CS 630, Fall 2024, Homework 1
Due Wednesday, September 18, 2024, 11:59 pm EST, via Gradescope

Homework Guidelines

Collaboration policy Collaboration on homework problems, with the exception of programming assignments and reading quizzes, is permitted, but not encouraged. If you choose to collaborate on some problems, you are allowed to discuss each problem with at most 5 other students currently enrolled in the class. Before working with others on a problem, you should think about it yourself for at least 45 minutes. Finding answers to problems on the Web or from other outside sources (including generative AI tools or anyone not enrolled in the class) is strictly forbidden.

You must write up each problem solution by yourself without assistance, even if you collaborate with others to solve the problem. You must also identify your collaborators. If you did not work with anyone, you should write "Collaborators: none." It is a violation of this policy to submit a problem solution that you cannot orally explain to an instructor or TA.

Typesetting Solutions should be typed and submitted as a PDF file on Gradescope. You may use any program you like to type your solutions. LaTeX, or "Latex", is commonly used for technical writing (`overleaf.com` is a free web-based platform for writing in Latex) since it handles math very well. Word, Google Docs, Markdown or other software are also fine.

Solution guidelines For problems that require you to provide an algorithm, you must provide:

1. pseudocode and, if helpful, a precise description of the algorithm in English. As always, pseudocode should include
   - A clear description of the inputs and outputs
   - Any assumptions you are making about the input (format, for example)
   - Instructions that are clear enough that a classmate who hasn't thought about the problem yet would understand how to turn them into working code. Inputs and outputs of any subroutines should be clear, data structures should be explained, etc.

   If the algorithm is not clear enough for graders to understand easily, it may not be graded.
2. a proof of correctness
3. an analysis of running time and space

You may use algorithms from class as subroutines. You may also use facts that we proved in class.

You should be as clear and concise as possible in your write-up of solutions. A simple, direct analysis is worth more points than a convoluted one, both because it is simpler and less prone to error and because it is easier to read and understand.

Problem 1 ***Database restore*** *(10 points)*

In the heart of a bustling metropolis, you find yourself working as a data analyst at one of the city's most prestigious banks. Each day, hundreds of clients make transactions, and you are tasked with recording every detail a meticulous job to keep the financial wheels turning. But one fateful evening, disaster strikes. A technical glitch sweeps through the bank's system, erasing every last bit of transactional data for the past m days from your records. The once carefully organized files of n clients now lay in digital ruin. Fortunately, despite the chaos, a glimmer of hope emerges. Thanks to a clever data-preserving mechanism, not all is lost. For each client, youve managed to salvage the sum total of their transactions over the entire period. Additionally, you have one more crucial piece of information: for each of the m days, you possess the grand total of all transactions that took place across all clients. Armed with these fragments, your task is clear but challenging: to restore the detailed transaction data for each client on every day, breathing life back into the banks records and restoring order from the brink of disaster. Can you untangle this web and retrieve the missing data, client by client, day by day?

As *input* you are given a table C for the clients, such that $C[i]$ contains the total Dollar amount of transactions for client i. We also get a table D, such that $D[j]$ is the total amount on day j.

As *output* your algorithm should return a table M, such that $M[i][j]$ contains the transaction amount of client i on day j or return that there doesn't exist such a database.

Problem 2 ***Decreased flow*** *(10 points)*

We are given a network flow graph $G(V, E, s, t, c)$ and a maximum flow f on this network. That is, for each edge (u, v) we are given the amount of flow $f(u, v)$.

1. (*not to be handed in*) As a warm up, think of it how we could verify that the given flow is indeed a max flow. That is, it can't be increased anymore. One way is to run Ford-Fulkerson from scratch and compare the value of the flow. We can do it in fact much faster, in $O(|V| + |E|)$ time. What is this faster approach?

2. Now we know for a fact that f is indeed a maximum flow. Let (x, y) be a specific edge, and suppose that we decrease the capacity of this edge by one. That is, if the current capacity is $c(x, y)$, then the new capacity is $c'(x, y) = c(x, y) - 1$. Given $value(f)$ in the original flow network, describe in words what values the maximum flow can take in the decreased graph. Also explain what property the edge (x, y) has for the different max flow values.

3. Design an algorithm that given $G(V, E, s, t, c)$, the max flow f and the edge (x, y) as input computes the max flow f' in the decreased graph. (Don't confuse the max flow f' with the $value(f')$, the former is a function assigning values to each edge, the latter is a single number.) Your algorithm should run faster than rerunning Ford-Fulkerson from scratch. For full credit it should run in $O(|V| + |E|)$.