Graphs : Flows and Cuts[1]

In a directed graph G, given two vertices we can do DFS (or BFS) to figure out whether there is a path from s to t. What if we want more? What if we wanted multiple paths from s to t which didn't "touch each other", that is, shared edges. Such an issue naturally arises when the graph G is a telecommunication network, and one maintains a back-up path in case one of the edges in the first path fails. Or, perhaps we use both the paths to obtain more throughput when sending packets from s to t. What is the maximum number of such ***disjoint paths*** from s to t? Can we answer these questions?

Remarkably, the answer is yes, and this is covered by the study of *flows* in graphs. This problem is the cornerstone of two areas of mathematics — *combinatorial optimization* and *linear programming* — areas that have literally changed the world and deserve multiple courses by themselves. Let's begin by defining the problem.

Remark: *Before we begin, here is a notation we use throughout this lecture and the remainder of the course. Suppose we have a real number $x(e) \in \mathbb{R}$ associated with every edge e of the graph. And suppose $B \subseteq E$ is a subset of edges. Then we use the shorthand $x(B)$ to denote the sum $\sum_{e \in B} x(e)$.*

1 Flows in a graph

Given a directed graph $G = (V, E)$, a flow is just an assignment of values to edges satisfying certain constraints. The picture to keep in mind when thinking of flows is actually of "pipes" instead of edges, and the "assignment" is the rate at which some "continuous" fluid (let us say water, but it could be data, oil, or what have you) is flowing through these pipes. The water could be coming from somewhere (sources), it could be going somewhere (sinks), there could be accumulation (excesses), and individual pipes may not be able to handle more than some rate (capacities). All these jargon is formalized below.

Definition 1 (s, t Flow Network, or simply, Flow Network)**.** A flow network (G, s, t, u) consists of a directed graph $G = (V, E)$, two specified vertices s, t, and a capacity function $u : E \to \mathbb{Z}_{\geq 0}$ on edges of the graph. The vertex s is called the *source* vertex and the vertex t is called the *sink* vertex.

Definition 2 (Feasible Flow)**.** A feasible/valid/standard *flow* in a flow network is an assignment $f : E \to \mathbb{R}_{\geq 0}$ satisfying the following two constraints:

- (Capacity Constraints) For every edge $e \in E$, $0 \leq f(e) \leq u(e)$.
- (Conservation Constraints) For every vertex $v \neq \{s, t\}$, $\sum_{(u,v) \in E} f(u, v) = \sum_{(v,w) \in E} f(v, w)$.

The first constraint limits the amount of flow any edge by the capacity. The second says that for any non-source, non-sink vertex, the total flow coming into a vertex v is the total flow leaving a vertex v. If you have taken a physics course, then the current in a circuit satisfies the conservation constraints except at the two ends of the battery. Indeed, another jargon for current is "electrical flow".

Figure 1 shows an example of a network. The capacities $u(e)$ are shows in bold blue. The flow values are shown in green.

[1] *Lecture notes by Deeparnab Chakrabarty. Last modified : 19th Mar, 2022*
These have not gone through scrutiny and may contain errors. If you find any, or have any other comments, please email me at deeparnab@dartmouth.edu. Highly appreciated!

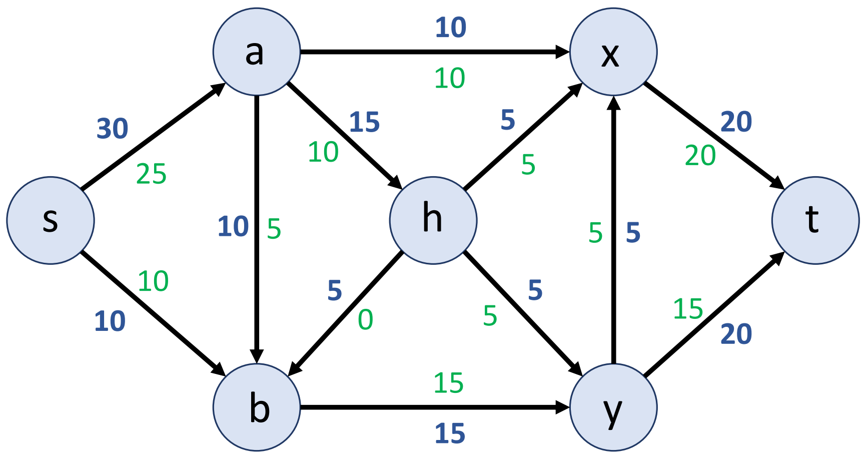

Figure 1: Example of a feasible/valid flow in a flow network. The bold blue numbers are the capacities $u(e)$, and the green numbers are the flow $f(e)$. Verify that conservation constraints hold.

Definition 3 (Excess)**.** Given any assignment $f : E \to \mathbb{R}_{\geq 0}$, we define the following excess_f vector for *every* vertex $v \in V$.

$$\mathsf{excess}_f(v) = \sum_{(u,v)\in E} f(u,v) - \sum_{(v,w)\in E} f(v,w)$$

That is, the $\mathsf{excess}_f(v)$ is the total "in-flow" into v, that is, the total flow coming into v minus the total "out-flow" from v, that is, the total flow coming out of v. Taking the water metaphor again, $\mathsf{excess}_f(v)$ is the rate at which surplus water accumulates at the vertex v if the flow is governed by the assignment f.

Claim 1. Given any assignment $f : E \to \mathbb{R}_{\geq 0}$, we have $\sum_{v\in V} \mathsf{excess}_f(v) = 0$.

Proof. Before proving this, let us see that the claim is consistent with the water metaphor: it says that the total accumulation over all nodes of the vertices must equal the total deficit. This should make sense: if there is water accumulating at some point, then water must be draining at some other point. Perhaps this helps in visualizing the proof below.

By definition, $\mathsf{excess}_f(v) = \sum_{(u,v)\in E} f(u,v) - \sum_{(v,w)\in E} f(v,w)$, and therefore

$$\sum_{v\in V} \mathsf{excess}_f(v) = \sum_{v\in V} \left(\sum_{(u,v)\in E} f(u,v) - \sum_{(v,w)\in E} f(v,w) \right)$$

Now fix any edge (x,y) and consider the flow $f(x,y)$. In the double-summation present in the RHS of the above equation, how many times does the term $f(x,y)$ appear? Yes, you are correct: exactly *two* times. Once it appears when we consider the outer summation for $x \in V$ where it appears in the inner summation $-\sum_{(x,w)\in E} f(x,w)$ with a ***negative*** sign. It appears once more when we consider the outer summation for $y \in V$ where it appears in the inner summation $\sum_{(u,y)\in E} f(u,y)$ with a ***positive*** sign. Positive and negative cancel each other, and we get a 0. Make sure you understand this proof before moving on. $\square$

Going back to the definition of a feasible flow, we see that a flow is a feasible flow if (a) capacity constraint holds at every edge, and (b) the excess at any *non-source, non-sink* vertex is exactly 0. Furthermore, by Claim 1, the excess at the sink t is equal to the negative of the excess at the source s. This value is called the value of the flow: the total "water" flowing into the sink.

Definition 4 (Value of a flow)**.** The *value* of a feasible flow is the total excess at the sink, which equals the negative of the total excess at the source. This will be denotes as $\mathsf{val}(f)$.

Remark: *Although we have been talking about directed graphs, the above also makes sense for undirected graphs. However, remember that flows are* directed *constructs. That is, $f(u,v) = 1$ doesn't imply $f(v,u) = 1$. More precisely, the flow is defined on ordered pairs of V. In an undirected graph, we could have both $f(u,v)$ and $f(v,u)$ non-zero, but their sum must add up to less than that edge's capacity. For simplicity, we will stick to directed graphs.*

Now we are ready to state the *maximum flow problem*.

MAX s,t FLOW
Input: A flow network (G,s,t,u).
Output: A feasible flow $f : E \to \mathbb{R}_{\geq 0}$ of *maximum* value.

What does flows have to do with "disjoint paths" in a directed graph (the illustrative problem we started with)? To see the connection, consider *unit capacities* on every edge of G. That is, $u(e) = 1$ for all $e \in E$. Note that if there are k edge-disjoint paths from s to t in the graph, then we can send a flow of value k from s to t: just assign $f(e) = 1$ to all edges participating in the paths. Do you see why this is a flow of value k? Figure 2 shows an illustration.

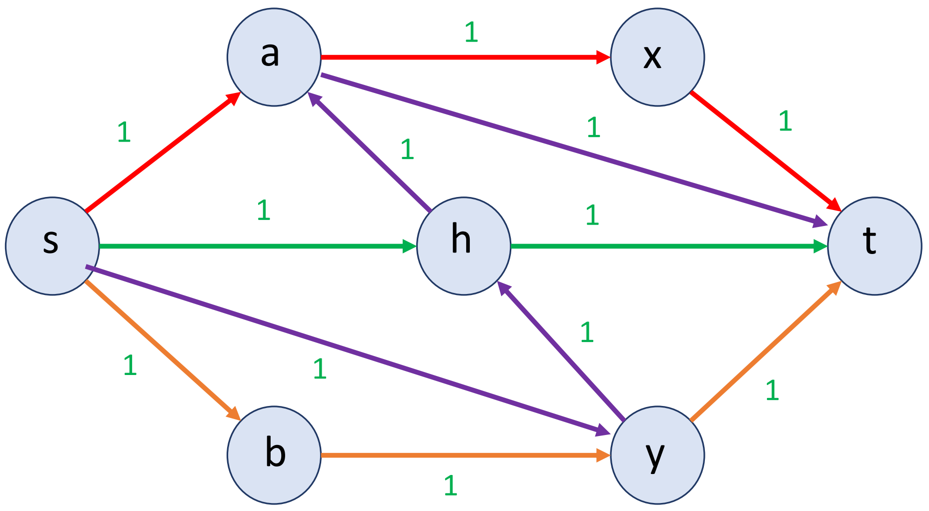

Figure 2: Four edge-disjoint s,t paths shown in four different colors. They are (s,a,x,t), (s,h,t), (s,y,h,a,t), and (s,b,y,t). A flow of value 4 also shown.

Contrapositively, given a feasible flow of value k, if all the $f(e)$'s were 0 or 1, then it is not too hard to see that they lead to k edge-disjoint paths. We start from s taking any edge with $f(e) = 1$. This leads us to a vertex v. If v is not t, then by conservation there must exist another out-edge with $f(e) = 1$. Repeating this will take us to t. Then we delete this path and repeat. Why should flow values be 0 or 1? A priori, there is no reason. However, later on we will prove that this indeed is the case. In sum, the maximum flow problem on unit capacity graphs will solve the disjoint paths problem.

2 Cuts in a graph

Now for something completely different. Till now, we have been given a graph G and we want to find a path, or disjoint paths, between two vertices s and t. Next, we look at a dual problem – we want to *remove* edges from the graph such that after the removal, s and t *cannot* have a path between them. Imagine s to be a source of "infection", and t is a vital node which needs protection. The edges of the graph show how the infection can spread. We want to remove edges so that the infection can't reach the vital node t from s. Of course, we would like to remove as few edges as possible. Can we figure out what is the smallest number of edges we need to remove? For an illustration, look at Figure 2 (ignore the colors) and try figuring out how what is the minimum number of edges that needs to be removed to disconnect s from t.

The *minimum* s, t*-cut problem* generalizes the above problem when each edge has a cost (or a capacity), and the goal is to disconnect s from t deleting as small a cost of edges as possible. There is a reason, which will become clear soon, why we call the costs capacities.

Definition 5. Given a flow network (G, s, t, u), a subset $F \subseteq E$ of edges is an s, t*-cut* if there is no path from s to t in $G \setminus F$. The *capacity* of the cut $F \subseteq E$ is defined to be $\mathsf{cap}(F) := \sum_{e \in F} u(e)$. An s, t-cut $F \subseteq E$ is *minimal* if any strict subset $F' \subsetneq F$ is *not* an s, t-cut.

Every ***minimal*** s, t-cut can ***equivalently*** be represented as the *boundary* edges of a subset of vertices.

Definition 6. Given a directed graph $G = (V, E)$ and a a subset $S \subseteq V$ vertices, the *out-boundary* of S, denoted as $\partial^+ S$, is defined to be

$$\partial^+ S = \{(x, y) \in E : x \in S, y \notin S\}$$

Remark: *There is an analogous definition of*

$$\partial^- S = \{(x, y) \in E : x \notin S, y \in S\}$$

which is the t, s *cut edges.*

The following claim shows the equivalence.

Claim 2. The out-boundary of any subset $S \subseteq V$ such that $s \in S, t \notin S$ is an s, t-cut. Any *minimal* s, t-cut is the out-boundary of some subset $S \subseteq V$ with $s \in S, t \notin S$.

Proof. Fix a subset $S \subseteq V$ with $s \in S, t \notin S$. Let $F := \partial^+ S$. Consider the graph $H := G \setminus F$. Suppose, for contradiction, there is a path from s to t in H. Let this path be $s = x_0, x_1, \ldots, x_k = t$ where each (x_i, x_{i+1}) is an edge in H. Since $x_0 \in S$ and $x_k \notin S$, there must exist some i such that $x_i \in S$ and $x_{i+1} \notin S$. This implies (x_i, x_{i+1}) is an edge in $\partial^+ S$. But this implies $(x_i, x_{i+1}) \notin H$. Contradiction.

For the other direction, suppose $F \subseteq E$ is a minimal s, t-cut. Consider the vertices S that are reachable from s in $G \setminus F$. We claim that $F = \partial^+ S$. To see $\partial^+ S \subseteq F$, consider any edge $(u, v) \in \partial^+ S$. Since u is reachable from s in $G \setminus F$, if the edge $(u, v) \notin F$, that is, $(u, v) \in G \setminus F$, then v would be reachable from s in $G \setminus F$ as well. This contradicts that $v \notin S$. To see $F \subseteq \partial^+ S$, we use the first part to say that (a) $\partial^+ S$ is an s, t-cut, and then since it is a subset of F and F is *minimal*, we must have $F = \partial^+ S$. $\square$

Remark: *Henceforth, when we talk about s, t-cuts, we may just think of them as out-boundaries of some subset S which contains s but doesn't contain t.*

MIN s, t CUT
Input: A flow network (G, s, t, u).
Output: An s, t-cut of *minimum* capacity. Equivalently, $\min_{S \subseteq V} \mathsf{cap}\,(\partial^+ S)$.

One trivial example of an s, t-cut is given by all the out-edges incident on s, that is $\partial^+\{s\}$, or all the in-edges incident on t, that is, $\partial^-\{t\}$. These are *upper bounds* on the value of the minimum cut. I hope you can see examples where these *may not* be the minimum cuts. For instance, consider the picture in the left of Figure 3. You may be tempted to say that the minimum s, t-cut has value 4 (all edges are unit capacity, say) since s has four edges leaving, and t has four edges coming in. But the minimum cut is value is 1.

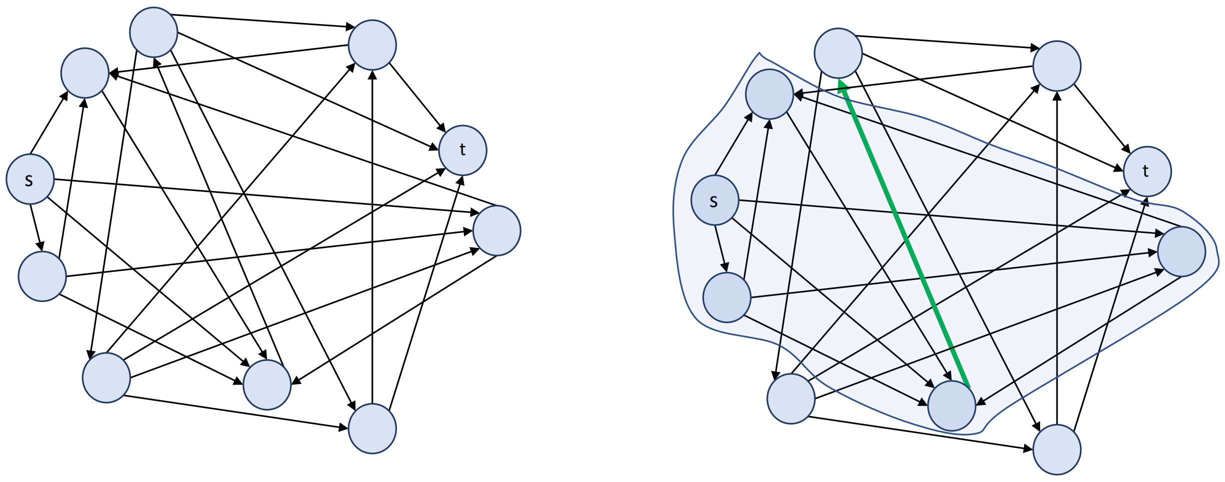

Figure 3: The removal of the green edge (marked in bold) disconnects s from t. This is because it is the only edge going from inside the set S marked as an amoeba, to outside the set S.

3 Flows and Cuts: A Duality

One of the most fascinating results in algorithms is that the above max s,t flow problem and the min s,t cut problem are actually one and the same. Or more correctly, they are two sides of the same coin. They are duals of each other. We are going to build this over the next few lectures. In this lecture, we are going to look at the "easy" direction. We show that given any network (G, s, t, u), the value of *any* feasible s,t flow must be at most the capacity of *any* s,t cut. And in particular, the *max-flow is at most the min-cut.*

Lemma 1. Let f be ***any*** feasible s,t flow and $\partial^+ S$ be ***any*** s,t cut. Then

$$\mathsf{val}(f) := \ \mathsf{excess}_f(t) \leq u(\partial^+ S) \ \ =: \mathsf{cap}(S)$$

Proof. Since we know that $\mathsf{excess}_f(t) = -\mathsf{excess}_f(s)$, the lemma is equivalently asking us to show that

$$u(\partial^+ S) + \mathsf{excess}_f(s) \geq 0$$

To get this, let's add the excesses for every $v \in S$. Why? This is because we wish to argue about the relation between $\mathsf{excess}_f(s)$ and $u(\partial^+ S)$, and the edges participating in the latter may be "far away" from the vertex s. Rather, they involve vertices on the "boundary" of the set S and somehow we need to "propagate" their information to the vertex s which may be deep inside the set. As in the proof of Claim 1 (which I hope you understand well), we get

$$\sum_{v\in S} \mathsf{excess}_f(v) = \sum_{v\in S}\left(\sum_{(u,v)\in E} f(u,v) - \sum_{(v,w)\in E} f(v,w)\right) = \sum_{(x,y)\in E} \left(f(x,y)\cdot \mathbf{1}_{y\in S} - f(x,y)\cdot \mathbf{1}_{x\in S}\right)$$

where $\mathbf{1}_{x\in S}$ is the indicator variable for $x \in S$ and takes value 1 if $x \in S$ and 0 if $x \notin S$. Now note that since f is feasible, the LHS is precisely $\mathsf{excess}_f(s)$. On the other hand, the RHS is precisely $f(\partial^- S) - f(\partial^+ S)$, that is, the total flow on the $\partial^- S$ edges minus the total flow on the $\partial^+ S$ edges. Together, we get,

$$\mathsf{excess}_f(s) = f(\partial^- S) - f(\partial^+ S)$$

See Figure 4, and the caption below the picture, for an illustration of this.
To finish the proof of the lemma, we use the following observations.

Observation 1. (a) Since $f(e) \geq 0$, we get $f(\partial^- S) \geq 0$, (b) since $f(e) \leq u(e)$, we get $f(\partial^+ S) \leq u(\partial^+ S)$.

Taking this together, we get

$$\mathsf{excess}_f(s) \geq -u(\partial^+ S) \ \ \Rightarrow \ \ u(\partial^+ S) + \mathsf{excess}_f(s) \geq 0$$

which completes the proof of the lemma. □

The obvious corollary to the above fact is that the maximum feasible s,t flow in any graph is at most the minimum s,t cut.

Theorem 1 (Weak Duality)**.** In any graph network (G, s, t, u), the maximum s,t flow is at most the minimum s,t cut value.

There is an important consequence of the (proof of) the above lemma which will be used to prove the magical fact: that the maximum s,t flow ***equals*** the minimum s,t cut.

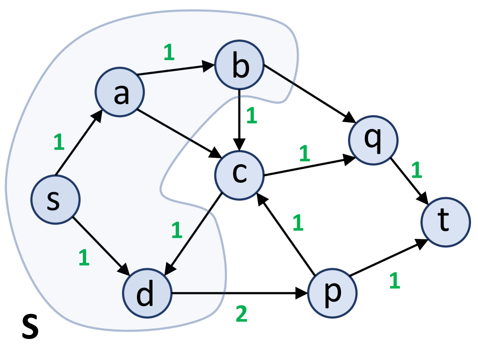

Figure 4: The set $S = \{s, a, b, d\}$. The flow is shown in green numbers. If an edge has no number, the flow is 0. The $\mathsf{excess}_f(s)$ is precisely -2, the negative of the flow leaving s. $f(\partial^+ S) = 3$, on the edges (d, p) and (b, c). The incoming flow into S is $f(\partial^- S) = 1$ on the edge (c, d). The difference is the excess at s.

Theorem 2 (Corollary to Lemma 1)**.** Suppose f is a feasible s, t flow f, and S is an s, t cut S such that

a. $f(e) = u(e)$ for all $e \in \partial^+ S$
b. $f(e) = 0$ for all $e \in \partial^- S$

Then f is a ***maximum*** s, t flow, S is a ***minimum*** s, t cut, and their values are the same.

Proof. The assumptions above imply that the inequalities in Observation 1 are indeed equalities. That would imply $\mathsf{excess}_f(s) + u(\partial^+ S) = 0$ implying $\mathsf{val}(f) = \mathsf{val}(S)$. □