Lecture Notes for CSCI 311: Algorithms

Set 19-Flows & Cuts

Professor Talmage

April 30, 2025

---

1 Definitions

We next want to consider a way to look at the capacity of a graph. This is discussed in terms that apply most directly to plumbing, with value flowing through the graph as if each edge is a pipe. This type of model applies to many different scenarios, though, such as traffic on a road network, people moving through a building, etc.

Consider a weighted digraph $G = (V, E, c : E \to \mathbb{R}, s, t)$, where c is a weight function, which we here interpret as an edge's *capacity*, and $s, t \in V$, s is a *source*, t is a *sink*. We will assume all nodes are reachable from s, as any unreachable nodes are of no interest to us. We can also represent missing edges as having no capacity, $(u, v) \notin E \Rightarrow c(u, v) = 0$, so assume every possible edge is in E. We call this a *flow network*.

Definition 1. A *flow* is a function $f : V \times V \to \mathbb{R}^{\geq 0}$ s.t.

1. Flow respects capacity: $\forall u, v \in V, 0 \leq f(u, v) \leq c(u, v)$.
2. Flow is conserved at each node: $\forall u \in V \setminus \{s, t\}, \sum_{v \in V} f(v, u) = \sum_{v \in V} f(u, v)$.

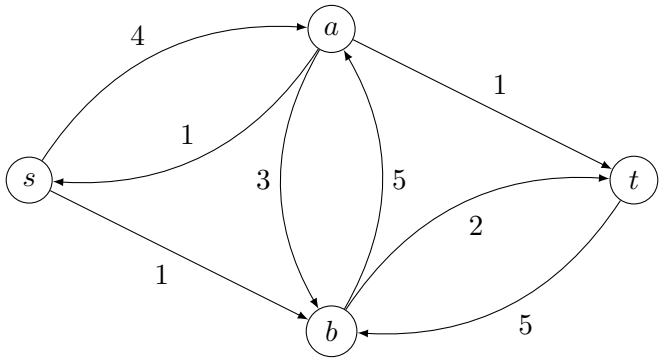
We define the *value* of a flow as the net flow out of the source $|f| = \sum_{v \in V} f(s, v) - \sum_{v \in V} f(v, s)$. This will also equal the net flow into the sink.

Problem: Given a flow network, find a flow with maximum value.

Input: Flow network $G = (V, E, c, s, t)$.

Output: Flow f on G with maximum value.

Exercise: Find the maximum flow you can on the following network.



Exercise: How do you know the flow you found is maximum?

One possibility is to observe that all the edges entering t together have capacity 3, so there is no way to get more than 3 flow into t. We will generalize this observation soon.

2 Algorithm Outline

Similar to what we saw for Minimum Spanning Trees, we will first look at an outline of an algorithm for finding maximum flows, then refine it to be fully defined.

Algorithm 1 Ford-Fulkerson Algorithm Outline for Max Flow

```
1: function FORD-FULKERSON(G, s, t)
2:     initialize f to 0 on all edges
3:     while there exists an augmenting path p in the residual graph G_f do
4:         augment flow f along p
5:     end while
6:     return f
7: end function
```

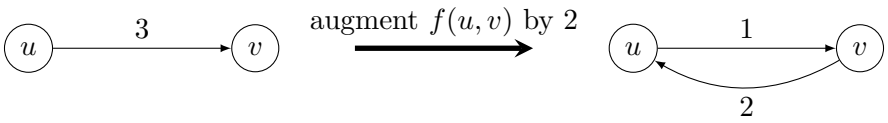
We need to define a few terms here, but first let us understand the intuition. A residual graph represents the remaining, unused capacity of the graph. An augmenting path in such a graph is a route along which we can move more flow from the source to the sink. Augmenting a flow is adding that additional flow to our flow function. In other words, `Ford-Fulkerson` says "As long as we can find a way to get more flow through the network, add that to our flow function." While this may seem obvious, this particular breakdown is something we can tackle algorithmically, and we will prove that it will, in fact, lead to an optimal solution.

Definition 2. Given a flow f, the *residual capacity* of edge (u, v) is

$$c_f = \begin{cases} c(u,v) - f(u,v) & f(u,v) > 0 \\ c(u,v) + f(v,u) & f(v,u) > 0 \end{cases}$$

The *residual graph* $G_f = (V, E, c_f, s, t)$.

What this is doing is tracking how much additional flow we could send between each pair of nodes. As we add flow from u to v, the residual capacity from u to v decreases, since we are using some of that capacity, so there is less available. But, simultaneously, the residual capacity from v to u *increases*. This is because we could send more flow back from v to u, which is equivalent to decreasing the flow from u to v. For example,



Definition 3. An *augmenting path* p of a flow f in flow network G is a path from s to t in G_f. The residual capacity of p is $c_f(p) = \min\{c_f(u,v) \mid (u,v) \in p\}$.

It is important to note that the capacity of a path is the smallest capacity of any of the edges on the path. This is quite different from the weight of a path, as we have considered for the shortest path problem. Here, the smallest capacity edge along the path is a bottleneck, limiting how much flow we can send along this route.

Claim 1. *If p is an augmenting path of f in G, the flow*

$$f_p(u,v) = \begin{cases} c_f(p) & (u,v) \in p \\ 0 & (u,v) \notin p \end{cases}$$

can be added to f to yield a valid flow.

That is, if we have an augmenting path, we can add the capacity of that path to the flow along each edge of the path without exceeding any edge's capacity. This is *augmenting* a flow.

3 Implementation

Since `Ford-Fulkerson` did not completely specify how to complete each of its steps, we need to give those specifics to implement an algorithm. The `Edmunds-Karp` implementation is one way to fully define the behavior of the `Ford-Fulkerson` outline:

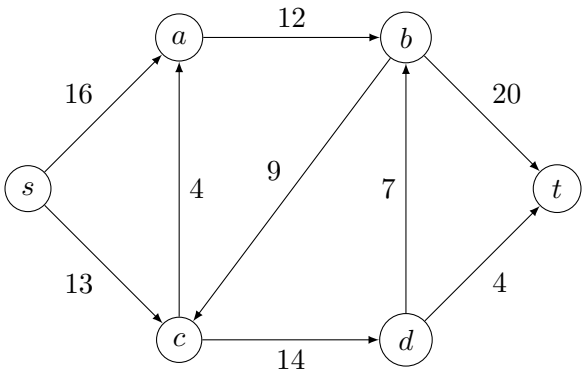
1. Use BFS to find augmenting paths $s \rightsquigarrow t$ in G_f.
2. Choose the path with fewest edges by using BFS on an unweighted version of the graph.
3. Augment the flow with this path and repeat until there is no path from $s \rightsquigarrow t$.

Complexity: Breadth-First Search requires $O(|E|)$ time, since we know there at least as many edges as vertices. The number of iterations is $O(|V||E|)$–since the shortest distance in the residual graph will increase each time, each edge can only be the lowest-capacity edge on an augmenting path a few times. This gives a total of $O(|V||E|^2)$. There are other implementations which improve this to $O(|V|^3)$, so this is not optimal, but we do not have time to consider them.

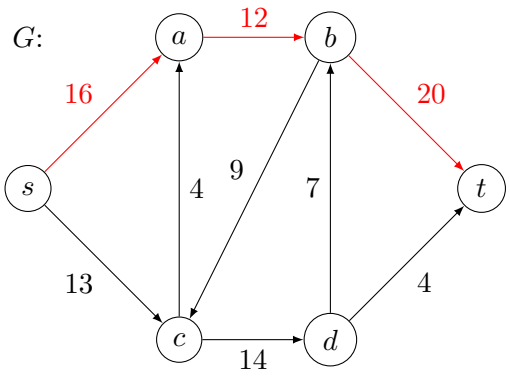
Aside: While we defined flow networks to have only one source and one sink, real networks may have many of each. We can model such network with our definition, though, by adding a new "supersource" with edges to each real source and "supersink" with edges from each real sink. These new edges have infinite capacity to avoid placing any new constraints on the network. Max-flow on this single-source, single-sink graph is now the same as max-flow on the original graph.

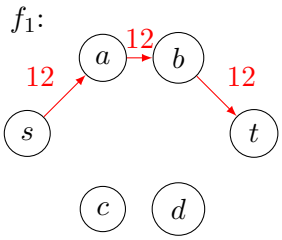
Exercise: Discuss with a neighbor and convince yourself that this construction does not change the maximum flow value.

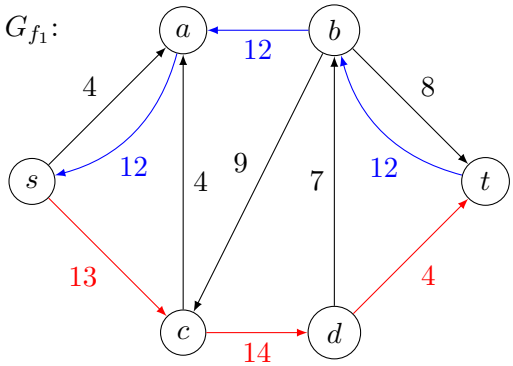
Example: Consider the following graph. We will run the `Edmunds-Karp` algorithm on it.

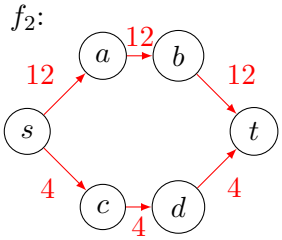


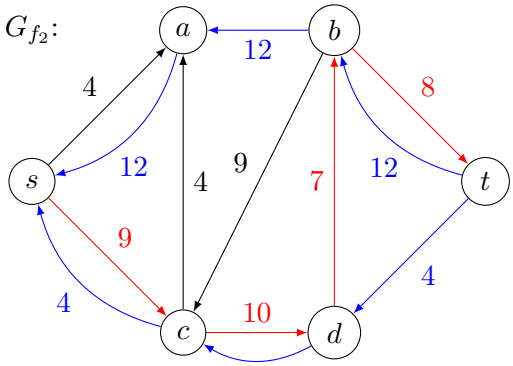
See the next page for a step-by-step construction of a max flow. We need to track residual graphs as we add more flow. Red edges indicate augmenting paths. Blue edges are back edges added to the residual graph.

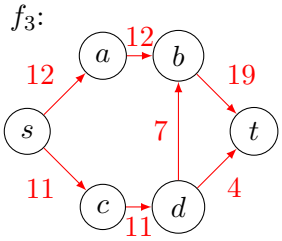


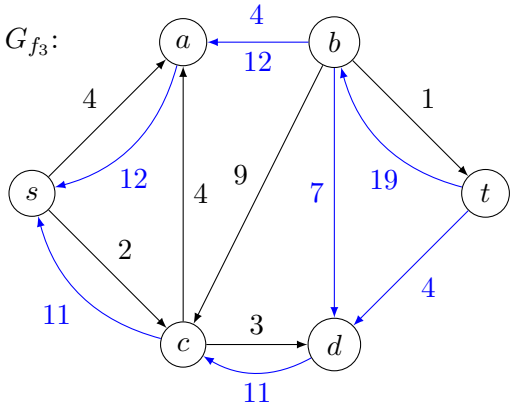






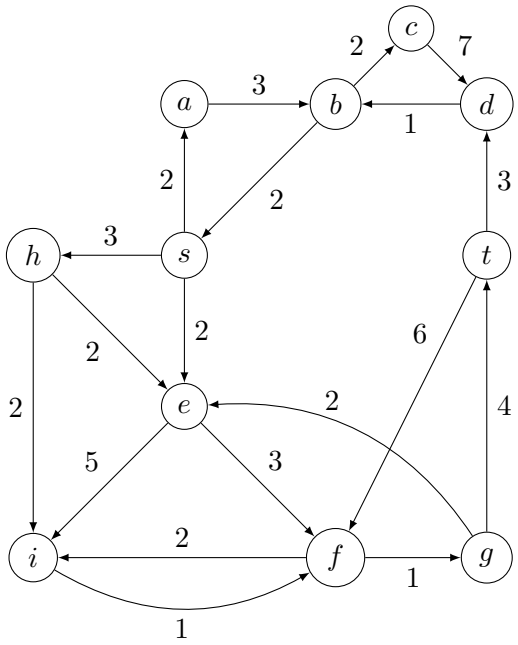






There are no more augmenting paths, so we have a maximum flow, with value $|f_3| = 23$.

Exercise: Run the Edmunds-Karp algorithm on the following graph.



4 Max-Cut, Min-Flow

Recall that a *cut* of a graph $G = (V, E)$ is a subset $S \subseteq V$. An s, t-*cut* of a flow network is a cut s.t. $s \in S$, $t \in V \setminus S$. The *capacity* of an s, t-cut in a flow network is

$$c(S, V \setminus S) = \sum_{\substack{(u,v) \in E \\ u \in S \\ v \in V \setminus S}} c(u, v)$$

In an s, t-cut S, the net sum of flow values on all values crossing the cut is $|f|$, so $|f| \leq c(S, v \setminus S)$, for any s, t-cut. This leads us to the following theorem:

Theorem 1. *In a flow network $G = (V, E, c, s, t)$, TFAE:*

1. *f is a max flow.*
2. *There are no augmenting paths in G_f.*
3. *There exists a cut $(S, V \setminus S)$ s.t. $|f| = c(S, V \setminus S)$.*

Sketch. We prove this by showing a cycle of implications:

- $(1 \Rightarrow 2)$: If f is a max flow, then there cannot be an augmenting path, or we could augment f.
- $(2 \Rightarrow 3)$: If there are no augmenting paths, then we can split V into S and $V \setminus S$ s.t. there is no path from S to $V \setminus S$ in the residual graph, by letting S be the vertices reachable from s in the residual graph. This means that $|f| = c(S, V \setminus S)$.
- $(3 \Rightarrow 1)$: Since every cut is an upper bound on flow, if the flow equals the capacity of some cut, then it is maximum.

□