Concordia University
Dept. of Computer Science and Software Engineering

COMP 691 – Online Algorithms and Competitive Analysis
Winter 2019

$\mathcal{P}$roject

Submission deadline: Monday April 1st at 23:55 **You can work in groups of 2.**

1 Project Description

In this project you will implement the Work Function Algorithm and the Greedy Algorithm for the k-Server problem on graph metrics, evaluate and compare their performance on different inputs and report your findings. In the process you will consult a couple of surveys on the k-Server problem ([2, 1]) and learn more about this problem.

Graph metric is defined based on an undirected and unweighted graph $G = (V, E)$. The distance between two vertices x and y is defined as the length of a shortest unweighted path between x and y.

2 Parameters of the Problem

In this project, you will deal only with the special case of 4 servers, i.e., $k = 4$. The number of vertices in the graph metric is going to be at most 50, i.e., $|V| \leq 50$. The number of requests is at most 1000.

3 Graph Format

In this project you will be reading/writing graphs from files. This requires a certain format. You will be working with Network Repository (`http://networkrepository.com`) – one of the largest freely available online databases of graphs. Thus, you will use the Matrix Market Coordinate (MMC for short) Format to represent undirected unweighted graphs, since this format is most popular on Network Repository. Undirected graphs in this format are represented as follows.

```
% comment line 1
% comment line 2
n n m
u1 v1
u2 v2
...
um vm
```

More specifically, any line starting with % is a comment and should be ignored. The first line that does not start with % contains 3 integers n, n, m. The integer n is repeated twice and stands for the number of vertices $|V|$. The integer m stands for the number of edges $|E|$. The next m lines contain description of edges as pairs of vertices. For example, the following example shows how a graph with 4 nodes and 6 edges $\{1,2\}, \{1,3\}, \{2,3\}, \{1,4\}, \{2,4\}, \{3,4\}$ can be represented as follows:

```
%MatrixMarket matrix coordinate pattern symmetric
4 4 6
2 1
3 1
3 2
4 1
4 2
4 3
```

Note that vertex names are 1-based, i.e., $1 \le u_i \le n$ and $1 \le v_i \le n$. Also note that each edge appears only once as a pair (u_i, v_i). Since we are working with undirected graphs you should remember to add two pairs (u_i, v_i) and (v_i, u_i) to your datastructure, if that's how you are representing the graph.

More information on this format can be found at

`http://networkrepository.com/format-info.php`

Lastly, we note that not all files on Network Repository follow this format. If there is a graph from Network Repository that you want to use in this project but it's in the wrong format, then you have to either manually convert it or write a program to convert it to this format. Alternatively, you can simply find a graph that satisfies this format.

4 Implementation Details

You must use one of the following programming languages: C, C++, Java, Python. You are allowed to use only the libraries that come together with the standard distribution of a compiler/interpreter (for example, STL for C++ is allowed). Anything else has to be implemented by you.

You are free to structure your implementation the way you want, but you must have the following functions:

computeDistancesFromFile(*filename*). Reads graph information from the file *filename* in the MMC format and computes $|V| \times |V|$ array D of pairwise distances. In other words, $D[x][y]$ = length of shortest path between x and y.

computeOPT(D, r, C_0). Input consists of two arrays and an initial configuration C_0. The first array D is the array of pairwise distances that would be produced by computeDistancesFromFile() function. The second array is the sequence of requests r such that

$r[i] \in \{1, 2, \ldots, |V|\}$ is the i^{th} request. This function should return the *value* of OPT on this sequence starting in configuration C_0. You must use work function to compute OPT.

computeWFA(D, r, C_0). The inputs for this function are the same as for the computeOPT() function. This function should return the *value* of the Work Function Algorithm on the sequence r starting in configuration C_0.

computeGreedy(D, r, C_0). The inputs for this function are the same as for the computeOPT() function. This function should return the *value* of the Greedy algorithm on the sequence r starting in configuration C_0.

The signatures of the above functions can be different from the suggested above. If you would like to pass some auxiliary information around, you can add it as an additional parameter. If you would like to return more than one thing, you can also do that. In addition to the above functions you will probably find it useful to write many helper methods.

Source code should be well documented with proper comments and names of variables. The code should be easy to read. I will be checking the code for correctness and readability.

In addition to the above methods, there must be a way to run your code so that it generates *ALL* the results that will appear in your writeup. You can do this by writting a small wrapper around the four methods mentioned above. I should be able to receive your submission, read the README file, understand how to compile your code (if there is a need for it), then run a single command, e.g., */.a.out*, and it would reproduce all figures and data used in your writeup. If your code doesn't compile, or doesn't reproduce similar results, you will lose a significant portion of the points.

5 Evaluation Details

Your evaluation will consists of two parts.

First part is evaluation of algorithms with respect to *random* graph metrics. The random Erdős-Rényi graph is denoted as $G(n, p)$ and is generated as follows. The set of vertices V is fixed to be $\{1, 2, \ldots, n\}$ and the set of edges E is generated randomly. Each edge $\{i, j\}$ is included in E with probability p and excluded from E with probability $1 - p$. In particular, $G(n, p)$ graphs have $\binom{n}{2}p$ edges on average.

You should run your implementation on $G(n, p)$ with different values of $n \leq 50$, different values of p, and different initial configurations C_0. You should also consider different ways of generating input sequences. Some suggestions: (a) generate the input sequence by sampling each new request uniformly at random from $[n]$ and independently of all other requests; (b) generate the input sequence randomly but with some dependence (maybe the input sequence arises out of some Markov chain process); (c) generate the input sequence by picking 5 nodes from V uniformly at random and always requesting one of the 5 nodes such that Greedy does not have a server present at the requested node at the time of the request; (d) anything else you might want to try. For each value of p, n, and initial configuration, you might need to run your implementation multiple times until the sample average of the results starts to stabilize. Observe if there is any relationship between the performance of the algorithms Greedy, WFA, and OPT and p, or n, or the choice of initial configuration, or the choice of the way to generate the input sequence, or the length of the input sequence. Summarize your results in a figure as time series or in a table. There is a lot of freedom in how you can choose your experiments and how you can run them.

For this part, you might want to write an analogue of computeDistancesFromFile function that computes all pairwise distances based on your internal datastructure representation of the graph, as opposed to reading it from file.

Second part is evaluation of algorithms with respect to benchmarks from Network Repository (`http://networkrepository.com`). Find 5 graphs on the Network Repository with at most 50 and no less than 20 nodes. Perform similar experiments as for the Erdős-Rényi model, but with respect to the 5 graphs you found on the network repository. Report your findings in a figure or a table.

6 Writeup Details

Write a report of at least 5 and no more than 7 pages (not including title and bibliography) explaining what you have done, what difficulties you had during implementation, results that you found. You can also suggest improvements to WFA and future progress. Your writeup must have the following sections:

Title, Author(s) Name(s), Date. Self-explanatory.

Introduction. Brief description of the problem, its significance, the main message of the paper, and briefly mention the key results.

Background. Contains some history about the k-Server problem. Koutsoupias survey [2] is helpful for this.

Implementation details. This is where you describe important implementation details. You don't have to explain every aspect of your code. You only have to describe the structure of your code and what were the most challenging aspects of the implementation and how you overcame them.

Experimental Setup. This is where you explain details of your experimental setup. What architecture did you use? What versions of compilers/interpreters? How fast was the CPU? How many cores? How much RAM? What graphs did you choose for the evaluation? Why?

Results. This is where you present your findings, time-series plots, tables, and comment on what you see. You need to analyze how the performance of WFA compares to Greedy, any patterns that you found, any general observations. The deeper and the more interesting observations the better your grade will be.

Future directions. This is where you explain in which direction you can extend this project. Mention interesting open problems, ideas you came across, etc. If you had more time, what things would you have liked to try?

Bibliography. Cite anything that you have used, including survey papers and Network Repository.

The writeup must be typed up and must be in the PDF format. You are strongly encouraged to use LaTeXfor that, although MS Word or other word processing software can also be used for this purpose.

Just as a guide, you could allocate page limits to the sections as follows: Intro and Background together occupy 1 page, Implementation Details and Experimental Setup together occupy 1 page, Results occupies 2.5 pages, and Future Directions occupies 0.5 pages.

7 What to Hand In

You are allowed to work in groups of 2. If you are working in a group, **exactly one person** from the group must submit the assignment. DO NOT MAKE 2 SUBMISSIONS FOR THE SAME GROUP.

You should create a single .zip archive and upload it to Moodle before the deadline. The file must contain the following items:

1. Source code – could be a single file or multiple files if you decided to split your implementation into several modules.
2. Files from the network repository. These are the graphs that you used as benchmarks for your evaluation in addition to $G(n,p)$.
3. PDF of the writeup.
4. README file – this should be a text file explaining the structure of your submitted .zip archive and instructions on how I can reproduce your results.

Submissions significantly deviating from the above standard could lose points.

References

[1] Aris Floratos and Ravi Boppana. The on-line k-server problem. , Courant Institute of Mathematical Sciences, NYU, 1997.

[2] Elias Koutsoupias. The k-server problem. *Comput. Sci. Rev.*, 3(2):105–118, May 2009.