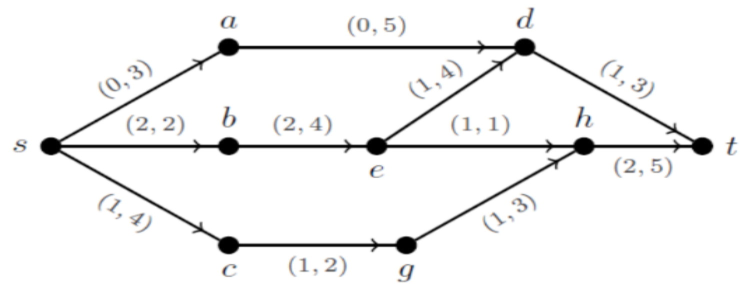
NETWORK FLOW

- This section will focus on a new application for digraphs, one in which items are sent through a network.
- These networks often model physical systems, such as sending water through pipelines or information through a computer network.
- The digraphs we investigate will need a starting and ending location, though there is no requirement for the network to be acyclic.

Below is an example of a network. Each arc is given a two-part label. The first component is the flow along the arc and the second component is the capacity.



G_8

- The names of the starting and ending vertices are reminiscent of a system of pipes with water coming from the source, traveling through some configuration of the piping to arrive at the sink (ending vertex).
- Using this analogy further, we can see that some restraints need to be placed on the flow along an arc.
- For example, flow should travel in the indicated direction of the arcs, no arc can carry more than its capacity, and the amount entering a junction point (a vertex) should equal the amount leaving.

Definition 4.28 For a vertex v, let $f^-(v)$ represent the total flow entering v and $f^+(v)$ represent the total flow exiting v. A flow is ***feasible*** if it satisfies the following conditions:

(1) $f(e) \geq 0$ for all edges e

(2) $f(e) \leq c(e)$ for all edges e

(3) $f^+(v) = f^-(v)$ for all vertices other than s and t

(4) $f^-(s) = f^+(t) = 0$

- The notation for in-flow and out-flow mirrors that for in-degree and out-degree of a vertex, though here we are adding the flow value for the arcs entering or exiting a vertex.
- The requirement that a flow is non-negative indicates the flow must travel in the direction of the arc, as a negative flow would indicate items going in the reverse direction.
- The final condition listed above requires no in-flow to the source and no out-flow from the sink.
- This is not necessary in theory, but more logical in practice and simplifies our analysis of flow problems.

The network G_8 shown above satisfies the conditions for a feasible flow. In general, it is easy to verify if a flow is feasible.

Our main goal will be to find the best flow possible, called the maximum flow.

Maximum Flow:

Definition 4.29 The ***value*** of a flow is defined as $|f| = f^+(s) = f^-(t)$, that is, the amount exiting the source which must also equal the flow entering the sink. A ***maximum flow*** is a feasible flow of largest value.

Intuitive Idea to Find Maximum Flow:

Look back at the flow shown in the network G_8 above, which has a value 3. If we compare the flow and capacity along the arcs, we should see many locations where the flow is below capacity. However, finding a maximum flow is not as simple as putting every arc at capacity this would likely violate one of the feasibility criteria.

- ✓ For example, if we had a flow of 5 along the arc *ad*, we would need the flow along *dt* to also equal 5 to satisfy criteria (3). But in doing so we would violate criteria (2) since the capacity of *dt* is 3.

Question: The main question in regard to network flow is that of optimization, what is the value of a maximum flow?

Answer: We will discuss an algorithm that not only finds a maximum flow but also provides proof that a larger flow cannot be found.

Slack & Chain:

Definition 4.30 Let f be a flow along a network. The ***slack*** k of an arc is the difference between its capacity and flow; that is, $k(e) = c(e) - f(e)$.

- Slack will be useful in identifying locations where the flow can be increased.
- For example, in the network above $k(sa) = 3$, $k(sc) = 3$, and $k(sb) = 0$ indicates that we may want to increase flow along the arcs sa and sc but no additional flow can be added to sb.

The difficult part is determining where to make these additions.

- To do this we will build special paths, called chains, that indicate where the flow can be added.

Definition 4.31 A ***chain*** K is a path in a digraph where the direction of the arcs are ignored.

- In the network shown above, both sadt and sadeht are chains, though only sadeht is not a directed path since it uses the arc ed in the reverse direction.
- We now have all the needed elements for finding the maximum flow. The algorithm below is similar to Dijkstra's Algorithm.

AUGMENTING FLOW ALGORITHM:

The format of the Augmenting Flow Algorithm, described below, is an adaption from that given in [79] and based on the Ford-Fulkerson and Edmonds-Karp Algorithms.

- Vertices will be assigned two-part labels that aid in the creation of a chain on which the flow can be increased.
- The first part of the label for vertex y will indicate one of two possibilities: x− if there is a positive flow along y → x, or x+ if there is slack along the arc x → y, where the former scenario may allow a reduction along the arc yx in order for additional flow along another edge out of x, whereas the latter scenario may allow more flow along the arc xy itself.
- The second part of the label will indicate the amount of flow that could be adjusted along the arc in question.

Augmenting Flow Algorithm

Input: Network $G = (V, E, c)$, with designated source s and sink t, and each arc is given a capacity c.

Steps:

1. Label s with $(-, \infty)$

2. Choose a labeled vertex x.

   (a) For any arc yx, if $f(yx) > 0$ and y is unlabeled, then label y with $(x^-, \sigma(y))$ where $\sigma(y) = \min\{\sigma(x), f(yx)\}$.

   (b) For any arc xy, if $k(xy) > 0$ and y is unlabeled, then label y with $(x^+, \sigma(y))$ where $\sigma(y) = \min\{\sigma(x), k(xy)\}$.

3. If t has been labeled, go to Step (4). Otherwise, choose a different labeled vertex that has not been scanned and go to Step (2). If all labeled vertices have been scanned, then f is a maximum flow.

4. Find an $s - t$ chain K of slack edges by backtracking from t to s. Along the edges of K, increase the flow by $\sigma(t)$ units if they are in the forward direction and decrease by $\sigma(t)$ units if they are in the backward direction. Remove all vertex labels except that of s and return to Step (2).

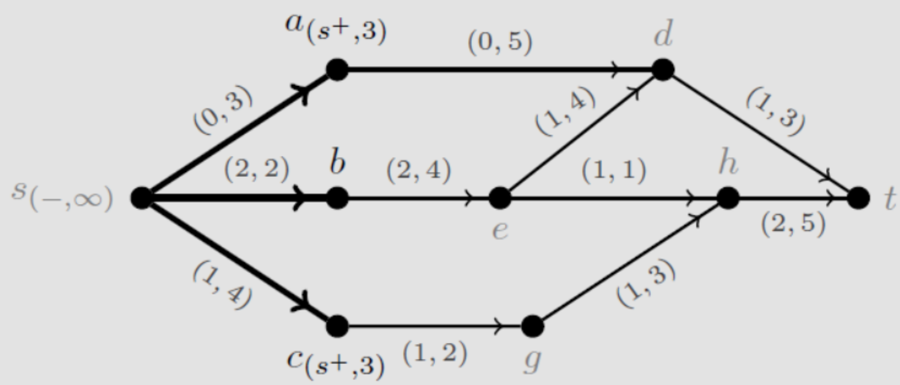
Output: Maximum flow f.

Note: It is important that in Step 2 when we are labeling the neighbors of a vertex *x* that we first consider arcs into *x* from unlabeled vertices that have positive flow (part a) and then the arcs out of *x* to unlabeled vertices with positive slack (part b). These are used to find a chain from *s* to *t* onto which flow can be added.

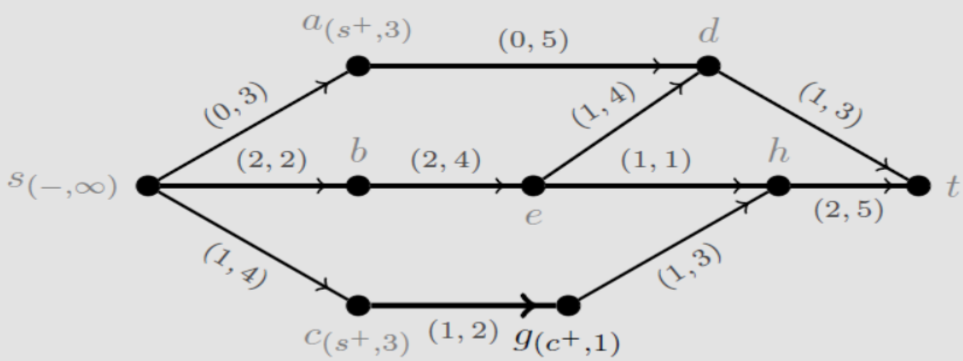
Example 4.6 Apply the Augmenting Flow Algorithm to the network G_8 on page 189.

Solution: As before, the edges under consideration in a given step will be shown in bold.

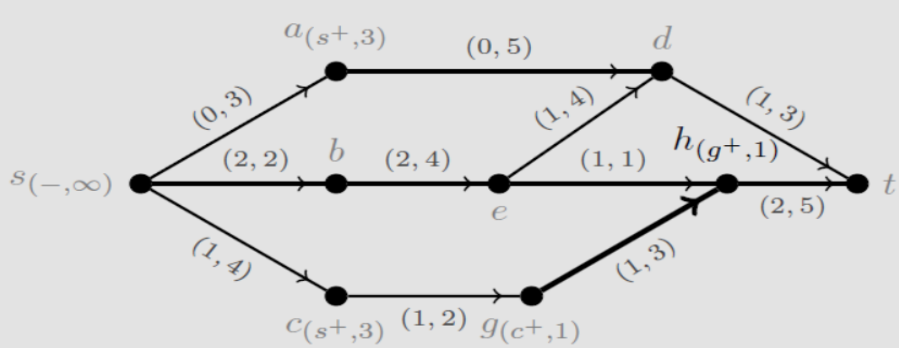
Step 1: Label s as $(-, \infty)$ and let $x = s$. As there are no arcs to s we will only consider the arcs out of s, of which there are three: sa, sb, and sc, which have slack of 3, 0, and 3, respectively. We label a with $(s^+, 3)$, b is left unlabeled since there is no slack on sb, and c is labeled $(s^+, 3)$.



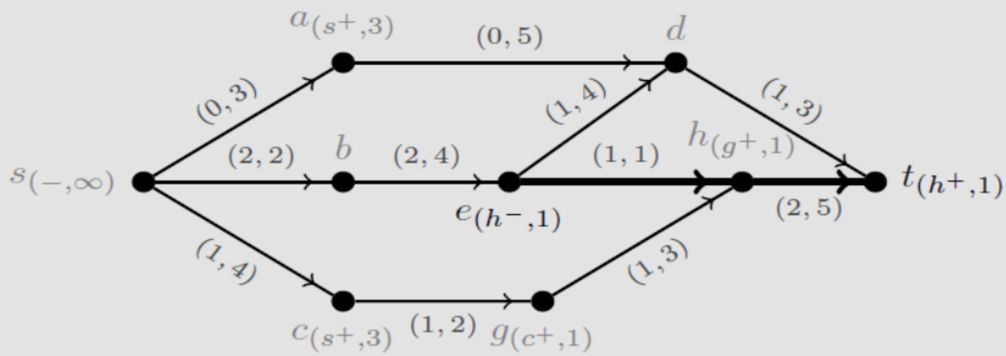
Step 2: As t is not labeled, we will scan either a or c; we choose to start with c. Since the only arc going into c is from a labeled vertex, we need only consider the edges out of c, of which there is only one, cg, with slack of 1. Label g as $(c^+, 1)$ since $\sigma(g) = \min\{\sigma(c), k(cg)\} = \min\{3, 1\} = 1$.



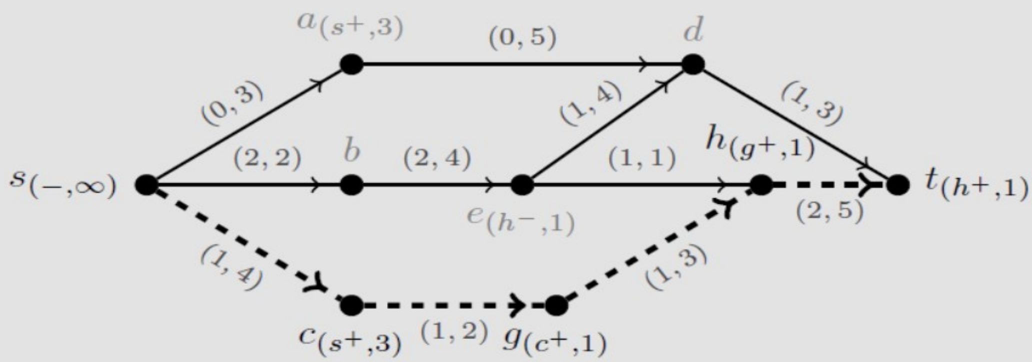
Step 3: We can scan either a or g; we choose g. We need only consider the edges out of g, of which there is only one, gh, with slack of 2. Label h as $(g^+, 1)$ since $\sigma(h) = \min\{\sigma(g), k(gh)\} = \min\{1, 2\} = 1$.



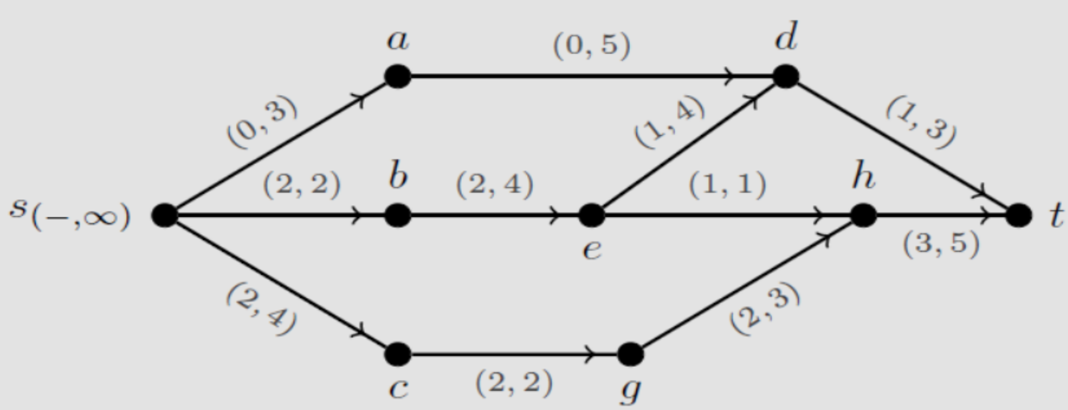
Step 4: We can scan either a or h; we choose h. There is one unlabeled vertex with an arc going into h, namely e, which gets a label of $(h^-, 1)$ since $\sigma(e) = \min\{\sigma(h), f(eh)\} = \min\{1, 1\} = 1$. The only arc out of h is ht, with slack of 3. Label t as $(h^+, 1)$ since $\sigma(t) = \min\{\sigma(h), k(ht)\} = \min\{1, 3\} = 1$.



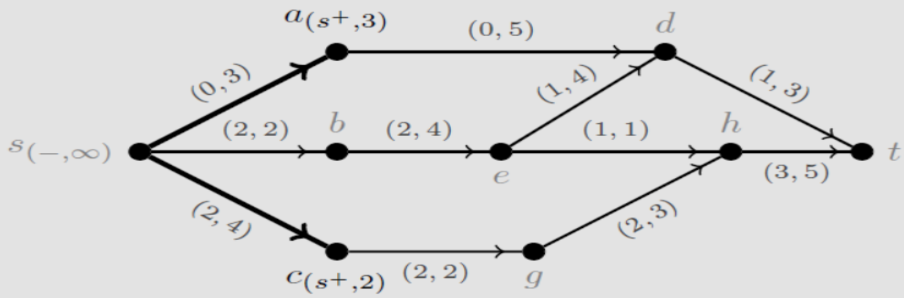
Step 5: Since t is now labeled, we find an $s - t$ chain K of slack edges. Backtracking from t to s gives the chain $s\,c\,g\,h\,t$ as shown below.



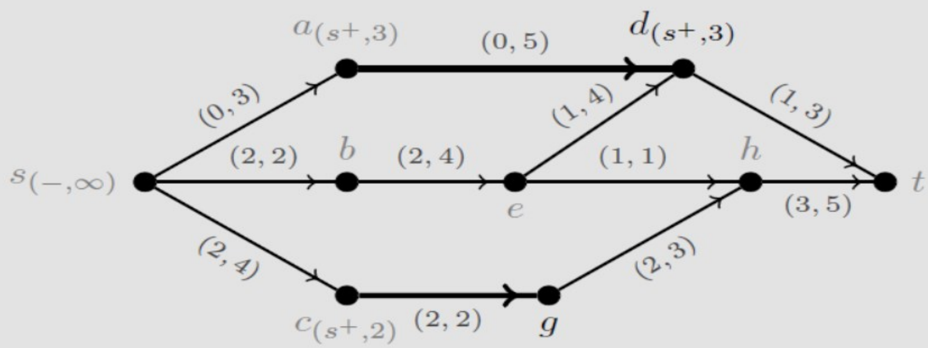
We increase the flow by $\sigma(t) = 1$ units along each of these edges since all are in the forward direction. We update the network flow and remove all labels except that for s.



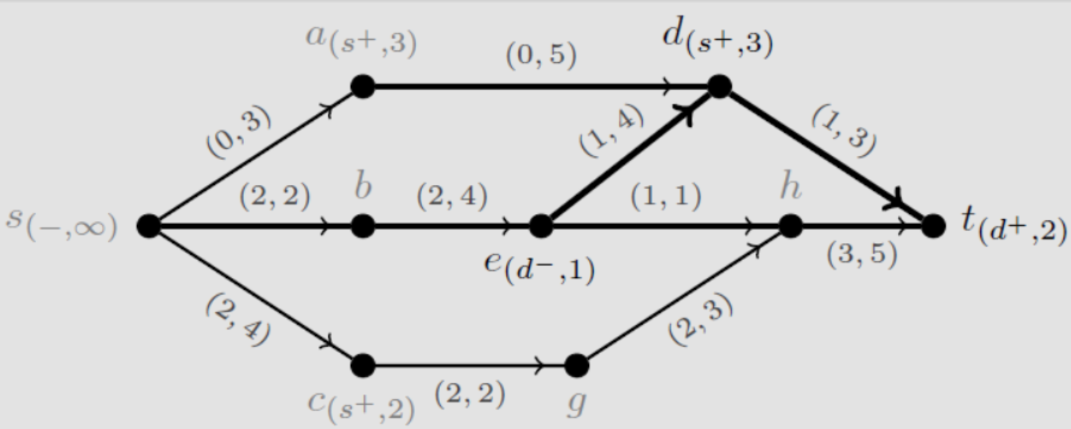
Step 6: As before we will only label the vertices whose arcs from s have slack. We label a with $(s^+, 3)$ and c with $(s^+, 2)$.



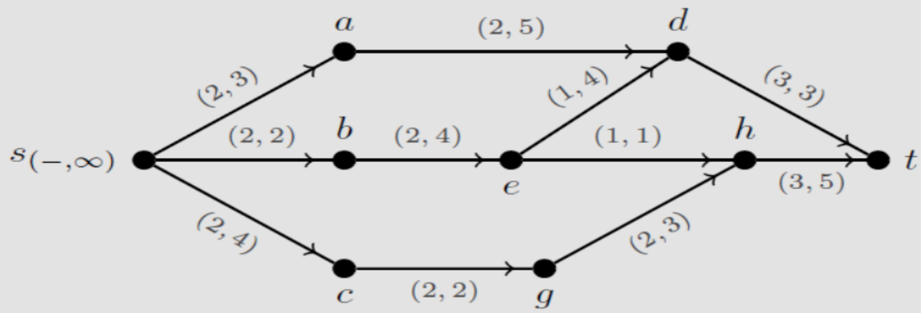
Step 7: We scan either a or c; we begin with c. The only arc from c is to g, but since there is no slack we do not label g. Scanning a we only consider the arc ad, which has slack 5. Label d with $(a^+, 3)$ since $\sigma(d) = \min\{\sigma(a), k(ad)\} = \min\{3, 5\} = 3$.



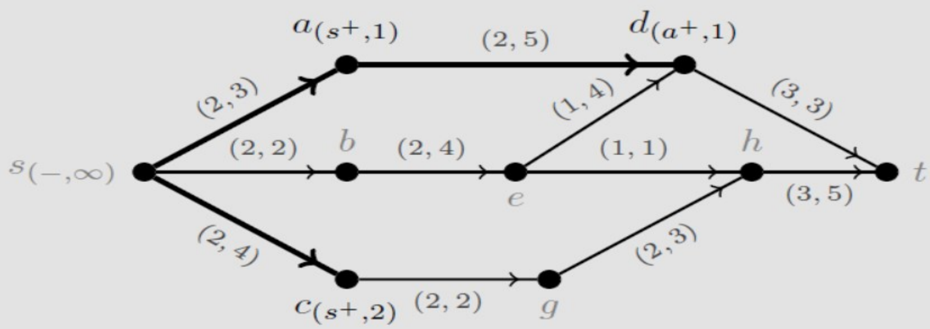
Step 8: Our only unscanned labeled vertex is d . Since e has an arc to d with positive flow, label e as $(d^-, 1)$ since $\sigma(e) = \min\{\sigma(d), f(ed)\} = \min\{3, 1\} = 1$. Also label t with $(d^+, 2)$ since $\sigma(t) = \min\{\sigma(g), k(dt)\} = \min\{3, 2\} = 2$.



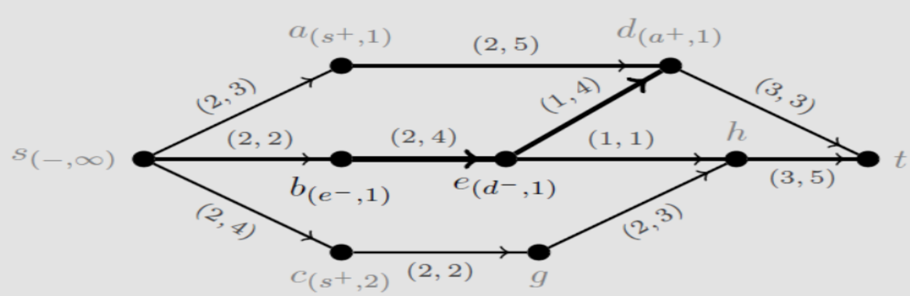
Step 9: Since t is again labeled, we find an $s-t$ chain K' of slack edges. Backtracking from t to s gives the chain $s\,a\,d\,t$, and we increase the flow by $\sigma(t) = 2$ units along each of these edges since all are in the forward direction. We update the network flow and remove all labels except that for s.



Step 10: The process of again assigning labels is quite similar to the steps above. Some σ values now change along the last chain from which we adjusted the flow. Instead of working through the individual steps, we will pick up after the label for d has been assigned, which is shown below.



Label e as $(d^-, 1)$ as before since there is flow along the arc ed, but t is not given a label due to no slack along the arc dt. Once this is complete, b will be given a label of $(e^-, 1)$ since $\sigma(b) = \min\{\sigma(e), f(be)\} = \min\{1, 2\} = 1$. No label will be assigned to h since there is no slack along the arc eh. Recall that g also remains unlabeled since cg has no slack.



At this point there are no further vertices to label and so f must be a maximum flow, with a value of 6.

Remark: When the Augmenting Flow Algorithm halts, a maximum flow has been achieved.

- ✓ The main idea will be to determine a barrier through which all flow must travel and as a consequence, the maximum flow cannot exceed the barrier with minimum size.
- ✓ The source and sink will be on opposite sides of this barrier, which is more commonly called a ***cut***.

Definition 4.32 Let P be a set of vertices and $\overline{P}$ denote those vertices not in P (called the complement of P). A ***cut*** $(P, \overline{P})$ is the set of all arcs xy where x is a vertex from P and y is a vertex from $\overline{P}$. An $\boldsymbol{s-t}$ ***cut*** is a cut in which the source s is in P and the sink t is in $\overline{P}$.

Remark:

A cut is a separating set that is reminiscent of an edge-cut, except that we are not concerned with disconnecting a graph but rather describing all arcs originating from one portion of the digraph and ending in the other portion.

Example:

In the network G_8 above, if we let $P = \{s, a, e, g\}$ then $\bar{P} = \{b, c, d, h, t\}$ and

$(P, \bar{P}) = \{sb, sc, ad, ed, eh, gh\}$.

- Note that *be* is not part of the cut even though *b* and *e* are in opposite parts of the vertex set (namely *b* is in *P* and *e* is in $\bar{P}$) since the arc travels in the wrong direction with regards to the definitions of *P* and $\bar{P}$.
- As this cut acts as a barrier to increasing the values of a flow, when we discuss the value of a cut we are in fact concerned with the capacities along these arcs rather than their flow.
- Thus, the value of a cut is referred to as its capacity.

Definition 4.33 The ***capacity*** of a cut, $c(P, \overline{P})$, is defined as the sum of the capacities of the arcs that comprise the cut.

- The cut given above has a capacity 18 and therefore indicates that all feasible flows of G_8 must have a value at most 18.
- Obviously, this is not the best bound for the maximum flow since our work above seems to indicate that the maximum flow of $G8$ has value of 6.
- In fact, two easy cuts often provide more useful initial bounds on the value of a flow; the first is where P only consists of the source and the second is where P consists of every vertex except the sink.
- In the example above, if we let P = {s} then the capacity of this cut is c(P, $\overline{P}$) = 9 and if $\overline{P}$ ={s, a, b, c, d, e, g, h} then it has capacity c(P, $\overline{P}$) = 8, which are much closer to our conjecture that the maximum flow is 6.

We showed that the minimum size of a separating set equals the maximum number of internally disjoint paths. A similar result holds for flows and cuts in a network. In fact, Menger's theorem is really just an undirected version of the Max Flow–Min Cut Theorem below, which was published about thirty years after Menger's result.

Theorem 4.34 (Max Flow–Min Cut) In any directed network, the value of a maximum $s-t$ flow equals the capacity of a minimum $s-t$ cut.

- The difficulty in using this result to prove a flow is maximum is in finding
- the minimum cut.
- We can use the vertex labeling procedure to obtain our minimum cut.

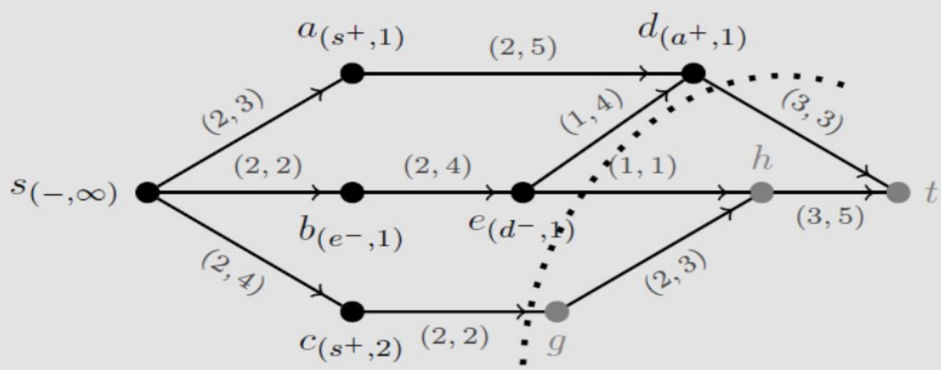
Min-Cut Method

1. Let $G = (V, A, c)$ be a network with a designated source s and sink t and each arc is given a capacity c.
2. Apply the Augmenting Flow Algorithm.
3. Define an $s-t$ cut $(P, \overline{P})$ where P is the set of labeled vertices from the final implementation of the algorithm.
4. $(P, \overline{P})$ is a minimum $s-t$ cut for G.

- ✓ By finding a flow and cut with the same value, we now have proof that the flow is indeed maximum.

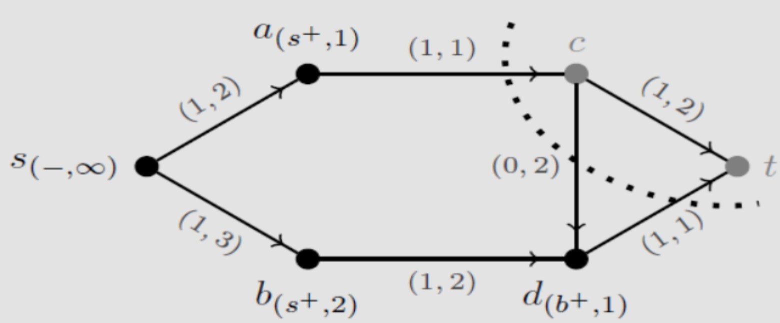
Example 4.7 Use the Min-Cut Method to find a minimum $s-t$ cut for the network G_8 on page 189 and the network G_9 from Example 4.5.

Solution: The final labeling from the implementation of the Augmenting Flow Algorithm on G_8 produced the following network.



The Min-Cut Method sets $P = \{s, a, b, c, d, e\}$ and $\overline{P} = \{g, h, t\}$. The arcs in the cut are $\{dt, eh, cg\}$, making the capacity of this cut $c(P, \overline{P}) = 3 + 1 + 2 = 6$. Since we have found a flow and cut with the same value, we know the flow is maximum and the cut is minimum.

The final labeling in the network G_9 from Example 4.5 gives $P = \{s, a, b, d\}$ and $\overline{P} = \{c, t\}$. The arcs in the cut set are $\{ac, dt\}$. Note, cd is not in the cut since the arc is in the wrong direction. The capacity of this cut is $c(P, \overline{P}) = 2$, and since we have found a flow and cut with the same value, we know the flow is maximum and the cut is minimum.



Remark:

In practice, we can perform the Augmenting Flow Algorithm and the Min-Cut Method simultaneously, thus finding a maximum flow and providing a proof that it is maximum (through the use of a minimum cut) in one complete procedure.