Graph theory: flows and networks

Yulia Burkatovskaya

Department of Information Technologies

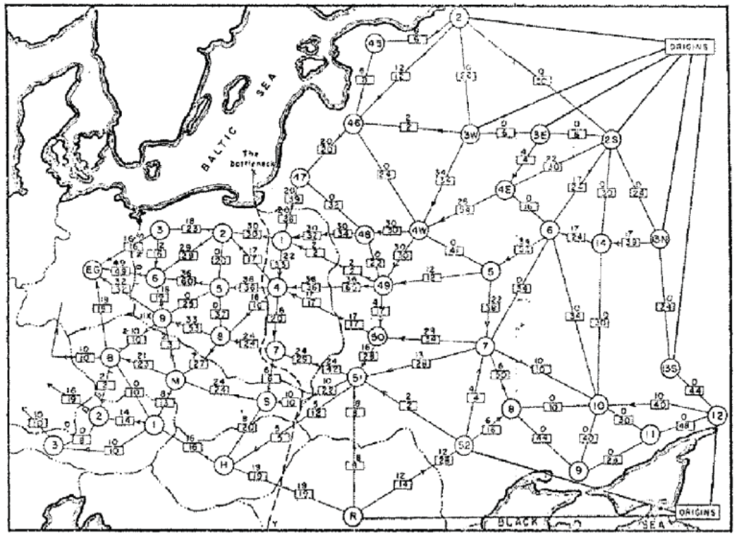
Associate professor

6. Flows and networks

- Network and flow
- Ford-Fulkerson theorem
- Maximum flow
- Minimum-cost flow

Flows and networks

Soviet Rail Network, 1955

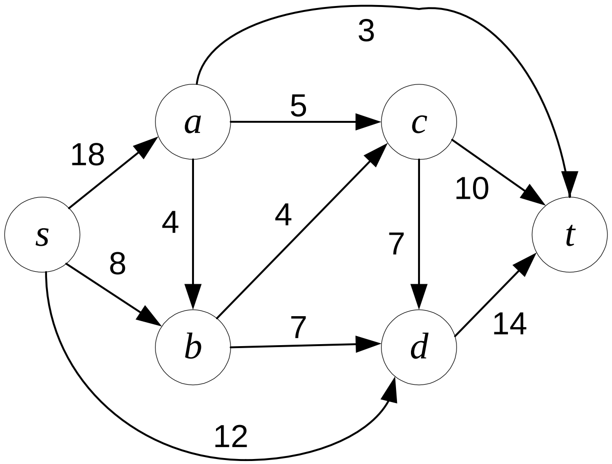


Reference: *On the history of the transportation and maximum flow problems.* Alexander Schrijver in Math Programming, 91: 3, 2002.

6.1. Network and flow

- A **flow network (network)** $G(V,E,C)$ is a directed graph, where each edge $(u,v)\in E$ has a nonnegative **capacity** $c(u,v)\geq 0$.
- If $(u,v)\notin E$, we assume that $c(u,v)=0$.
- There are two distinct vertices: a **source** s with $d^{-}(s)=0$ and a **sink** t with $d^{+}(t)=0$.

Example.



Network and flow

- A **flow** in G: a real-valued function f: $E \rightarrow R$ satisfying the following two properties:
- **Capacity constraint**: For all $(u,v) \in E$,

  we require $f(u,v) \leq c(u,v)$.
- **Flow conservation:** For all $v \in V \setminus \{s,t\}$,

  we require div(f,u)=0.

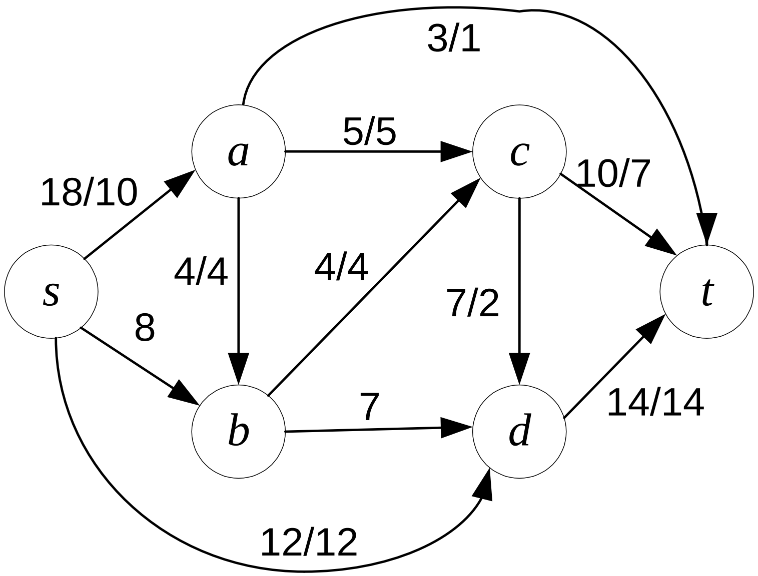
$$\operatorname{div}(f, v) = \sum_{v:\{u,v\} \in E} f(u, v) - \sum_{v:\{v,u\} \in E} f(v, u).$$

- The source divergence is the **value of the flow**

$$w(f) = \operatorname{div}(f, s)$$

Network and flow

Example. A flow with the value 22.



Problem: to find a flow with the maximum value.

Network and flow

Let S and T be two disjoint subsets of V;

- $s \in S$, $t \in T$;
- $S \cup T = V$.

Then the **cut** $P(S,T)$ is the set of edges joining vertices from S and vertices from T;

- $P^{+}(S,T)=\{(u,v): u \in S, t \in T\}$;
- $P^{-}(S,T)=\{(u,v): u \in S, t \in T\}$.

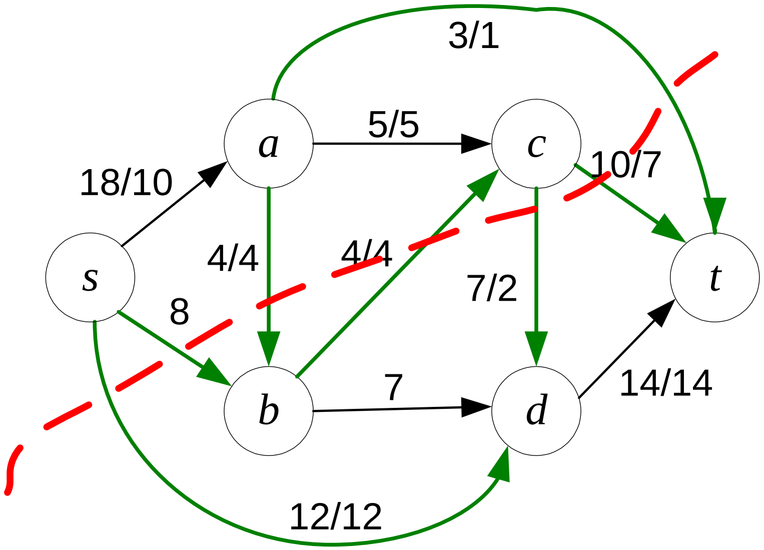
$$F(P) = \sum_{e \in P} f(e),$$

$$C(P) = \sum_{e \in P+} C(e).$$

Network and flow

Example.

- $S=\{s,a,c\}$;
- $P=\{b,d,t\}$;
- $P^{+}=\{sd, sb, ab, at, cd, ct\}$;
- $P^{-}=\{bc\}$;
- $C(P)=12+8+4+3+7+10=44$.



6.2. Ford-Fulkerson theorem

Theorem. In any s-t network there exists a feasible flow *f** and an s-t cut *P* such that

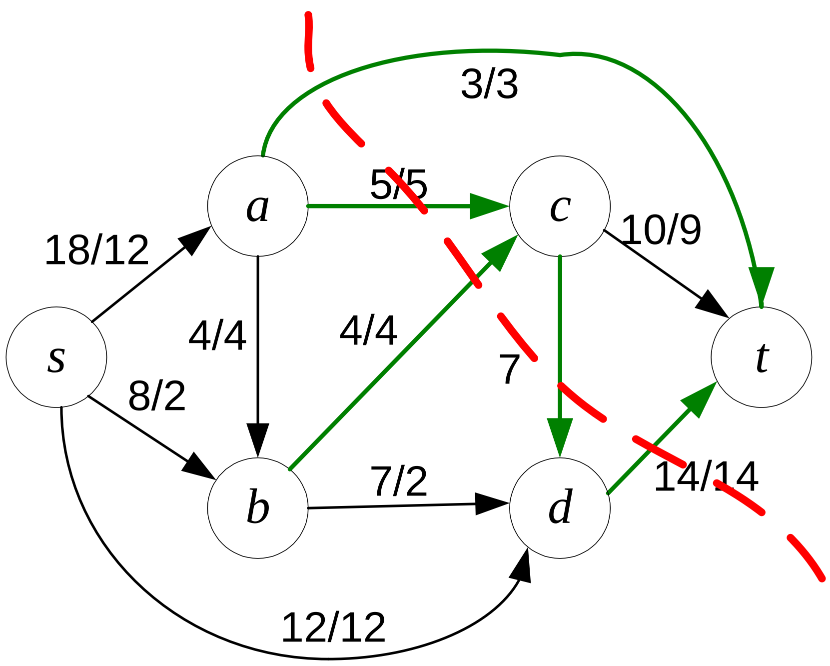
- (1) the flow equals the capacity of the cut,
- (2) on any arc belonging to P^+, this flow equals the capacity of the arc, and
- (3) on any arc, that would belong to P^-, the flow equals zero.

Also known as **max-flow min-cut theorem**.

$$w(f^*) = \max_f w(f) = \min_P C(P).$$

Ford-Fulkerson theorem

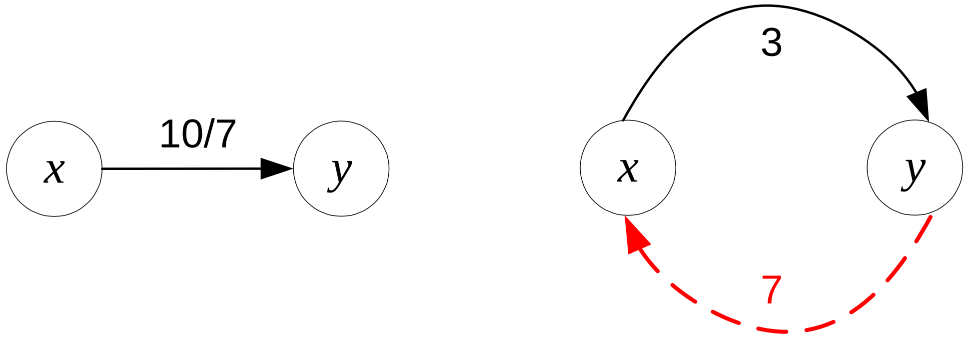
Example. The maximum flow with the value 26.



6.3. Maximum flow

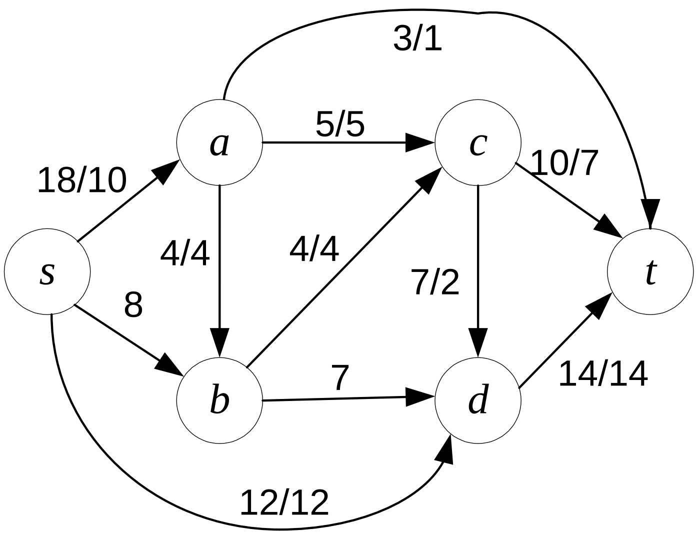
Residual network. Given a graph G and a flow f in it, we form a new flow network G_f that has the same vertex set of G and that has two arcs for each arc of G:

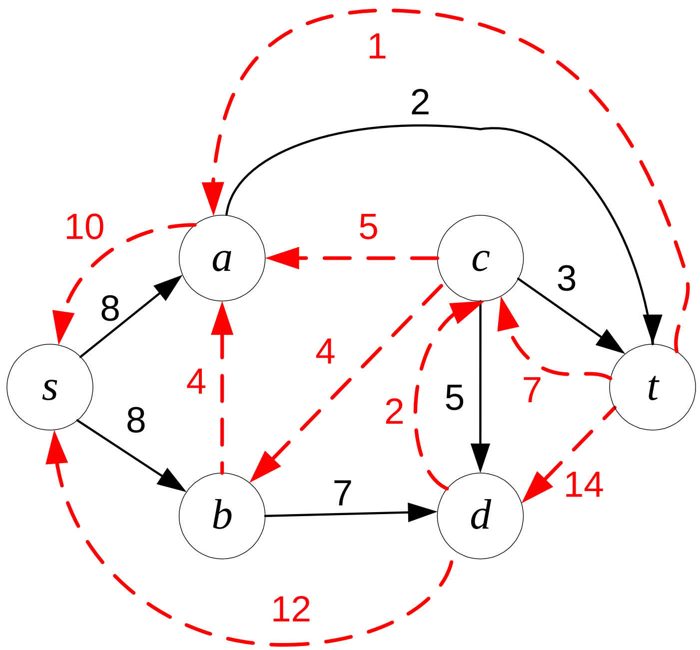
- An arc $e = (x,y)$ of G that carries non-zero flow $f(e)$ and has capacity $C(e)$ spawns a **"forward arc"** of G_f with capacity $C(e)–f(e)$ (the room remaining), if $C(e)–f(e)>0$; else the "forward arc" does not exist;
- and a **"backward arc"** (y,x) of G_f with capacity $f(e)$ (the amount of previously routed flow that can be undone).



Maximum flow

Example.





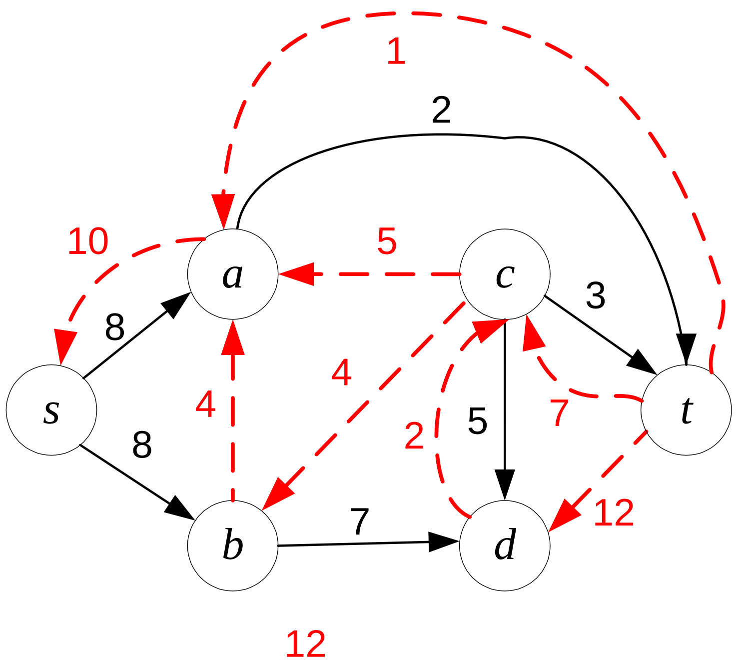
Maximum flow

Ford-Fulkerson algorithm.

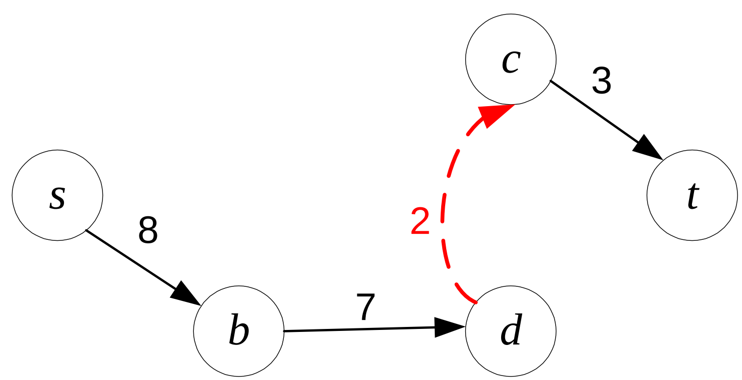
- *Start.* Given a network $G(V,E,C)$, set $f(e)=0$ for all arcs; $G_f=G$.
- *Step 1.* Find a path <s,t> in Gf. If there are no such paths then go to the End.
- *Step 2*. Calculate the value δ=min{C(e)} where {e} is the set of arcs of the path. Change the flow along the path:
  - if (x,y) is a forward arc then increase the flow in (x,y) by δ;
  - if (x,y) is a backward arc then decrease the flow in (y,x) by δ.
- *Step 3*. Update the residual network Gf. Go to Step 1.
- *End.* The current flow is the maximum flow.

Maximum flow

Example.

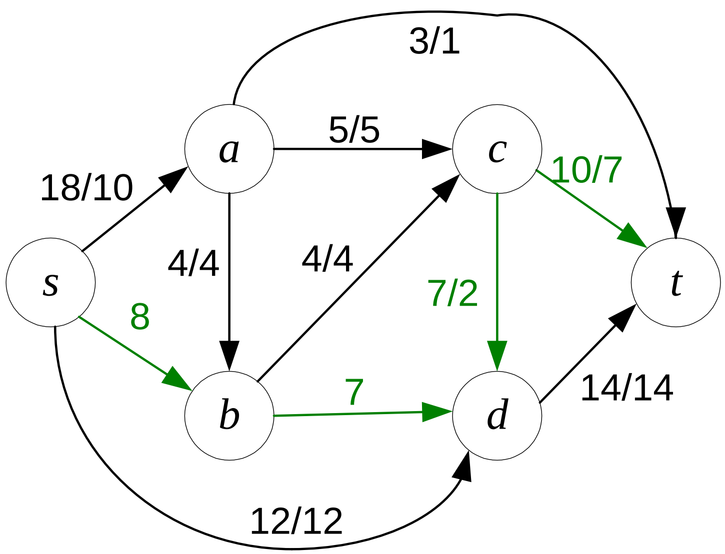


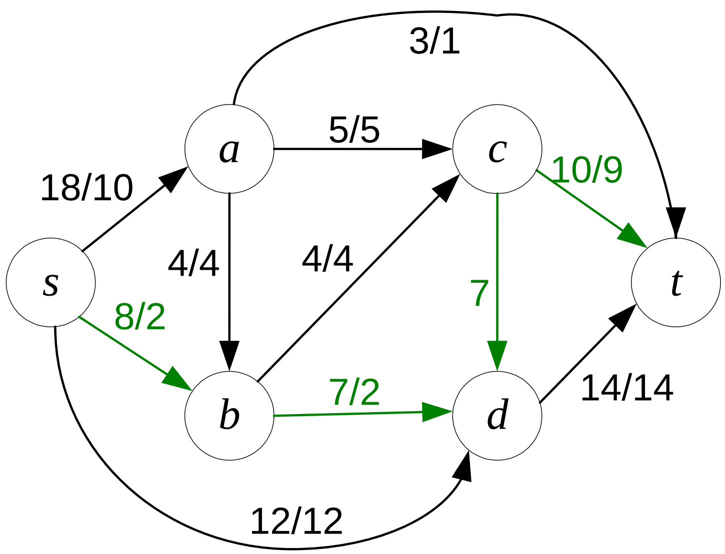
δ=min{8,7,2,3}=2



Maximum flow

Example.



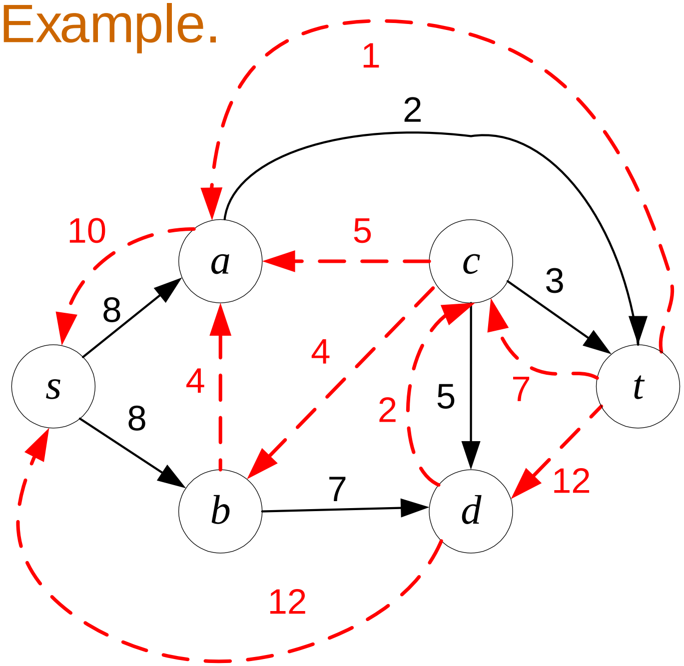


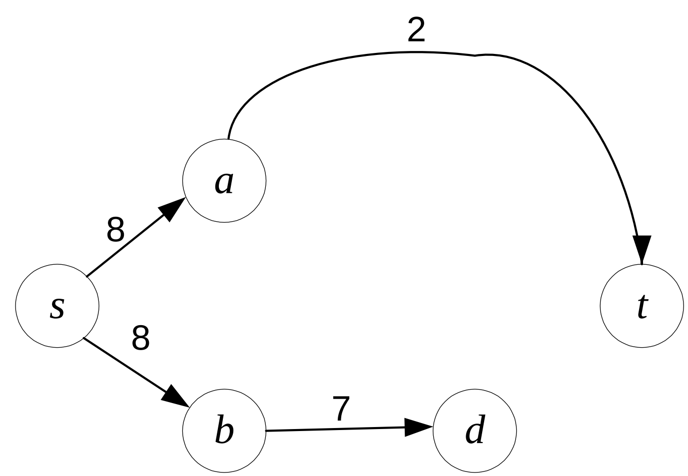
Maximum flow

Edmond-Karp algorithm.

- Very similar to Ford-Fulkerson algorithm.
- To find an **augmenting path** <s,t> in G_f, use breadth-first search.

Example.





Maximum flow

Dinic algorithm.

Level graph.

- We assign **levels** to all nodes, level of a node is shortest distance (in terms of number of edges) of the node from source.
- In the level graph, we find in general more than one augmenting path <*s*,*t*> in G_f.
- We send multiply flows; so, it works better than Edmond-Karp algorithm.

Blocking flow.

- A flow is **blocking flow** if no more flow can be sent using level graph, i.e., no more s-t path exists such that path vertices have current levels 0, 1, 2… in order.

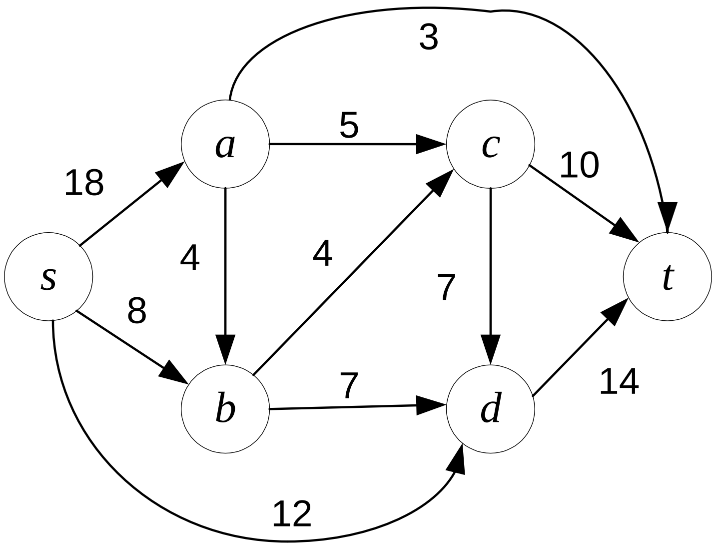
Maximum flow

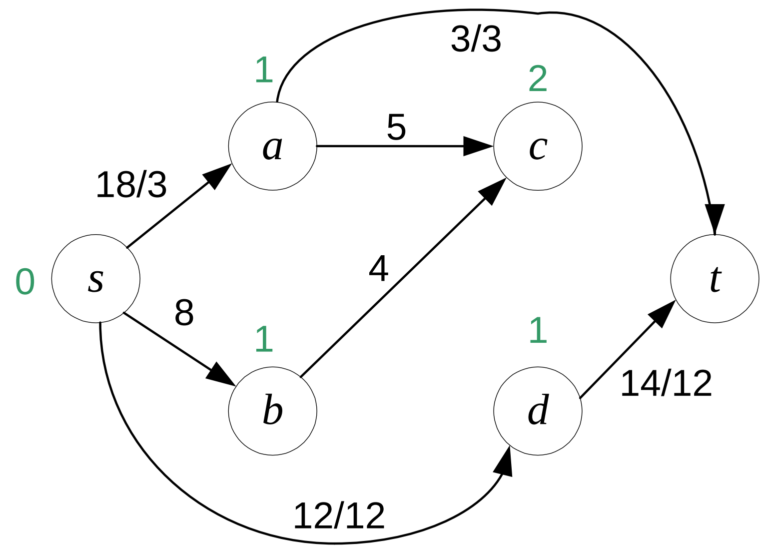
Dinic algorithm.

- *Start.* Given a network $G(V,E,C)$, set $f(e)=0$ for all arcs; initialize $G_f=G$.
- *Step 1.* By using BFS, construct the level graph in G_f. If the sink t is not included into the level graph, go to the End.
- *Step 2*. Find the blocking flow in G_f. Add it to the current flow in G.
- *Step 3*. Update the residual network G_f. Go to Step 1.
- *End.* The current flow is the maximum flow.

Maximum flow

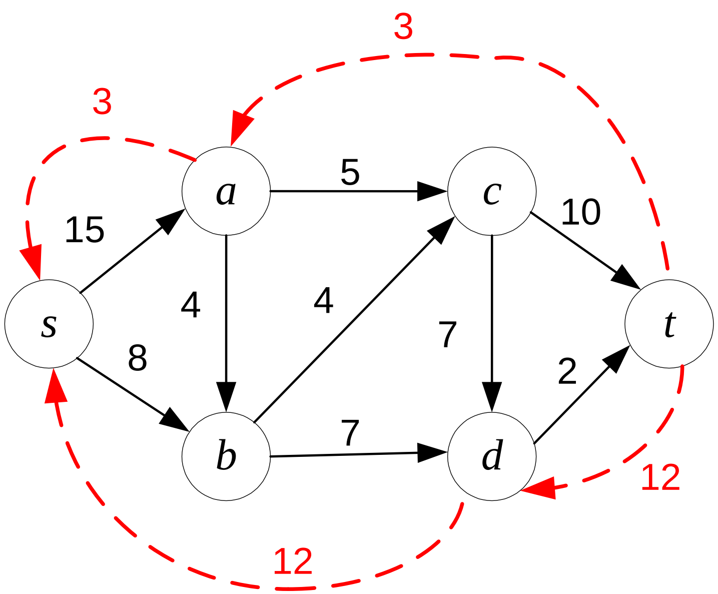
Example. Left picture represents a network with zero flow. Right picture represents the level graph and the blocking flow. Levels are shown in green.

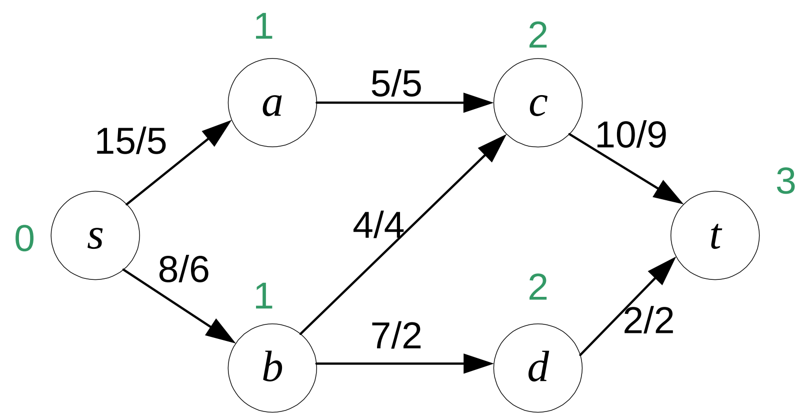




Maximum flow

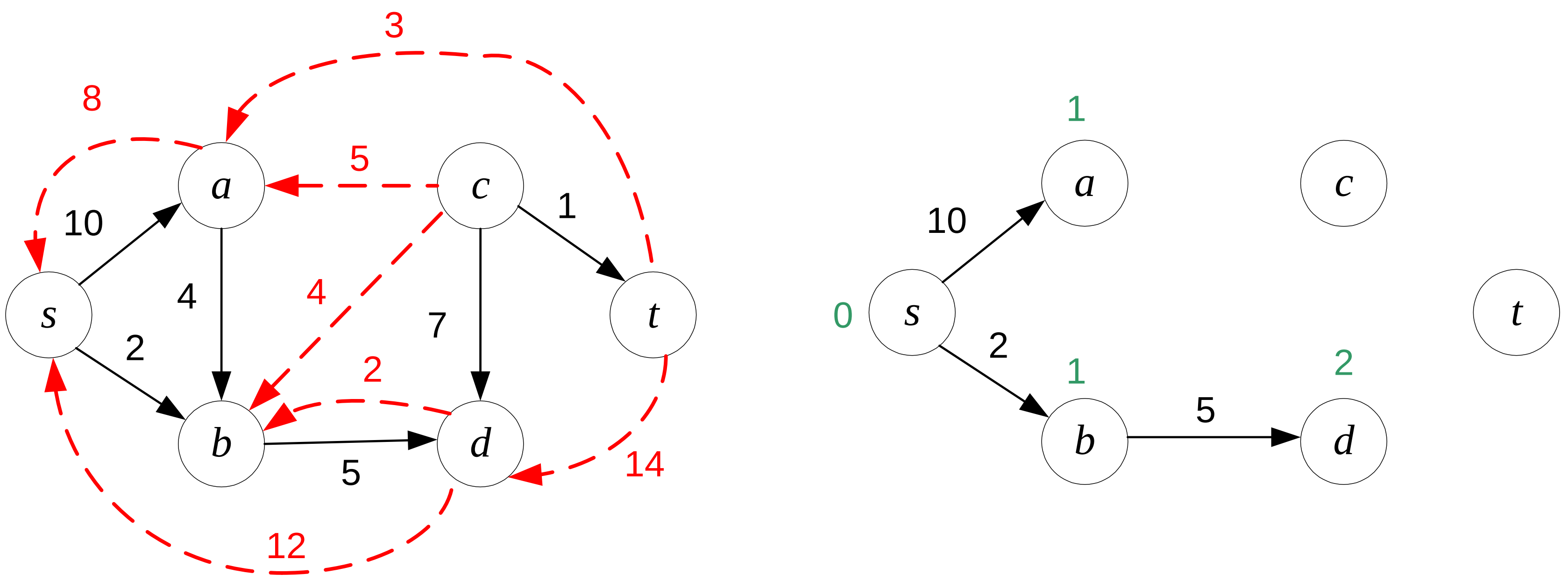
Example.





Maximum flow

Example. The sink is not reachable.



Maximum flow

- Computational complexity

Ford-Fulkerson	Edmond-Karp	Dinic
$O(nm^2)$	$O(nm^2)$	$O(n^2m)$

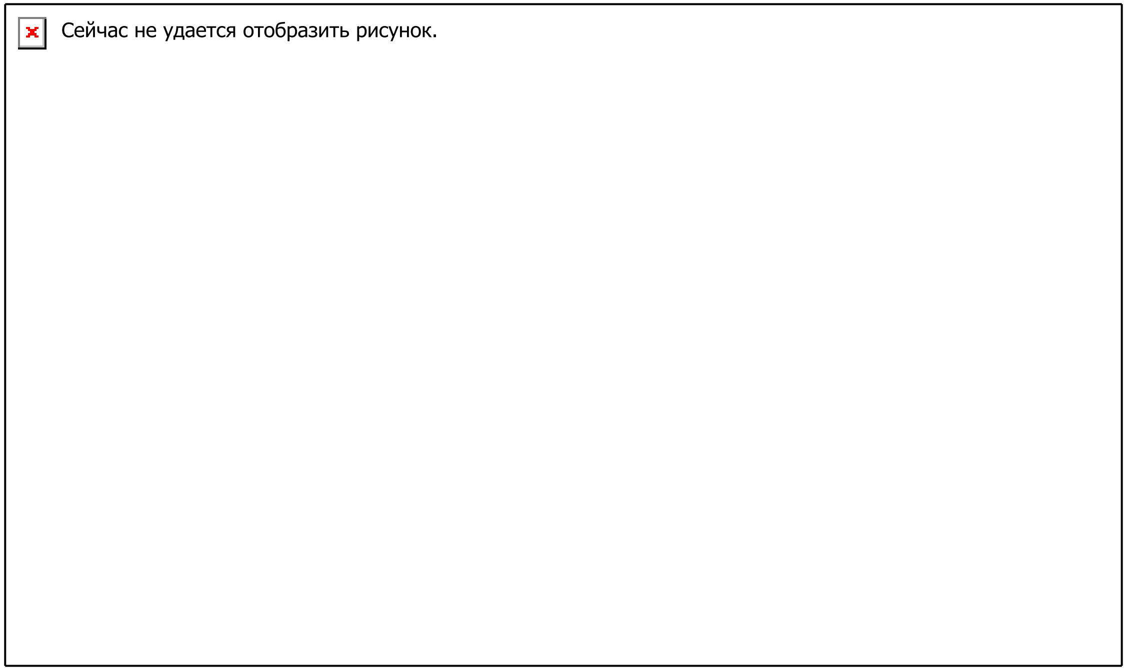
6.4. Minimum-cost flow

Let $d(u,v)$ be the **cost** that must be paid per unit of flow that goes through the arc.

Minimum-cost flow problem

- Given a network and the desirable value of flow θ, to find a flow with the value θ and the minimum cost; or to decide that none exists.

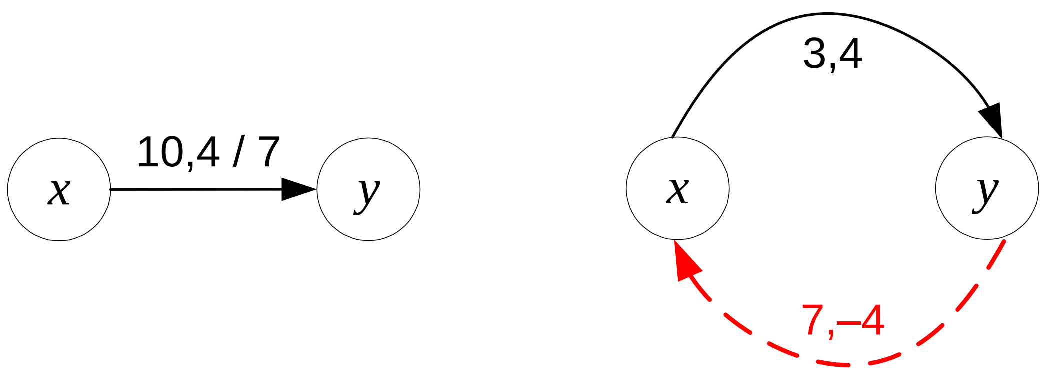
Minimum-cost flow



Minimum-cost flow

Residual network. Given a graph *G* and a flow *f* in it, we form a new flow network G_f that has the same vertex set of G and that has two arcs for each arc of G:

- An arc $e = (x,y)$ of *G* that carries non-zero flow $f(e)$ and has capacity $C(e)$ spawns a **"forward arc"** of G_f with capacity $C(e)–f(e)$ and with the cost $d(e)$, if $C(e)–f(e)>0$; else the "forward arc" does not exist;
- and a **"backward arc"** (y,x) of G_f with capacity $f(e)$ and with the cost $–d(e)$.



Minimum-cost flow

Negative-cost cycle condition.

- A flow has the minimum cost, iff the residual network contains no negative-cost cycles.
- If there is a negative-cost cycle, the flow can be change along the cycle; the obtained flow has the same value and smaller cost.

Minimum-cost flow

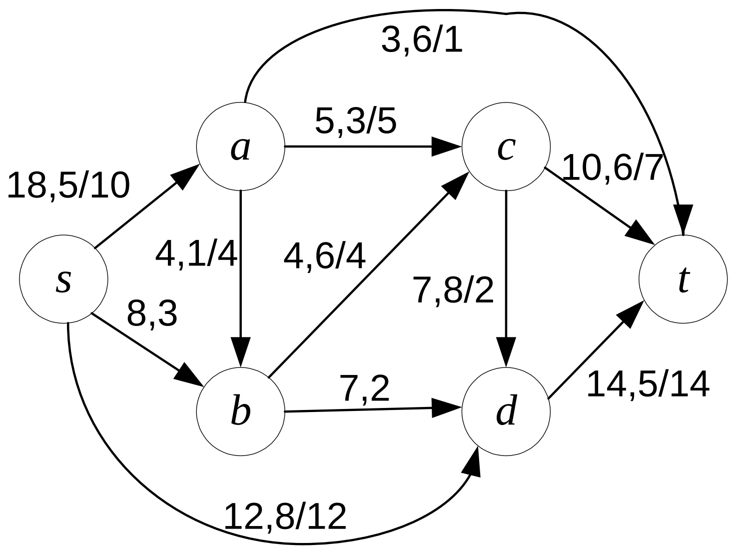
Cycle-chancelling algorithm.

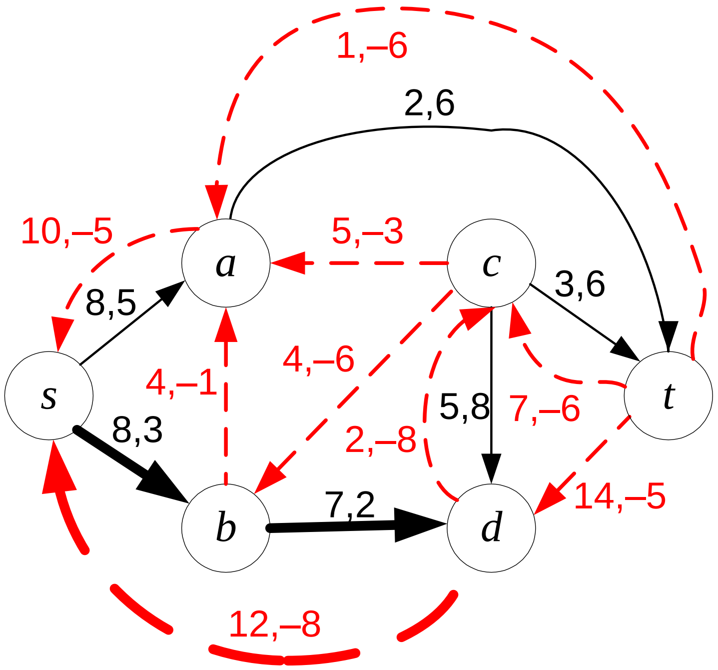
- *Start.* Given a network $G(V,E,C,D)$ and the value θ.
- *Step 1.* Find a feasible flow of the value θ (use any maximum-flow algorithm). If there is not any, the solution does not exist, go to End.
- *Step 2*. Update the residual network G_f.
- *Step 3.* Find a negative-cost cycle μ. If there in not any, a minimum-cost flow is obtained, go to End.
- *Step 4*. Calculate the value δ=min{C(e)} where {e} is the set of arcs of the cycle. Change the flow along the cycle:
  - if (x,y) is a forward arc then increase the flow in (x,y) by δ;
  - if (x,y) is a backward arc then decrease the flow in (y,x) by δ.

  Go to Step 2.
- *End.*

Minimum-cost flow

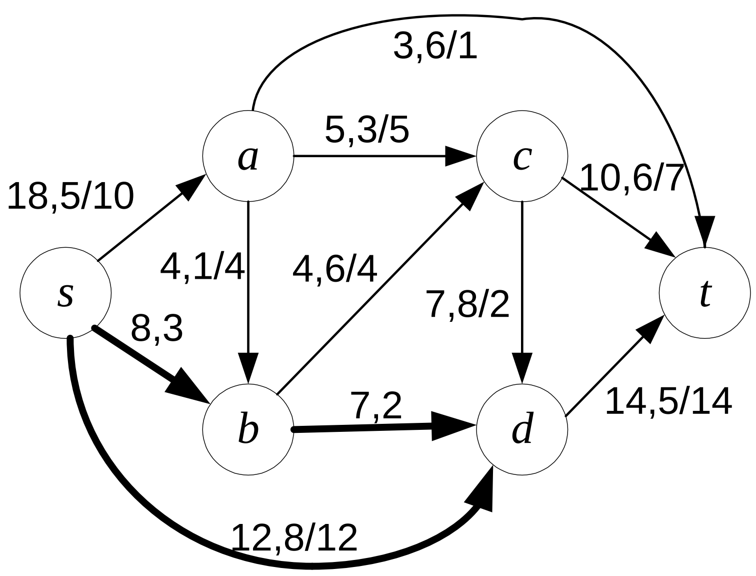
Example. μ=sbds (3+2–8=–3); δ=min{8,7,12}=7.

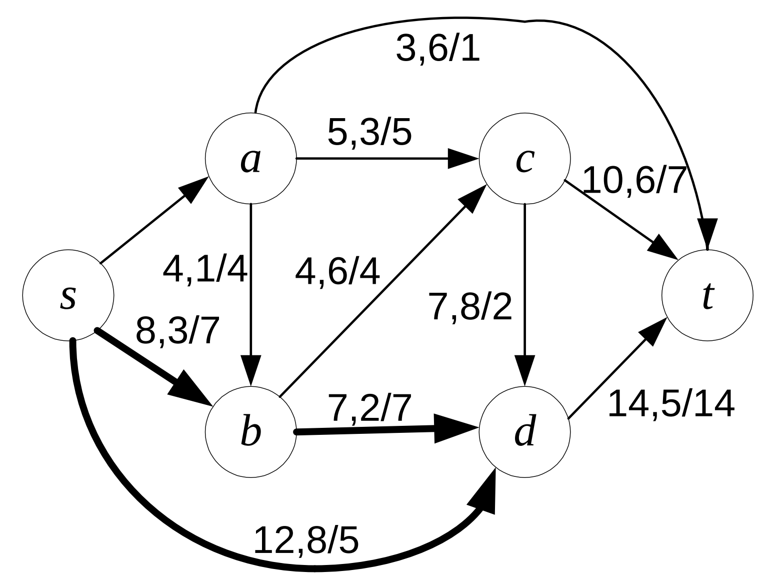




Minimum-cost flow

Example. Change the flow in the cycle.





Minimum-cost flow

Shortest-path algorithm.

- *Start.* Given a network $G(V,E,C,D)$ and the value θ, the residual network Gf=G.
- *Step 1.* Find the shortest path <s,t> (path with the minimum cost) in Gf. If there is not any, the solution does not exist, go to End.
- *Step 2*. Calculate the value δ=min{C(e)} where {e} is the set of arcs of the cycle. If δ > θ then set δ = θ.
- *Step 3.* Change the flow along the path:
  - if (x,y) is a forward arc then increase the flow in (x,y) by δ;
  - if (x,y) is a backward arc then decrease the flow in (y,x) by δ.
- *Step 4*. Decrease θ by δ. If θ=0 then a minimum-cost flow is constructed, go to the End.
- *Step 5.* Update the residual network G_f. Go to Step 1.
- *End.*

Minimum-cost flow

Example. The first shortest path is *sbdt*; the second is *sat*.