asg3` and copy all files to your own `~/IT427/asg3`.

Two parts of submission: Programs (50 points) and Reports (10 points)

Submission details are same as the previous assignment. Run the submission script with the submission number changed to **3**, but you can use the same secret name as follow:

```
bash /home/ad.ilstu.edu/cli2/Public/IT427/submit427.sh peekapoo 3
```

Note that, since I will keep updating `submit427.sh` for different assignment, you have to run the script from my `/home/ad.ilstu.edu/cli2/Public/IT427/` directly for the most recent updated version, i.e., don't copy it to your own directory. Also, once you run `submit427.sh`, don't make any change on the target directory, including compile and run your programs. That will mess up the permission. You will lose significant points if you fail to follow the instructions.

1. Programs: 40 points. Submission on Linux server.

   The score is based on the correctness and the programming style, which includes efficiency, appropriateness of data structures, and documentation of your programs. At the beginning of every program file, put a section of comments including (1) your full name, (2) student ID, (3) a pledge of honesty that you do not copy/modify from other's codes and (4) a declaration of your copyright that no one else should copy/modify the codes. You will receive:

   (a) 95 $\sim$ %: No error with a good programming style.
   (b) 80 $\sim$ %: Minor error and fair programming style.
   (c) 60 $\sim$ %: Some error and not so good but acceptable programming style.
   (d) 40 $\sim$ %: Too many error and bad programming style, but meaningful.
   (e) 20 $\sim$ %: Compilable but not working and the program must show reasonable trying.
   (f) 0 $\sim$ %: : Fail to meet any of aforementioned qualities or plagiarism involved.

2. Report: 20 points. Submission through Canvas.

   You have to write up a report with a cover page and prepare it in pdf format. You don't have to put program output on the report as I will run and exam your program directly on some different input files.

   You report should contain algorithm descriptions, summary of the methods, data structures, and efficiency analysis on time and space in details in terms of big-O notations. If there is any difficulties encountered in this assignment, you can report it.

   Regarding the complexity analysis, you should not simply copy the textbook analysis to your report, because it is not related to your program and the efficiency of your implementation may not meet the theoretical results on the textbook. Thus, you have to point out which part of your program that causes the main time/space complexity bounded by the big-O function in your analysis. For example, if your time complexity is $O(|V| \times |E|)$, clearly indicate where the loops or recursions involving V and E occur and provide an explanation for why multiplication is used instead of addition in the analysis.

   Moreover, every operation has a cost. Unless it is trivial like a simple assignment, which is simply $O(1)$, you have to justify the complexity of such operation. For example, what is the complexity of "finding the minimum weights from a set of edges"? The answer being $O(1)$ or $O(n)$ makes a very different result in this assignment.

   If your analysis is not clearly related to your program and provides sufficient justification, your report score will not be higher than 50%