2801ICT Computing Algorithms – Assignment 3 (10%)

(May 27, 2019)

This assignment must be done individually. The submission date is as specified in the Course Profile (**May 27, 2019**) and the submission method will be communicated during trimester.

Background

Melanie is a second-year computer science student currently studying at Griffith School of ICT. One day, Melanie's father came to her and told her that he wants to travel from Southport to Brisbane CBD and asks Melanie to recommend exactly K-shortest loopless paths for some reason. Luckily, Melanie is undertaking the "2801ICT: Computing Algorithms" course convened by Dr. Saiful Islam. Though, Melanie knows both Dijkstra's algorithm and Bellman Ford's algorithm very well as these two algorithms have been taught in Week 10, she is not exactly sure how to solve this K-**shortest loopless paths** problem for the Queensland road network graph. She then discussed this problem with Dr. Saiful. Dr. Saiful thinks that this is an interesting problem and it should be solved by each student in his class including Melanie as part of Assignment 3. Melanie's father is happy if and only if the first recommended path is the **shortest path** between Southport to Brisbane CBD and the **rest of the $K-1$ paths are approximated shortest paths** as these paths are his backup paths.

Problem Description

The shortest path problem is about finding a path between **2** vertices in a graph such that the total sum of the edges weights is minimum. This problem could be solved easily using (BFS) if all edge weights were **1**, but here weights can take any value. There are TWO popular algorithms in the literature for computing the shortest paths from the source vertex to all other vertices in a graph: (i) Dijkstra's Algorithm; and (ii) Bellman Ford's Algorithm. The Bellman Ford's algorithm exploits the idea that a shortest path contains at most $n-1$ edges, and a shortest path cannot have a cycle. The Bellman Ford's algorithm works for graphs with negative weight cycle(s). The time complexity of Dijkstra's Algorithm with min-priority queue is $O(V + E \log V)$. On the other hand, the time complexity of Bellman Ford algorithm is relatively high $O(V.E)$, in case $E = V^2$, $O(V^3)$. Both Dijkstra's and Bellman Ford's algorithms unable to solve the this K-shortest loopless paths (KSP) problem. There exist many works in the literature to solve KSP problem. Some useful references are given at the end of this document.

In this problem, you are given 2 integers (N, M), N is the number of vertices, M is the number of edges. You will also be given a_i, b_i, w_i where a_i and b_i represents an edge from a vertex a_i to a vertex b_i and w_i respresents the weight of that edge. Finally, you will be given two vertices s and d, where s denotes the source vertex (Southport) and b denotes the destination vertex (Brisbane CBD) and the value of K.

For each input file, print the length of the K-shortest loopless paths separated by commas and these paths must include the **shortest path between s and b** and the **rest of the $K-1$ paths are approximated shortest paths**.

Sample Input: Consider the following directed graph as an example.

6 9

C D 3

C E 2

D F 4

E D 1

E F 2

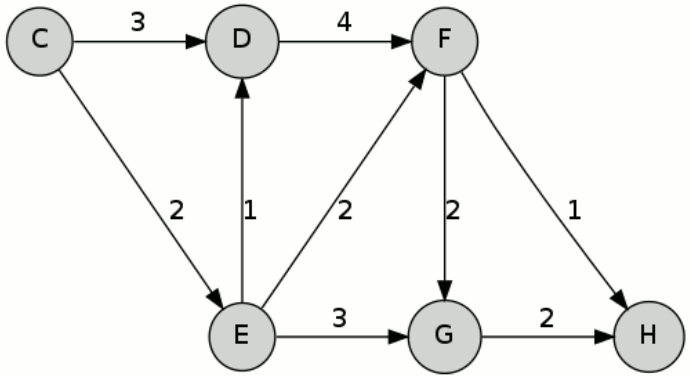
E G 3

F G 2

F H 1

G H 2

C H 3



Sample Output: 5, 7, 8

The 3 shortest paths from source **C** to destination **H** are as follows:

P_1 = C→E→F→H, Cost (P_1) = 5

P_2 = C→E→G→H, Cost (P_2) = 7

P_3 = C→D→F→H, Cost (P_3) = 8

Results for Java Implementation (Yen Algorithm)

Input (k=?)	Output	CPU Time (millisecs)
2	1038.57, 1042.38	1286
3	1038.57, 1042.38, 1043.02	2007
5	1038.57, 1042.38, 1043.02, 1043.29, 1044.05	7767
7	1038.57, 1042.38, 1043.02, 1043.29, 1044.05, 1044.12, 1044.63	10897
10	1038.57, 1042.38, 1043.02, 1043.29, 1044.05, 1044.12, 1044.63, 1044.90, 1045.73, 1045.97	20345

*sample input file (finalInput.txt is uploaded at L@G, you can change the k-value in the last line)

Notes

1. You must design your own algorithm and write the program code submitted. The program must be able to be run from the command line passing the path to the input.txt file as an argument. The naming convention for your program file is: <student_number>_ k_shortest_paths.py (or

<student_number>_ k_shortest_paths.cpp or <student_number>_ k_shortest_paths.java) (without the <>) and try and keep the program within a single file.
2. You must produce a Power-point presentation describing your algorithm (including the pseudocode), innovation, the results of the problem (sample input.txt is provided) and performance analysis.

Marking

This assignment is worth 10% of your final grade. The assignment will be marked out of 100 and marks will be allocated as follows:

1. (**Algorithmic Design**) Overview of the problem, Algorithm Description, Algorithm Pseudo-Code and Innovation (describe how did you innovate): 5+10+10+20**=45 marks**
2. (**Implementation**) Quality of code (e.g. appropriate naming conventions, code comments, class structure, method structure, appropriate use of data structures etc.): **15 marks**
3. (**Result and Algorithm Analysis**) The results of the problem (finalInput.txt is provided) and performance (time-complexity) analysis: 10+20**=30 marks**
4. (**Presentation**) Power-point presentation: **10 marks**

*Innovation - your own design of the algorithm (solution approach) for the given problem. The solution approach must not exist in the literature.

Submission

Student should submit a zip file (<student_number>_Algorithm_Assignment_3.zip) consisting of the following:

1. A document covering 1 and 3: <student_number>_Algorithm_Assignment_3.pdf,

2. Source code file: <student_number>_ k_shortest_paths.py (cpp/java) and

3. A power point presentation file: <student_number>_Algorithm_Assignment_3.ppt (/pptx).

Please note that all submitted assignments will be analysed by a plagiarism detector including the submitted source code of the program. If any form of plagiarism is found, it will be dealt with in accordance with university policy: https://www.griffith.edu.au/academic-integrity/academic-misconduct.

Useful Articles for KSP problems

1. Yen, J.Y., 1971. Finding the k shortest loopless paths in a network. *management Science*, 17(11), pp.712-716.
2. Eppstein, D., 1998. Finding the k shortest paths. *SIAM Journal on computing*, 28(2), pp.652-673.
3. Martins, E.D.Q.V., Pascoal, M.M.B. and Dos Santos, J.L.E., 1998. The k shortest paths problem.
4. Liu, H., Jin, C., Yang, B. and Zhou, A., 2018. Finding top-k shortest paths with diversity. *IEEE Transactions on Knowledge and Data Engineering*, 30(3), pp.488-502.