CS624 - Analysis of Algorithms

Max Flow

November 26, 2025

The problem

- $G = (V, E)$ is a directed graph with two distinguished vertices, s (the "source") and t (the "sink" or "target"). We also assume that
  1. There is at least one path from s to t. In fact, even more is true: for each node u in the graph, there is at least one path from s through u to t.
  2. There are no "antiparallel" edges. That is, if $e = (u, v)$ is an edge from vertex u to vertex v, then there is no edge from v to u. (there could be a *path* from v to u.)

The problem

- This last item ("no antiparallel edges") is not a big deal.
- In fact, if we have a graph with edges (u, v) and (v, u), we can just introduce a new vertex w and replace the edge (v, u) by the two edges (v, w) and (w, u). It won't change anything.

Definition

A *flow network* is a directed graph $G = (V, E)$ with two distinguished vertices s and t such that

- for each vertex u there is at least one path from s through u to t, and
- there are no antiparallel edges in the graph.

Flows

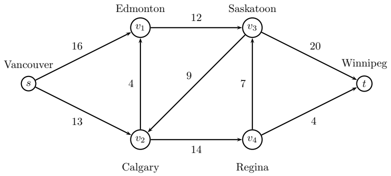
Definition

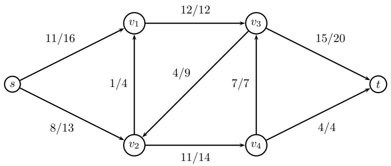
A flow in the directed graph G as above is an assignment of a non-negative real number to each edge of the graph.

- We denote the assignment by $f : E \to \{r \in \mathbb{R} : r \geq 0\}$. Thus, for each edge e, $f(e)$ is a non-negative real number.
- If $e = (u, v)$, then we may (by an abuse of notation) also write $f(u, v)$ for $f(e)$ – they will mean the same thing.
- We think of $f(e)$ as the amount of some substance passing over edge e per unit time (current or water for example).
- We might also think of a transportation network like a railroad lines. And then for each edge e, $f(e)$ represents the amount of something that was being transported over e in some unit of time.

Network Flow Example

- We think of $f(e)$ as the amount of some substance passing over edge e per unit time (current or water for example).
- We might also think of a transportation network like a railroad lines. And then for each edge e, $f(e)$ represents the amount of something that was being transported over e in some unit of time.
- Left: Flow network. Right: Flow inside the network/capacity.





A conservation property

- Remember we are transporting some substance from s to t.
- So it must be that the function f satisfies the following constraint:
  Conservation: At every node u in the graph other than nodes s and t, the flow into u must exactly equal the flow out of u:
  $\sum_{v \in V} f(v,u) = \sum_{v \in V} f(u,v)$
- This constraint is really a sort of conservation property – it says that the amount being transported is conserved at each vertex.
- We need to be able to express this constraint mathematically. To do this, it is most convenient to extend the function f to all pairs of nodes (u, v), even of (u, v) is not an edge in the graph.

Two Ways To Define The conservation property

extend f so it remains non-negative, We extend the function f so that if (u, v) is not an edge, then we simply set $f(u, v) = 0$. With that notation, we can write our constraint as follows: for each node u different from s and t, $\sum_{v \in V} f(v, u) = \sum_{v \in V} f(u, v)$.
This is in fact the way our text writes this constraint.

extend f so that it can also be negative. if (u, v) is a (directed) edge in the graph, then we know already that (v, u) is not. We define $f(v, u) = -f(u, v)$. And for all other pairs (x, y) such that neither (x, y) nor (y, x) is an edge, we define $f(x, y) = f(y, x) = 0$.
The idea here is that we can think of a flow along (u, v) as a “negative” flow along (v, u).

A conservation property

- With this notation, we can write our constraint more simply: for each node u different from s and t,

$$\sum_{v \in V} f(u, v) = 0$$

- Both of these conventions are completely equivalent.
- The first convention has the advantage that in the beginning, anyway, it can seem somewhat easier to understand what is going on.
- The second one has the advantage that the mathematics turns out to be much simpler.
- We have to pick one. Both are widely used. We are going to use the second one.

Definition of Flow

Definition

A flow f on the graph G with source and target vertices s and t is any function mapping pairs of vertices of G to $\mathbb{R}$ such that it satisfies the following three conditions:

1: f lives on edges. If u and v are not joined by an edge (in either direction), then $f(u, v) = 0$.

2: f is skew-symmetric. For all vertices u and v, $f(u, v) = -f(v, u)$

3: f satisfies a conservation property. For all vertices v other than s and t, $\sum_{u \in V} f(u, v) = 0$

Flows can be added

Definition

If f and g are two flows, we define their sum $f + g$ in the obvious way:

$$(f + g)(u, v) = f(u, v) + g(u, v)$$

Definition

If f is a flow, the *value* $|f|$ of the flow is the flow out of the source. To be precise,

$$|f| = \sum_{v \in V} f(s, v)$$

Notice that the flow out of s is equal to the total flow into t.

Cuts

Definition

A *cut* in G is a partition of the vertex set V into two sets X and $\overline{X}$ such that $s \in X$ and $t \in \overline{X}$.
If f is a flow and $(X, \overline{X})$ is a cut, the *flow across the cut* is defined to be

$$f(X, \overline{X}) = \sum_{v \in X, w \in \overline{X}} f(v, w)$$

Lemma

If f is a flow and $(X, \overline{X})$ is a cut, the flow across the cut is equal to the flow value $|f|$.

Proof

First, note that

$$\begin{aligned}\sum_{\substack{v\in X\\ w\in V}} f(v,w) &= \sum_{w\in V} f(s,w) + \sum_{\substack{v\in X\setminus\{s\}\\ w\in V}} f(v,w)\\ &= |f| + 0 = |f|\end{aligned}$$

The last sum in the first line vanishes because v is neither s nor t, and so we can apply the conservation property:

$$\begin{aligned}\sum_{\substack{v\in X\setminus\{s\}\\ w\in V}} f(v,w) &= \sum_{v\in X\setminus\{s\}} \sum_{w\in V} f(v,w)\\ &= \sum_{v\in X\setminus\{s\}} 0 = 0\end{aligned}$$

because (since v is neither s nor t) the inner sum vanishes by the conservation property)

Proof

Using this, we see then that

$$\begin{aligned} f(X,\overline{X}) &= \sum_{\substack{v\in X\\ w\in\overline{X}}} f(v,w) \\ &= \sum_{\substack{v\in X\\ w\in V}} f(v,w) - \sum_{\substack{v\in X\\ w\in X}} f(v,w) \\ &= |f| - 0 \\ &= |f| \end{aligned}$$

where here the second sum in the second line vanishes because of skew-symmetry.

The capacity of an edge

- To make our problem realistic, we assume that each edge has a maximum capacity that it can carry.
- Therefore, the flow along that edge can be no more than the capacity of the edge.
- So there is a *capacity function* $c(e)$ which assigns to each edge a non-negative real number.
- Therefore, we insist that for each edge e,

$$0 \leq f(e) \leq c(e)$$

- The above is referred to as "capacity constraint" in the text.
- If (v, w) is a (directed) edge, we also use the notation $c(v, w)$ for the capacity of that edge.
- If (v, w) is not an edge (and in particular, if it is a "reverse edge"), then we set $c(v, w) = 0$.

The capacity of an edge

- We can also talk about the *capacity of a cut*: if $(X, \overline{X})$ is a cut, we define its capacity $c(X, \overline{X})$ by

$$c(X, \overline{X}) = \sum_{\substack{v \in X \\ w \in \overline{X}}} c(v, w)$$

- It can be shown that if $(X, \overline{X})$ is a cut, then

$$f(X, \overline{X}) \leq c(X, \overline{X})$$

or equivalently,

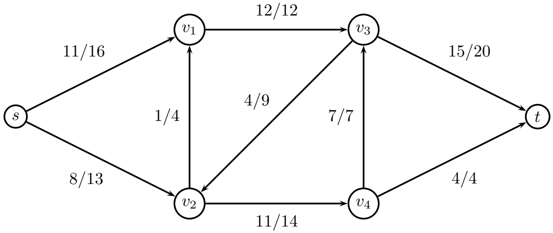
$$|f| \leq c(X, \overline{X})$$

The actual statement of the problem

- Given a flow network with a capacity function, we want to find the flow respecting that capacity and having the greatest possible value.
- We call such a flow a *maximum flow*. We're trying to find a maximum flow.

The Ford-Fulkerson Method

- Given a flow graph, find a pathway (an augmenting path) of unused capacity and increase the flow along that pathway.
- Repeat until no such pathways are found.
- Notice that overall flow can sometimes be *increased* by *decreasing* flow along certain edges
- Because they flow in the "wrong" direction or move capacity to a part of the network that can't handle it as well.
- See whether you can find an example in the graph shown.



The Ford-Fulkerson Method – Residual Networks

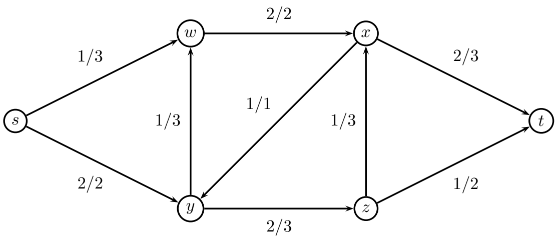
- Ford Fulkerson manages this by constructing a parallel network of the available or *residual capacity*.
- Given a flow f in a network G = (V, E), consider a pair of vertices $u, v \in V$. How much additional flow can we push directly from u to v? This is the residual capacity:

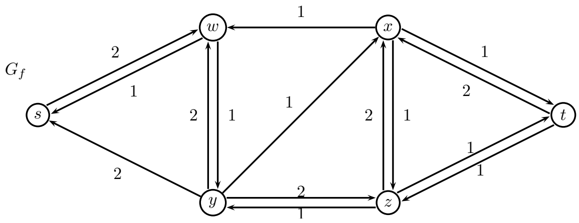
$$c_f(u,v) = \begin{cases} c(u,v) - f(u,v) & \text{if } (u,v) \in E \\ f(v,u) & \text{if } (v,u) \in E \\ 0 & \textit{otherwise} \end{cases}$$

The Ford-Fulkerson Method – Residual Networks

- The first case says that if we have not used the full capacity $c(u, v)$ of an edge $(u, v) \in E$ then we can increase it by the difference.
- The second case says that if we are using $f(v, u)$ of the capacity of $(v, u) \in E$ then we have the residual "capacity" of reversing (cancelling) that much flow in the reverse direction (u, v) (notice that the letters are swapped).
- Otherwise there is no residual capacity between u and v.
- We record these capacities in the residual network $G_f = (V, E_f)$, where $E_f = \{(u, v) \in V \times V : c_f(u, v) > 0\}$ Each edge of the residual network can admit a positive flow.

Residual Network – Example





Residual Network – Example

- G_f says that we can add two more units from s to w in G or we can take one unit back.
- Every edge $(u, v) \in E_f$ corresponds to $(u, v) \in E$ or $(v, u) \in E$ or both.
- So, $|E_f| \leq 2|E|$
- A residual network is similar to a flow network, except that it may contain antiparallel edges.
- We can define flow in a residual network that satisfies the definition of flow, but with respect to c_f in G_f.

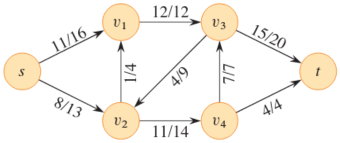
Augmentation and Augmenting Paths

- Given flows f in G and f' in G_f, define the augmentation of f by f', $f \uparrow f'$, to be a function $V \times V \to \mathbb{R}$:

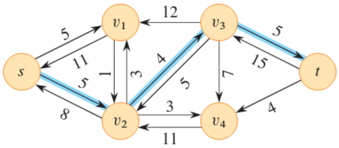
$$(f \uparrow f')(u,v) = \begin{cases} f(v,u) + f'(u,v) - f'(v,u) & \text{if } (v,u) \in E \\ 0 & \textit{otherwise} \end{cases}$$

  for all $u, v \in V$.
- Increase the flow on (u,v) by $f'(u,v)$, but decrease it by $f'(v,u)$ because pushing flow on the reverse edge in the residual network decreases the flow in the original network.
- **Lemma:** Given a flow network G, a flow f in G, and a residual network G_f, let f' be a flow in G_f. Then $f \uparrow f'$ is a flow in G with value $|f \uparrow f'| = |f| + |f'|$.

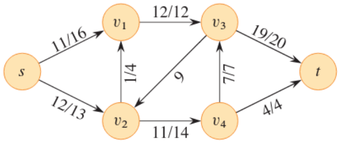
Augmenting Paths – Example



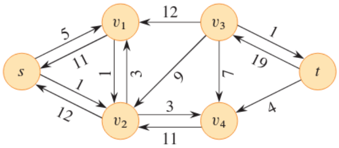
(a)



(b)



(c)



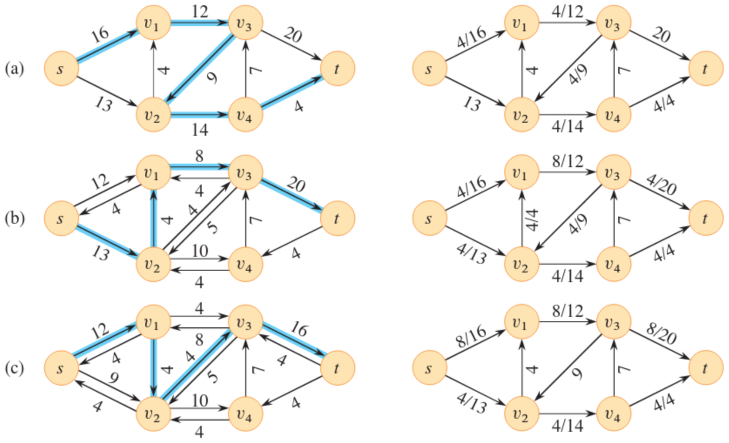
(d)

Augmenting paths Algorithm

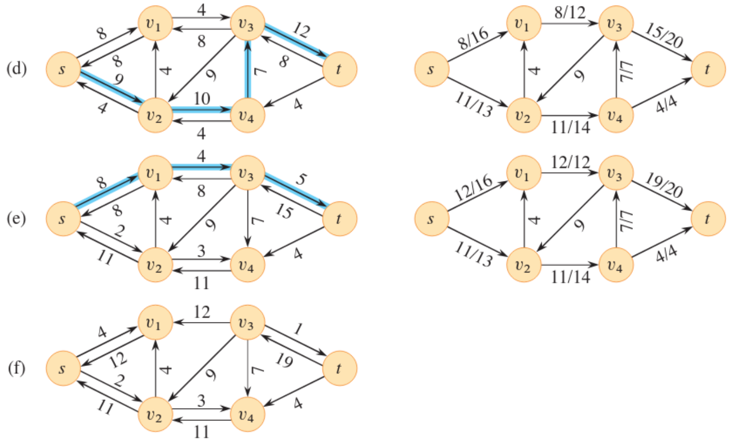
So we have the beginnings of an algorithm here:

- Start with any allowable flow.
- From it, produce the residual graph.
- Use that residual graph to find any augmenting path.
- Add the augmenting path to the original flow to produce a new flow. The new flow will still be allowable and will have a greater value.
- Repeat this whole process until nothing more can be done.

Ford-Fulkerson – Example I



Ford-Fulkerson – Example II



Does the algorithm really work?

There are two questions that come up immediately:

1. Is it possible that we might have a non-maximum flow f, but nevertheless there was no augmenting path for f? If there was, then of course the algorithm would fail to yield a maximum flow.
2. Is it possible that the algorithm fails in some other way? For instance, maybe it never ends.

The max-flow min-cut theorem

- The second question actually has a pretty curious answer.
- But the first question can be answered right away, based on a remarkable theorem due to Ford and Fulkerson, who laid the foundations for this whole theory and wrote the original book on the subject.

Theorem (The max-flow min-cut theorem)

If $G = (V, E)$ is a flow network with capacity function c and source and sink nodes s and t, the following statements are equivalent:

1. *f is a maximum flow.*
2. *There is no augmenting path for f.*
3. *There is some cut $(X, \overline{X})$ for which*

$$|f| = c(X, \overline{X})$$

Proof

(1) $\implies$ (2). If there is an augmenting path p for f, then we can increase the flow value.

(2) $\implies$ (3). Suppose there is no augmenting path for f. Let X be the set of vertices reachable from s on a path on which each edge has positive residual value. Trivially, X includes s, and by our assumption, X does not include t. Thus, $(X, \overline{X})$ is a cut. Further, if (v, w) is an edge in the original graph that crosses the cut (i.e., with $v \in X$ and $w \in \overline{X}$), we must have $f(v, w) = c(v, w)$, since otherwise w would also be in X. Thus, we have

$$|f| = \sum_{\substack{v \in X \\ w \in \overline{X}}} f(v, w) = \sum_{\substack{v \in X \\ w \in \overline{X}}} c(v, w) = c(X, \overline{X})$$

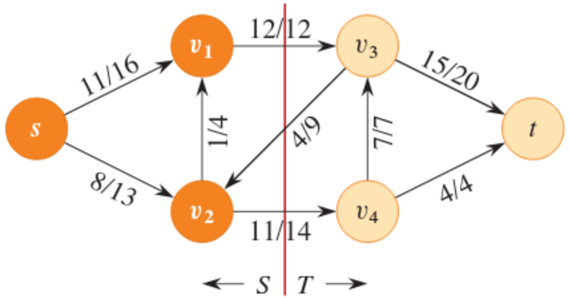
Proof (Cont.)

(3) $\implies$ (1). We know that in any case, $|f| \leq c(X, \overline{X})$ (See exercise in class notes). The fact that with this cut they are equal means that there is no flow with a greater value than f.

- Note that as a consequence of this theorem, we know that the value of the maximum flow in the network is equal to minimum capacity of any cut. That's where the theorem gets its name.
- So this answers the first of our questions: if f is not maximum, we know that there will be at least one augmenting flow, so the algorithm can make progress.

A Cut Example

- The value of the flow is 19.
- The capacity of the cut is 26.



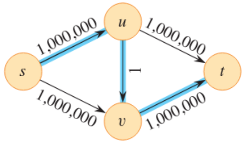
Termination and correctness

- We need to know if the algorithm is guaranteed to terminate. Here's where things get complex.
- First of all, let us suppose that all the capacities of the network are integers. In that case, there is something we can say, provided we make two very natural assumptions:
- We assume that we start with a flow that is identically 0. (We'll always assume this in any case; it's the natural way to start.)
- We assume that when we pick an augmenting path, we make the flow along that path as large as possible.
- That is, we let it be the minimum of the residual values of each edge on the flow.

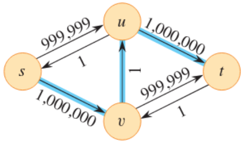
Termination and correctness

- In this case, the augmenting flow value will of necessity be an integer, and so we will always be increasing the value of the flow by an integer each time we pick an augmenting path.
- Since the maximum flow value is finite, this process has to stop.
- This also shows that the maximum flow is an integral flow–the flow along each edge is an integer.
- However, there are many flow networks that arise in practice in which the capacities are not necessarily integers.
- This is not really a problem. If they are rational numbers, then we can always reason in exactly the same way by using as our "minimal unit" the fraction which is 1 over the least common denominator of all the capacities. Exactly the same argument then works.

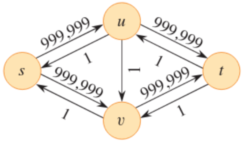
Inefficient Example



(a)



(b)



(c)

Termination and correctness

- The problem of irrational numbers is peculiar, though. Ford and Fulkerson actually gave an example where, by picking the augmenting paths in a particularly stupid manner, even though the flow along these augmenting paths was chosen to be as big as possible –the following would happen:
- The process would not terminate. Of course the values of the successive flows would keep increasing, but they would always be less than the maximum possible value.
- It's even worse than that. Because the values of the successive flows keep increasing, they have a limit. But this limit is strictly less than the value of the maximum flow in the network.

Termination and correctness

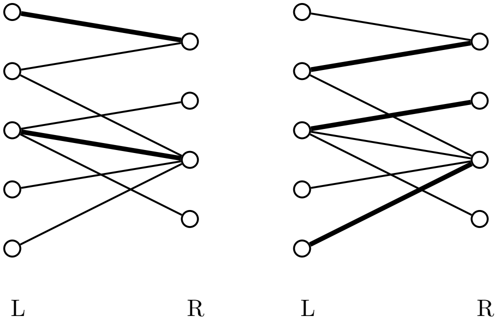
- We can always assume that our capacities are rational, and so our algorithm – however we pick the augmenting paths – will definitely terminate and give the maximum flow.
- However, there still is the question of efficiency: how well can we pick the augmenting paths so that we have to repeat the iterations of the algorithm as few times as possible?
- This question has been considered over many years, and successively better algorithms have been produced.
- Our textbook goes into this a little but we will stop here.

Application– Bipartite Matching

- Maximum Flow can also be used to solve problems that don't look like flow problems. Here is an example.
- Suppose we want to maximize ...
- The number of boys and girls who can dance, given a list of who is willing to dance with whom (a.k.a. the "marriage problem").
- The number of classes that can be scheduled, given a list of which classes can be held in which rooms.
- The number of tasks that can be performed by some machines, given that some tasks can only be performed by some of the machines.
- We make a bipartite graph $G = (V, E)$ where $V = L \cup R$ such that all edges go between L and R.

Bipartite Matching

- A matching is a subset of the edges $M \subseteq E$ such that for all $v \in V$, zero or one edges of M are incident on v.
- 0: v is unmatched; 1: v is matched; > 1 is not allowed.
- Here is one example with two solutions.
- **Goal:** Find the matching(s) of maximum cardinality: $|M| \geq |M'|$ for all matchings M'.



Bipartite Matching

- Add two nodes, s and t.
- Connect s to all the nodes on the L side.
- Connect all the nodes on the R side to t.
- Assign $c(u, v) = 1$ for all $(u, v) \in E$.
- Find maximum flow.