CSE 417 25au Homework 7: Network Flows

Released: Friday, November 14, 2025 @ 11:30am

First due by: Wednesday, November 26, 2025 @ 11:59pm

Last resubmissions by: Friday, December 5, 2025 @ 11:59pm

Instructions

For Problems 13, 13X.1, and 14, you have four options for submission:

- **Film a video in which you explain your solution.** See the Homework Guide for more details.
- **Use LaTeX to type your solutions.** A template is provided in the "Tasks" page of the course website, if you like.
- **Use Google Docs or Microsoft Word to type your solutions.** If doing so, please use the Equation Editor to ensure that any equations are legible and easy to read.
- **Handwrite your solutions on paper or digitally.** Please write neatly and if on paper, scan in black/white mode, not grayscale.

We prefer either video or LaTeX, but accept any of the 4 options.

A few more reminders:

- **Submit all problems on Canvas.** Each problem should have its own submission. *Do not* submit one large file containing answers to several problems.
- **Suggested word counts are rough guidelines.** We won't actually count, but if your writing is verbose to the point of obscuring your main argument, we may ask you resubmit more concisely.
- **Review the collaboration policy in the syllabus**. Collaboration is encouraged but strict rules apply, and remember to cite your collaborators.
- If you don't finish in time, we encourage you to be honest and just upload what you have so far. Resubmission won't cost you anything, and we can give you timely feedback on your partial progress by submitting on time.

Happy problem solving!

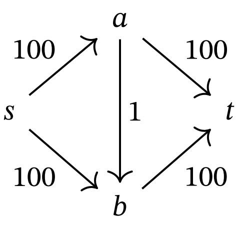
Problem 13: Improving Ford–Fulkerson

The purpose of this problem is to reinforce and extend the Ford–Fulkerson algorithm for network flows discussed in class.

Recall the Ford–Fulkerson algorithm described in class to solve the network flow problem, given a directed graph $G = (V, E)$ with capacities $c(e)$ on edges and two special vertices s and t. Stated succinctly, it is:

```
1: Initialize the flow $f$ to 0 everywhere.
2: Initialize the residual graph $G_f$ to the original graph $G$.
3: while $G_f$ contains a path from $s$ to $t$ do
4:     Let $P$ be any path from $s$ to $t$.
5:     Update the flow $f$ to push as much extra flow along $P$ as possible.
6:     Update the residual graph $G_f$ according to the new flow.
```

While the Ford–Fulkerson algorithm works to find a max flow, as you saw in class, there are some instances in which some bad choices for the path P could result in the algorithm taking a long time. For example, recall that the following graph could takes 200 iterations in the worst case, but only 2 iterations for a smart choice of augmenting paths.



The way to fix this issue is by always picking P to be an *unweighted* shortest path from s to t (i.e. fewest number of edges), which you could find, for instance, via BFS. This problem will guide you through analyzing this idea.

1. Note that the residual graph G_f has edges both added and deleted throughout the algorithm. Explain why if you use BFS to find a path, then for any vertex v, the unweighted distance from s to v will never decrease (i.e. v can only get farther away from s). (suggested 50–100 words)

   (Hint: Recall that BFS sorts the vertices into levels, so that any shortest path from s to t goes from s in level 0, to a vertex in level 1, to vertex in level 2, and so on. When you create an edge (u, v), what can you say about the levels of u and v?)

2. Recall that the *bottleneck* of an augmenting path P is the minimum residual capacity along the path. Call an edge in P a *bottleneck edge* if its residual capacity is the bottleneck value of P. (Every augmenting path has a bottleneck edge, but some may have more than one if two edges both have the least residual capacity.)

   Let (u, v) be an edge. Show that (u, v) will only be a bottleneck edge at most $n/2$ times during the Ford–Fulkerson with BFS algorithm, where n is the number of vertices in the graph. (suggested 70–120 words)

   (Hint: What happens to (u, v) during the iteration in which it is a bottleneck edge?

What has to happen before it can be a bottleneck edge again? Then use the previous part.)

3. Every iteration of the Ford–Fulkerson algorithm takes $O(m)$ time, where m is the number of edges in the original graph G. Use the previous part to analyze the maximum number of iterations of the Ford–Fulkerson algorithm with BFS. Then, what is the overall running time of the algorithm, in big-O? (suggested 30–70 words)

Video submissions are encouraged to take 3–6 minutes total.

Problem 13X: Improving Ford–Fulkerson (extensions)

This is an extension problem that doesn't build so much on the ideas of Problem 13, but builds on the same Ford–Fulkerson background as Problem 13. **Pick one** of the following to complete:

1. Consider the following instance of a network flow, where $\phi = (\sqrt{5} - 1)/2 \approx 0.618$ is the positive solution to the quadratic equation $\phi^2 + \phi - 1 = 0$.

$$\begin{tikzcd}
 & & a \arrow[d, "\phi"'] \arrow[dr, "100"] & \\
s \arrow[urr, "100"] \arrow[r, "100"'] \arrow[dr, "100"'] & b \arrow[r, "1"] \arrow[d, "1"] & c \arrow[r, "100"] & t \\
 & d \arrow[urr, "100"'] & &
\end{tikzcd}$$

   It turns out that for a particularly bad choice of augmenting paths, the Ford–Fulkerson algorithm could take *infinite* time, never terminating, and that the value of the flow never gets anywhere close to the maximum flow of 201.

   The bad sequence of augmenting paths is as follows. First, augment by $s \to b \to c \to t$, pushing 1 flow. Then, repeat these four augmenting paths over and over again.

   - $s \to a \to c \to b \to d \to t$
   - $s \to b \to c \to a \to t$
   - $s \to a \to c \to b \to d \to t$
   - $s \to d \to b \to c \to t$

   Draw out these augmenting paths and the resulting residual graphs for yourself, to see what happens. Then answer the following questions. (No need to include your drawings, but feel free to if you want.)

   (a) In the kth iteration of the loop (the first iteration being $k = 1$), how much flow is pushed through by each augmenting path? Be sure to simplify your answers using the fact that $\phi^2 + \phi - 1 = 0$ (in other words $1 - \phi = \phi^2$). Say how you used this step, but no need for a very long explanation. (suggested ≤ 50 words)

   (b) Recall that $1 + x + x^2 + x^3 + \cdots = 1/(1 - x)$, the formula for an infinite geometric series. Use this to calculate the limit of the total flow that these augmenting paths push through, as the number of iterations tends to infinity. Don't forget to include the first path that pushed 1 flow. As a sanity check, your answer should be much less than the actual maximum flow of 201. (suggested ≤ 50 words)

(c) Could the same problem happen if your capacities are *rational numbers* (fractions of integers)? Why or why not? (suggested ≤ 50 words)

(d) Could the same problem happen if you insist on using BFS to find an augmenting path, like in the previous problem? (Not necessarily the graph above, maybe with a different graph?) Why or why not? (suggested ≤ 50 words)

Video submissions are encouraged to take 3–6 minutes total.

2. Ford–Fulkerson is not the only algorithm that solves the maximum flow problem. It was the first major algorithm to do so, but many improvements have been made over the years. The most recent big improvement just happened in 2022, achieves almost linear time: $O(m^{1+\epsilon})$ time for all $\epsilon > 0$. (This means almost, but not exactly linear time. For example, the function $m \log m$ is $O(m^{1+\epsilon})$ for all $\epsilon > 0$. The actual running time for their algorithm was a more complicated expression.)

   Read about this breakthrough in this news report from Quanta. Then, join the discussion on Canvas by responding to the following prompt, raising other questions of your own choosing, or replying directly to other students' responses. (suggested 150–300 words)

   > Continued research in algorithms progresses in many directions. In some instances, researchers chase gains in big-O running time, sometimes incurring large constant overhead, as you also saw in homework Problem 5X.1. Other times, researchers focus on improving implementations for realistic-sized problem instances, and are happy with, e.g. 50% performance improvement without any big-O savings. How do you value each type of research? Would you personally be interested in one over the other?

   You are *not* required to cite the article or any other sources, but be sure to demonstrate an accurate understanding of big-O. If you discuss any technical information that is not in the lecture slides or this reading, please cite your sources. **Resubmissions will not be available for this part.**

Problem 14: Traffic modeling

The purpose of this problem is to practice modeling with network flows.

In most cities, traffic congestion only happens when two roads intersect. Red lights cause traffic, and not much else for most of the time. A very rough approximation is that the number of vehicles that can pass through an intersection in any direction per hour is proportional to the number of vehicle lanes there, including turning lanes and through lanes in all directions. You are given the road network of a city as a graph $G = (V, E)$, as well as the number of lanes $c(v)$ at each intersection $v \in V$. Assume that this city only has one-way roads.

Suppose each lane adds a capacity of 400 vehicles per hour, which is typical in cities. Now, many people are trying to reach the football stadium at intersection f from a set of starting intersections H, homes where people live. The capacities of f and all intersections in H do matter. Your job is to compute how many vehicles per hour can move from the set of starting intersection H to t.

Design a new graph $G' = (V', E')$ based on the original graph G, on which you will run the Ford–Fulkerson algorithm. Be sure to specify the capacities of every *edge* (which may be infinite) that you create, as well a single starting vertex s and a single ending vertex t. The value of the flow output by Ford–Fulkerson on your new graph G' should be exactly the number of vehicles per hour that can move from the set of starting intersections H to f in the original graph G.

Just describing your graph is sufficient. No need for a full proof of correctness. (suggested 50–100 words, or 2–4 minute video)