TCSS 543A: Advanced Algorithms
Empirical study project on network flow
Group Membership Due: Friday November 17, 2023 at 1:30pm
Informal Status Update Due: Monday Dec 4, 2023 at 1:30pm
Slides and Presentations Due: Wednesday Dec 6, 2023 at 1:30pm
Code and Documentation Due: Friday Dec 8, 2023 at 1:30pm
Group Report Due: Sunday Dec 10, 2023 at 1:30pm

1 Overview

Problem description In this project, you will be conducting an empirical study to see which of the network flow algorithms is better than the others. It is unlikely that you will discover that one algorithm is better than the others for all inputs. In fact, our goal in this project is to see if you can figure out for what kinds of graphs one algorithm does better than the others. The algorithms you will implement and run experiments on are: the Ford-Fulkerson algorithm (Ch 7.1), the scaling Ford-Fulkerson algorithm (Ch 7.3), and the preflow-push algorithm (Ch 7.4). You are expected to implement the algorithms yourself, as opposed to using existing code that you might have found online.

Logistics Each project team should have 3-4 members. Note that I reserve the right to assign different grades to students in the same team based on the division of labor. The project counts for 25% of your final grade. The project will be graded out of a total of 25 points.

- Group membership: 1 point
- Status update: 1 point
- Slides, presentation and demo: 6 points
- Code and documentation: 7 points
- Report: 10 points

Code You can choose to do this project either in Java or Python. Java code for input graph generation and data structures are provided on Canvas. You are free to choose and/or modify the code provided. No code is available for Python. In order to compare the performance across different network flow algorithms, you are required to use the same programming language (Java or Python) in all your benchmarking experiments.

Input graphs You might consider the following when generating input graphs:

1. Random graphs (each edge is generated with probability p).
2. Mesh graphs.
3. Bipartite graphs.
4. The range of capacities on the edges.

2 Timeline

Fri Nov 17 at 1:30pm Submit group membership. One submission per group.

Mon Dec 4 at 1:30pm Informal project update and feedback opportunity.

Wed Dec 6 at 1:30pm Slides, presentation and demo due via Canvas. One submission per group.

Fri Dec 8 at 1:30pm Code and documentation are due via Canvas. One submission per group.

Sun Dec 10 at 1:30pm Your group report due via Canvas. One submission per group.

3 Deliverables

Informal Status Update; Monday Dec 4 Each team is encouraged to give a quick status update (no slides required), and ask questions during class on December 4. Your team can choose a representative to give the update in class. The goal of this status update is to help each other out and to clarify any doubts about the project.

Slides, presentation and demo (1 upload per team; Dec 6, 1:30pm) Post your slides and video presentation on the Canvas Discussion board. The idea is that you can learn from each other by watching other group's presentation. Make sure to include a demo that shows how you run the code in your presentation video. Document the division of labor (i.e. who did what) on your slides.

Code and documentation (1 upload per team; Dec 8, 1:30pm) To be submitted on the course Canvas. Your network flow algorithms must be implemented in Java or Python. Submit a `zip` file.

- Submit everything we need to run your code. If your code is in Java, you are encouraged to submit source code, compiled `.class` files and makefile.
- The code needs to be heavily documented explaining each approach, with the appropriate input and output. Conclusion should be well explained based on the results obtained at the end.
- Generate input graphs and submit your input testing files.
- Submit an additional documentation file in PDF. Explain clearly in your documentation **how to compile and/or run your code**. Document the version of software and the OS you use to test the code. In addition, your documentation should include the structure of your code. Name each routine and describe in 1-2 sentences what each routine does. This file should also show the output of your code using each of your input testing files.
- Your implementation should output the value of the flow of the input graphs using each of the three methods.
- Add screenshots of your output as part of the zip files with detailed explaination of each output

In addition to your own input testing files, we will test your implementation using additional input files. Points will be deducted if your submission does not follow the above specifications.

Group report (1 upload per team; Dec 10, 1:30pm) In addition to your code for your empirical study, you are expected to write a report on your work. The submitted report should be typeset using any common software and submitted as a PDF. The report is not quite an academic paper, but it follows roughly the same format. Your report should be divided into sections as follows:

- Introduction: A description of the problem (network flow), what the goals of the study are, and a brief description of the results.
- Methodology: A description of the code, what algorithms you implemented, implementation difficulties, etc. You should also describe how you tested your code, how you generated your input instances (e.g., how you generated your graphs), etc. For example, for network flow, how did you choose your s and t? You should also say something about how many times you ran your algorithm on input instances, what range of graph sizes (number of vertices and number of edges) you used, and how you systematically investigated the algorithms.
- Results: What did you discover about your algorithms? Which was faster? Was one or more algorithm more sensitive to the number of edges in the graph, the number of vertices in the graph (or in network flow, the value C)?
- Future work: If you were to do the experiments over again, what would you do differently? What future experiments can you think of that might be interesting, given what you learned from this project? Are there any applications or data that you would be interested to apply your implementation to?
- Division of labor: Describe the parts of the project each team member worked on (which algorithm(s) you implemented, which input graphs you tested, who you worked with on what parts, etc.) and what parts of the project your teammates worked on.
- Lessons learned: What you learned from the project.
- References: Make sure to cite the source of your materials, including the textbook. If you get inspired by other group's presentation, please document and cite here.

You are recommended (but not required) to use the ACM template. Formatting guidelines: up to 6 pages, double column, ACM Proceedings format available at `https://www.overleaf.com/latex/templates/association-for-computing-machinery-acm-sig-proceedings-template/bmvfhcdnxfty`. In case you need more than 6 pages, consider splitting your material in a main paper and an appendix.