AMC 18102 Graph Theory	**Department of Mathematics & Computing**
Winter 2019-2020	**Indian Institute of Technology (ISM), Dhanbad**
Lecture: Network Flows-I	April 06-17, 2020
Instructor: **D. Pradhan** `<dina@iitism.ac.in>`	Prepared by: D. Pradhan

1.1 Flow in a network

Definition 1. A *network* or s, t-*network* $N = (G, s, t, c)$ consists of a simple digraph $G = (V, E)$ with two distinguished vertices s, t, called the *source* and *sink* respectively, and a capacity function $c : E \longrightarrow \mathbb{R}^+$. We may assume that G is a simple digraph: it has no loops, and that for every $x, y \in V$ there is at most one edge of the form $\overrightarrow{xy}$ and at most one edge of the form $\overrightarrow{yx}$.

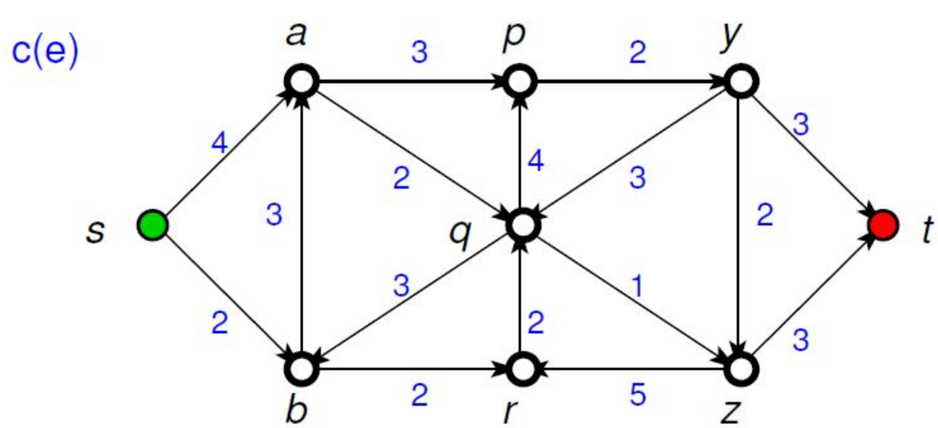


Figure 1.1: An example of a network and its capacity function

We want to think of a network as modeling a situation where stuff (data, traffic, liquid, electrical current etc.) is flowing from source s to sink t. The capacity of an edge is the amount of stuff that can flow through it (or perhaps the amount of stuff per unit time). This is a very general model that can be specialized to describe cuts, connectivity, matchings and other things in directed and undirected graphs.

A **flow** on N is a function $f :\longrightarrow \mathbb{R}$ that satisfies the constraints

(1.1) $$0 \leq f(e) \leq c(e) \text{ for all } e \in E \text{ (the \textit{capacity constraints})}$$

(1.2) $$f_{\text{in}}(v) = f_{\text{out}}(v) \text{ for all } v \in V \setminus \{s,t\} \text{(the \textit{conservation constraints}),}$$

where for every $v \in V$, $f_{\text{in}}(v) = \sum_{e=\overrightarrow{uv}} f(e)$ and $f_{\text{out}}(v) = \sum_{e=\overrightarrow{vu}} f(e)$

The function $f(e)$ is of course a flow. Here is a nontrivial example. We will consistently use blue for capacities and red for flows.

Note that the conservation constraints say that flow cannot accumulate at any internal vertex.

The **value** $|f|$ of a flow f is the net flow into the sink:

(1.3) $$|f| = f_{\text{in}}(t) - f_{\text{out}}(t) = f_{\text{out}}(s) - f_{\text{in}}(s)$$

To see the second equality, note that

$$\sum_{e \in E} f(e) = \sum_{v \in V} f_{\text{in}}(v) = \sum_{v \in V} f_{\text{out}}(v)$$

and by the conservation constraints, most of the summands cancel, leaving only

$$f_{\text{in}}(s) + f_{\text{in}}(t) = f_{\text{out}}(s) + f_{\text{out}}(t)$$

from which the second equality easily follows. Since we are concerned with maximizing $|f|$, we typically assume that s has no in-edges and t had no out-edges, so that (1.3) can be simplified to

(1.4) $$|f| = f_{\text{in}}(t) = f_{\text{out}}(s)$$

The flow f shown in Figure 1.2 has $|f| = 3$.

Max Flow Problem: Given a source-sink network (G, s, t, c), find a flow of maximum value.

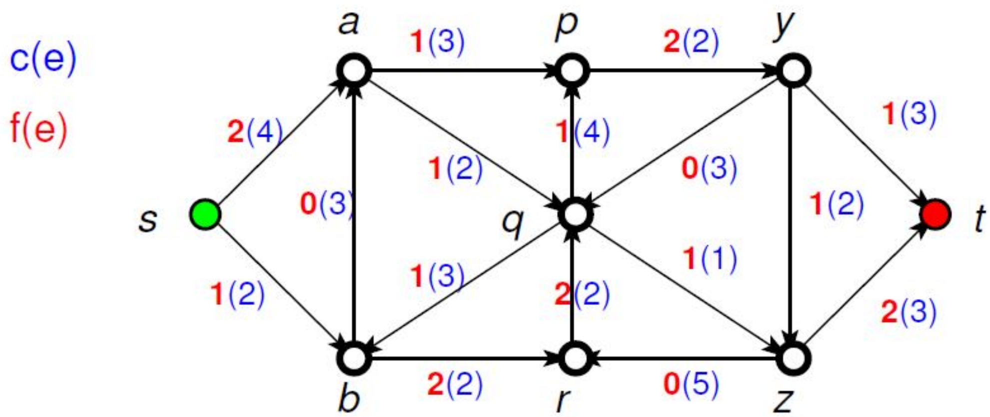


Figure 1.2: A capacity function and a compatible flow

We need a way of increasing the value of a given flow f, or showing that no such way exists. (This ought to remind you of the Augmenting Path Algorithm.) The naive way is to look for an "f-augmenting path"-an s, t-path $P \subseteq N$ in which no edge of P is being used to its full capacity, that is, such that $f(e) < c(e)$ for all $e \in P$. In this case, we can increase all flows along the path by some nonzero amount ε so as to preserve the conservation and capacity constraints, and increase the value of the flow by ε.

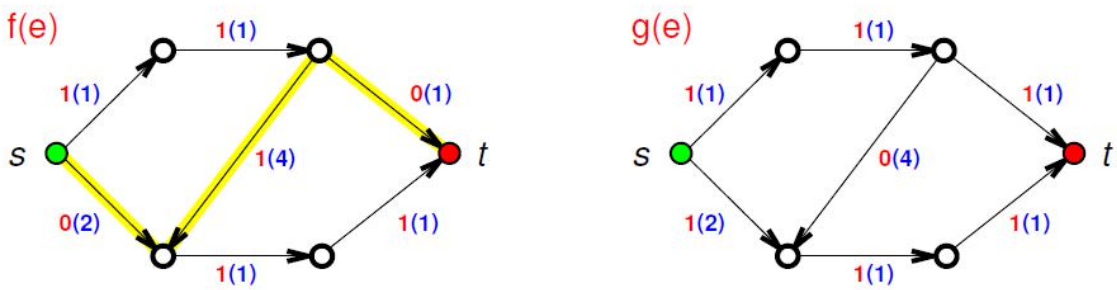


Figure 1.3: Two "maximal" flows, one with an augmenting path (highlighted).

However, there can be nonmaximum flows where no such path P exists. Consider the network shown in Figure 1.3. Continuing the analogy with matchings and the APA (aug path algo), the flow f on the left is "**maximal**", in the sense that there does not exist any flow f_1 such that $|f_1| > |f|$ and $f_1(e) \geq f(e)$ for every $e \in E$. However, it is not maximum: $|f| = 1$, while the flow g on the left has $|g| = 2$.

There is a more general way to increase flow: Allow the augmenting path P to contain edges that point in the wrong direction, but along which the flow is nonzero. As far as the conservation constraints and the value of the flow is concerned, decreasing "backward" flow is equivalent to increasing "forward" flow. The "forward" edges a, c are not being used to full capacity, and the "backward" edge b contains flow "in the wrong direction". So we can define a new flow $\tilde{f}$ by

- $\tilde{f}(a) = f(a) + 1$ $\qquad$ $\tilde{f}(b) = f(b) - 1$ $\qquad$ $\tilde{f}(c) = f(c) + 1$;
- $\tilde{f}(e) = f(e)$ for all other edges e

Then $\tilde{f}$ satisfies the capacity and conservation constraints, and $|\tilde{f}| = |f| + 1$. (In fact $\tilde{f} = g$.)

Definition 2. Let f be a flow in an s, t-network $N = (G, s, t, c)$. Let P be an s, t-path in G, which may include backward edges $\overleftarrow{e}$ as well as forward edges $\overrightarrow{e}$ in P. The **tolerance** of an edge $e \in P$ is defined as

$$\varepsilon(e) = \begin{cases} c(e) - f(e), & \textit{if } \overrightarrow{e} \in P, \\ f(e), & \textit{if } \overleftarrow{e} \in P. \end{cases}$$

The proof of the following lemma is then completely routine.

Lemma 1. *If P is an f-augmenting path, then the function $\tilde{f}$ defined by*

$$\tilde{f}(e) = \begin{cases} f(e) + \varepsilon, & \textit{if } \overrightarrow{e} \in P \\ f(e) - \varepsilon, & \textit{if } \overleftarrow{e} \in P \\ f(e), & \textit{otherwise} \end{cases}$$

is a flow (i.e., it satisfies the capacity and conservation constraints), and $|\tilde{f}| = \varepsilon + |f|$.

The dual problem to the Max-Flow problem is the Min-Cut problem.

Definition 3. Let $N = (G, s, t, c)$ be an s, t-network. A **source-sink cut** is a directed cut of the form $[S, T] = \{xy \in E(G), x \in P, y \in T\}$, where $V(G) = S \cup T, s \in S$, and $t \in T$. (Note that this is a directed graph, so we only include edges from S to T, not from T to S.) The **capacity** of the cut is $c(S, T) = \sum_{e \in [S,T]} c(e)$.

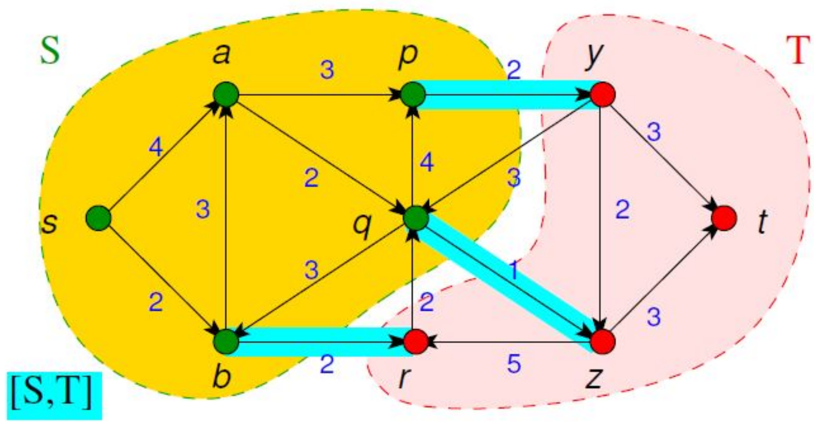


Figure 1.4: Explaining a source-sink cut

In Figure 1.4, $S = \{s, a, b, p, q\}$ (gold) and $T = \overline{S} = \{t, r, y, z\}$ (pink). The resulting source-sink cut is $[S, T] = \{\overrightarrow{br}, \overrightarrow{py}, \overrightarrow{z}\}$ (highlighted in cyan), so $c(S, T) = 2 + 1 + 2 = 5$. Note that the T, S-edges $\overrightarrow{yq}$ and $\overrightarrow{rq}$ are not considered part of the cut.

> **Min-Cut Problem:** Find a source-sink cut of minimum capacity.

A source-sink cut can be thought of as a bottleneck: a channel through which all ow must pass. Therefore, the capacity of any cut should be an upper bound for the maximum value of a flow - this is the "weak duality" inequality, analogous to the easy directions of results such as the König-Egerváry theorem and the various versions of Menger's theorem.

For a flow f and a vertex set $A \subseteq V$, define,

(1.5) $$f(A, \overline{A}) = \sum_{\vec{e} \in [A, \overline{A}]} f(e) - \sum_{\vec{e} \in [\overline{A}, A]} f(e)$$

Lemma 2. *Let f be a flow, and let $A \subseteq V$. Then*

(1.6) $$f(A, \overline{A}) = \sum_{w \in A} (f_{\text{out}}(w) - f_{\text{in}}(w)).$$

In particular, if $[S, T]$ is a source-sink cut, then

(1.7) $$f(S, T) = |f| \leq c(S, T).$$

That is, the Max-Flow and Min-Cut problems are weakly dual.

Proof.

$$\begin{aligned} f(A,\overline{A}) &= \sum_{\vec{e}\in[A,\overline{A}]} f(e) - \sum_{\vec{e}\in[\overline{A},A]} f(e) \\ &= \sum_{e=\overrightarrow{vw},v\in A} f(e) - \sum_{e=\overrightarrow{vw},w\in A} f(e) \\ &= \sum_{w\in A}\left(\sum_{e:\text{head}(e)=w} f(e) - \sum_{e:\text{tail}(e)=w} f(e)\right) \\ &= \sum_{w\in A}(f_{\text{out}}(w) - f_{\text{in}}(w)) \end{aligned}$$

establishing (1.6).

In particular, if $[S,T]$ is a source-sink cut and f is any flow, then $f(S,T) = \sum_{w\in S} f_{\text{out}}(w) - f_{\text{in}}(w) = f_{\text{out}}(s) = |f|$, but on the other hand $f(S,T) \le c(S,T)$ by the capacity constraints (1.1). □

In fact, the Max-Flow and Min-Cut problems are strongly dual. They can be solved simultaneously in finite time by the following simple but very powerful algorithm, due to Ford and Fulkerson.

Ford-Fulkerson algorithm:

Input: a network $N = (G,s,t,c)$
Output: a maximum flow f and minimum s,t-cut $[S,T]$

Initialize $f(e) = 0$ for all e.
Repeat:
  Let S be the set of all vertices reachable from s by an f-augmenting path.
  If $t \in S$ ("breakthrough"), then increase flow along some augmenting path until breakthrough does not occur.
Return the flow f and the cut $[S,\overline{S}]$.

Theorem 1 (The Max-Flow/Min-Cut Theorem -"MFMC")**.** *The Ford-Fulkerson algorithm finishes in finite time and computes a maximum flow and a minimum cut.*

Proof. Since everything in sight is an integer, each instance of breakthrough increases $|f|$ by at least 1. Therefore, the algorithm will terminate in a number of steps equal to or less than the minimum capacity of an s, t-cut.

Let f and $[S, T]$ be the output of the Ford-Fulkerson algorithm. The fact that breakthrough did not occur means that every forward edge of $[S, T]$ is being used to full capacity, and no backward edge has positive flow. That is,

$$f(e) = c(e) \;\; \forall \;\; \overrightarrow{e} \in [S, T] \quad \text{and} \quad f(e) = 0 \;\; \forall \;\; \overleftarrow{e} \in [S, T].$$

But this says exactly that

$$|f| = f(S, T) = c(S, T)$$

and so by weak duality, f is a maximum flow and $[S, T]$ is a minimum source-sink cut. $\square$

Example 1. *Let N be the network shown in Figure 1.5.*

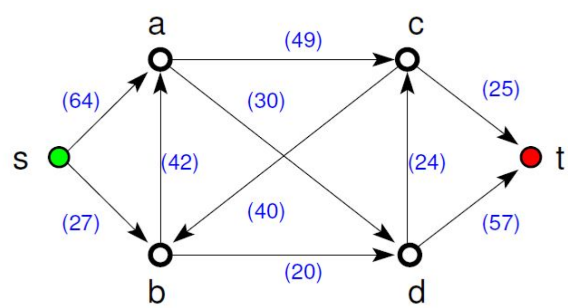
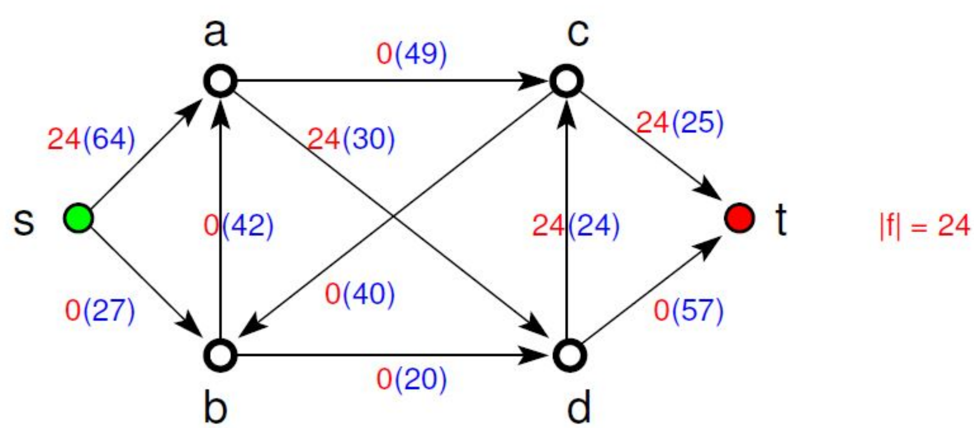


Figure 1.5: Explaining Ford-Fulkerson algorithm

Initialize f to be the zero flow and work through the algorithm. Note that there may be several possible augmenting paths in each iteration, so in that sense the algorithm is not deterministic.

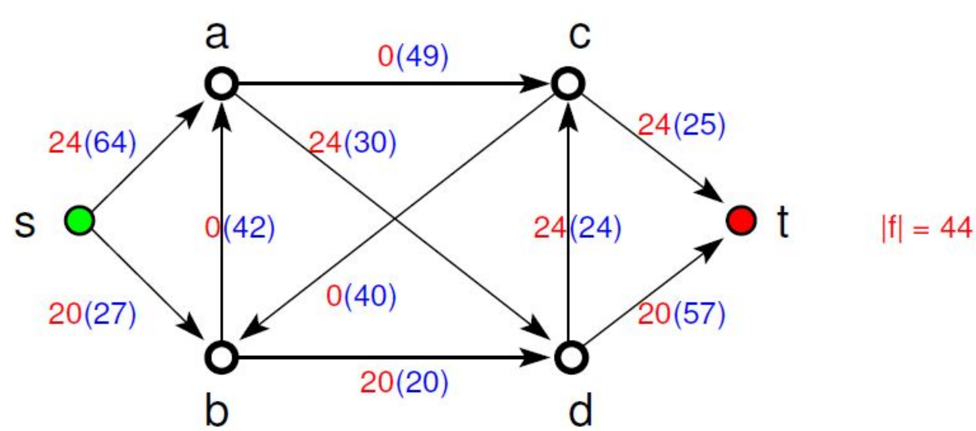
Step 1:

- Augmenting path: $P = s, a, d, c, t$
- Edge tolerances: $\varepsilon(sa) = 64, \varepsilon(ad) = 30, \varepsilon(dc) = 24, \varepsilon(ct) = 25$
- Path tolerance: $\min\{64, 30, 24, 25\} = 24$



Step 2:

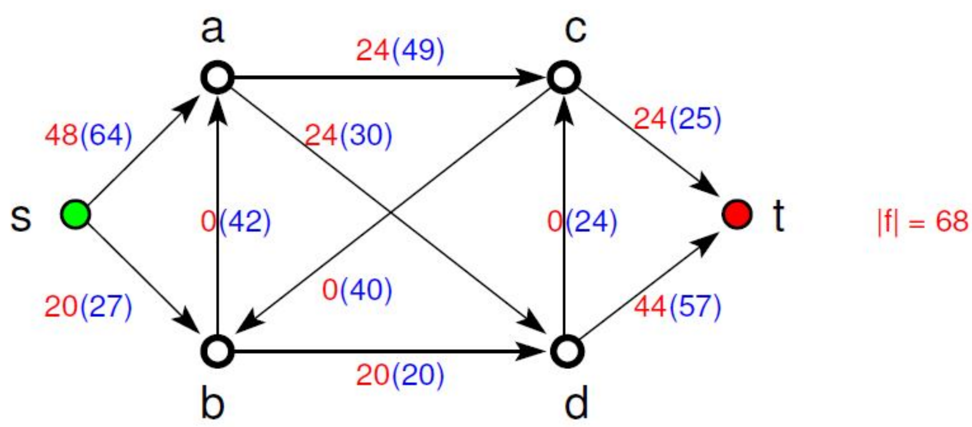
- Augmenting path: $P = s, b, d, t$
- Tolerance: $\varepsilon(P) = \min\{27, 20, 57\} = 20$



Step 3:

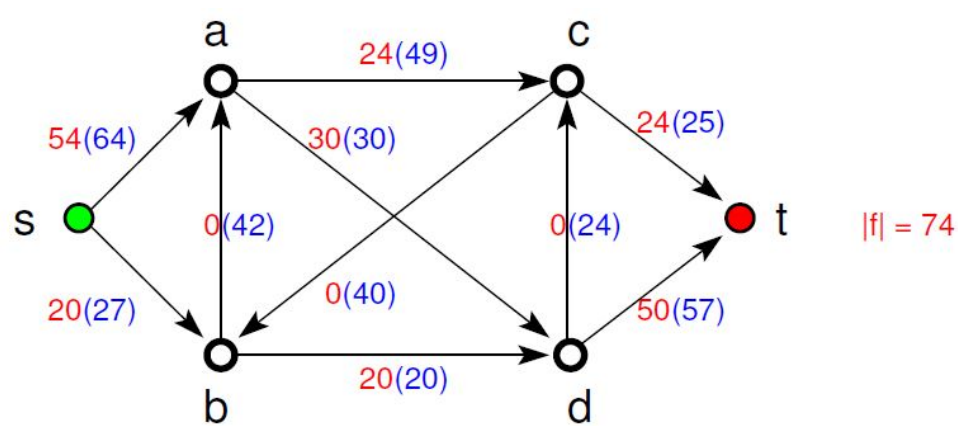
- Augmenting path: $P = s, a, c, d, t$. Note that $\overrightarrow{dc} \in E$, so we have a backward edge.

- Tolerance: $\varepsilon(P) = \min\{40, 49, 24, 37\} = 24$. Note that $\varepsilon(\overleftarrow{cd}) = f(\overrightarrow{dc}) = 24$
- Net flow: Add 24 to $f(\overrightarrow{sa}), f(\overrightarrow{ac}), f(\overrightarrow{dt})$; subtract 24 from $f(\overrightarrow{dc})$

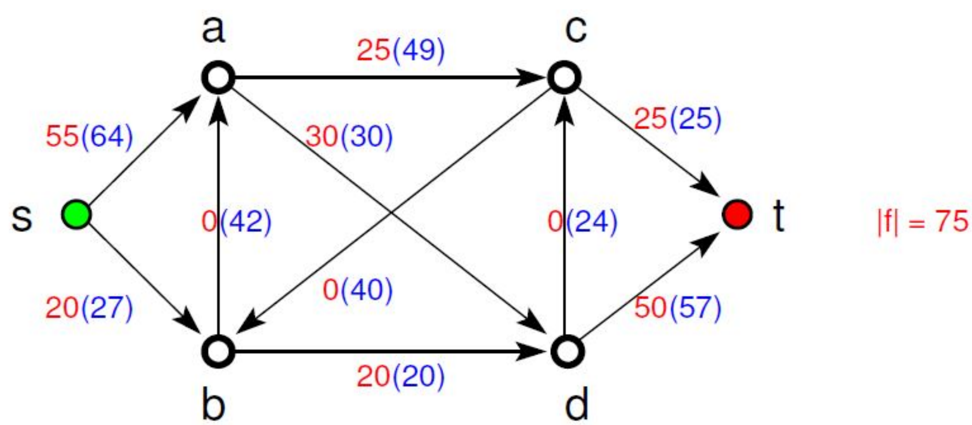


Step 4:

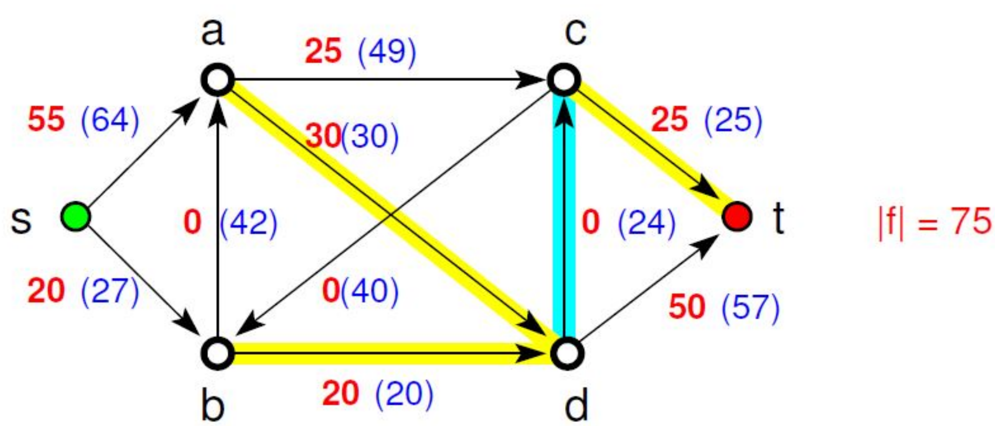
- Augmenting path: $P = s, a, d, t$. Note that $\overrightarrow{dc} \in E$, so we have a backward edge.
- Tolerance: $\varepsilon(P) = \min\{16, 6, 13\} = 6$.



Step 5:
- Augmenting path: $P = s, a, c, t$. Note that $\overrightarrow{dc} \in E$, so we have a backward edge.
- Tolerance: $\varepsilon(P) = \min\{64 - 54, 49 - 24, 25 - 24\} = 1$.



Step 6: At this point in the algorithm, breakthrough fails, since every edge in the cut $[S = \{s, a, b, c\}, T = \{d, t\}$ is either a forward edge being used at capacity (yellow), or a backward edge with flow 0 (blue).



Moreover, $c(S, T) = c(\overrightarrow{ad}) + c(\overrightarrow{bd}) + c(\overrightarrow{ct}) = 75 = |f|$. So f is a maximum flow and $[S, T]$ is a minimum cut. The algorithm has succeeded.