Lec # 20, 21 & 22

Example 6.4.3.

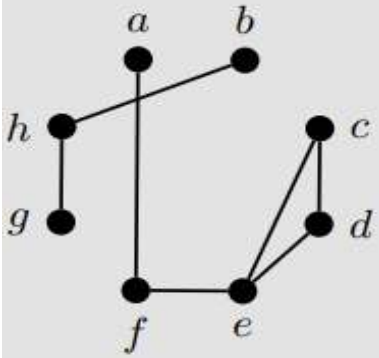
(i) The graph K_2 or any other connected graph containing a cut-vertex is 1-connected but not 2-connected.

(ii) Every cycle C_n, $n \geq 3$, is k-connected, for $k = 1, 2$, but not 3-connected.

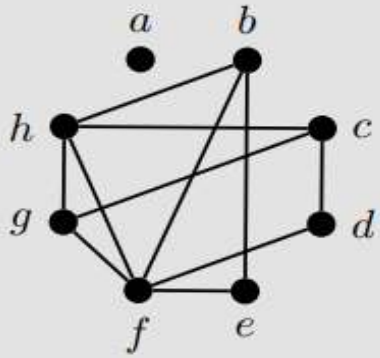
(iii) Every wheel W_n, $n \geq 4$, is k-connected, for $k = 1, 2, 3$, but not 4-connected.

Example 4.2 Find $\kappa'(G)$ for each of the graphs shown on page 169.

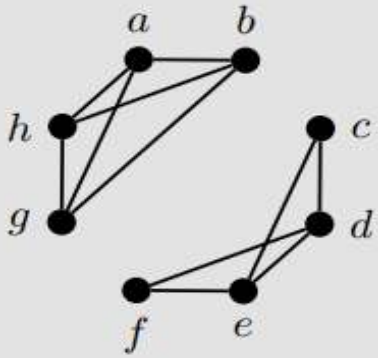
Solution: There are many options for a single edge whose removal will disconnect G_1 (for example af or dg). Thus $\kappa'(G_1) = 1$. For G_2, no one edge can disconnect the graph with its removal, yet removing both ab and ah will isolate a and so $\kappa(G_2) = 2$. Similarly $\kappa'(G_3) = 2$, since the removal of bc and cg will create two components, one with vertices a, b, g, h and the other with c, d, e, f.



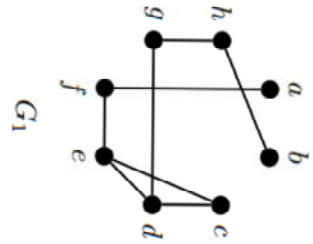
$G_1 - dg$



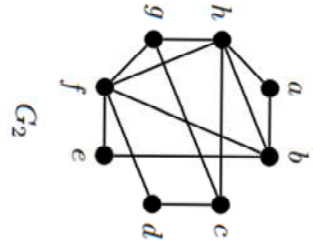
$G_2 - \{ab, ah\}$



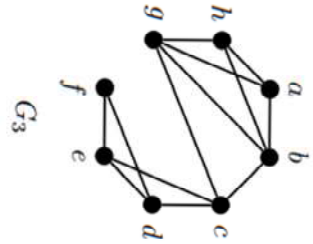
$G_3 - \{bc, cg\}$



G_1



G_2



G_3

Theorem 4.5 (Whitney's Theorem) For any graph G, $\kappa(G) \leq \kappa'(G) \leq \delta(G)$.

Theorem 6.3.1. (Whitney (1932)) *For any graph G of order $n \geq 1$,*

$$0 \leq \kappa(G) \leq \kappa'(G) \leq \delta(G) \leq n - 1.$$

Connectivity & Paths

Theorem 4.6 A vertex v is a cut-vertex of a graph G if and only if there exist vertices x and y such that v is on every $x - y$ path.

Proof: First suppose v is a cut-vertex in a graph G. Then $G-v$ must have at least two components. Let x and y be vertices in different components of $G - v$. Since G is connected, we know there must exist an $x - y$ path in G that does not exist in $G - v$. Thus v must lie on this path.

Conversely, let v be a vertex and suppose there exist vertices x and y such that v is on every $x - y$ path. Then none of these paths exist in $G - v$, and so x and y cannot be in the same component of $G - v$. Thus G must have at least two components and so v is a cut-vertex.

Theorem 4.7 An edge e is a bridge of G if and only if there exist vertices x and y such that e is on every $x - y$ path.

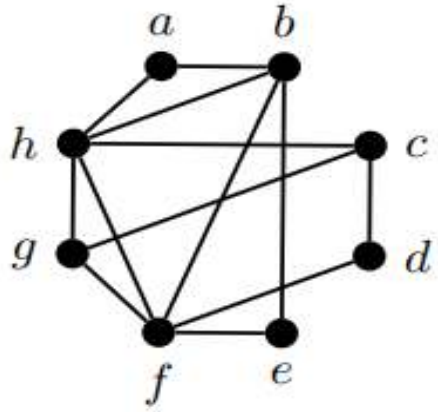
Theorem 4.8 Every nontrivial connected graph contains at least two vertices that are not cut-vertices.

Theorem 4.9 An edge e is a bridge of G if and only if e lies on no cycle of G.

Definition 4.10 Let P_1 and P_2 be two paths within the same graph G. We say these paths are

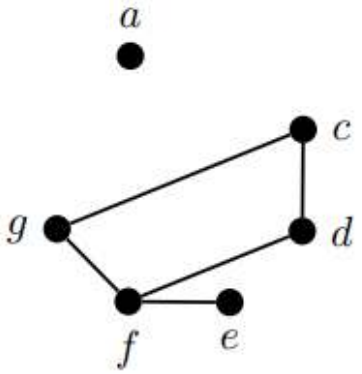
- ***disjoint*** if they have no vertices or edges in common.
- ***internally disjoint*** if the only vertices in common are the starting and ending vertices of the paths.
- ***edge-disjoint*** if they have no edges in common.

Definition 4.11 Let x and y be two vertices in a graph G. A set S (of either vertices or edges) ***separates*** x and y if x and y are in different components of $G - S$. When this happens, we say S is a separating set for x and y.

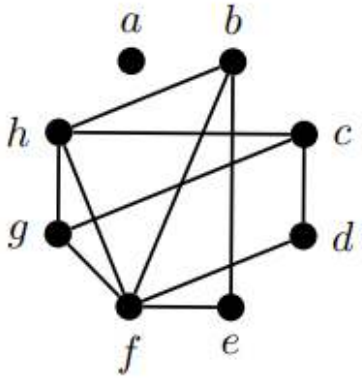


G_2

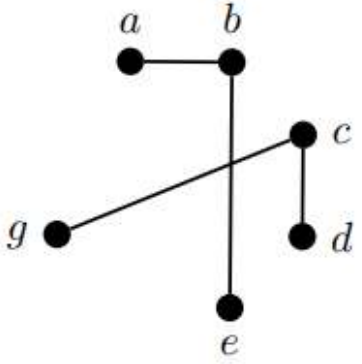
Note that a cut-set may or may not be a separating set for a specific pair of vertices. Consider the graph G_2 from page 169 (and reproduced on the next page). We have already shown that $\{b, h\}$ is a cut-set and $\{ab, ah\}$ is an edge-cut. But if we want to separate b and c then we cannot use b in the separating set, and using the edges ab and ah will only isolate a, leaving b and c in the same component. However, we can separate b and c using the vertices $\{f, h\}$ and the edges $\{cd, cg, ch\}$. Note that you cannot separate b and c with fewer vertices or edges (try it!).



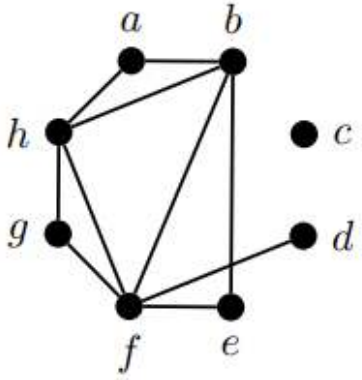
$G_2 - \{b, h\}$



$G_2 - \{ab, ah\}$



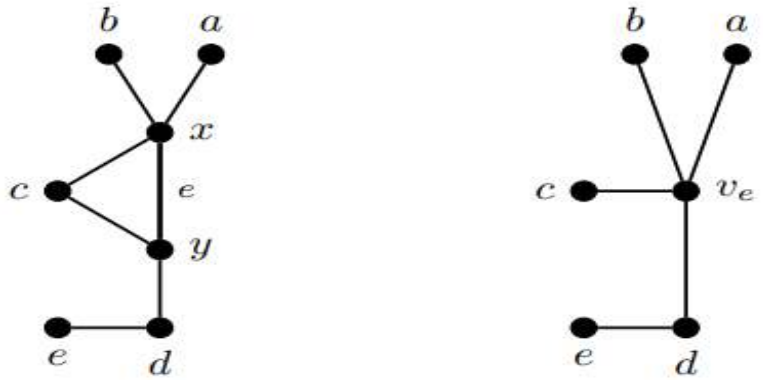
$G_2 - \{f, h\}$



$G_2 - \{cd, cg, ch\}$

Contraction

Definition 4.12 Let $e = xy$ be an edge of a graph G. The ***contraction*** of e, denoted G/e, replaces the edge e with a vertex v_e so that any vertices adjacent to either x or y are now adjacent to v_e.



G_4 G_4/e

Contracting an edge creates a smaller graph, both in terms of the number of vertices and edges, but keeps much of the structure of a graph in tact. In particular, contracting an edge cannot disconnect a graph (see Exercise 4.22).

Menger's Theorem

Theorem 4.13 (Menger's Theorem) Let x and y be nonadjacent vertices in G. Then the minimum number of vertices that separate x and y equals the maximum number of internally disjoint $x - y$ paths in G.

Theorem 4.14 A nontrivial graph G is k-connected if and only if for each pair of distinct vertices x and y there are at least k internally disjoint $x - y$ paths.

Theorem 4.15 Let x and y be distinct vertices in G. Then the minimum number of edges that separate x and y equals the maximum number of edge-disjoint $x - y$ paths in G.

Theorem 4.16 A nontrivial graph G is k-edge-connected if and only if for each pair of distinct vertices x and y there are at least k edge disjoint $x - y$ paths.

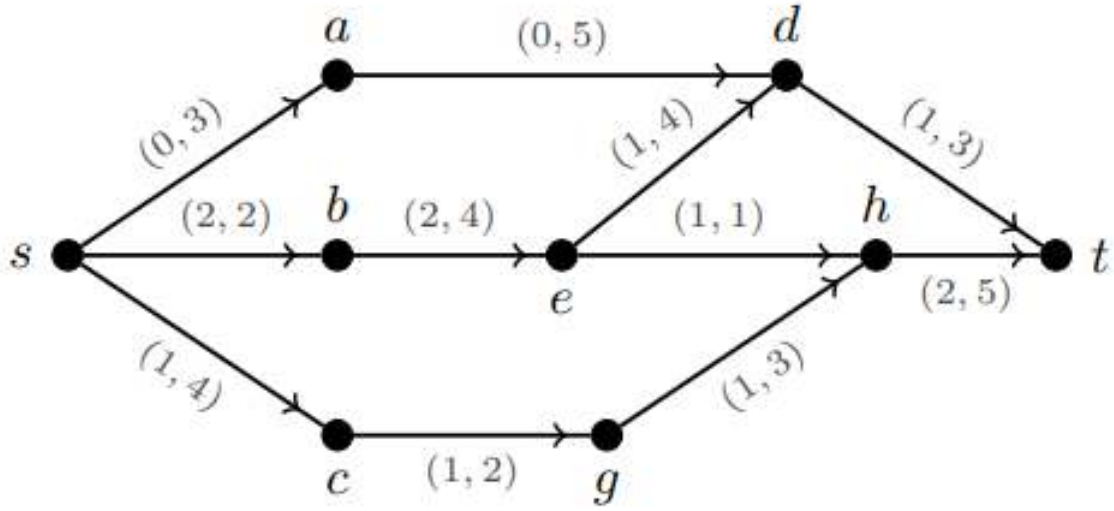
National University of computer and emerging sciences

Foundation for Advancement of Science and Technology

Network Flow

Definition 4.27 A ***network*** is a digraph where each arc e has an associated nonnegative integer $c(e)$, called a ***capacity***. In addition, the network has a designated starting vertex s, called the ***source***, and a designated ending vertex t, called the ***sink***. A ***flow*** f is a function that assigns a value $f(e)$ to each arc of the network.

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

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

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.

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, $k(e) = c(e) - f(e)$.

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

The Maximum Flow Problem

A digraph with an integer-valued function c (known as the **capacity function**) defined on its set of arcs is called a **capacitated network.** Two vertices in the network are specially designated: the **source,** with indegree zero, and the **sink,** with outdegree zero. Every other vertex is an **intermediate vertex.** We assume that the set of vertices is $\{1, 2, \ldots, n\}$ in which vertex 1 is the source and vertex n is the sink. If $e = (i, j)$ is an arc in G, the integer $c(e) = c(i, j)$ is the **capacity** of the arc. It is assumed that the capacity of each arc is nonnegative. A **flow** f in the network is an integer-valued function defined on its set of arcs such that $0 \leq f(e) \leq c(e)$ for each arc e. The integer $f(e)$ is the **flow along arc** e**.** The sum of the flows along all the arcs directed to vertex i is the **inflow into** i**,** and the sum of the flows along all the arcs directed from vertex i is the **outflow from** i**.** A flow is a **feasible flow** if it satisfies the **conservation condition:** the inflow into i is equal to the outflow from i for every vertex i other than the source and the sink. If f is a feasible flow in a capacitated network G, the **value** $f(G)$ **of the flow** is the outflow from the source. A feasible flow in a capacitated network such that the value of the flow is as large as possible is called a **maximum flow** in the network. The problem of finding a feasible flow in a network such that its flow value is maximum is known as the maximum flow problem.

[If there are arcs in either direction between a pair of vertices in a network, a new vertex can be inserted on one of the two arcs, replacing that arc by two arcs in the same direction with the same capacity. So without loss of generality, it can be assumed that the digraph is asymmetric; that is, for distinct vertices i and j, not both (i, j) and (j, i) are arcs.]

Augmenting Flow Algorithm

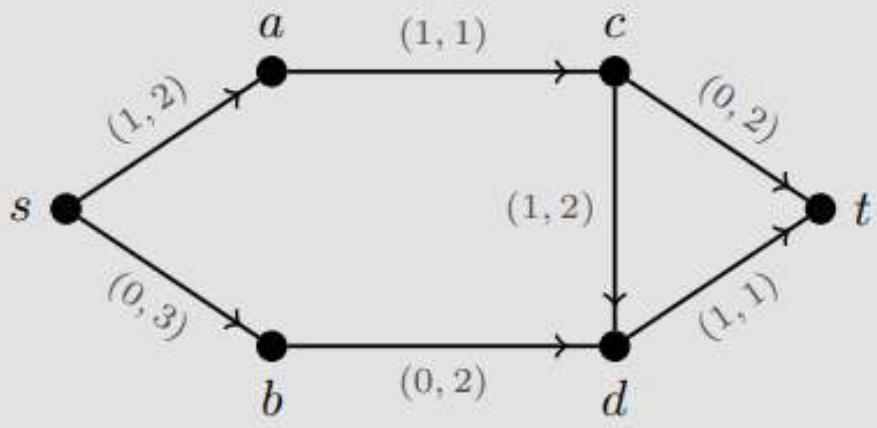
Input: Network $G = (V, E, c)$, with designated source s and sink t, and each arc is given a capacity c.

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.

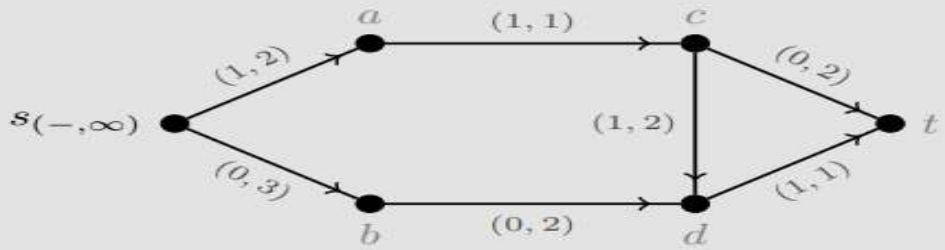
Example 4.5 Apply the Augmenting Flow Algorithm to the network G_9 shown below.



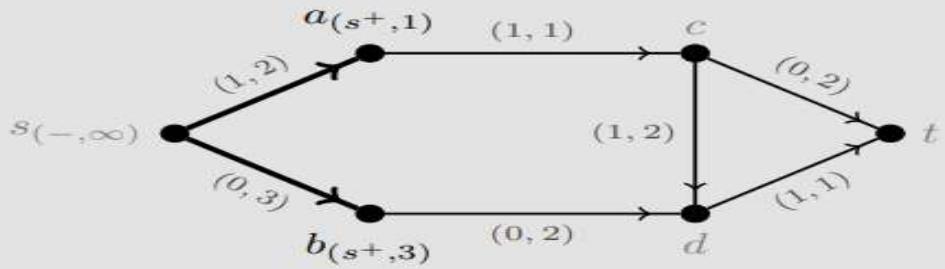
G_9

Solution: We will show edges under consideration in bold.

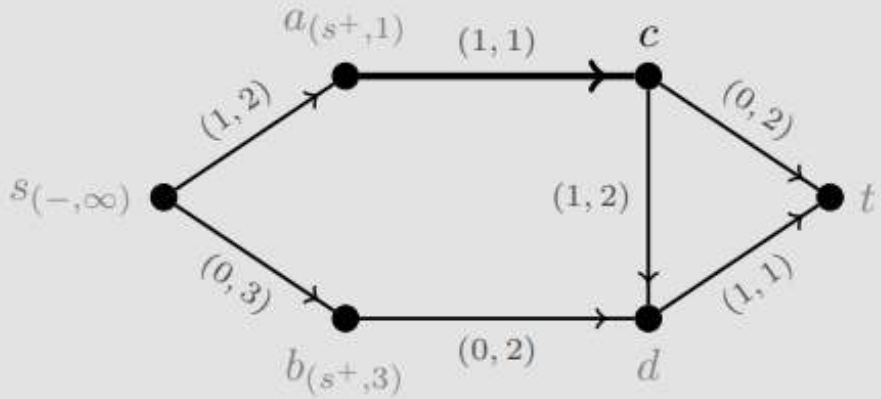
Step 1: Label s as $(-, \infty)$.



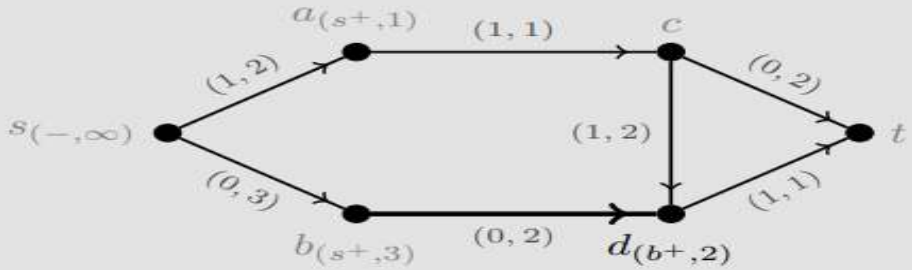
Step 2: Let $x = s$. As there are no arcs to s we will only consider the arcs out of s, of which there are two: sa and sb, which have slack of 1 and 3, respectively. We label a with $(s^+, 1)$, b is labeled $(s^+, 3)$.



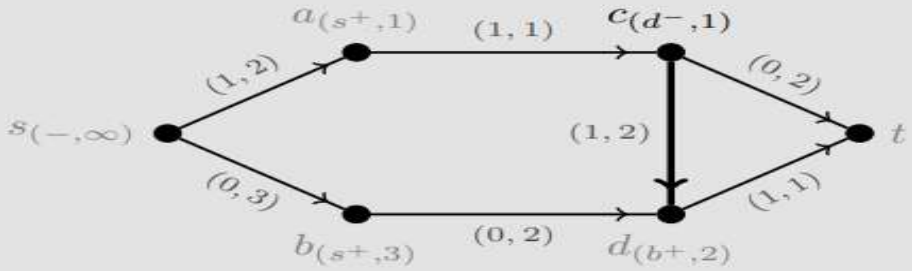
Step 3: As t is not labeled, we will scan either a or b; we choose to start with a, (so $x = a$ in the algorithm). Since the only arc going into a is from a labeled vertex, we need only consider the edges out of a, of which there is only one, ac, which has no slack. Thus c is left unlabeled at this stage.



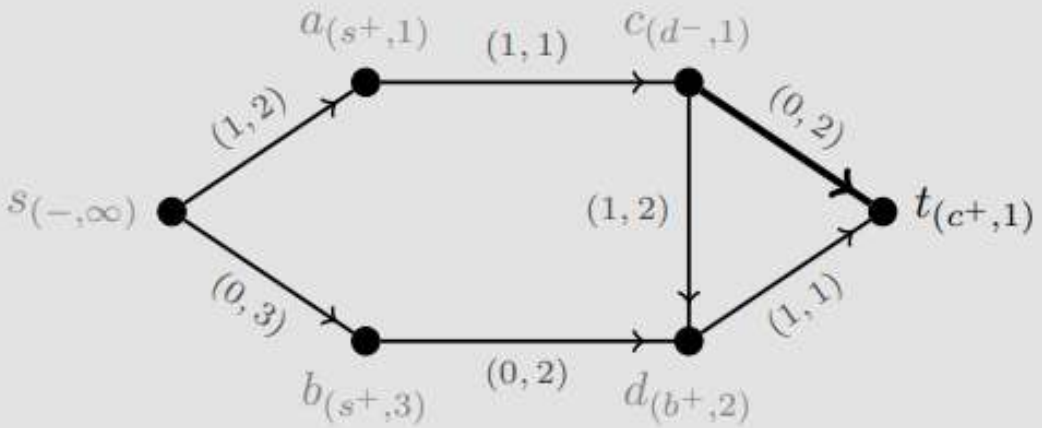
Step 4: As t is not labeled, we will scan b (so $x = b$ in the algorithm). Since the only arc going into b is from a labeled vertex, we need only consider the edges out of b, of which there is only one, bd, with slack of 2. Label d as $(b^+, 2)$ since $\sigma(d) = \min\{\sigma(b), k(bd)\} = \min\{3, 2\} = 2$.



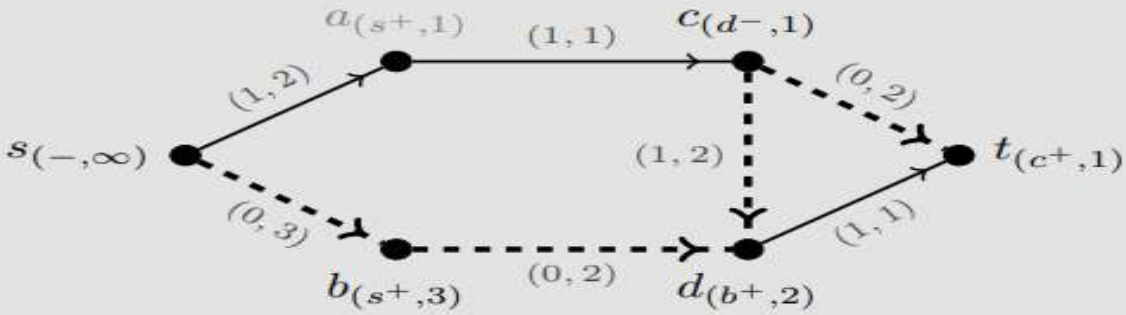
Step 5: As t is not labeled, we will scan d (so $x = b$ in the algorithm). There is one unlabeled vertex with an arc going into d, namely c, which gets a label of $(d^-, 1)$ since $\sigma(c) = \min\{\sigma(d), f(cd)\} = \min\{2, 1\} = 1$. The only arc out of d is dt, which has no slack.



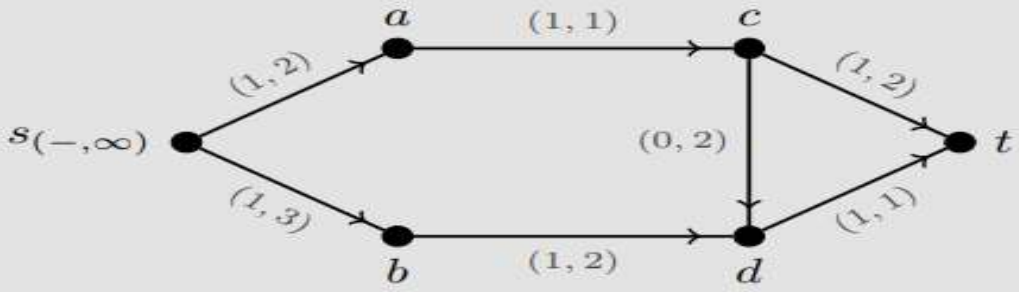
Step 6: As t is not labeled, we will scan c (so $x = c$ in the algorithm). Since the only arc going into c is from a labeled vertex, we need only consider the edges out of c to an unlabeled vertex, of which there is only one, ct, with slack of 2. Label t as $(c^+, 1)$ since $\sigma(t) = \min\{\sigma(c), k(ct)\} = \min\{1, 2\} = 1$.



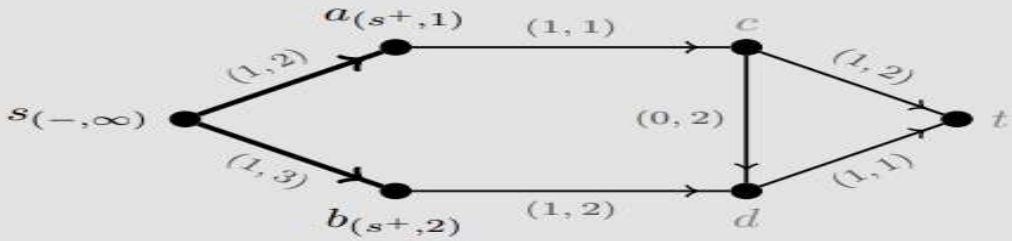
Step 7: Since t is now labeled, we find an $s-t$ chain K of slack edges. Backtracking from t to s gives the chain $s\,b\,d\,c\,t$.



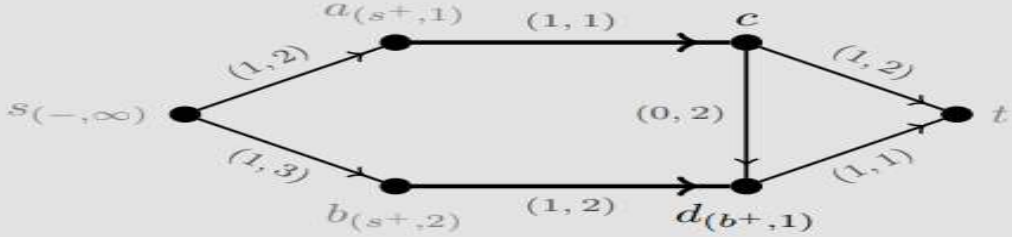
We increase the flow by $\sigma(t) = 1$ units along each of the edges in the forward direction (namely sb, bd, ct) and decrease by 1 along the edges in the backward direction (namely cd). We update the network flow and remove all labels except that for s.



Step 8: As before we will only label the vertices whose arcs from s have slack. We label a with $(s^+, 1)$ and b with $(s^+, 2)$.



Step 9: We scan either a or b; we begin with a. The only arc from a is to c, but since there is no slack we do not label c. Scanning b we only consider the arc bd, which has slack 1. Label d with $(b^+, 1)$ since $\sigma(d) = \min\{\sigma(b), k(bd)\} = \min\{2, 1\} = 1$.



Scanning d will not assign a label to t since there is no slack along the arc dt, and c also remains unlabeled since cd has no flow. At this point there are no further vertices to label and so f must be a maximum flow of G_9, with a value of 2.