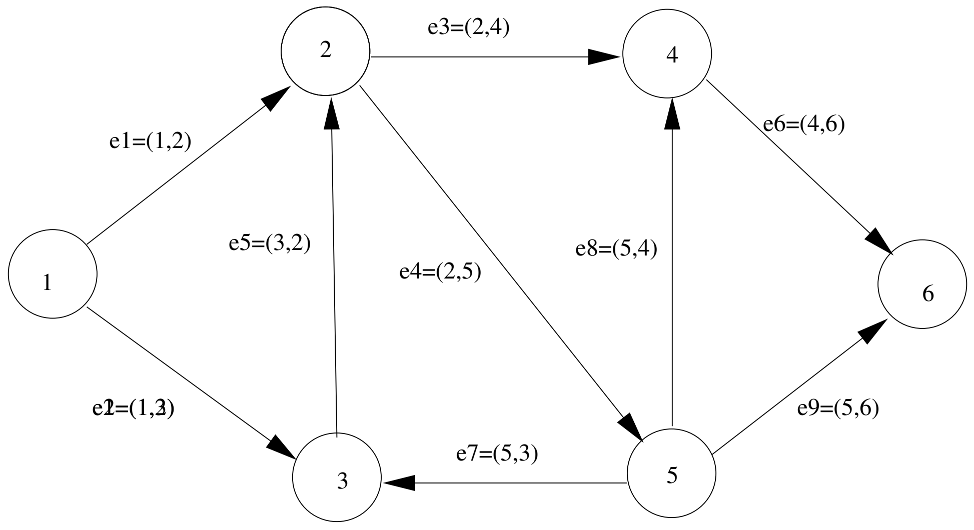

FIGURE 1. A simple network

1. NETWORK FLOWS

1.1. **Flows.** Let $G = (V, E)$ be a network with $|V| = n$ and $|E| = m$. A *flow* f on G is any function of the arcs E of the network, $f : E \to \mathbb{R}^m_+$.

For example, in the network pictured above, we can define

$$f = (5, 6, 3, 4, 3, 7, 2, 4, 5)^T \tag{1.1}$$

so that $f(e_3) = f((2, 4)) = 3$ and $f(e_7) = f((5, 3)) = 2$. A flow on the arc $e_5 = (3, 2)$ is meant to suggest the movement of some underlying quantity from node v_3 to node v_2. Depending on the type of network one is modeling, the flow might represent power, water, cars, goods, people, cash, etc... . In this spirit, we can think of the total flow into a node as the sum of the flows on those arcs whose head is incident to the node minus the flows on those arcs whose tails are incident to the node. Formally, we write

$$\text{total flow into node } v := \sum_{e\in\delta^{\text{in}}(v)} f(e) - \sum_{e\in\delta^{\text{out}}(v)} f(e)\ ,$$

where

$$\delta^{\text{in}}(v) := \{(w, v)\ |(w, v) \in E\} \quad \text{and} \quad \delta^{\text{out}}(v) := \{(v, w)\ |(v, w) \in E\}\,.$$

Similarly, we have

$$\text{total flow out of node } v := \sum_{e\in\delta^{\text{out}}(v)} f(e) - \sum_{e\in\delta^{\text{in}}(v)} f(e) = -(\text{total flow into node } v)\ .$$

We say that the *flow is conserved at node* v if

$$\sum_{e\in\delta^{\text{in}}(v)} f(e) = \sum_{e\in\delta^{\text{out}}(v)} f(e),$$

or, equivalently, the net flow into v is zero. To illustrate these ideas, consider the network given in Figure 1 and the flow on this network given by (1.1). We have that

$$\text{total flow into } v_2 = (5+3) - (3+4) = 1,$$

with

$$\delta^{\text{in}}(v_2) = \{e_1, e_5\} \quad \text{and} \quad \delta^{\text{out}}(v_2) = \{e_3, e_4\} .$$

The relationship between flows and and nodes is nicely expressed using the node-arc incidence matrix of a network. For example, consider the node-arc incidence matrix for the network given in Figure 1,

$$A = \begin{bmatrix} -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & -1 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} .$$

If we now multiply the flow given in (1.1) by A, we get

$$Af = \begin{bmatrix} -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & -1 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} 5 \\ 6 \\ 3 \\ 4 \\ 3 \\ 7 \\ 2 \\ 4 \\ 5 \end{pmatrix} = \begin{pmatrix} -11 \\ 1 \\ 5 \\ 0 \\ -7 \\ 12 \end{pmatrix} .$$

Note that each component of the vector Af on the right hand-side of this expession equals the net inflow to the corresponding vertex, i.e.

$$\begin{aligned} \text{total flow into } v_1 &= -11 \\ \text{total flow into } v_2 &= 1 \\ \text{total flow into } v_3 &= 5 \\ \text{total flow into } v_4 &= 0 \\ \text{total flow into } v_5 &= -7 \\ \text{total flow into } v_6 &= 12 . \end{aligned}$$

In particular, if for each $v \in V$, we define $A_{v,\cdot}$ to be the row of the node-arc incidence matrix corresponding to the vertex v, then

$$\text{total flow into } v = A_{v,\cdot} f .$$

Hence, the flow f is conserved at node $v \in V$ if $A_{v,\cdot} f = 0$. We say that a node $v \in V$ is a *source* if the total flow into v is negative, and we say that v is a sink if the total flow into v is positive.

Given a subset of nodes $S \subset V$ and a flow on f on E, we define the total flow into S to be the sum of the total flows into each node of S:

$$\begin{aligned}
\text{total flow into } S \ &:= \ \sum_{v \in S} (\text{total flow into } v) \\
&= \ \sum_{v \in S} \left[\sum_{e \in \delta^{\text{in}}(v)} f(e) - \sum_{e \in \delta^{\text{out}}(v)} f(e) \right] \\
&= \ \sum_{v \in S} A_{v,\cdot} f \ .
\end{aligned}$$

We can further simplify this expression using matrix algebra. Given a node $v \in V$ and arc $e \in E$, recall that $A_{v,\cdot}$ is the row of the node-arc incidence matrix associated with the node v. Similarly, let $A_{\cdot,e}$ denote the column of A associated with the arc e and let $A_{v,e}$ denote the element of A associated with node v and arc e so that

$$A_{v,e} = \begin{cases} -1 & \text{if } e = (v,w) \in E, \\ 1 & \text{if } e = (w,v) \in E, \text{ and} \\ 0 & \text{if neither } e = (v,w) \text{ or } e = (w,v) \text{ are in } E. \end{cases}$$

If the nodes and arcs are ordered so that $V = \{v_1, v_2, \ldots, v_n\}$ and $E = \{e_1, e_2, \ldots, e_m\}$, then we write $A_{ij} := A_{v_i,e_j}$, $A_{i,\cdot} := A_{v_i,\cdot}$, and $A_{\cdot,j} := A_{\cdot,e_j}$ for $i = 1, 2, \ldots, n$, $j = 1, 2, \ldots, m$, and $\hat{S} = \{i \mid v_i \in S\}$. Define the *support* for S to be the function of the nodes z_S given by

$$z_S(v) := \begin{cases} 1 & \text{if } v \in S, \text{ and} \\ 0 & \text{if } v \notin S, \end{cases}$$

and, again, if the nodes are ordered,

$$z_{\hat{S}}(i) := \begin{cases} 1 & \text{if } i \in \hat{S}, \text{ and} \\ 0 & \text{if } i \notin \hat{S}. \end{cases}$$

Then the total flow into S is given by

$$\begin{aligned}
\text{total flow into } S \ &= \ \sum_{v \in S} A_{v,\cdot} f \\
&= \ z_S^T \begin{pmatrix} A_{1,\cdot} f \\ A_{2,\cdot} f \\ \vdots \\ A_{n,\cdot} f \end{pmatrix} \\
&= \ z_{\hat{S}}^T A f \qquad (1.2) \\
&= \ (z_{\hat{S}}^T A_{\cdot,1}, z_{\hat{S}}^T A_{\cdot,2}, \ldots, z_{\hat{S}}^T A_{\cdot,m}) f \\
&= \ (\sum_{i \in \hat{S}} A_{i,1}, \sum_{i \in \hat{S}} A_{i,2}, \ldots, \sum_{i \in \hat{S}} A_{i,m}) f \ .
\end{aligned}$$

Consider the terms $\sum_{i\in\hat{S}} A_{i,j},\ j = 1, 2, \ldots, m$ in the final expression for the total flow. If $e_j = (v_i, v_k)$ with both $i, k \in \hat{S}$, then $\sum_{i\in\hat{S}} A_{i,j} = 0$ since the column $A_{\cdot,j}$ contains exactly one 1 and one -1 with both these indices occurring in the index set $\hat{S}$. Consequently, for $e = (v, w) \in E$, we must have

$$z_S^T A_{\cdot,e} = z_S^T A_{\cdot,(v,w)} = \begin{cases} 0 & \text{if } v, w \in S, \\ 0 & \text{if } v, w \in E \setminus S, \\ 1 & \text{if } w \in S \text{ and } v \notin S, \text{ and} \\ -1 & \text{if } v \in S \text{ and } w \notin S. \end{cases}$$

Therefore the expression for the total flow into S can be simplified to

$$\begin{aligned} \text{total flow into } S &= \sum_{(v,w)\in E} f((v,w)) z_S^T A_{\cdot,(v,w)} \\ &= \sum_{e\in\delta^{\text{in}}(S)} f(e) - \sum_{e\in\delta^{\text{out}}(S)} f(e)\ , \end{aligned}$$

where

$$\delta^{\text{in}}(S) := \{(v, w) \in E\ |w \in S,\ v \notin S\} \quad \text{and} \quad \delta^{\text{out}}(S) := \{(v, w) \in E\ |v \in S,\ w \notin S\}\ . \tag{1.3}$$

Finally, observe that

$$\begin{aligned} \text{total flow out of } S &= -(\text{total flow into } S) \\ &= \sum_{e\in\delta^{\text{out}}(S)} f(e) - \sum_{e\in\delta^{\text{in}}(S)} f(e). \end{aligned} \tag{1.4}$$

1.2. **Capacitated Networks and the Max-Flow Problem.** A *capacitated network* $G = (V, E, \mathsf{u})$ is a network having node set V and arc set E along with a non-negative function u of the arcs: $\mathsf{u} : E \to \mathbb{R}_+$. The function u is called a flow capacity on E. A feasible flow on this capacitated network is a flow on E satisfying

$$0 \le f(e) \le \mathsf{u}(e) \quad \forall\ e \in E\ .$$

We now describe the *max-flow problem* for capacitated networks. Let $s, t \in V$ with $s \neq t$ be given. We refer to s as the *source* and t as the *sink*. A flow f on $G = (V, E, \mathsf{u})$ is said to be an s-t flow if f is feasible and is conserved at all nodes other than s and t:

$$0 \le f(e) \le \mathsf{u}(e) \quad \forall\ e \in E \quad \text{and} \quad A_{v,\cdot} f = 0 \quad \forall\ v \in V \setminus \{s, t\}\ . \tag{1.5}$$

The *value* of an s-t flow is the total flow out of s:

$$\text{value}(f) := \text{ total flow out of } s\ = -A_{s,\cdot} f\ .$$

Note that for an s-t flow the total flow out of s necessarily equals the total flow into t. Indeed, since $0 = z_V^T A$, we have from (1.5) that

$$0 = z_V^T A f = \sum_{v\in V} A_{v,\cdot} f = A_{s,\cdot} f + A_{t,\cdot} f,$$

so that value$(f) = -A_{s,\cdot}f = A_{t,\cdot}f$. The max-flow problem on (V, E, u, s, t) is to maximize the value of the flow over all s-t flows :

$$\textbf{Max-Flow Problem:} \qquad \begin{array}{ll} \text{maximize} & -A_{s,\cdot}f \\ \text{subject to} & A_{v,\cdot}f = 0 \quad \forall\, v \in V \setminus \{s,t\} \\ & 0 \le f(e) \le \mathsf{u}(e) \quad \forall\, e \in E\ . \end{array}$$

Note that the max-flow problem is always feasible since the zero flow is always feasible. Moreover, the max-flow problem is always bounded by $A_{t,\cdot}u$, where u is the flow capacity. Finally observe that the max-flow problem is a linear program, and so by the Strong Duality Theorem of Linear Programming there always exist solutions to both the max-flow problem and it dual with the optimal values of these problems coinciding.

1.3. **Capacitated Networks and the Min-Cut Problem.** Let $G = (V, E, \mathsf{u})$ be a capacitated network and let $s, t \in V$ with $s \neq t$. An s-t cut in G is a set of arcs in E of the form $\delta^{\text{out}}(S)$ for a subset of nodes $S \subset V$ with $s \in S$ and $t \notin S$. Recall that the set $\delta^{\text{out}}(S)$, defined in (1.3), is given by

$$\delta^{\text{out}}(S) := \{(v,w) \in E \mid v \in S,\ w \notin S\}\ .$$

Note that any set of arcs of the form $\delta^{\text{out}}(S)$ necessarily forms a directed cut in the capacitated network $G = (V, E, \mathsf{u})$. That is, when the arcs in $\delta^{\text{out}}(S)$ are removed, there is no directed path from from a node in S to any node not in S. The *capacity* of an s-t cut is the sum of the capacities of the arcs in the cut:

$$C(\delta^{\text{out}}(S)) = \sum_{e \in \delta^{\text{out}}(S)} \mathsf{u}(e)\ .$$

A minimum s-t cut is an s-t cut of minimum capacity. The min-cut problem is the problem of finding a minimum capacity cut:

$$\textbf{Min-Cut Problem:} \qquad \begin{array}{ll} \text{minimize} & C(\delta^{\text{out}}(S)) \\ \text{subject to} & S \subset V, s \in S, \text{ and } t \notin S\ . \end{array}$$

In the next lemma, we establish a relationship between the capacity of an s-t cut and the value of an s-t flow .

Lemma 1.1 (Max-Flow Min-Cut Inequality)**.** *For any node set $S \subset V$ such that $s \in S$ and $t \notin S$, and any s-t flow f we have*

$$\textit{value}(f) = \sum_{e \in \delta^{\text{out}}(S)} f(e) - \sum_{e \in \delta^{\text{in}}(S)} f(e) \le \sum_{e \in \delta^{\text{out}}(S)} \mathsf{u}(e) = C(\delta^{\text{out}}(S))\ . \tag{1.6}$$

Proof. The inequality in (1.6) follows immediately from the first equation in (1.6) since $0 \le f(e) \le \mathsf{u}(e)$ for all $e \in E$. Therefore we need only prove the first equation in (1.6). For this, note that

$$\begin{aligned}
\text{value}(f) &= \sum_{e\in\delta^{\text{out}}(s)} f(e) - \sum_{e\in\delta^{\text{in}} s} f(e) \\
&= -A_{s,\cdot}f - \sum_{v\in S\setminus\{s\}} A_{v,\cdot}f \\
&= -z_S^T Af \\
&= \text{total flow out of } S \\
&= \sum_{e\in\delta^{\text{out}}(S)} f(e) - \sum_{e\in\delta^{\text{in}}(S)} f(e),
\end{aligned}$$

where the second equality follows by the conservation of flow at each node in $S \setminus \{s,t\}$, the third and fourth equality follows from (1.2), and the final equality follows from (1.4). □

Lemma 1.1 immediately yields the following corollary.

Corollary 1.2. *The optimal value in the max-flow problem is bounded above by the optimal value in the min-cut problem. Moreover, if f is an s-t flow and $\delta^{\text{out}}(S)$ is an s-t cut for which*

$$\textit{value}(f) = C(\delta^{\text{out}}(S)),$$

then f solves the max-flow problem and $\delta^{\text{out}}(S)$ solves the min-cut problem.

1.4. **Flow Augmenting Paths.** We now consider an algorithm for solving the max-flow problem. The key idea is the notion of a *flow augmenting path.* Consider the following simple capacitated network where the labels on the arcs are the capacities. We wish to solve the max-flow problem on this network beginning with the zero flow, i.e. $f(e) = 0\ \forall\, e \in V$. In this case we take $s = v_1$ and $t = v_4$. The first step is to find an s-t directed path, or s-t path , in the network, that is, a directed path from s to t. Once this path is found, we then push as much flow from s to t along this path subject to the arc capacities.

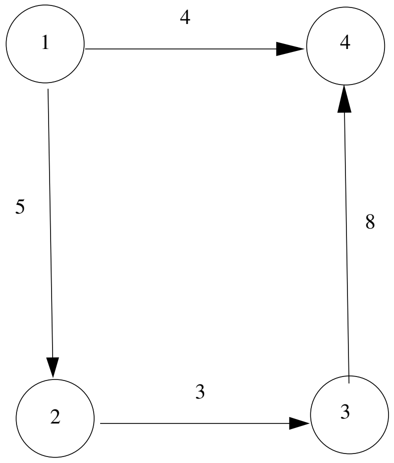

Figure 2. First augmenting path example

Let's try this on the network in Figure 2. Set the initial flow on E to be

$$f_0 = (f_0((1,2)), f_0((2,3)), f_0((3,4)), f_0((1,4)) = (0,0,0,0) \ .$$

Next, beginning at $s = v_1$, find a directed flow from s to t. Using a greedy approach, we choose the arc $(1,2)$ as the first edge in the path since it has the highest flow capacity. The s-t path is then completely determined and is given by

$$P_0 : \qquad s = v_1(1,2)v_2(2,3)v_3(3,4)v_4 = t \ .$$

This path is called a *flow augmenting path* for the the flow f_0, or equivalently, an f_0-augmenting path. The maximum flow that can be pushed along this path equals the minimum of the flow capacities of the edges in the path:

$$\gamma = \min\{\mathsf{u}((1,2)), \mathsf{u}((2,3)), \mathsf{u}((3,4))\} = \min\{5,3,8\} = 3 \ .$$

Pushing this flow along the path P_0 induces a flow on G given by

$$p_0 = (p_0((1,2)), p_0((2,3)), p_0((3,4)), p_0((1,4)) = (3,3,3,0) \ .$$

Adding the flow along this path to our initial flow f_0, we obtain the s-t flow

$$f_1 = f_0 + p_0 = (0,0,0,0) + (3,3,3,0) = (3,3,3,0) \ ,$$

with a total out flow from $s = v_1$ of

$$\text{value}(f_1) = \text{value}(f_0) + \text{value}(p_0) = 0 + 3.$$

Next we find another s-t path along which we can push more flow. While it is possible to push 2 more units of flow from v_1 to v_2, we cannot push any more flow from v_2 to v_3 since we are already at capacity on the arc $(2,3)$. On the other hand, we can push 4 units of flow directly from $s = v_1$ to $t = v_4$ along the arc $(1,4)$. That is, we can push 4 units of flow along the s.t. path $P_1 : \ s = v_1(1,4)v_4 = t$. This path is an f_1-augmenting path and it yields the flow

$$p_1 = (p_1((1,2)), p_1((2,3)), p_1((3,4)), p_1((1,4)) = (0,0,0,4) \ .$$

Adding the flows f_1 and p_1 gives the flow

$$f_2 = f_1 + p_1 = (3,3,3,0) + (0,0,0,4) = (3,3,3,4),$$

with a total out flow from $s = v_1$ of

$$\text{value}(f_2) = \text{value}(f_1) + \text{value}(p_1) = 3 + 4 = 7.$$

Repeating this process, we try to find another s-t path along which we can push more flow. Starting at $s = v_1$ we can only push more flow as far as v_2 at which point we can no longer push any more flow. That is, there does not exist an f_2-augmenting path, and we cannot push flow beyond the two vertices $R = \{v_1, v_2\}$. Note that

$$\delta^{\text{out}}(R) = \{(2,3),(1,4)\}, \text{ with } C(\delta^{\text{out}}(R)) = \mathsf{u}((2,3)) + \mathsf{u}((1,4)) = 3 + 4 = 7 \ .$$

That is, we have found an s-t cut , $\delta^{\text{out}}(R)$ such that

$$\text{value}(f_2) = 7 = C(\delta^{\text{out}}(R)) \quad !$$

Therefore, by Corollary 1.2, the flow f_2 solves the max-flow problem and the cut $\delta^{\text{out}}(R)$ solves the min-cut problem for the capacitated network in Figure 2.

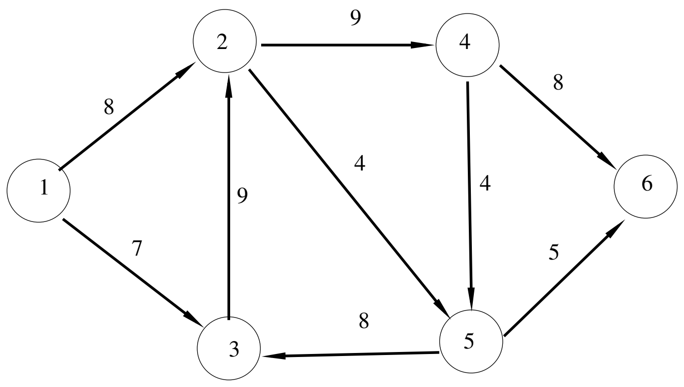

FIGURE 3. Second augmenting path example

Clearly, the graph in Figure 2 is exceptionally trivial, and so we could have arrived at the solution to this max-flow problem as a simple direct observation. However, the manner in which we laboriously constructed the optimal flow is instructive since it outlines a procedure that can be followed on a much more general graph. In brief, the procedure is to successively determine flow augmenting paths that allow us to push more and more flow until no such path exits. This yields a set of vertices beyond which we can no longer push any flow and these vertices constitute a an s-t cut whose capacity equals the value of the current flow thus establishing the optimality of both. The following table recaps all of the information used to obtain this flow.

e	$(1,2)$	$(1,4)$	$(2,3)$	$(3,4)$	flow
u	5	4	3	8	value
f_0	0	0	0	0	0
p_0	3	3	3	0	3
$f_1 = f_0 + p_0$	3	3	3	0	3
p_1	0	0	0	4	4
$f_2 = f_1 + p_1$	3	3	3	4	7

To cement our understanding of this procedure, let us apply it to the somewhat more complicated capacitated network described in Figure 3 with $s = v_1$ and $t = v_6$. To simplify the presentation, we index the arcs as follows:

$$e_1 = (1,2),\ e_2 = (1,3),\ e_3 = (2,4),\ e_4 = (2,5),$$
$$e_5 = (3,2),\ e_6 = (4,5),\ e_7 = (4,6),\ e_8 = (5,3),\ e_9 = (5,6)\ .$$

Again we begin with a the zero flow $f_0 = 0$ which is zero on all arcs. Next we find an f_0-augmenting path P_0. Following the greedy approach used above, we look for the arc exiting $s = v_1$ having greatest capacity, and at each successive vertex choosing the arc with greatest capacity until $t = v_6$ is reached. This gives the path

$$P_0 : \qquad v_1 e_1 v_2 e_3 v_4 e_8 v_6\ ,$$

with path capacity

$$\gamma = \min\{\mathsf{u}(e_1), \mathsf{u}(e_3), \mathsf{u}(e_8)\} = \min\{8, 9, 8\} = 8\ .$$

The corresponding f_0-augmenting flow is

$$p_0 = (8,0,8,0,0,0,8,0,0)^T \ .$$

Adding p_0 to f_0 gives the new flow

$$f_1 = f_0 + p_0 = (8,0,8,0,0,0,8,0,0)^T \ .$$

The value of f_1 is value$(f_1) = 8$:

e	$(1,2)$	$(1,3)$	$(2,4)$	$(2,5)$	$(3,2)$	$(4,5)$	$(4,6)$	$(5,3)$	$(5,6)$	flow
u	8	7	9	4	9	4	8	8	5	value
f_0	0	0	0	0	0	0	0	0	0	0
p_0	8	0	8	0	0	0	8	0	0	8
$f_1 = f_0 + p_0$	8	0	8	0	0	0	8	0	0	8

Repeating this procedure, we choose the f_1-augmenting path

$$P_1 : \qquad v_1 e_2 v_3 e_6 v_2 e_4 v_5 e_9 v_6 \ ,$$

with path capacity

$$\gamma = \min\{\mathsf{u}(e_2), \mathsf{u}(e_6), \mathsf{u}(e_4), \mathsf{u}(e_9)\} = \min\{7,9,4,5\} = 4 \ .$$

The corresponding f_1-augmenting flow is

$$p_1 = (0,4,0,4,4,0,0,0,4)^T \ .$$

Adding p_1 to f_1 gives the new flow

$$f_2 = f_1 + p_1 = \begin{pmatrix} 8\\0\\8\\0\\0\\0\\8\\0\\0 \end{pmatrix} + \begin{pmatrix} 0\\4\\0\\4\\4\\0\\0\\0\\4 \end{pmatrix} = \begin{pmatrix} 8\\4\\8\\4\\4\\0\\8\\0\\4 \end{pmatrix} \ .$$

The value of f_2 is value$(f_2) = $ value$(f_1) + $ value$(p_1) = 8 + 4 = 12$:

e	$(1,2)$	$(1,3)$	$(2,4)$	$(2,5)$	$(3,2)$	$(4,5)$	$(4,6)$	$(5,3)$	$(5,6)$	flow
u	8	7	9	4	9	4	8	8	5	value
f_0	0	0	0	0	0	0	0	0	0	0
p_0	8	0	8	0	0	0	8	0	0	8
$f_1 = f_0 + p_0$	8	0	8	0	0	0	8	0	0	8
p_1	0	4	0	4	4	0	0	0	4	4
$f_2 = f_1 + p_1$	8	4	8	4	4	0	8	0	4	12

We repeat again, but this time something slightly different occurs. Now we can push more flow along e_2 up to the residual (or remaining) capacity $\mathsf{u}(e_2) - f_2(e_2) = 7 - 4 = 3$. Then we can push it along e_3 up to the the residual capacity $\mathsf{u}(e_3) - f_2(e_3) = 9 - 8 = 1$. Then along

e_8 up to the residual capacity $\mathsf{u}(e_8) - f_2(e_8) = 4 - 0 = 4$. Finally, we can push it along e_9 up to the the residual capacity $\mathsf{u}(e_9) - f_2(e_9) = 5 - 4 = 1$. This gives an f_2-augmenting path of

$$P_2 : \qquad v_1 e_2 v_3 e_6 v_2 e_3 v_4 e_7 v_5 e_9 v_6 \ ,$$

with path capacity

$$\gamma = \min\{\mathsf{u}(e_2) - f_2(e_2), \mathsf{u}(e_3) - f_2(e_3), \mathsf{u}(e_8) - f_2(e_8), \mathsf{u}(e_9) - f_2(e_9)\} = \min\{3, 1, 4, 1\} = 1 \ .$$

The corresponding f_2-augmenting flow is

$$p_2 = (0, 1, 1, 0, 1, 1, 0, 0, 1)^T \ .$$

Adding p_2 to f_2 gives the new flow

$$f_3 = f_2 + p_2 = \begin{pmatrix} 8 \\ 4 \\ 8 \\ 4 \\ 4 \\ 0 \\ 8 \\ 0 \\ 4 \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 8 \\ 5 \\ 9 \\ 4 \\ 5 \\ 1 \\ 8 \\ 0 \\ 5 \end{pmatrix} \ .$$

The value of f_3 is value(f_3) = value(f_2) + value$(p_2) = 12 + 1 = 13$:

e	$(1,2)$	$(1,3)$	$(2,4)$	$(2,5)$	$(3,2)$	$(4,5)$	$(4,6)$	$(5,3)$	$(5,6)$	flow
u	8	7	9	4	9	4	8	8	5	value
f_0	0	0	0	0	0	0	0	0	0	0
p_0	8	0	8	0	0	0	8	0	0	8
$f_1 = f_0 + p_0$	8	0	8	0	0	0	8	0	0	8
p_1	0	4	0	4	4	0	0	0	4	4
$f_2 = f_1 + p_1$	8	4	8	4	4	0	8	0	4	12
p_2	0	1	1	0	1	1	0	0	1	1
$f_3 = f_2 + p_2$	8	5	9	4	5	1	8	0	5	13

Repeating, we find that due to capacity limits we can only push flow through the nodes $R = \{v_1, v_2, v_3\}$ where

$$\delta^{\text{out}}(R) = \{e_3, e_4\} \ , \text{ with } C(\delta^{\text{out}}(R)) = \mathsf{u}(e_3) + \mathsf{u}(e_4) = 9 + 4 = 13 \ .$$

Therefore, value$(f_3) = 13 = C(\delta^{\text{out}}(R))$ so Corollary 1.2 tells us that f_3 solves the max-flow problem and $\delta^{\text{out}}(R)$ solves the max-cut problem.

There are many ways to solve the max-flow problem using this algorithm and not all ways will provide the same solution. The method differs depending on how one choses the flow augmenting path at each iteration. Sometimes one must backtrack the flow in order to make a correction to a flow previously allocated. In order to illustrate how this works, we again sole the max-flow problem for the capacitated graph in Figure 3, but this time making different choices in the flow augmenting paths.

Again $s = v_1$ and $t = v_6$ and we take the initial flow to be the zero flow $f_0 = 0$. Our first flow augmenting path is

$$P_0: \qquad v_1(1,3)v_3(3,2)v_2(2,4)v_4(4,5)v_5(5,6)v_6,$$

with $\gamma_0 = \min\{7, 9, 9, 4, 5\} = 4$. The resulting f_0-augmenting flow p_0 and f_1 are given in the following table:

e	$(1,2)$	$(1,3)$	$(2,4)$	$(2,5)$	$(3,2)$	$(4,5)$	$(4,6)$	$(5,3)$	$(5,6)$	flow
u	8	7	9	4	9	4	8	8	5	value
f_0	0	0	0	0	0	0	0	0	0	0
p_0	0	4	4	0	4	4	0	0	4	4
$f_1 = f_0 + p_0$	0	4	4	0	4	4	0	0	4	4

We now choose the flow augmenting path

$$P_1: \qquad v_1(1,3)v_3(3,2)v_2(2,4)v_4(4,6)v_6,$$

with $\gamma_1 = \min\{7-4, 9-4, 9-4, 8\} = 3$. The resulting f_1-augmenting flow p_1 and f_2 are given in the following table:

e	$(1,2)$	$(1,3)$	$(2,4)$	$(2,5)$	$(3,2)$	$(4,5)$	$(4,6)$	$(5,3)$	$(5,6)$	flow
u	8	7	9	4	9	4	8	8	5	value
f_0	0	0	0	0	0	0	0	0	0	0
p_0	0	4	4	0	4	4	0	0	4	4
$f_1 = f_0 + p_0$	0	4	4	0	4	4	0	0	4	4
p_1	0	3	3	0	3	0	3	0	0	3
$f_2 = f_1 + p_1$	0	7	7	0	7	4	3	0	4	7

The next flow augmenting path we choose is

$$P_2: \qquad v_1(1,2)v_2(2,4)v_4(4,6)v_6,$$

with $\gamma_2 = \min\{8, 9-7, 8-3\} = 2$. The resulting f_2-augmenting flow p_2 and f_3 are given in the following table:

e	$(1,2)$	$(1,3)$	$(2,4)$	$(2,5)$	$(3,2)$	$(4,5)$	$(4,6)$	$(5,3)$	$(5,6)$	flow
u	8	7	9	4	9	4	8	8	5	value
f_0	0	0	0	0	0	0	0	0	0	0
p_0	0	4	4	0	4	4	0	0	4	4
$f_1 = f_0 + p_0$	0	4	4	0	4	4	0	0	4	4
p_1	0	3	3	0	3	0	3	0	0	3
$f_2 = f_1 + p_1$	0	7	7	0	7	4	3	0	4	7
p_2	2	0	2	0	0	0	2	0	0	2
$f_3 = f_2 + p_2$	2	7	9	0	7	4	5	0	4	9

The f_3-augmenting path we choose is

$$P_3: \qquad v_1(1,2)v_2(2,5)v_5(5,6)v_6$$

with $\gamma_3 = \min\{8-2, 4, 5-4\} = 1$. The resulting f_3-augmenting flow p_3 and f_4 are given in the following table:

e	$(1,2)$	$(1,3)$	$(2,4)$	$(2,5)$	$(3,2)$	$(4,5)$	$(4,6)$	$(5,3)$	$(5,6)$	flow
u	8	7	9	4	9	4	8	8	5	value
f_0	0	0	0	0	0	0	0	0	0	0
p_0	0	4	4	0	4	4	0	0	4	4
$f_1 = f_0 + p_0$	0	4	4	0	4	4	0	0	4	4
p_1	0	3	3	0	3	0	3	0	0	3
$f_2 = f_1 + p_1$	0	7	7	0	7	4	3	0	4	7
p_2	2	0	2	0	0	0	2	0	0	2
$f_3 = f_2 + p_2$	2	7	9	0	7	4	5	0	4	9
p_3	1	0	0	1	0	0	0	0	1	1
$f_4 = f_3 + p_3$	3	7	9	1	7	4	5	0	5	10

Obviously f_4 is not yet optimal since we know that the optimal value is 13, so there must be a way to push more flow along a flow augmenting path. But this time there will be a difference since we have incorrectly allocated some of the flow in f_4 so we must use an f_4-augmenting flow with at least one negative component. In general a flow augmenting path can push back along an arc only if there is already a positive flow in this arc. Moreover, in this case, we cannot push back more flow than is already flowing forward in this arc. The flow augmenting path we choose for f_4 is

$$P_4 : \qquad v_1(1,2)v_2(2,5)v_5(4,5)v_4(4,6)v_6 \ .$$

Note that here we move *backwards* along the arc $(4,5)$ so we will be pushing flow back along this arc. Corrently, we are pushing a flow of 4 along this arc so we cannot push more than a flow of 4 back along this arc. The computation of γ_4 is as follows:

$$\gamma_4 = \min\{8-2, 4, 4, 8-5\} = 3.$$

The resulting f_4-augmenting flow p_4 and f_5 are given in the following table:

e	$(1,2)$	$(1,3)$	$(2,4)$	$(2,5)$	$(3,2)$	$(4,5)$	$(4,6)$	$(5,3)$	$(5,6)$	flow
u	8	7	9	4	9	4	8	8	5	value
f_0	0	0	0	0	0	0	0	0	0	0
p_0	0	4	4	0	4	4	0	0	4	4
$f_1 = f_0 + p_0$	0	4	4	0	4	4	0	0	4	4
p_1	0	3	3	0	3	0	3	0	0	3
$f_2 = f_1 + p_1$	0	7	7	0	7	4	3	0	4	7
p_2	2	0	2	0	0	0	2	0	0	2
$f_3 = f_2 + p_2$	2	7	9	0	7	4	5	0	4	9
p_3	1	0	0	1	0	0	0	0	1	1
$f_4 = f_3 + p_3$	3	7	9	1	7	4	5	0	5	10
p_4	3	0	0	3	0	-3	3	0	0	3
$f_5 = f_4 + p_4$	6	7	9	8	7	1	8	0	5	13

which is optimal. Notice that this optimal flow is very different from the optimal flow we computed previously.

In the next section we make precise how the algorithm works, especially the procedure for pushing flow back along some arcs.

1.5. **The Ford - Fulkerson Algorithm for Max-Flows.** In this section we provide a formal statement of the algorithm introduced in the previous section and show that it solves the max-flow/min-cut problems. In order to make the presentation precise as well as applicable to arbitrary capacitated network, we need to introduce a number of additional concepts.

Definition 1.3. *Let $G = (V, E, \mathsf{u})$ be a capacitated network. Let $s, t \in V$ with $s \neq t$ and f an s-t flow on G. Set*

$$\begin{aligned} E^F &:= E, \quad \textit{and} && \textit{(forward arcs)} \\ E^B &:= \{(w, v) \,|(v, w) \in E\}\ , && \textit{(backward arcs)} \end{aligned}$$

and $E^A := E^F \cup E^B$. Define the f residual capacities to be

$$\mathsf{u}_f(e) := \begin{cases} \mathsf{u}(e) - f(e) & e \in E^F\ , \\ f(e) & e \in E^B\ . \end{cases}$$

Then the capacitated network $G_f := (V, E^A, \mathsf{u}_f)$ is called the f-augmenting capacitated network. An s-t path P_f in G_f is said to be an f-augmenting path if

$$\gamma(P_f) := \min\{\mathsf{u}_f(e) \,|e \in P_f\} > 0\ .$$

An f-augmenting path P_f on G_f defines an f-augmenting flow p_f on G by

$$p_f((u, v)) := \begin{cases} \gamma(P_f) & (u, v) \in E^F \textit{ and } (u, v) \in P_f, \\ -\gamma(P_f) & (v, u) \in E^B \textit{ and } (v, u) \in P_f, \\ 0 & (u, v) \notin P_f \textit{ and } (v, u) \notin P_f\ . \end{cases}$$

[Ford-Fulkerson Max-Flow Algorithm]

Initialization: Let $G = (V, E, \mathsf{u})$ be a capacitated network. Let $s, t \in V$ with $s \neq t$ and f_0 an s-t flow on G, e.g. f_0 can be taken to be the zero flow. Set $v_0 := \text{value}(f_0)$ and $k = 0$.

Step 1: [Find an f_k-augmenting path P_k.] Let P_k be an f_k-augmenting path in capacitated network G_{f_k}. That is, P_k is an s-t path in G_{f_k} for which

$$\gamma_k := \min\{\mathsf{u}_f(e) \,|e \in P_k\} > 0\ .$$

If no such path exists, then STOP.

Step 2: [Define a flow on P_k.] Define a flow p_k on G by

$$p_k((u, v)) := \begin{cases} \gamma_k & (u, v) \in E^F \text{ and } (u, v) \in P_k, \\ -\gamma_k & (v, u) \in E^B \text{ and } (v, u) \in P_k, \\ 0 & (u, v) \notin P_k \text{ and } (v, u) \notin P_k\ . \end{cases}$$

(Note that the flow p_k is not necessarily an s-t flow in G since it is not necessarily non-negative.)

Step 3: [Update the flow.] Set $f_{k+1} := f_k + p_k$, $v_{k+1} := v_k + \gamma_k$, and $k := k + 1$, and return to Step 1.

The algorithm as described differs from the two examples provided since it allows the possibility reallocating from one arc to another. That is, we allow the f_k-augmenting flows to take negative values. In many cases this is necessary in order to obtain an optimal flow. Before showing that this algorithm solves the max-flow problem, let us more closely examine the algorithm by showing that all of the flows that it construct are indeed s-t flows and that the value of these flows are given by the v_k's.

Consider the flow p_k defined in Step 2 of the algorithm. We first show that p_k conserves flow at all nodes other than s and t. If $v \neq P_k$, then the flow is conserved at v by definition since the flow in p_k only enters of leaves the nodes in P_k. Let $v \in P_k$ with $s \neq v \neq t$. Then there exists nodes $u, w \in V$ such that ue_1ve_2w where $e_1, e_2 \in E^A$ is the snippet of the path P_k passing through v. The following 4 scenarios are possible:

Path in G	flow out of v
$ue_1ve_2w = u(u,v)v(v,w)w$	$-\gamma_k + \gamma_k = 0$
$ue_1ve_2w = u(v,u)v(v,w)w$	$-\gamma_k + \gamma_k = 0$
$ue_1ve_2w = u(u,v)v(w,v)w$	$-\gamma_k - (-\gamma_k) = 0$
$ue_1ve_2w = u(v,u)v(w,v)w$	$-\gamma_k - (-\gamma_k) = 0$

Hence, p_k conserves flow at all nodes other than s and t. We now show that the value of p_k is γ_k. Suppose $sev \in P_k$ for some $e \in E^A$ and $v \in V$. The following two scenarios are possible:

Path in G	flow out of s
$sev = s(s,v)v$	$\text{value}(p_k) = \gamma_k$
$sev = s(v,s)v$	$\text{value}(p_k) = -(-\gamma_k) = \gamma_k$.

In particular, this implies that

$$\text{value}(f_{k+1}) = \text{value}(f_k + p_k) = -A_{s\cdot}(f_k + p_k) = -A_{s\cdot}f_k - A_{s\cdot}p_k = v_k + \gamma_k = v_{k+1} \ .$$

Finally, we show that $f_{k+1} = f_k + p_k$ is an s-t flow in G whenever f_k is an s-t flow in G. Since p_k conserves flow at all nodes other than s and t we need only check that f_{k+1} is feasible for G, that is, we need only show that $0 \leq f_{k+1}(e) \leq \mathsf{u}(e)$ for all $e \in E$. For each $e \in E$, the following two scenarios are possible:
For $(u, v) \in E^F$:

$$\begin{aligned} f_{k+1}((u,v)) &= f_k((u,v)) + p_k((u,v)) \leq f_k((u,v)) + (\mathsf{u}((u,v)) - f_k((u,v))) = \mathsf{u}((u,v)) \quad \text{and} \\ f_{k+1}((u,v)) &= f_k((u,v)) + p_k((u,v)) \geq f_k((u,v)) + 0 \geq 0 \ , \end{aligned}$$

For $(u, v) \in E^B$:

$$\begin{aligned} f_{k+1}((u,v)) &= f_k((u,v)) + p_k((u,v)) \leq f_k((u,v)) + 0 \leq \mathsf{u}((u,v)) \quad \text{and} \\ f_{k+1}((u,v)) &= f_k((u,v)) + p_k((u,v)) \geq f_k((u,v)) - f_k((u,v)) = 0 \ . \end{aligned}$$

Hence f_{k+1} is feasible for G, and so the Ford-Fulkerson Max-Flow Algorithm moves from one s-t flow to another s-t flow with strictly greater flow value.

We now show that the Ford-Fulkerson Max-Flow Algorithm terminates at an s-t flow f if and only if f solves the max-flow problem.

Theorem 1.4. *Let $G = (V, E, \mathsf{u})$ be a capacitated network and let $s, t \in V$ with $s \neq t$. The Ford-Fulkerson Max-Flow Algorithm applied to the max-flow problem on (G, s, t) terminates* *at an s-t flow f if and only if f solves the max-flow problem. Moreover, if the algorithm terminates at f, then f generates a vertex set $R \subset V$ such that the cut $\delta^{\mathrm{out}}(R)$ solves the min-cut problem with value*$(f) = C(\delta^{\mathrm{out}}(R))$.

Proof. By Step 1 of the algorithm, the Max-Flow Algorithm terminates at an s-t flow f if and only if G_f contains no f-augmenting path P_f. Clearly, if there is an f-augmenting path P_f, then f cannot be optimal since its value can be increased by pushing a positive flow along this path. That is, if f solves the max-flow problem, then there cannot exist an f-augmenting path.

To prove the theorem it remains to show that if there is no f-augmenting path, then f must solve the max-flow problem and this f also generates a cut solving the min-cut problem. We do both of these at the same time by showing that if there is no f-augmenting path, then f generates an s-t cut $\delta^{\mathrm{out}}(R)$ such value$(f) = C(\delta^{\mathrm{out}}(R))$. To this end, let f be an s-t flow in G for which there is no f-augmenting path. Let $R \subset V$ be the set of nodes in $v \in V$ for which there is an s-t path , P_v, in G_f such that

$$\min \left\{ \mathsf{u}_f(e) \;\middle|\, e \in P_v,\ e \in E^A \right\} > 0 \ .$$

By construction $\mathsf{u}_f(e) = 0$ for all $e = (v, w) \in \delta^{\mathrm{out}}_{G_f}(R)$ since otherwise $w \in R$. In particular, $t \notin R$ (here $\delta^{\mathrm{out}}_{G_f}(R)$ is the set of arcs leaving R in the capacitated network G_f). Observe that

$$\delta^{\mathrm{out}}_{G_f}(R) = \left\{ e = (v, w) \;\middle|\, (v, w) \in \delta^{\mathrm{out}}_G(R) \text{ or } (w, v) \in \delta^{\mathrm{in}}_G(R) \right\} .$$

Hence, if $e \in \delta^{\mathrm{out}}_G(R)$, then $e \in E^F$ in G_f with $0 = u_f(e) = u(e) - f(e)$. Similarly, if $e \in \delta^{\mathrm{in}}_G(R)$, then $e \in E^B$ with $0 = u_f(e) = f(e)$. Therefore, by Lemma 1.1,

$$\begin{aligned} \text{value}(f) &= \sum_{e \in \delta^{\mathrm{out}}_G(R)} f(e) - \sum_{e \in \delta^{\mathrm{in}}_G(R)} f(e) \\ &= \sum_{e \in \delta^{\mathrm{out}}_G(R)} \mathsf{u}(e) \\ &= C(\delta^{\mathrm{out}}_G(R)) \ , \end{aligned}$$

which, by Corollary 1.2, proves the theorem. $\square$