IT 427, Design and Analysis of Algorithms

Programming Assignment 8: Shortest paths with negative weights.

Due date: November 11, 2024, Monday, 11:59 PM 50 points (40 on programs, 10 on report)

In this assignment, we will revisit the shortest path problem. The difference is that the edges may have negative weights, which obsolete Dijkstra's algorithm. We will have to use a dynamic programming technique to find the shortest path.

Prepare your programs on Linux Server:

- Make your directory ∼`/IT427/asg8/` and all programs and needed files should be saved under this directory before run `submit427.sh` for submission.
- Check the contents of my `/home/ad.ilstu.edu/cli2/Public/IT427/asg8` and copy the graph file `nwdGraphs.txt` to your own directory.

Program requirement: Since Dijkstra's algorithm is more efficient and not all graphs in the input file have negative edges, we want to take this advantage by using Dijkstra's algorithm if no negative edge is found in the graph. In other words, your program is required to use Dijkstra's algorithm for graphs with all positive edges and automatically switch to dynamic programming for graphs with negative edges.

The name of the program should be `shortestPath.py` and I will run your program on our Linux server as follows.

```
python3 shortestPath.py nwdGraphs30.txt
```

If your program fails to run, you will get 0 point. Also, the sole purpose of this assignment is dynamic programming. If you simply implement Dijkstra's algorithm, which is Assignment 2 that you already get credit, you will get 0 point. In addition to the standard input file, I will test your program on some different graph files.

Input: The input file is formatted exactly same as Assignment 2 except the weights of edges that may be negative. Check `nwdGraphs30.txt` from `/home/ad.ilstu.edu/cli2/Public/IT427/asg8`, which contains 30 weighted directed graphs. This file will be the standard input file for you to test, but I will use another test file for grading. Note that, not all graphs have negative edges.

Output: For each graph, the output should show the shortest path from v_0 to v_{n-1} with incremental weight from v_0 to each vertex on the path. The output has to indicate which approach is used for each graph. The results should be shown on the screen in the following format:

```
Shortest Paths from vertex 0 to vertex n-1 in nwdGraphs30.txt, |V|=n

G1's shortest path from 0 to 14:
     Dijkstra's Algorithm
     ( 0, 8, 63.054) --> 63.054
     ( 8, 9, 31.743) --> 94.797
     ( 9, 14, 70.605) --> 165.402


G2's shortest path from 0 to 14:
     Dynamic Programming
     ( 0, 12, -4.456) --> -4.456
     (12, 8, 35.256) --> 30.800
     ( 8, 14, -5.413) --> 25.387


G3's shortest path from 0 to 19:
     Dynamic Programming
     ( 0, 5, -6.763) --> -6.763
     ( 5, 9, 3.461) --> -3.302
     ( 9, 6, -20.162) --> -23.464
     ( 6, 0, -2.712) --> -26.176
     ** 6 ==> 0 Enter a negative cycle.

G4's shortest path from 0 to 19:
     Dijkstra's Algorithm
     ( 0, 9, 81.681) --> 81.681
     ( 9, 6, 8.109) --> 89.790
     ( 6, 19, 21.284) --> 111.074
```

If there is no path from v_0 to v_{n-1}, show the message as follows:

```
G15's shortest path from 0 to 49:
     Dynamic Programming
     *** There is no path.
```

Note: If you check the graphs in `nwdGraphs30.txt`, you will find that G1 and G4 do not have negative edges, and therefore the program will use Dijkstra's algorithm to find the shortest paths as shown above, while G2 and G3 have negative edges, hence the dynamic programming technique is used. Also note that, when a graph have negative edges, there is a chance that the path may enter a negative loop (i.e., the total weight of the loop is negative). If such negative loop exists, we can keep repeating the negative loop to have a however small total weight, which is meaningless. Therefore, the program should detect this situation and abort the search as shown in G3 above.

Submission: Programs (40 points) and Reports (10 points)

Submission details are same as the previous assignment. Run the submission script with the submission number changed to 6, but you can use the same secret name as follow:

```
bash /home/ad.ilstu.edu/cli2/Public/IT427/submit427.sh peekapoo 8
```

Note: 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.

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

2. Report: 10 points. Submission through Canvas.

   You have to write up a report 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.

   The report should includes brief descriptions of your program, 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. If your analysis is not clearly related to your program with sufficient justification, your report score will not be higher than 50%.