

Algorithms for Programming Contests - Week 05

Prof. Dr. Javier Esparza
Pranav Ashok, A. R. Balasubramanian,
Tobias Meggendorfer, Philipp Meyer,
Mikhail Raskin,
`conpra@in.tum.de`

20. Mai 2020

Flow Network

Definition (Flow network)

A *flow network* is a directed graph $G = (V, E)$, where each edge (u, v) is assigned a nonnegative *capacity* $c(u, v) \geq 0$ and there are two designated vertices, the *source* s and the *target* t .

W.l.o.g., we will only consider flow networks without antiparallel edges, i.e. if $(u, v) \in E$, then $(v, u) \notin E$.

Definition (Flow)

For a given flow network $G = (V, E)$ with capacity function c , a *flow* is a function $f: E \rightarrow \mathbb{R}$ satisfying

$$\begin{aligned} \forall (u, v) \in E: \quad & 0 \leq f(u, v) \leq c(u, v) \\ \forall u \in V \setminus \{s, t\}: \quad & \sum_{\{v: (v, u) \in E\}} f(v, u) = \sum_{\{v: (u, v) \in E\}} f(u, v) \end{aligned}$$

Maximum Flow Problem

Definition (Flow value)

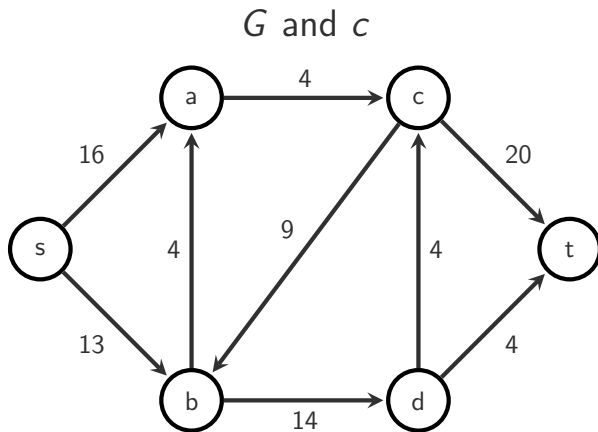
The *value* $|f|$ of a flow f is defined as

$$|f| = \sum_{\{v: (s,v) \in E\}} f(s,v) - \sum_{\{v: (v,s) \in E\}} f(v,s)$$

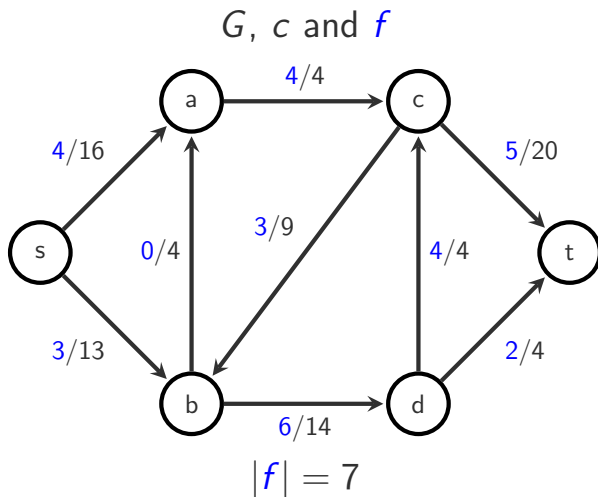
Definition (Maximum Flow Problem)

For a given flow network G with source s and target t , what is a flow f with maximal value $|f|$ over all flows?

Example: Flow network



Example: Flow network with flow



Reductions to maximum flow

Hints

- Minimum flow: send no flow at all.
- Multiple sources/sinks: add super-source/sink and edges with infinite capacity to other sources/sinks.
- Sources/sinks with supply constraints: add super-source/sink with edges to sources/sinks with corresponding capacity.
- Demands for sinks: check value of maximum flow.
- Vertex capacities: Split up vertex with edge of that capacity in between.
- Antiparallel edges: Insert vertex in between one edge (or use multi-edges when necessary).
- Undirected edges: Convert to two antiparallel directed edges.

Max-flow min-cut

Definition (Cut)

For a given flow network G with source s and target t , a *cut* $C = (S, T)$ is a partition of V into two subsets S and T such that $s \in S$ and $t \in T$. The *capacity* $c(S, T)$ of a cut (S, T) is defined as

$$c(S, T) = \sum_{(u,v) \in (S \times T) \cap E} c(u, v)$$

Theorem (Min-cut max-flow theorem)

The maximum value $|f|$ over all flows f is equal to the minimum capacity $c(S, T)$ over all cuts (S, T) .

Residual network

Definition (Residual capacity and residual network)

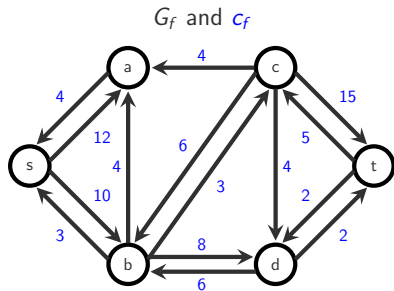
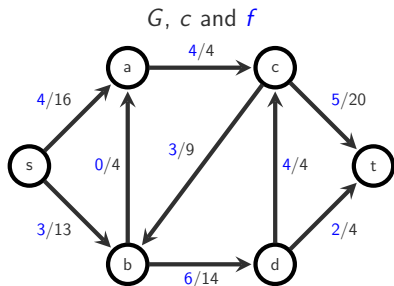
For a given flow network G and a flow f , and a pair of vertices $u, v \in V$, the *residual capacity* $c_f(u, v)$ is defined by

$$c_f(u, v) = \begin{cases} c(u, v) - f(u, v) & \text{if } (u, v) \in E \\ f(v, u) & \text{if } (v, u) \in E \\ 0 & \text{otherwise} \end{cases}$$

The *residual network* of G induced by f is $G_f = (V, E_f)$, where

$$E_f = \{(u, v) \in V \times V : c_f(u, v) > 0\}$$

Example: Residual network



Augmenting flow

Definition (Augmenting flows)

If f is a flow in a flow network G and f' is a flow in the corresponding residual network G_f , then the *augmentation* $f \uparrow f'$ of f by f' is a flow of G defined as

$$(f \uparrow f')(u, v) = \begin{cases} f(u, v) + f'(u, v) - f'(v, u) & \text{if } (u, v) \in E \\ 0 & \text{otherwise} \end{cases}$$

Lemma

If f is a flow in a flow network G and f' is a flow in the corresponding residual network G_f , then $f \uparrow f'$ is also a flow in G and

$$|f \uparrow f'| = |f| + |f'|$$

Augmenting path

Definition (Augmenting path)

Given a flow network G and a flow f , an *augmenting path* p is a simple path in the residual network G_f from s to t .

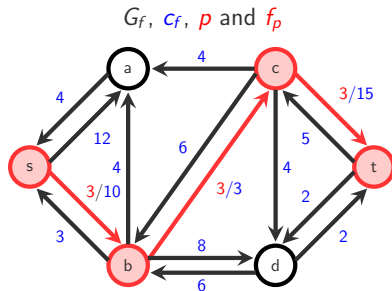
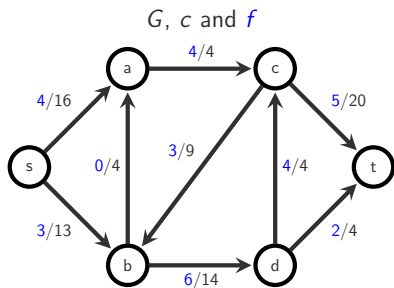
The *residual capacity* of an augmenting path p is given by

$$c_f(p) = \min\{c_f(u, v) : (u, v) \text{ is on } p\}$$

The flow f_p of an augmenting path p in G_f is defined as

$$f_p(u, v) = \begin{cases} c_f(p) & \text{if } (u, v) \text{ is on } p \\ 0 & \text{otherwise} \end{cases}$$

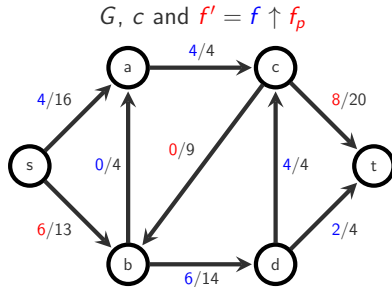
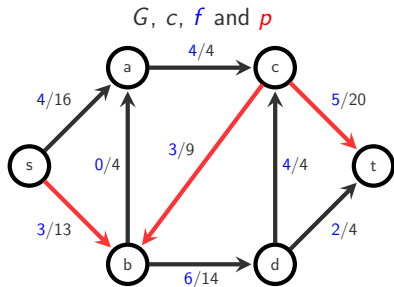
Example: Augmenting path



$$p = sbct$$

$$c_f(p) = 3$$

Example: Augmenting flow



$$p = sbct$$

$$c_f(p) = 3$$

Augmenting path algorithm (Ford-Fulkerson algorithm)

Algorithm 1 Ford-Fulkerson algorithm

```
▷ Initial flow is 0 ( $f \leftarrow 0$ )
for  $(u, v) \in E$  do
     $f(u, v) \leftarrow 0$ 
end for
while there exists a path  $p$  from  $s$  to  $t$  in the residual network  $G_f$  do
    ▷ Augment  $f$  by  $f_p$  ( $f \leftarrow f \uparrow f_p$ )
     $c_f(p) \leftarrow \min\{c_f(u, v) : (u, v) \text{ is on } p\}$ 
    for each edge  $(u, v)$  in  $p$  do
        if  $(u, v) \in E$  then
             $f(u, v) \leftarrow f(u, v) + c_f(p)$ 
        else
             $f(v, u) \leftarrow f(v, u) - c_f(p)$ 
        end if
    end for
end while
```

Analysis

How to decide with augmenting path to choose?

- Any path (with DFS): Ford-Fulkerson algorithm. Complexity $\mathcal{O}(|E| U)$ with integer capacities, where U is the value of the maximum flow. Possibly non-terminating for irrational capacities.
- Shortest path by number of edges (with BFS): Edmonds-Karp algorithm. Complexity $\mathcal{O}(|V| |E|^2)$.
- All shortest paths (blocking flows): Dinic's algorithm. Complexity $\mathcal{O}(|V|^2 |E|)$.

Blocking flow

Definition (Level graph)

Given a residual network $G_f = (V, E_f)$, let $d_{G_f}(s, v)$ be the length of the shortest path from s to v in G_f (by number of edges).

The *level graph* of G_f is the graph $G_L = (V, E_L, c_L)$, where

$$E_L = \{(u, v) \in E_f : d_{G_f}(s, v) = d_{G_f}(s, u) + 1\}$$
$$c_L(u, v) = \begin{cases} c_f(u, v) & \text{if } (u, v) \in E_L \\ 0 & \text{otherwise} \end{cases}$$

Definition (Blocking flow)

A *blocking flow* in the level graph G_L is a flow f such that every path from s to t in G_L contains a saturated edge, i.e., an edge (u, v) with $f(u, v) = c_L(u, v)$.

Dinic's algorithm

Algorithm 2 Dinic's algorithm

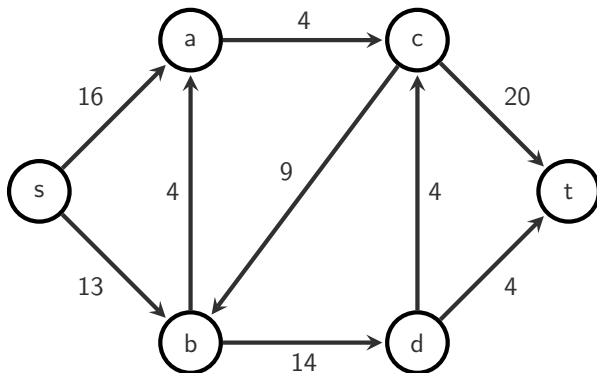
 $f \leftarrow 0$ **while** there exists a blocking flow f' in G_L with $|f'| > 0$ **do** $f \leftarrow f \uparrow f'$ **end while**

Finding a blocking flow

Algorithm 3 Finding blocking flows via DFS

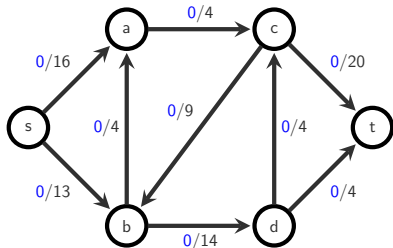
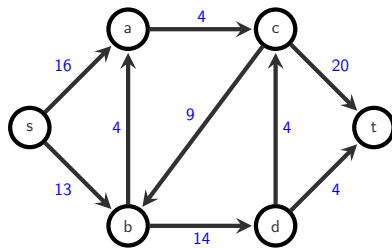
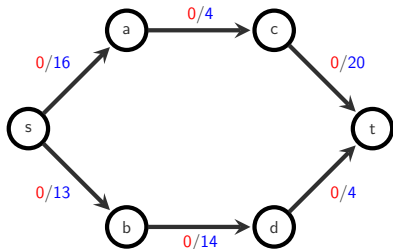
```
 $f' \leftarrow 0; p \leftarrow s; u \leftarrow s$   
while  $u \neq t$  do  
  while there is an edge  $(u, v) \in E_L$  with  $f'(u, v) < c_L(u, v)$  do  
     $p \leftarrow pv$   
     $u \leftarrow v$   
  end while  
  if  $u = t$  then  
     $f' \leftarrow f' \uparrow f_p; p \leftarrow s; u \leftarrow s$   
  else if  $u = s$  then  
    return  $f'$   
  else  
    let  $(v, w)$  be the last edge on  $p$ ; delete  $w$  from  $p$   
    delete  $(v, w)$  from  $E_L; u \leftarrow v$   
  end if  
end while
```






Dinic's algorithm (example)



- Maximum Flow

- Dinic's algorithm

 G, c and f  G_f and c_f  G_L, c_L and f' 

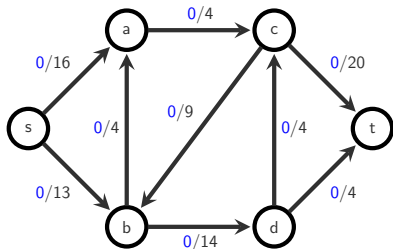
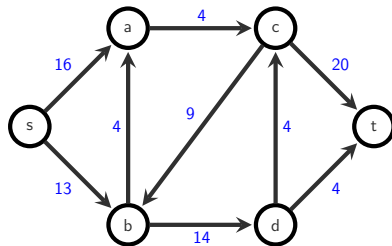
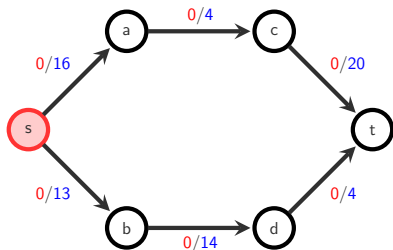
-  Vertex currently explored
-  Vertex on current path
-  Edge on augmenting path
-  Edge on current path
-  Saturated/deleted edge

Current operation:

Find blocking flow

- Maximum Flow

- Dinic's algorithm

 G, c and f  G_f and c_f  G_L, c_L and f' 

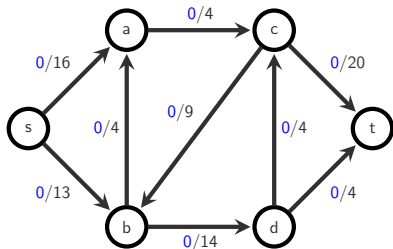
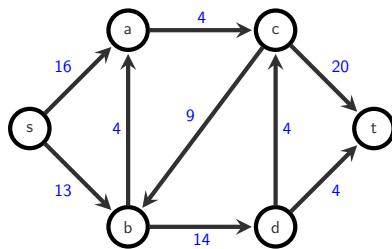
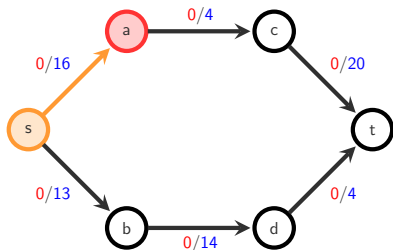
- Vertex currently explored
- Vertex on current path
- Edge on augmenting path
- Edge on current path
- Saturated/deleted edge

Current operation:

Find augmenting path

- Maximum Flow

- Dinic's algorithm

 G, c and f  G_f and c_f  G_L, c_L and f' 

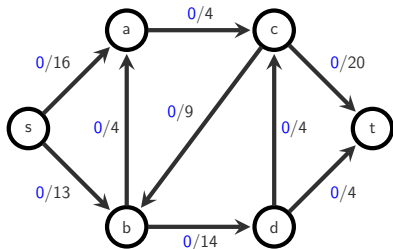
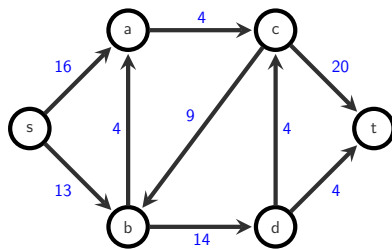
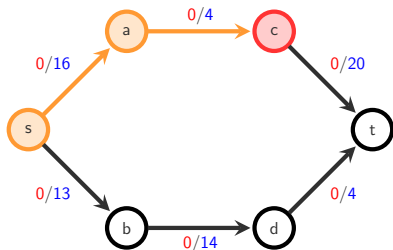
- Vertex currently explored
- Vertex on current path
- Edge on augmenting path
- Edge on current path
- Saturated/deleted edge

Current operation:

Find augmenting path

Maximum Flow

Dinic's algorithm

 G , c and f  G_f and c_f  G_L , c_L and f' 

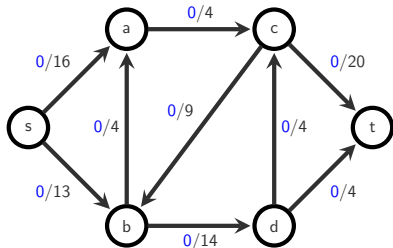
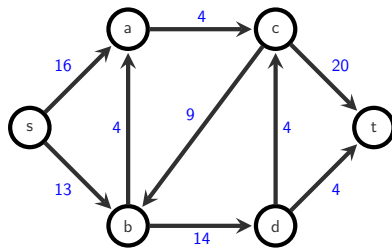
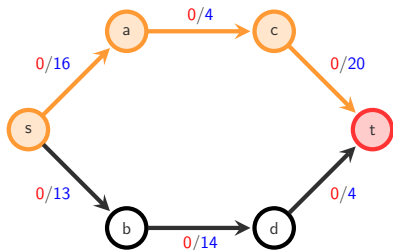
- Vertex currently explored
- Vertex on current path
- Edge on augmenting path
- Edge on current path
- Saturated/deleted edge

Current operation:

Find augmenting path

- Maximum Flow

- Dinic's algorithm

 G, c and f  G_f and c_f  G_L, c_L and f' 

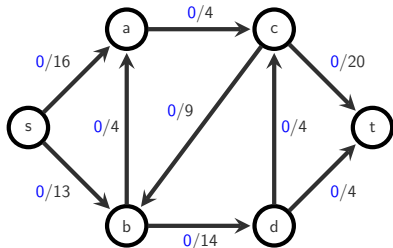
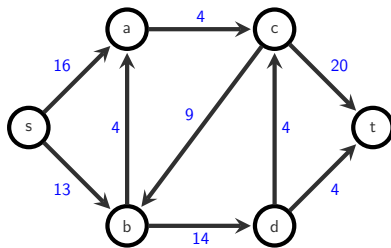
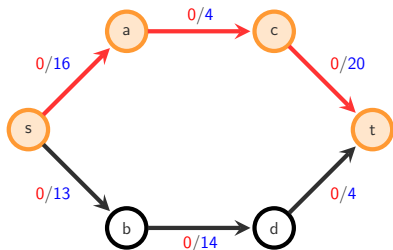
- v Vertex currently explored
- v Vertex on current path
- Edge on augmenting path
- Edge on current path
- Saturated/deleted edge

Current operation:

Find augmenting path

Maximum Flow

Dinic's algorithm

 G , c and f  G_f and c_f  G_L , c_L and f' 

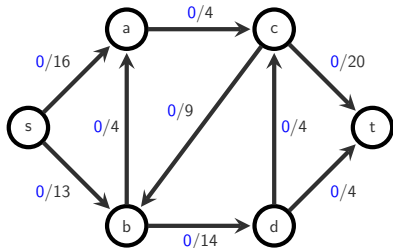
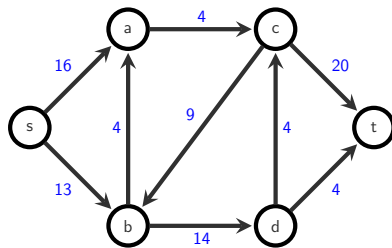
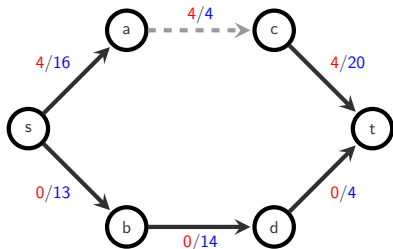
- v Vertex currently explored
- v Vertex on current path
- Edge on augmenting path
- Edge on current path
- Saturated/deleted edge






Current operation:

Augment f' by f_p

- Maximum Flow

- Dinic's algorithm

 G , c and f  G_f and c_f  G_L , c_L and f' 

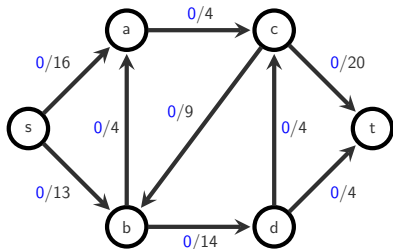
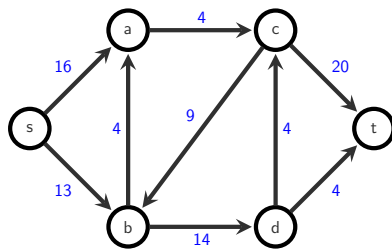
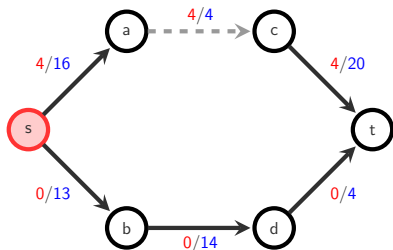
-  Vertex currently explored
-  Vertex on current path
-  Edge on augmenting path
-  Edge on current path
-  Saturated/deleted edge

Current operation:

Augment f' by f_p

- Maximum Flow

- Dinic's algorithm

 G , c and f  G_f and c_f  G_L , c_L and f' 

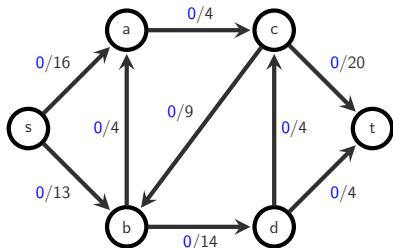
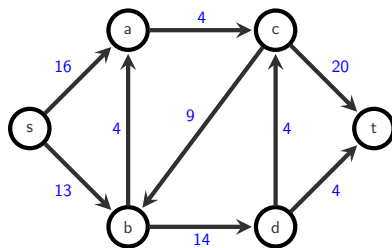
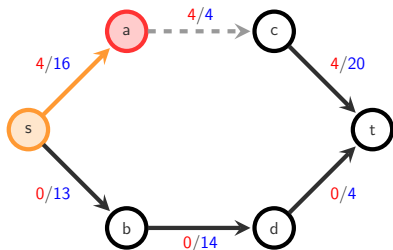
- Vertex currently explored
- Vertex on current path
- Edge on augmenting path
- Edge on current path
- Saturated/deleted edge

Current operation:

Find augmenting path

- Maximum Flow

- Dinic's algorithm

 G, c and f  G_f and c_f  G_L, c_L and f' 

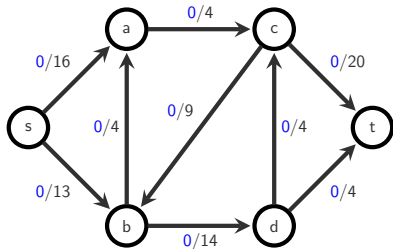
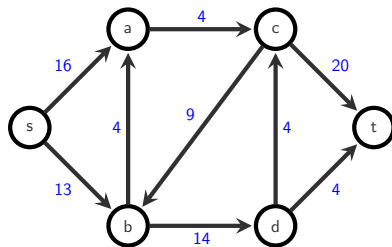
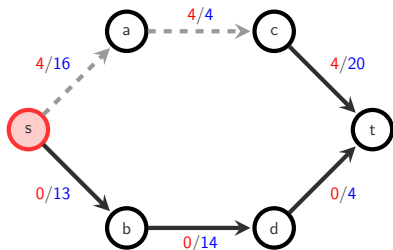
- Vertex currently explored
- Vertex on current path
- Edge on augmenting path
- Edge on current path
- Saturated/deleted edge






Current operation:

Find augmenting path

Maximum Flow

Dinic's algorithm

 G , c and f  G_f and c_f  G_L , c_L and f' 

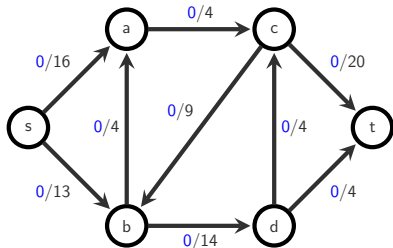
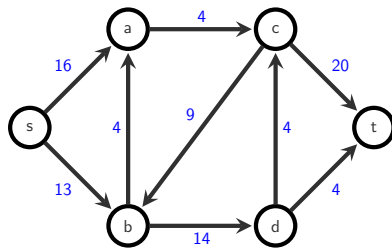
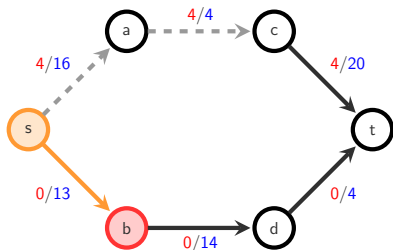
-  Vertex currently explored
-  Vertex on current path
-  Edge on augmenting path
-  Edge on current path
-  Saturated/deleted edge

Current operation:

Find augmenting path

Maximum Flow

Dinic's algorithm

 G , c and f  G_f and c_f  G_L , c_L and f' 

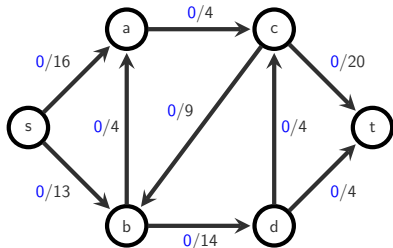
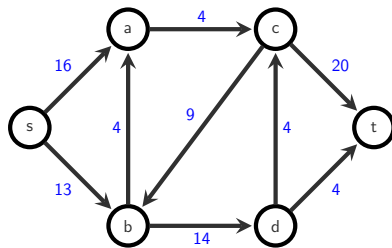
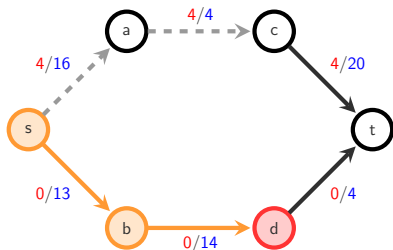
- v Vertex currently explored
- v Vertex on current path
- Edge on augmenting path
- Edge on current path
- Saturated/deleted edge

Current operation:

Find augmenting path

Maximum Flow

Dinic's algorithm

 G , c and f  G_f and c_f  G_L , c_L and f' 

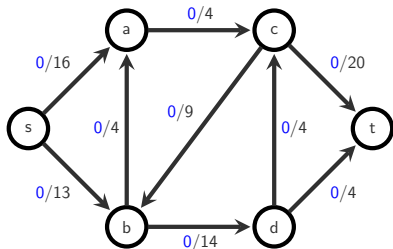
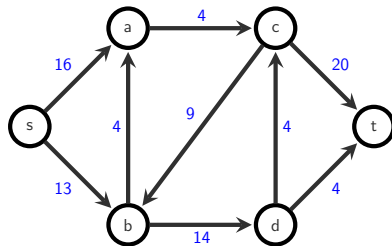
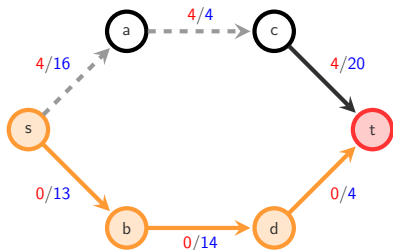
- v Vertex currently explored
- v Vertex on current path
- Edge on augmenting path
- Edge on current path
- Saturated/deleted edge

Current operation:

Find augmenting path

- Maximum Flow

- Dinic's algorithm

 G, c and f  G_f and c_f  G_L, c_L and f' 

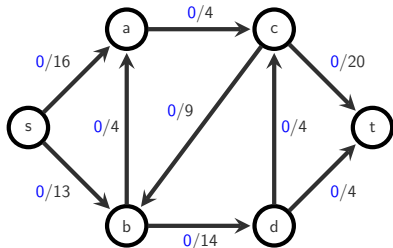
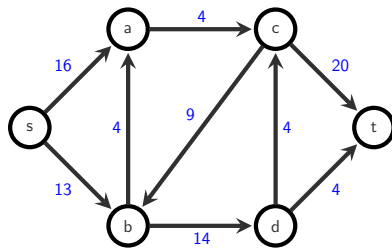
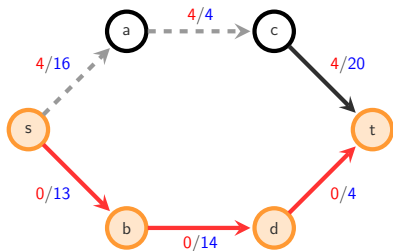
- v Vertex currently explored
- v Vertex on current path
- Edge on augmenting path
- Edge on current path
- Saturated/deleted edge

Current operation:

Find augmenting path

- Maximum Flow

- Dinic's algorithm

 G, c and f  G_f and c_f  G_L, c_L and f' 

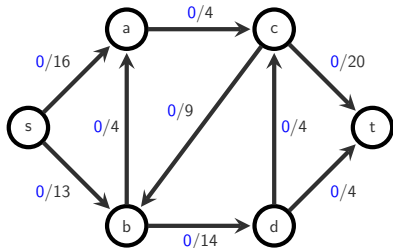
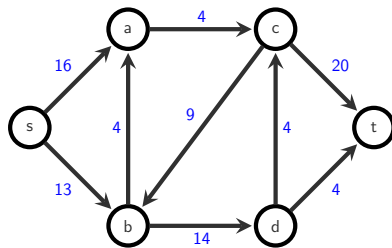
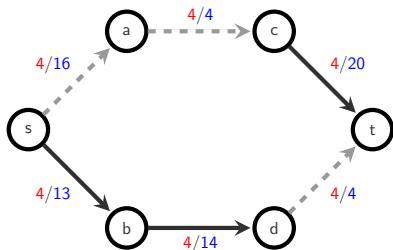
- Vertex currently explored
- Vertex on current path
- Edge on augmenting path
- Edge on current path
- Saturated/deleted edge






Current operation:

Augment f' by f_p

- Maximum Flow

- Dinic's algorithm

 G , c and f  G_f and c_f  G_L , c_L and f' 

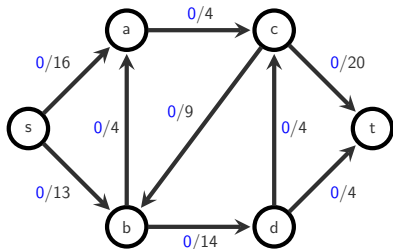
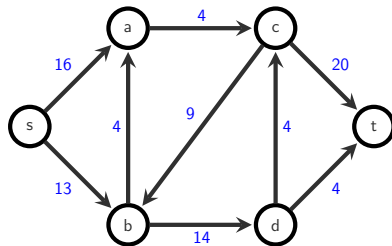
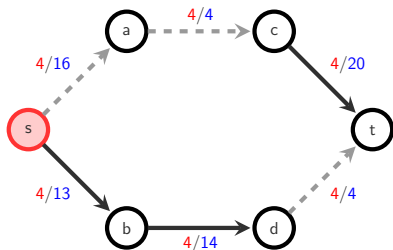
-  Vertex currently explored
-  Vertex on current path
-  Edge on augmenting path
-  Edge on current path
-  Saturated/deleted edge






Current operation:

Augment f' by f_p

Maximum Flow

Dinic's algorithm

 G , c and f  G_f and c_f  G_L , c_L and f' 

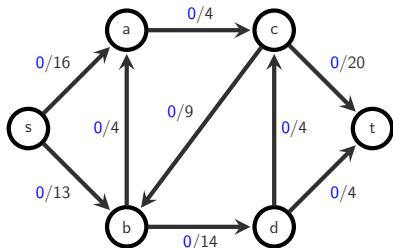
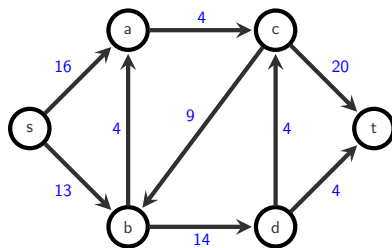
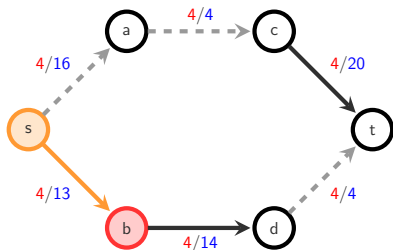
-  Vertex currently explored
-  Vertex on current path
-  Edge on augmenting path
-  Edge on current path
-  Saturated/deleted edge

Current operation:

Find augmenting path

- Maximum Flow

- Dinic's algorithm

 G, c and f  G_f and c_f  G_L, c_L and f' 

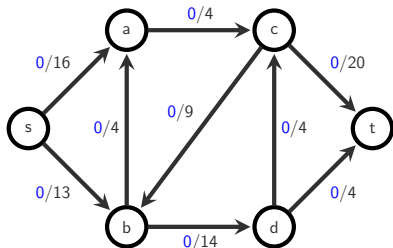
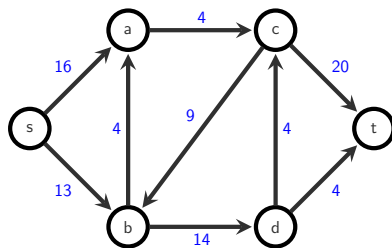
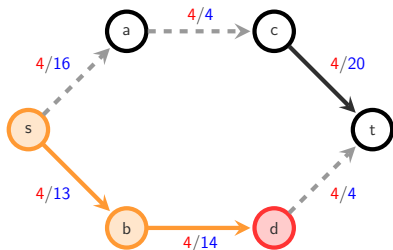
- Vertex currently explored
- Vertex on current path
- Edge on augmenting path
- Edge on current path
- Saturated/deleted edge

Current operation:

Find augmenting path

Maximum Flow

Dinic's algorithm

 G , c and f  G_f and c_f  G_L , c_L and f' 

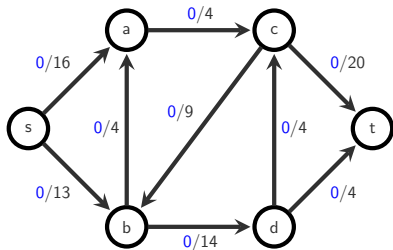
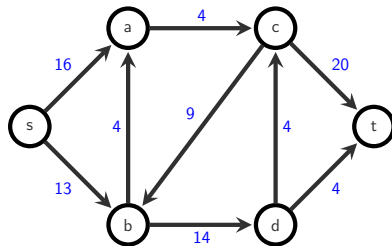
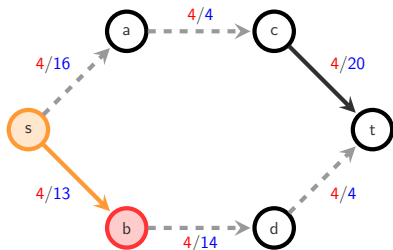
- v Vertex currently explored
- v Vertex on current path
- Edge on augmenting path
- Edge on current path
- Saturated/deleted edge

Current operation:

Find augmenting path

- Maximum Flow

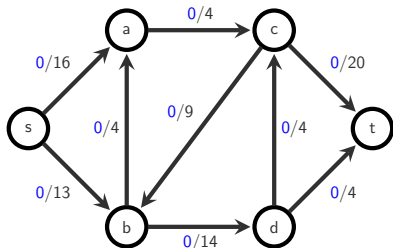
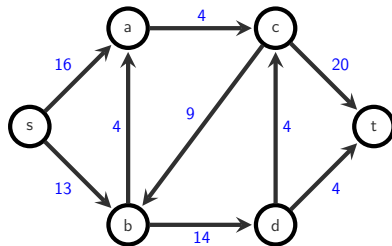
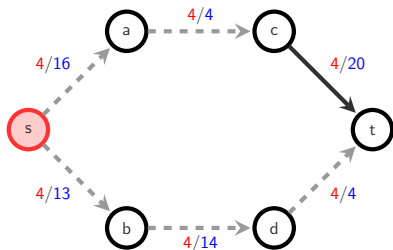
- Dinic's algorithm






 G , c and f  G_f and c_f  G_L , c_L and f' 

- Vertex currently explored
- Vertex on current path
- Edge on augmenting path
- Edge on current path
- Saturated/deleted edge

Current operation:

Find augmenting path

G , c and f  G_f and c_f  G_L , c_L and f' 

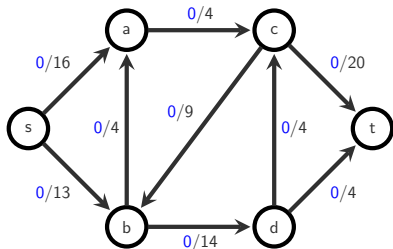
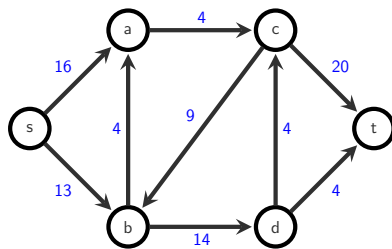
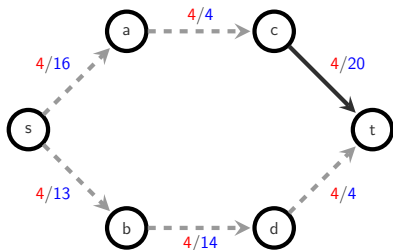
-  Vertex currently explored
-  Vertex on current path
-  Edge on augmenting path
-  Edge on current path
-  Saturated/deleted edge






Current operation:

Find augmenting path

- Maximum Flow

- Dinic's algorithm

 G, c and f  G_f and c_f  G_L, c_L and f' 

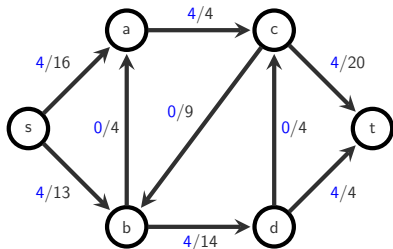
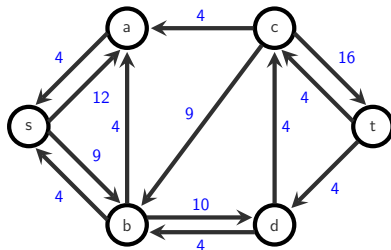
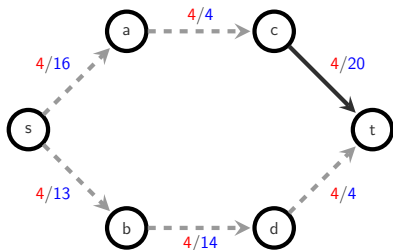
-  Vertex currently explored
-  Vertex on current path
-  Edge on augmenting path
-  Edge on current path
-  Saturated/deleted edge






Current operation:

Augment f by f'

Maximum Flow

Dinic's algorithm

 G , c and f  G_f and c_f  G_L , c_L and f' 

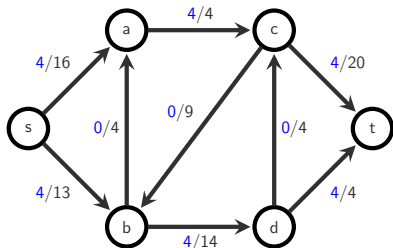
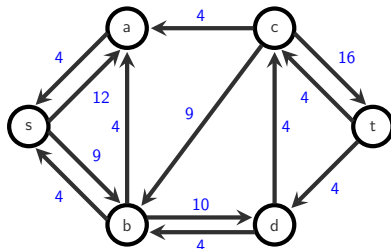
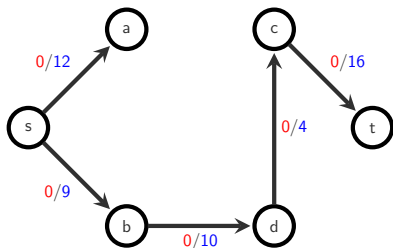
-  Vertex currently explored
-  Vertex on current path
-  Edge on augmenting path
-  Edge on current path
-  Saturated/deleted edge






Current operation:

Augment f by f'

Maximum Flow

Dinic's algorithm

 G , c and f  G_f and c_f  G_L , c_L and f' 

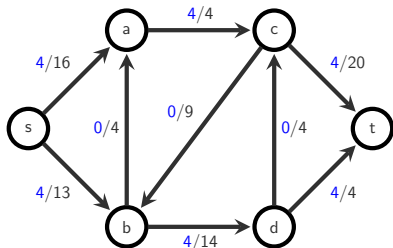
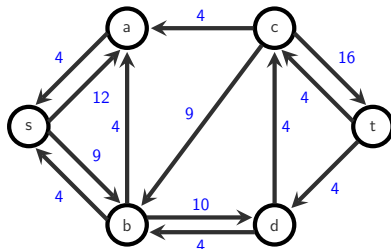
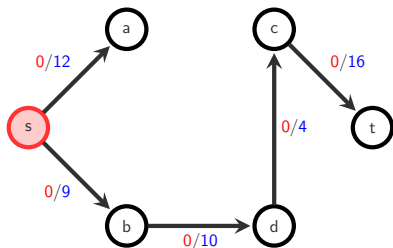
-  Vertex currently explored
-  Vertex on current path
-  Edge on augmenting path
-  Edge on current path
-  Saturated/deleted edge






Current operation:

Find blocking flow

Maximum Flow

Dinic's algorithm

 G , c and f  G_f and c_f  G_L , c_L and f' 

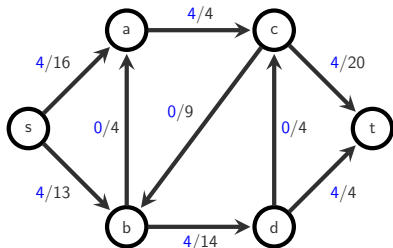
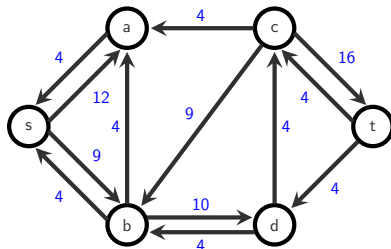
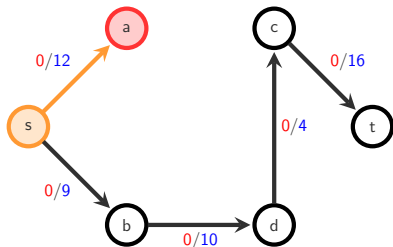
-  Vertex currently explored
-  Vertex on current path
-  Edge on augmenting path
-  Edge on current path
-  Saturated/deleted edge

Current operation:

Find augmenting path

Maximum Flow

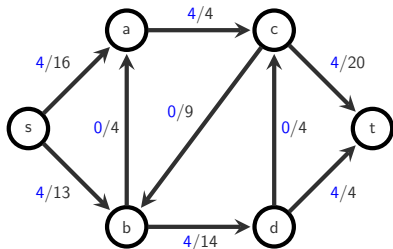
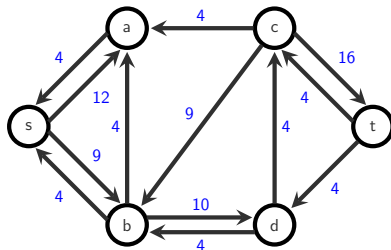
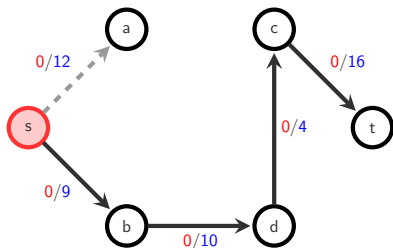
Dinic's algorithm






 G , c and f  G_f and c_f  G_L , c_L and f' 

- v Vertex currently explored
- v Vertex on current path
- Edge on augmenting path
- Edge on current path
- Saturated/deleted edge

Current operation:

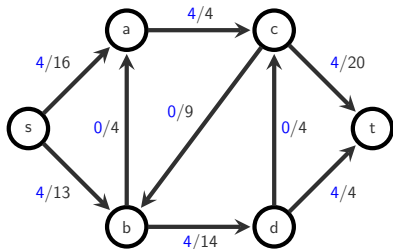
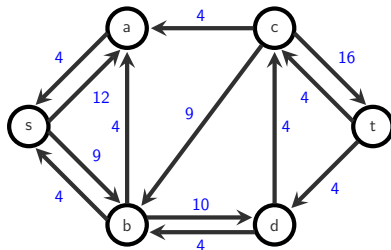
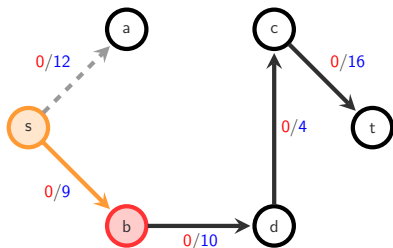
Find augmenting path

G , c and f  G_f and c_f  G_L , c_L and f' 

-  Vertex currently explored
-  Vertex on current path
-  Edge on augmenting path
-  Edge on current path
-  Saturated/deleted edge

Current operation:

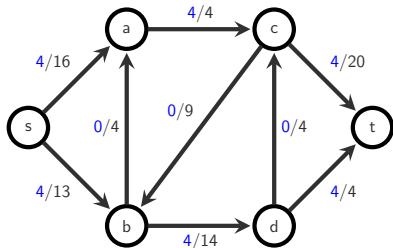
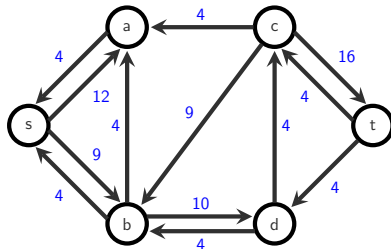
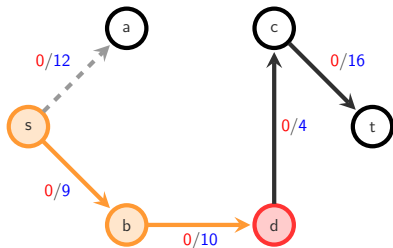
Find augmenting path

G , c and f  G_f and c_f  G_L , c_L and f' 

- v Vertex currently explored
- v Vertex on current path
- Edge on augmenting path
- Edge on current path
- Saturated/deleted edge

Current operation:

Find augmenting path

G , c and f  G_f and c_f  G_L , c_L and f' 

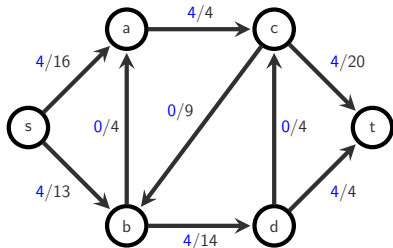
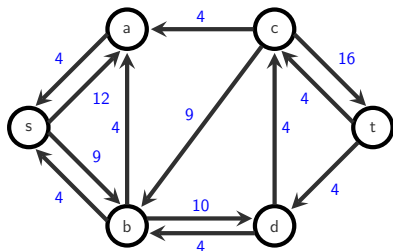
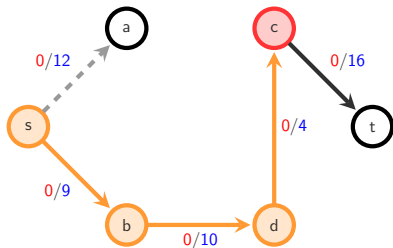
- v Vertex currently explored
- v Vertex on current path
- Edge on augmenting path
- Edge on current path
- Saturated/deleted edge

Current operation:

Find augmenting path

Maximum Flow

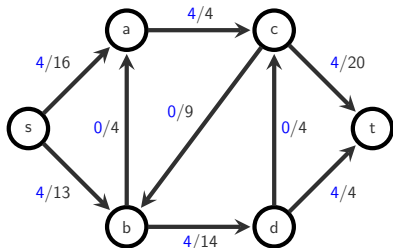
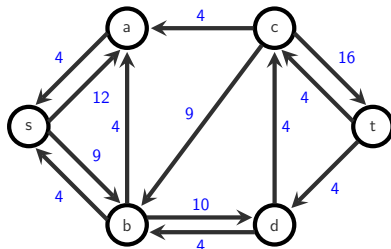
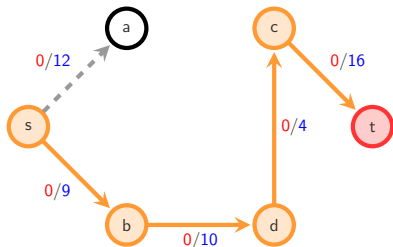
Dinic's algorithm

 G , c and f  G_f and c_f  G_L , c_L and f' 

- v Vertex currently explored
- v Vertex on current path
- Edge on augmenting path
- Edge on current path
- Saturated/deleted edge

Current operation:

Find augmenting path

G , c and f  G_f and c_f  G_L , c_L and f' 

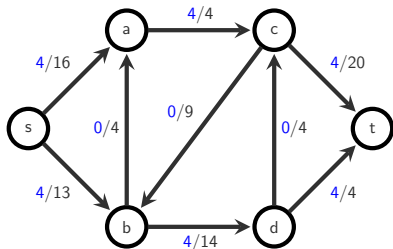
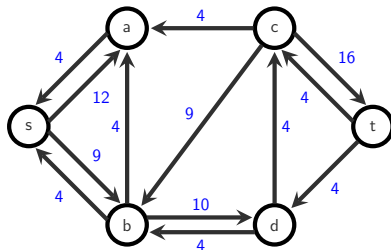
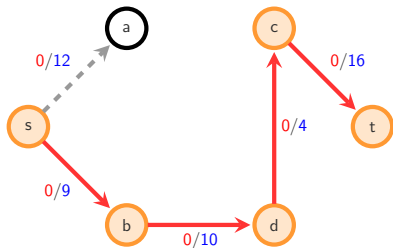
- v Vertex currently explored
- v Vertex on current path
- Edge on augmenting path
- Edge on current path
- Saturated/deleted edge






Current operation:

Find augmenting path

- Maximum Flow

- Dinic's algorithm

 G, c and f  G_f and c_f  G_L, c_L and f' 

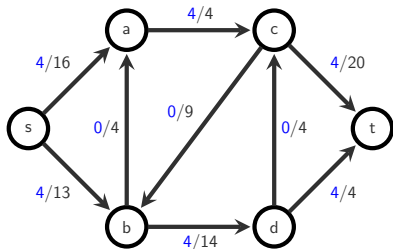
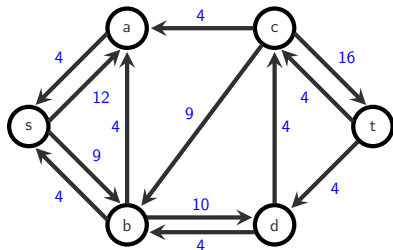
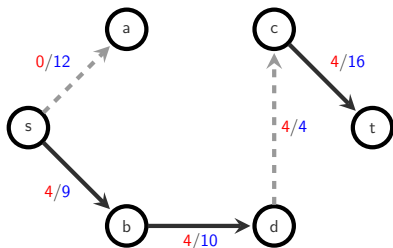
-  Vertex currently explored
-  Vertex on current path
-  Edge on augmenting path
-  Edge on current path
-  Saturated/deleted edge






Current operation:

Augment f' by f_p

Maximum Flow

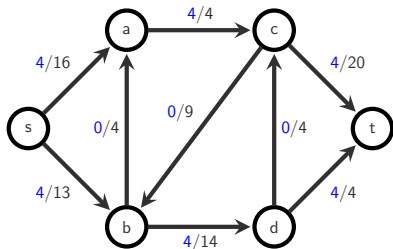
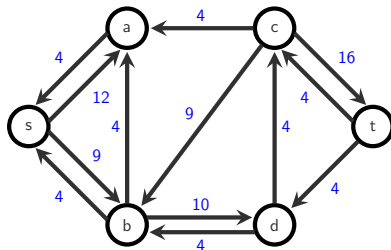
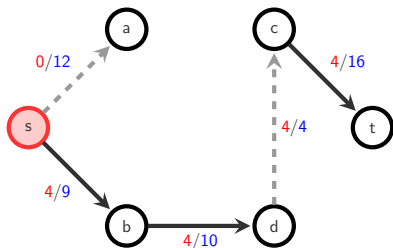
Dinic's algorithm

 G , c and f  G_f and c_f  G_L , c_L and f' 

-  Vertex currently explored
-  Vertex on current path
-  Edge on augmenting path
-  Edge on current path
-  Saturated/deleted edge

Current operation:

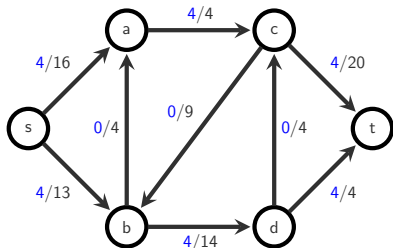
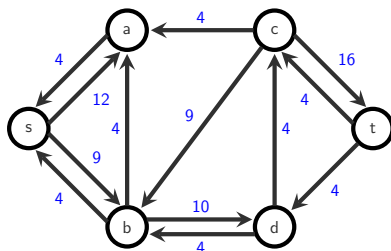
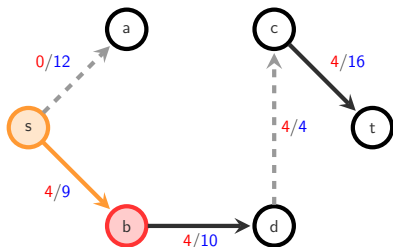
Augment f' by f_p

G , c and f  G_f and c_f  G_L , c_L and f' 

- v Vertex currently explored
- v Vertex on current path
- Edge on augmenting path
- Edge on current path
- Saturated/deleted edge

Current operation:

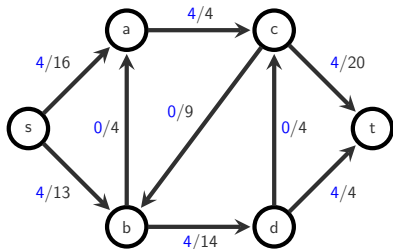
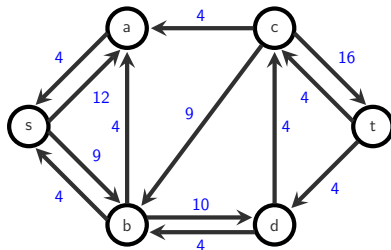
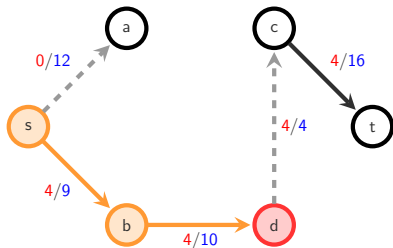
Find augmenting path

G , c and f  G_f and c_f  G_L , c_L and f' 

- v Vertex currently explored
- v Vertex on current path
- Edge on augmenting path
- Edge on current path
- Saturated/deleted edge

Current operation:

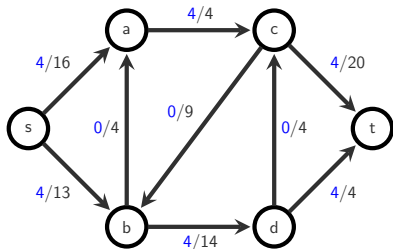
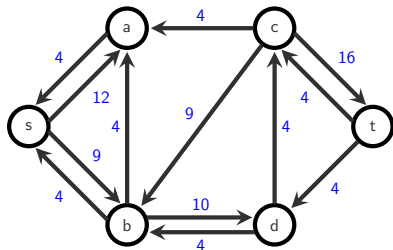
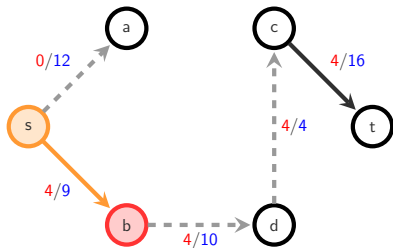
Find augmenting path

G , c and f  G_f and c_f  G_L , c_L and f' 

- v Vertex currently explored
- v Vertex on current path
- Edge on augmenting path
- Edge on current path
- Saturated/deleted edge

Current operation:

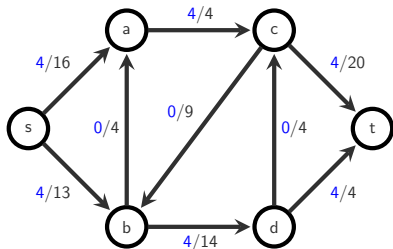
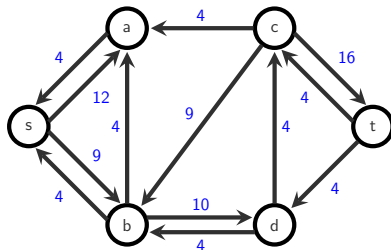
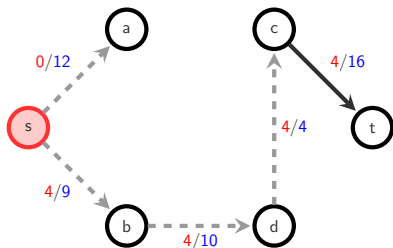
Find augmenting path

G , c and f  G_f and c_f  G_L , c_L and f' 

- v Vertex currently explored
- v Vertex on current path
- Edge on augmenting path
- Edge on current path
- Saturated/deleted edge

Current operation:

Find augmenting path

G , c and f  G_f and c_f  G_L , c_L and f' 

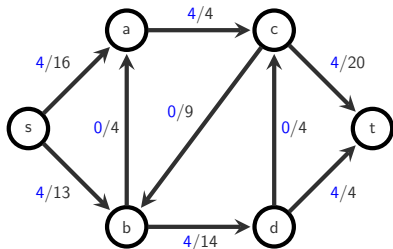
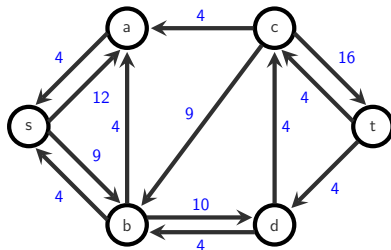
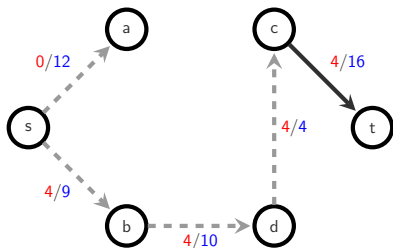
- v Vertex currently explored
- v Vertex on current path
- Edge on augmenting path
- Edge on current path
- Saturated/deleted edge






Current operation:

Find augmenting path

Maximum Flow

Dinic's algorithm

 G , c and f  G_f and c_f  G_L , c_L and f' 

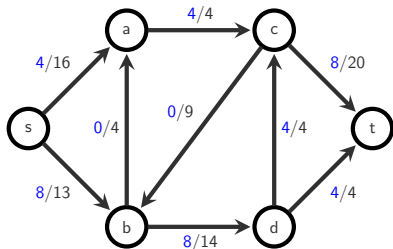
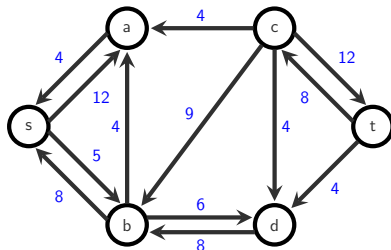
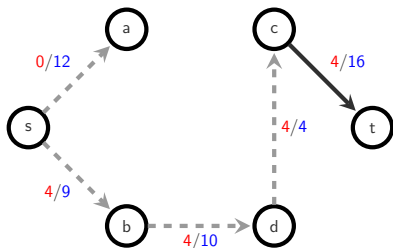
-  Vertex currently explored
-  Vertex on current path
-  Edge on augmenting path
-  Edge on current path
-  Saturated/deleted edge






Current operation:

Augment f by f'

Maximum Flow

Dinic's algorithm

 G , c and f  G_f and c_f  G_L , c_L and f' 

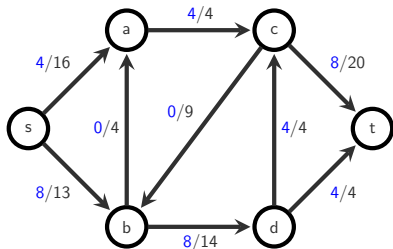
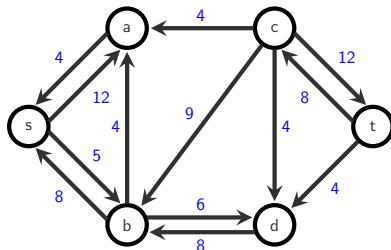
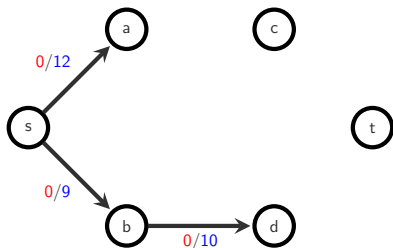
-  Vertex currently explored
-  Vertex on current path
-  Edge on augmenting path
-  Edge on current path
-  Saturated/deleted edge






Current operation:

Augment f by f'

Maximum Flow

Dinic's algorithm

 G , c and f  G_f and c_f  G_L , c_L and f' 

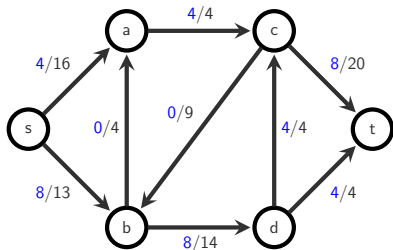
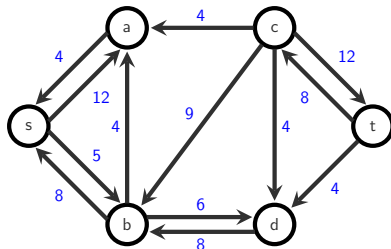
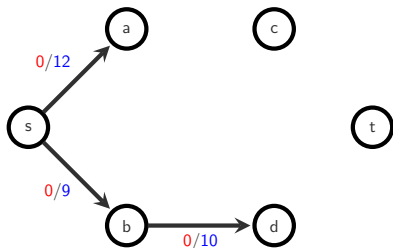
-  Vertex currently explored
-  Vertex on current path
-  Edge on augmenting path
-  Edge on current path
-  Saturated/deleted edge






Current operation:

Find blocking flow

Maximum Flow

Dinic's algorithm

 G , c and f  G_f and c_f  G_L , c_L and f' 

-  Vertex currently explored
-  Vertex on current path
-  Edge on augmenting path
-  Edge on current path
-  Saturated/deleted edge

No path from s to t :**Maximum flow found**

Push-relabel algorithms: Preflow

The class of *push-relabel* algorithms for maximum flow work by maintaining a *preflow* and pushing it along edges, while (re-)labeling vertices to determine where flow can be pushed.

Definition (Preflow)

For a given flow network $G = (V, E)$ with capacity function c , a *preflow* is a function $f: E \rightarrow \mathbb{R}$ satisfying

$$\begin{aligned} \forall (u, v) \in E: \quad & 0 \leq f(u, v) \leq c(u, v) \\ \forall u \in V \setminus \{s, t\}: \quad & \sum_{\{v: (v, u) \in E\}} f(v, u) - \sum_{\{v: (u, v) \in E\}} f(u, v) \geq 0 \end{aligned}$$

Push-relabel algorithms: Excess flow and height labels

Definition (Excess flow)

For a given flow network G and a preflow f , the *excess flow* $e(u)$ of a vertex u is given by

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

A vertex $u \in V \setminus \{s, t\}$ is said to be *overflowing* if $e(u) > 0$.

Definition (Height function)

For a given flow network G and a flow f , a function $h: V \rightarrow \mathbb{N}$ is a *height function* if $h(s) = |V|$, $h(t) = 0$ and $h(u) \leq h(v) + 1$ for every residual edge $(u, v) \in E_f$.

Push and relabel operations

Algorithm 4 Push operation

▷ Applies to $(u, v) \in E_f$ when u is overflowing and $h(u) = h(v) + 1$
 $\Delta_f(u, v) \leftarrow \min(e(u), c_f(u, v))$
if $(u, v) \in E$ **then**
 $f(u, v) \leftarrow f(u, v) + \Delta_f(u, v)$
else
 $f(v, u) \leftarrow f(v, u) - \Delta_f(u, v)$
end if
 $e(u) \leftarrow e(u) - \Delta_f(u, v)$
 $e(v) \leftarrow e(v) + \Delta_f(u, v)$

Algorithm 5 Relabel operation

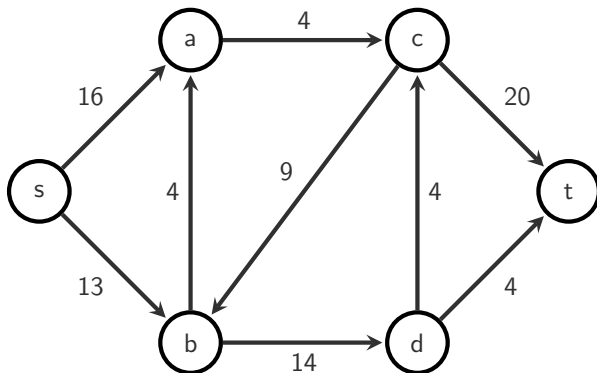
▷ Applies to u when u is overflowing and $h(u) \leq h(v)$ for all $(u, v) \in E_f$
 $h(u) \leftarrow 1 + \min\{h(v) : (u, v) \in E_f\}$

Push-relabel algorithm (Goldberg-Tarjan algorithm)

Algorithm 6 Push-relabel algorithm

```
for each vertex  $v \in V$  do  
     $h(v) \leftarrow 0$ ;  $e(v) \leftarrow 0$   
end for  
for  $(u, v) \in E$  do  
     $f(u, v) \leftarrow 0$   
end for  
 $h(s) \leftarrow |V|$   
for each vertex  $v \in sE$  do  
     $f(s, v) \leftarrow c(s, v)$   
     $e(v) \leftarrow e(v) + c(s, v)$   
     $e(s) \leftarrow e(s) - c(s, v)$   
end for  
while there is an applicable push or relabel operation do  
    select an applicable push or relabel operation and perform it  
end while
```

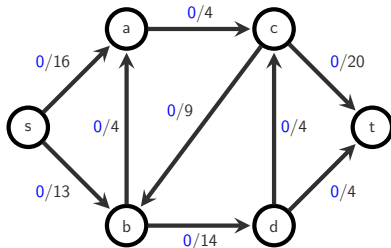
Push-relabel algorithm (example)



Maximum Flow

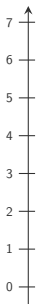
Push-relabel algorithm

G , c and f

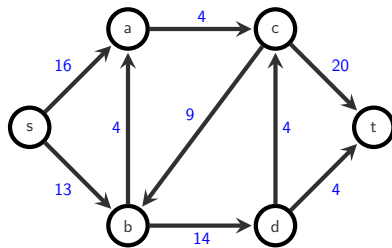


$h(v)$ and $e(v)$

$h(v)$



G_f and c_f



Currently selected vertex



Overflowing vertex



Edge selected for push



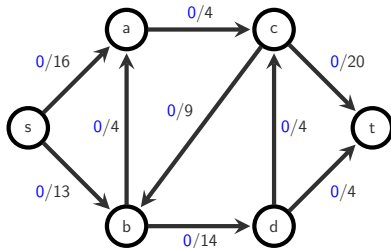
Edge available for push

Current operation:

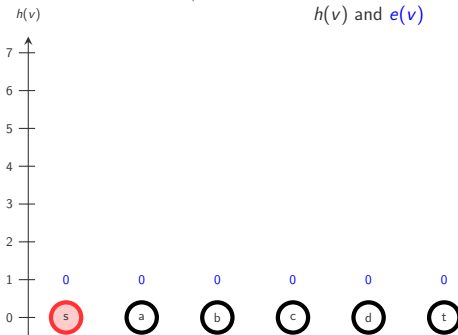
Maximum Flow

Push-relabel algorithm

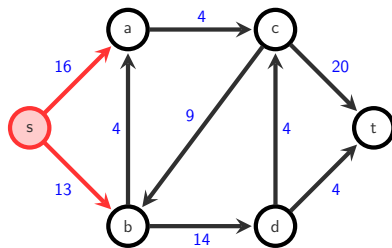
G , c and f



$h(v)$ and $e(v)$



G_f and c_f



- v Currently selected vertex
- v Overflowing vertex
- Edge selected for push
- Edge available for push

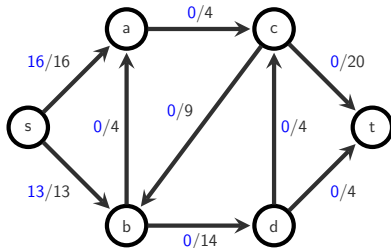
Current operation:

Initialize preflow

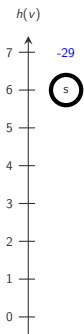
└ Maximum Flow

└ Push-relabel algorithm

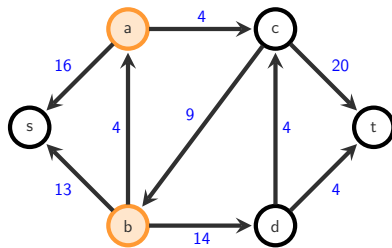
G, c and f



$h(v)$ and $e(v)$



G_f and c_f



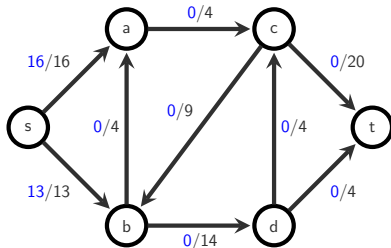
- v Currently selected vertex
- v Overflowing vertex
- Edge selected for push
- Edge available for push

Current operation:

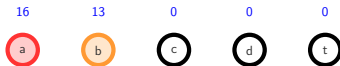
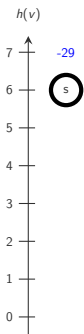
└ Maximum Flow

└ Push-relabel algorithm

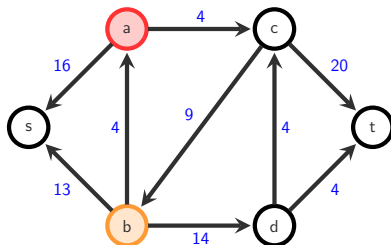
G, c and f



$h(v)$ and $e(v)$



G_f and c_f



- v Currently selected vertex
- v Overflowing vertex
- Edge selected for push
- Edge available for push

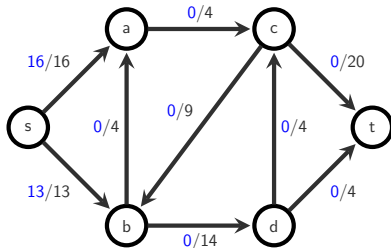
Current operation:

Relabel a

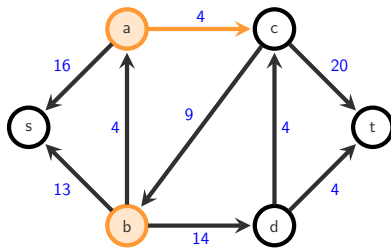
└ Maximum Flow

└ Push-relabel algorithm

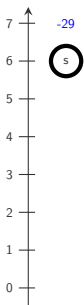
G, c and f



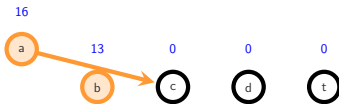
G_f and c_f



$h(v)$



$h(v)$ and $e(v)$



Currently selected vertex



Overflowing vertex



Edge selected for push



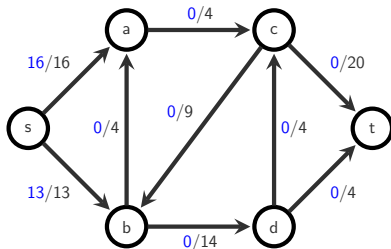
Edge available for push

Current operation:

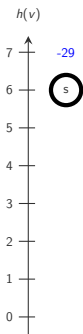
└ Maximum Flow

└ Push-relabel algorithm

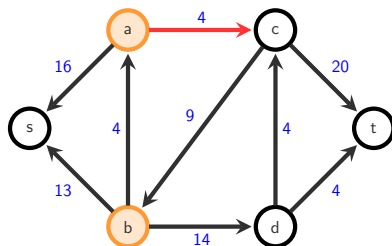
G, c and f



$h(v)$ and $e(v)$



G_f and c_f



Currently selected vertex



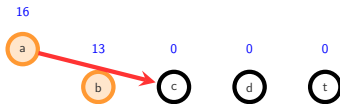
Overflowing vertex



Edge selected for push



Edge available for push



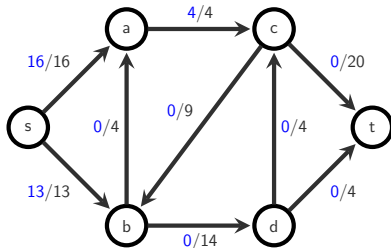
Current operation:

Push from a to c

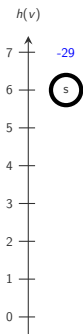
└ Maximum Flow

└ Push-relabel algorithm

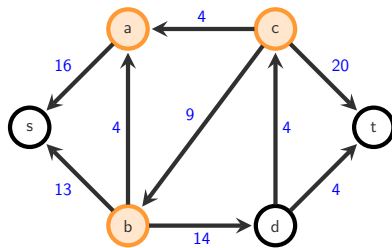
G, c and f



$h(v)$ and $e(v)$



G_f and c_f



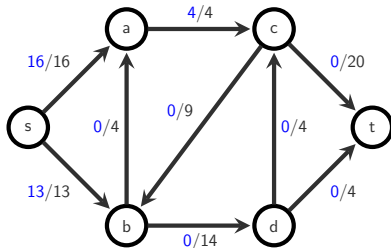
- v Currently selected vertex
- v Overflowing vertex
- Edge selected for push
- Edge available for push

Current operation:

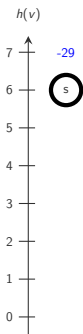
└ Maximum Flow

└ Push-relabel algorithm

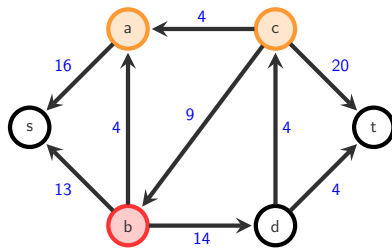
G, c and f



$h(v)$ and $e(v)$



G_f and c_f



- v Currently selected vertex
- v Overflowing vertex
- Edge selected for push
- Edge available for push

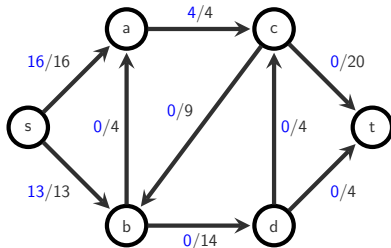
Current operation:

Relabel b

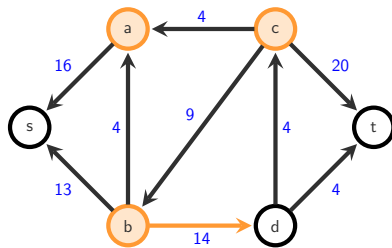
└ Maximum Flow

└ Push-relabel algorithm

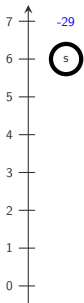
G, c and f



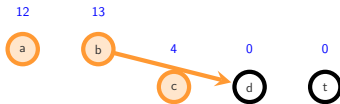
G_f and c_f



$h(v)$



$h(v)$ and $e(v)$



Currently selected vertex



Overflowing vertex



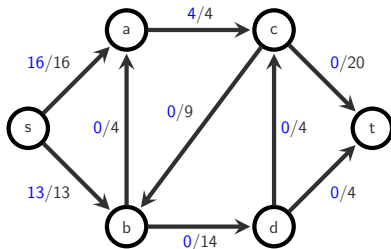
Edge selected for push



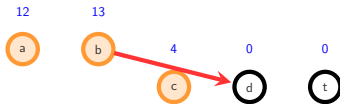
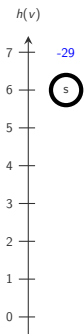
Edge available for push

Current operation:

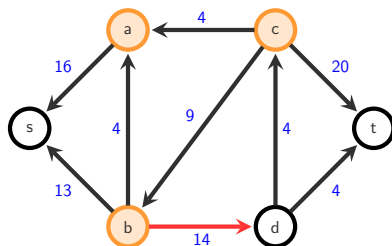
G , c and f



$h(v)$ and $e(v)$



G_f and c_f



- v Currently selected vertex
- v Overflowing vertex
- Edge selected for push
- Edge available for push

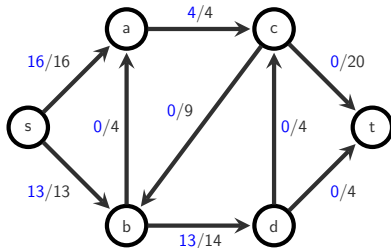
Current operation:

Push from b to d

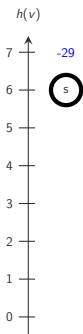
└ Maximum Flow

└ Push-relabel algorithm

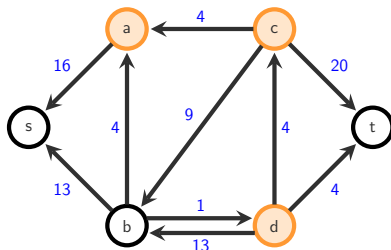
G, c and f



$h(v)$ and $e(v)$



G_f and c_f



Currently selected vertex



Overflowing vertex



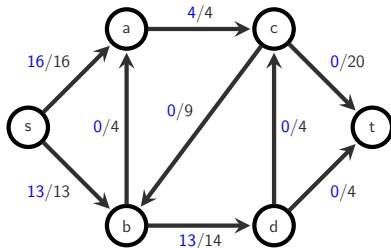
Edge selected for push



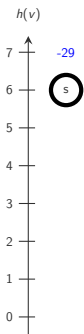
Edge available for push

Current operation:

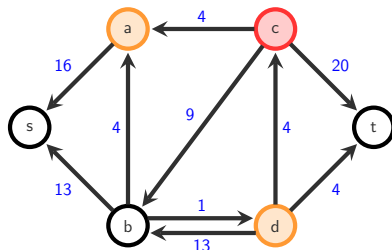
G, c and f



$h(v)$ and $e(v)$



G_f and c_f



- v Currently selected vertex
- v Overflowing vertex
- Edge selected for push
- Edge available for push

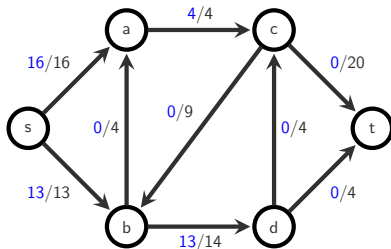
Current operation:

Relabel c

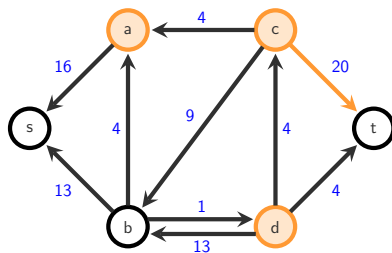
└ Maximum Flow

└ Push-relabel algorithm

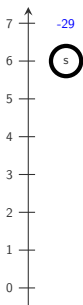
G, c and f



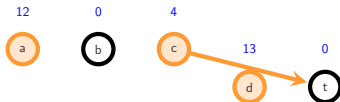
G_f and c_f



$h(v)$



$h(v)$ and $e(v)$



Currently selected vertex



Overflowing vertex



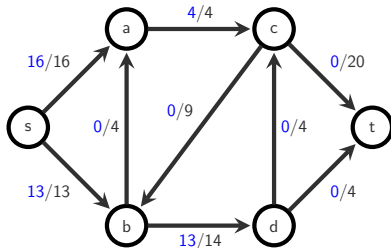
Edge selected for push



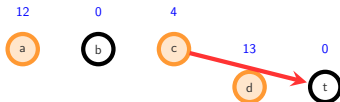
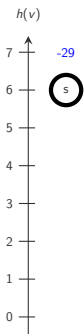
Edge available for push

Current operation:

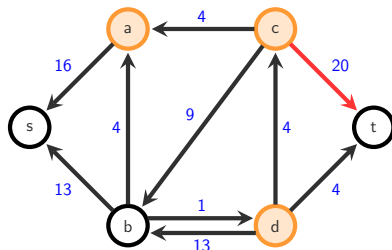
G, c and f



$h(v)$ and $e(v)$



G_f and c_f



- v Currently selected vertex
- v Overflowing vertex
- Edge selected for push
- Edge available for push

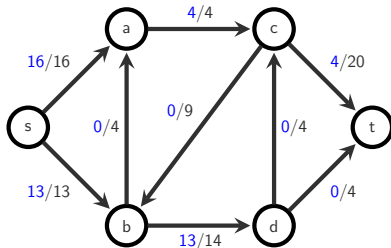
Current operation:

Push from c to t

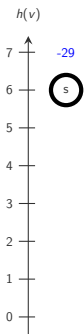
└ Maximum Flow

└ Push-relabel algorithm

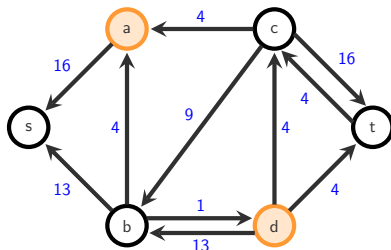
G, c and f



$h(v)$ and $e(v)$



G_f and c_f



Currently selected vertex



Overflowing vertex

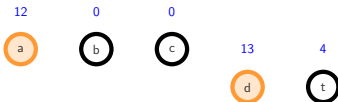


Edge selected for push



Edge available for push

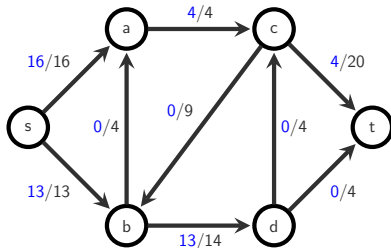
Current operation:



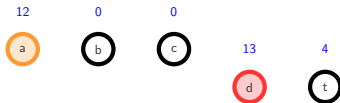
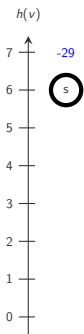
└ Maximum Flow

└ Push-relabel algorithm

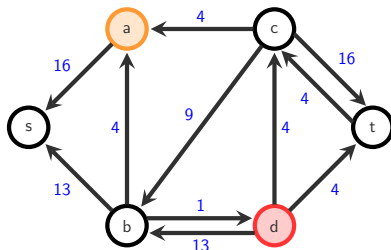
G, c and f



$h(v)$ and $e(v)$



G_f and c_f



Currently selected vertex



Overflowing vertex



Edge selected for push

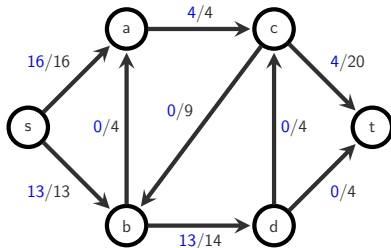


Edge available for push

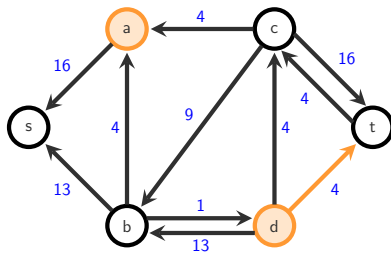
Current operation:

Relabel d

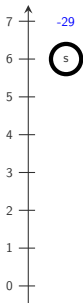
G , c and f



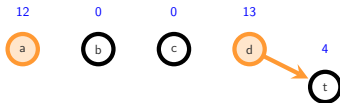
G_f and c_f



$h(v)$



$h(v)$ and $e(v)$



Currently selected vertex



Overflowing vertex



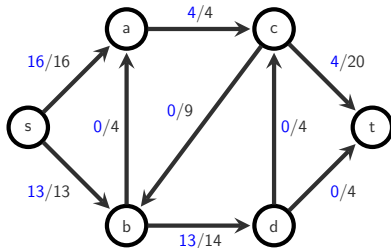
Edge selected for push



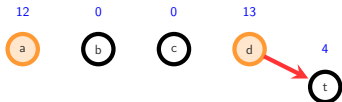
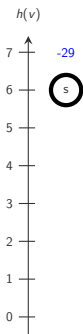
Edge available for push

Current operation:

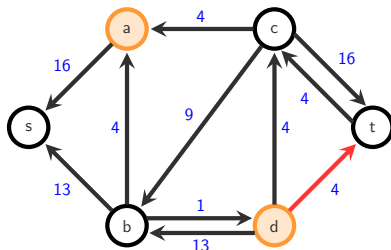
G , c and f



$h(v)$ and $e(v)$



G_f and c_f



- v Currently selected vertex
- v Overflowing vertex
- Edge selected for push
- Edge available for push

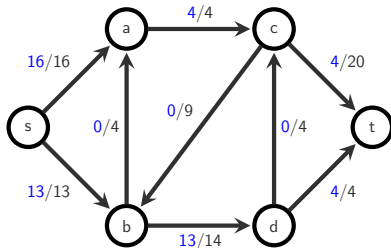
Current operation:

Push from d to t

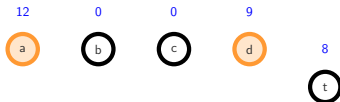
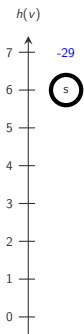
└ Maximum Flow

└ Push-relabel algorithm

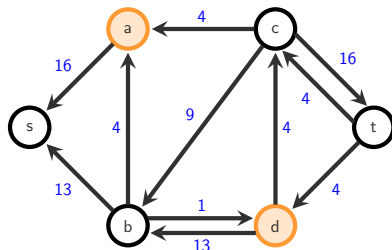
G, c and f



$h(v)$ and $e(v)$



G_f and c_f



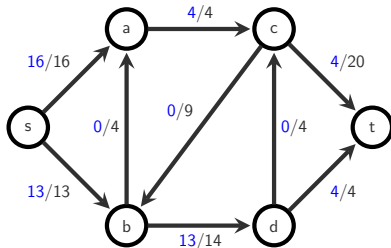
- v Currently selected vertex
- v Overflowing vertex
- Edge selected for push
- Edge available for push

Current operation:

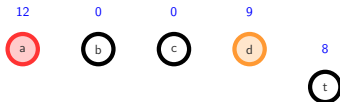
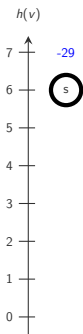
└ Maximum Flow

└ Push-relabel algorithm

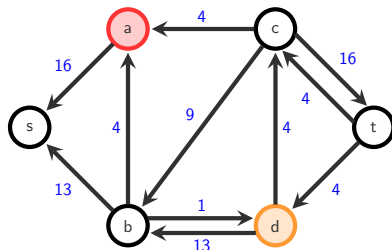
G, c and f



$h(v)$ and $e(v)$



G_f and c_f



Currently selected vertex



Overflowing vertex



Edge selected for push

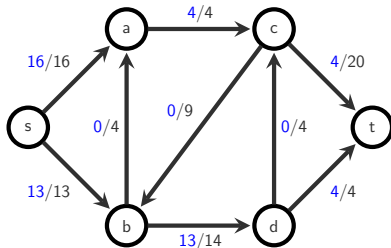


Edge available for push

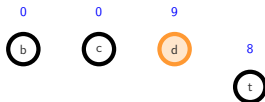
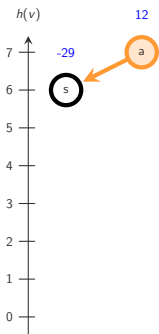
Current operation:

Relabel a

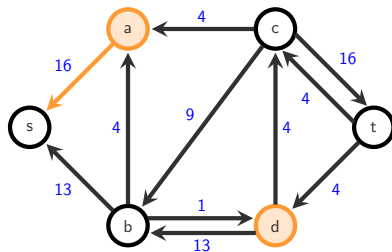
G, c and f



$h(v)$ and $e(v)$



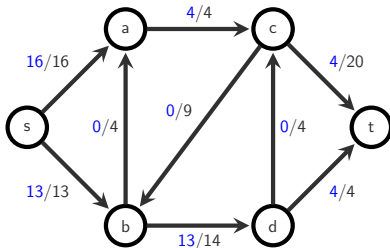
G_f and c_f



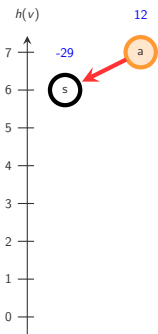
- v Currently selected vertex
- v Overflowing vertex
- Edge selected for push
- Edge available for push

Current operation:

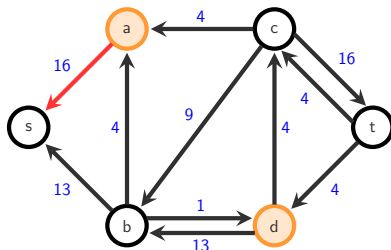
G, c and f



$h(v)$ and $e(v)$



G_f and c_f



- v Currently selected vertex
- v Overflowing vertex
- Edge selected for push
- Edge available for push

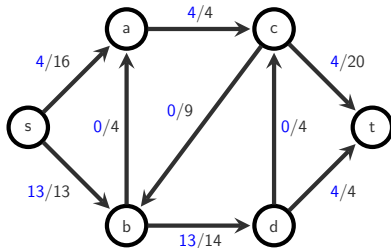
Current operation:

Push from a to s

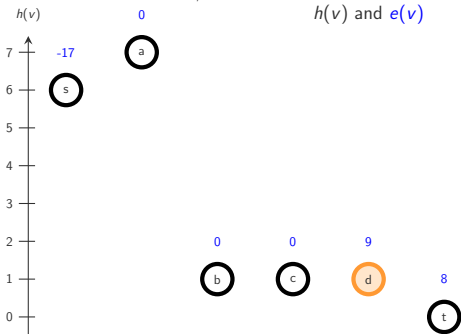
└ Maximum Flow

└ Push-relabel algorithm

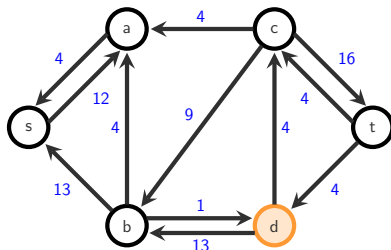
G, c and f



$h(v)$ and $e(v)$



G_f and c_f



Currently selected vertex



Overflowing vertex



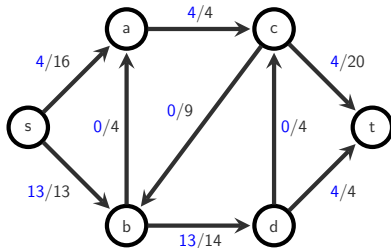
Edge selected for push



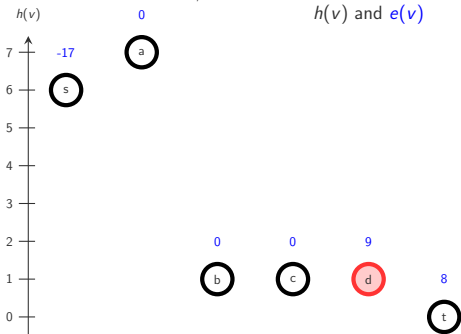
Edge available for push

Current operation:

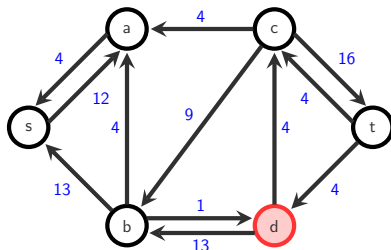
G, c and f



$h(v)$ and $e(v)$



G_f and c_f



Currently selected vertex



Overflowing vertex



Edge selected for push

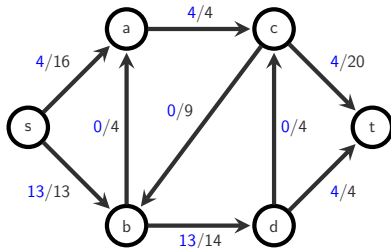


Edge available for push

Current operation:

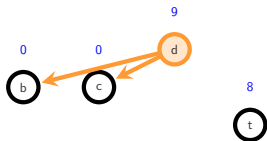
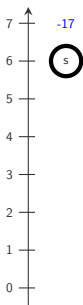
Relabel d

G, c and f

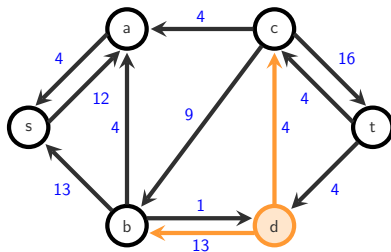


$h(v)$ and $e(v)$

$h(v)$



G_f and c_f



Currently selected vertex



Overflowing vertex



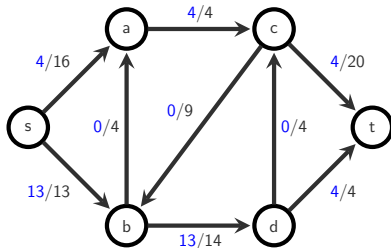
Edge selected for push



Edge available for push

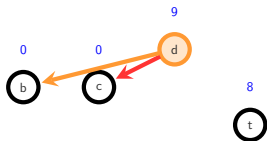
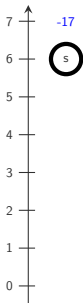
Current operation:

G, c and f

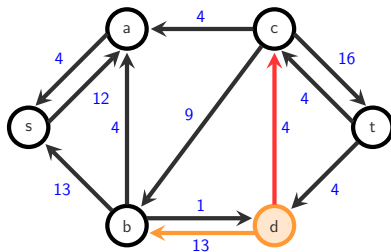


$h(v)$ and $e(v)$

$h(v)$



G_f and c_f



Currently selected vertex



Overflowing vertex



Edge selected for push

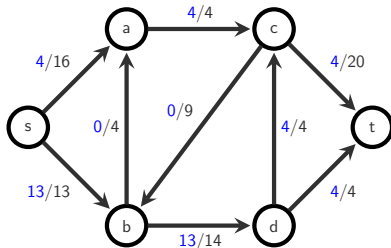


Edge available for push

Current operation:

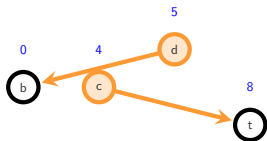
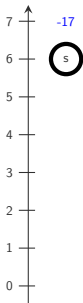
Push from d to c

G, c and f

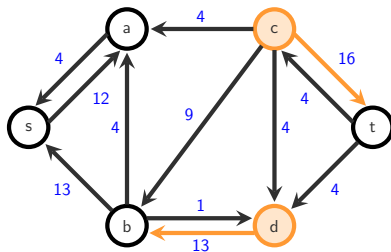


$h(v)$ and $e(v)$

$h(v)$



G_f and c_f



Currently selected vertex



Overflowing vertex



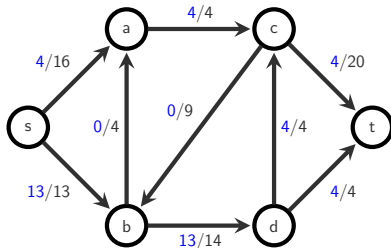
Edge selected for push



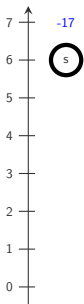
Edge available for push

Current operation:

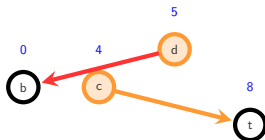
G, c and f



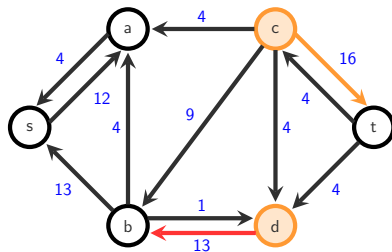
$h(v)$



$h(v)$ and $e(v)$



G_f and c_f



Currently selected vertex



Overflowing vertex



Edge selected for push

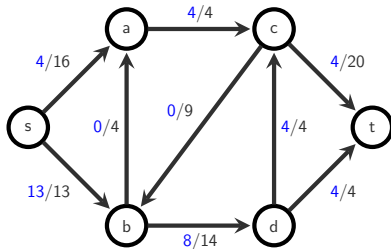


Edge available for push

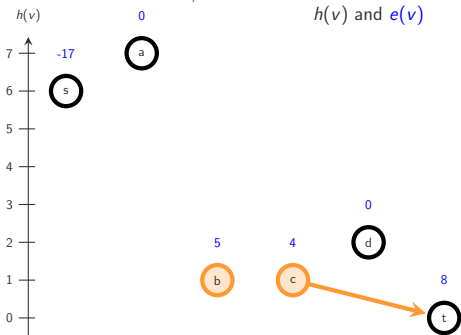
Current operation:

Push from d to b

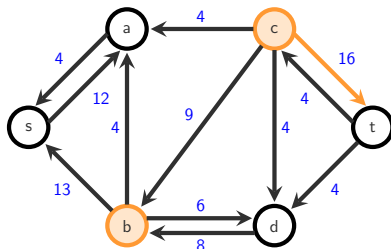
G , c and f



$h(v)$ and $e(v)$



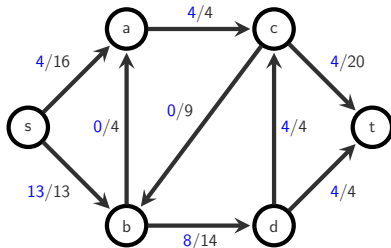
G_f and c_f



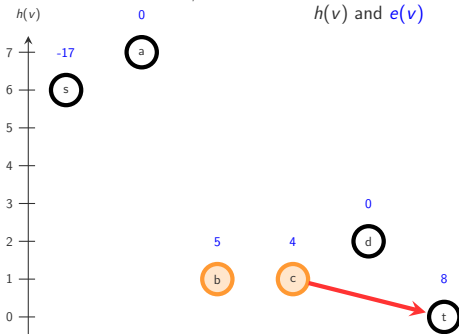
- v Currently selected vertex
- v Overflowing vertex
- Edge selected for push
- Edge available for push

Current operation:

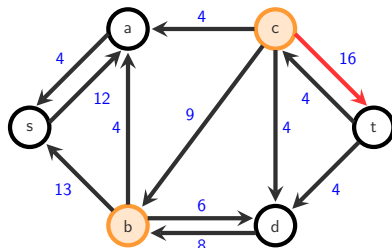
G, c and f



$h(v)$ and $e(v)$



G_f and c_f



- v Currently selected vertex
- v Overflowing vertex
- Edge selected for push
- Edge available for push

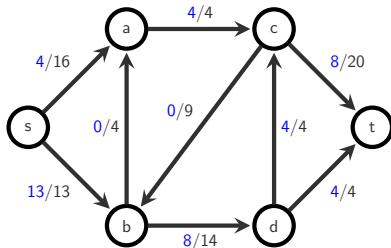
Current operation:

Push from c to t

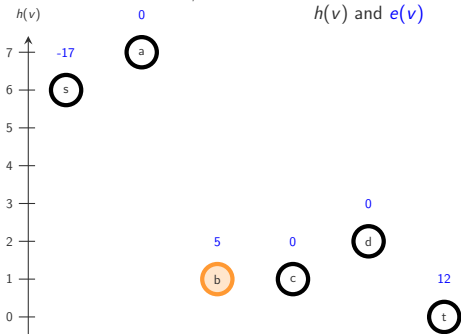
└ Maximum Flow

└ Push-relabel algorithm

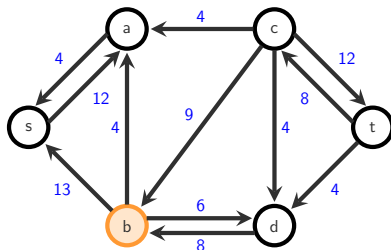
G , c and f



$h(v)$ and $e(v)$



G_f and c_f



Currently selected vertex



Overflowing vertex



Edge selected for push



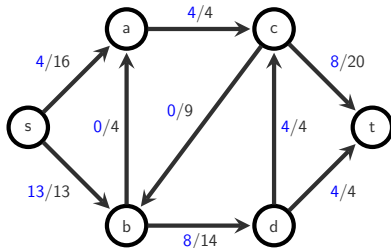
Edge available for push

Current operation:

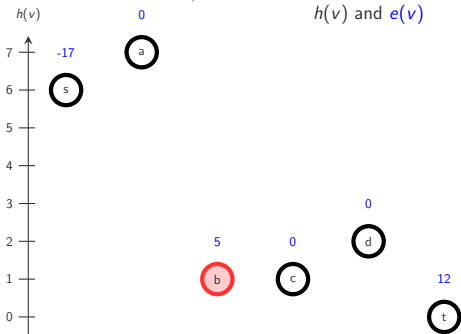
└ Maximum Flow

└ Push-relabel algorithm

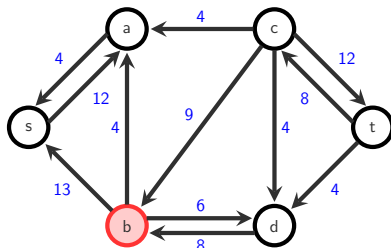
G, c and f



$h(v)$ and $e(v)$



G_f and c_f



Currently selected vertex



Overflowing vertex



Edge selected for push



Edge available for push

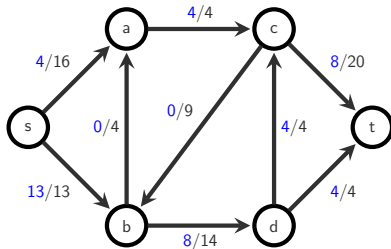
Current operation:

Relabel b

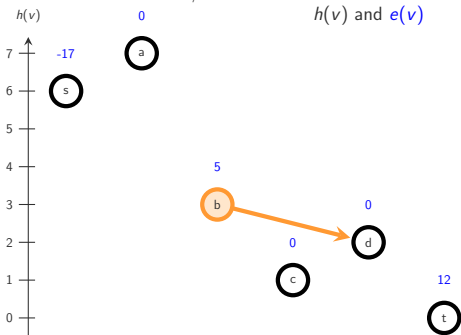
└ Maximum Flow

└ Push-relabel algorithm

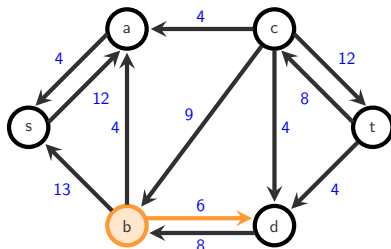
G, c and f



$h(v)$ and $e(v)$



G_f and c_f



Currently selected vertex



Overflowing vertex



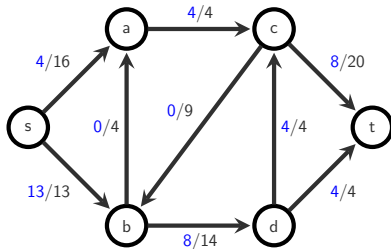
Edge selected for push



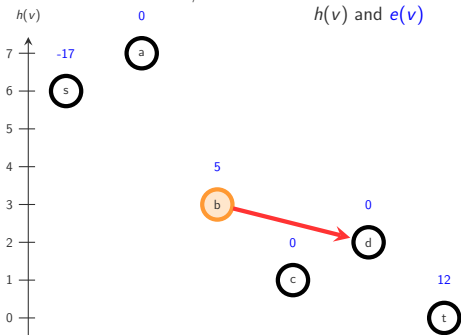
Edge available for push

Current operation:

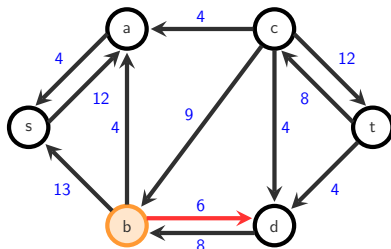
G, c and f



$h(v)$ and $e(v)$



G_f and c_f

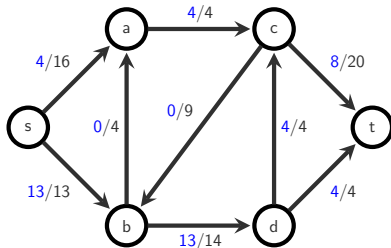


- v Currently selected vertex
- v Overflowing vertex
- Edge selected for push
- Edge available for push

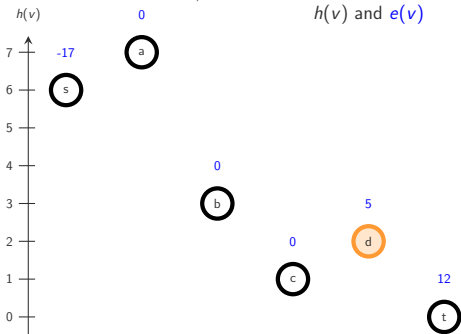
Current operation:

Push from b to d

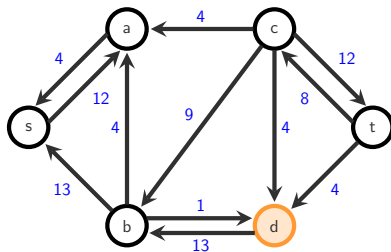
G, c and f



$h(v)$ and $e(v)$



G_f and c_f



Currently selected vertex



Overflowing vertex



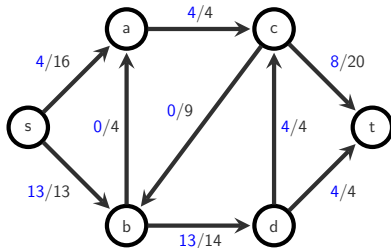
Edge selected for push



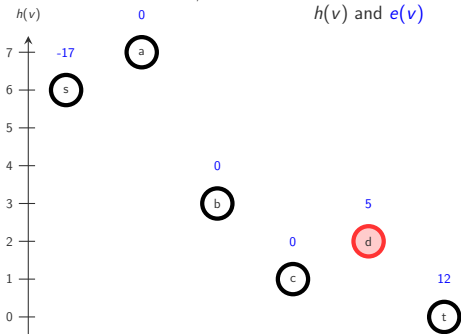
Edge available for push

Current operation:

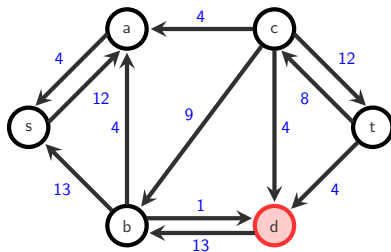
G, c and f



$h(v)$ and $e(v)$



G_f and c_f



Currently selected vertex



Overflowing vertex



Edge selected for push

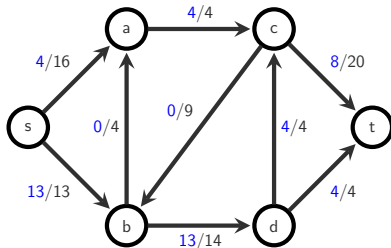


Edge available for push

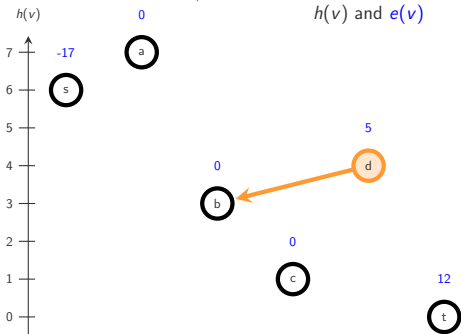
Current operation:

Relabel d

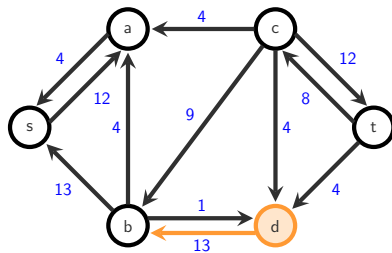
G , c and f



$h(v)$ and $e(v)$



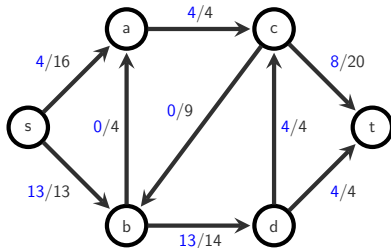
G_f and c_f



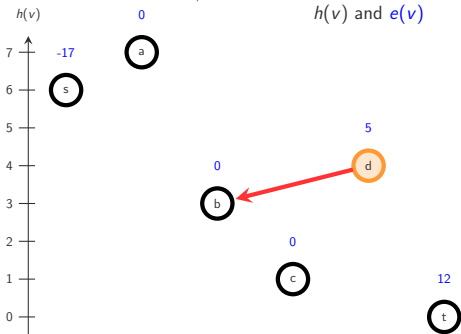
- v Currently selected vertex
- v Overflowing vertex
- Edge selected for push
- Edge available for push

Current operation:

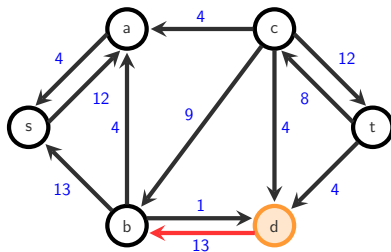
G, c and f



$h(v)$ and $e(v)$



G_f and c_f



Currently selected vertex



Overflowing vertex



Edge selected for push

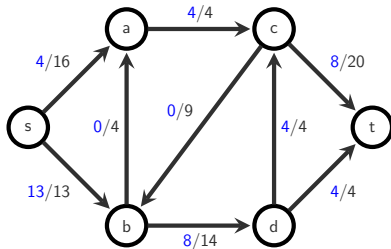


Edge available for push

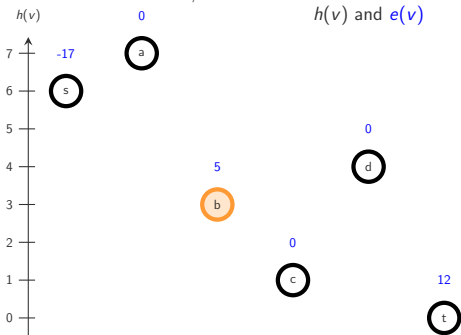
Current operation:

Push from d to b

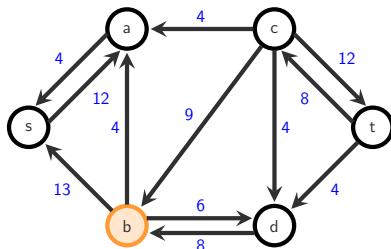
G, c and f



$h(v)$ and $e(v)$



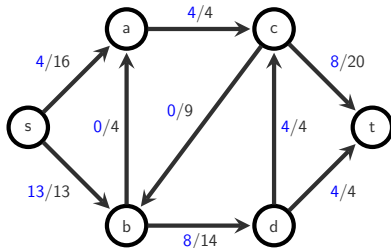
G_f and c_f



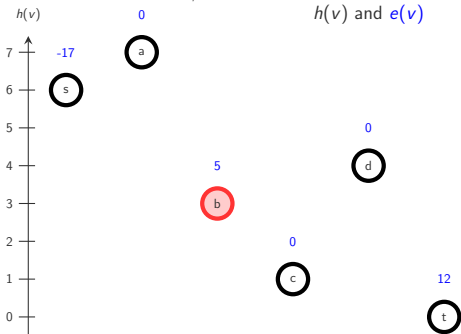
- v Currently selected vertex
- v Overflowing vertex
- Edge selected for push
- Edge available for push

Current operation:

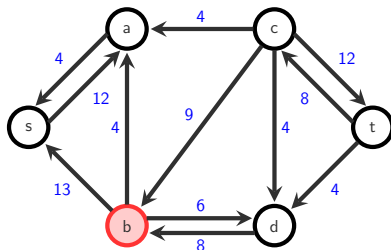
G, c and f



$h(v)$ and $e(v)$



G_f and c_f



Currently selected vertex



Overflowing vertex



Edge selected for push

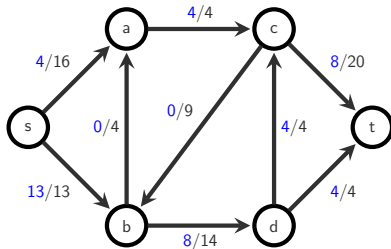


Edge available for push

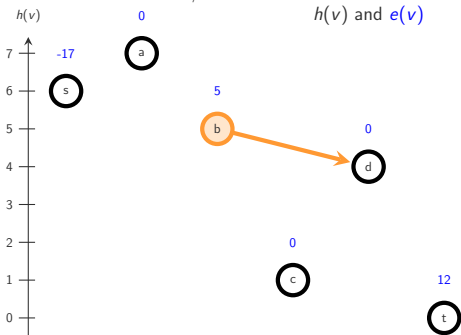
Current operation:

Relabel b

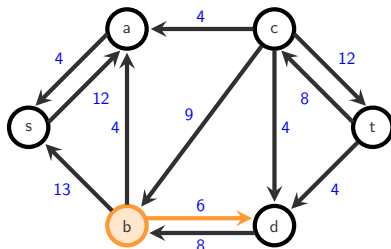
G, c and f



$h(v)$ and $e(v)$



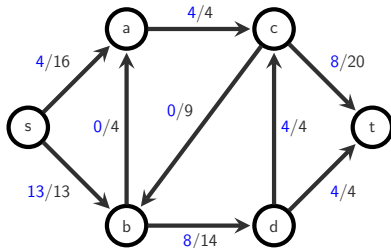
G_f and c_f



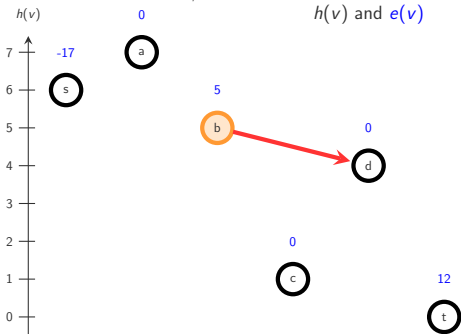
- v Currently selected vertex
- v Overflowing vertex
- Edge selected for push
- Edge available for push

Current operation:

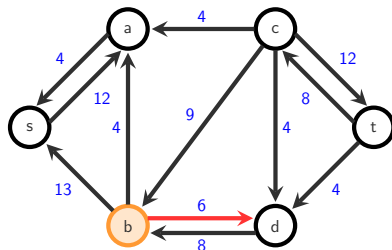
G, c and f



$h(v)$ and $e(v)$



G_f and c_f

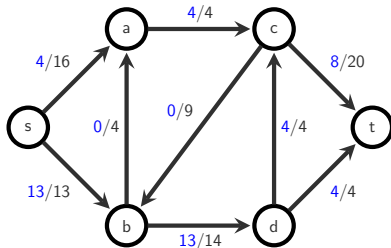


- v Currently selected vertex
- v Overflowing vertex
- Edge selected for push
- Edge available for push

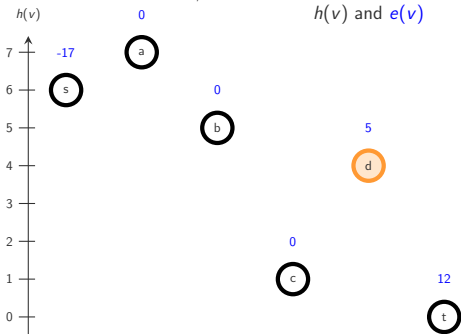
Current operation:

Push from b to d

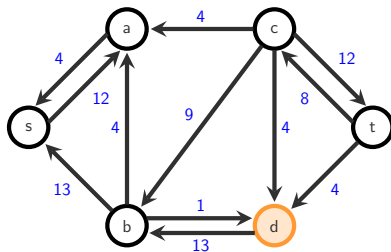
G, c and f



$h(v)$ and $e(v)$



G_f and c_f



Currently selected vertex



Overflowing vertex



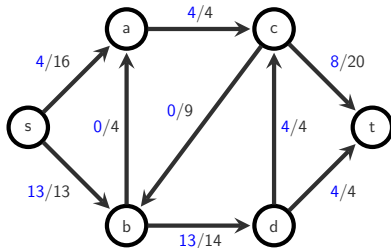
Edge selected for push



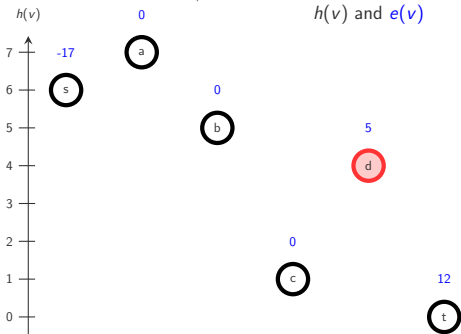
Edge available for push

Current operation:

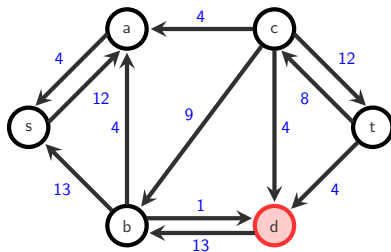
G , c and f



$h(v)$ and $e(v)$



G_f and c_f



Currently selected vertex



Overflowing vertex



Edge selected for push

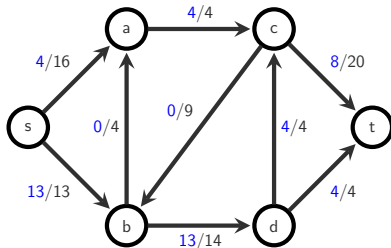


Edge available for push

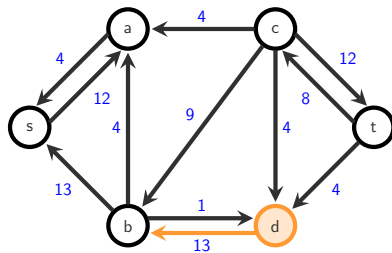
Current operation:

Relabel d

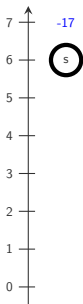
G , c and f



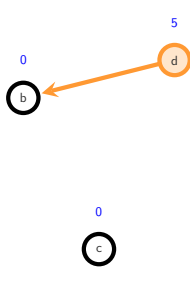
G_f and c_f



$h(v)$



$h(v)$ and $e(v)$



Currently selected vertex



Overflowing vertex



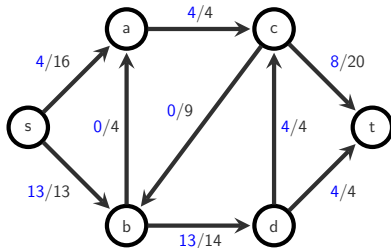
Edge selected for push



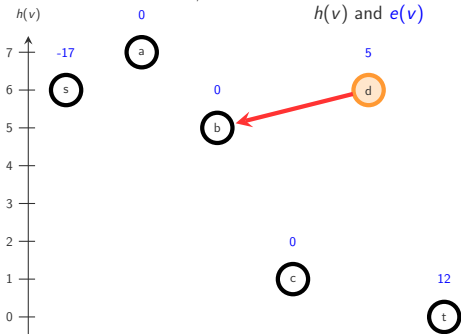
Edge available for push

Current operation:

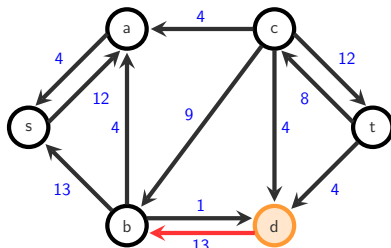
G, c and f



$h(v)$ and $e(v)$



G_f and c_f

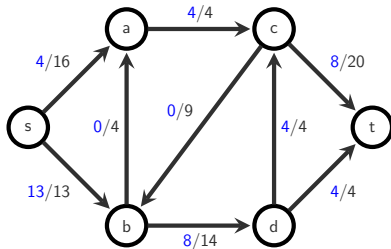


- v Currently selected vertex
- v Overflowing vertex
- Edge selected for push
- Edge available for push

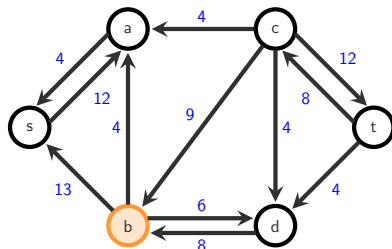
Current operation:

Push from d to b

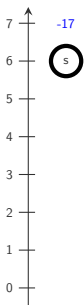
G, c and f



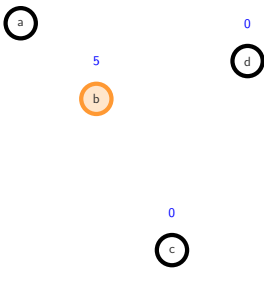
G_f and c_f



$h(v)$



$h(v)$ and $e(v)$



Currently selected vertex



Overflowing vertex



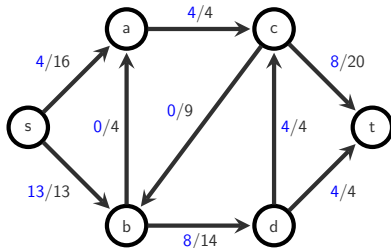
Edge selected for push



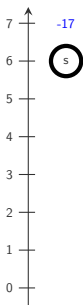
Edge available for push

Current operation:

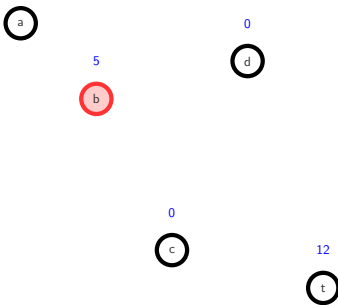
G, c and f



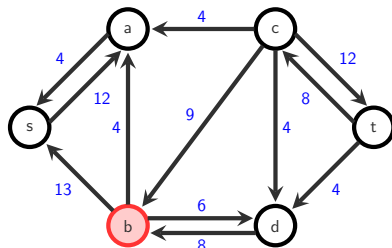
$h(v)$



$h(v)$ and $e(v)$



G_f and c_f



Currently selected vertex



Overflowing vertex



Edge selected for push

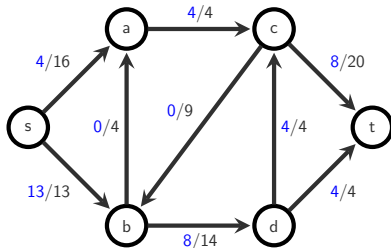


Edge available for push

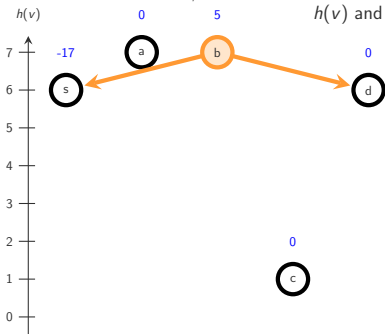
Current operation:

Relabel b

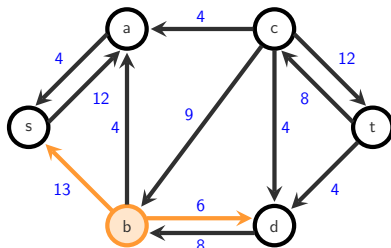
G , c and f



$h(v)$ and $e(v)$



G_f and c_f



Currently selected vertex



Overflowing vertex



Edge selected for push



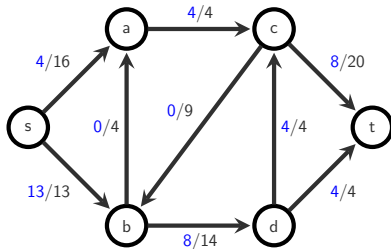
Edge available for push

Current operation:

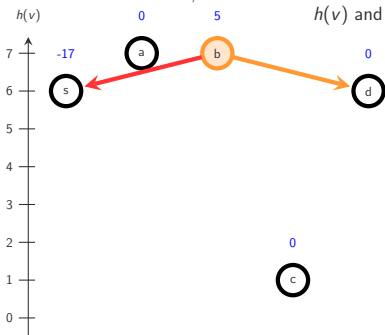
Maximum Flow

Push-relabel algorithm

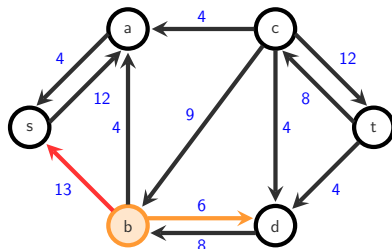
G , c and f



$h(v)$ and $e(v)$



G_f and c_f



- v Currently selected vertex
- v Overflowing vertex
- Edge selected for push
- Edge available for push

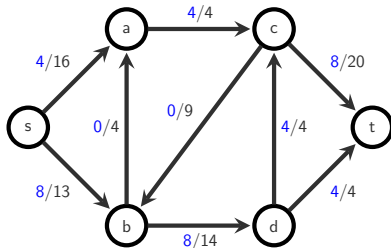
Current operation:

Push from b to s

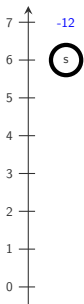
Maximum Flow

Push-relabel algorithm

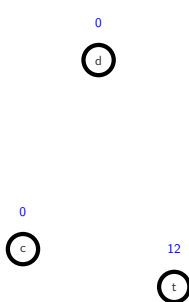
G, c and f



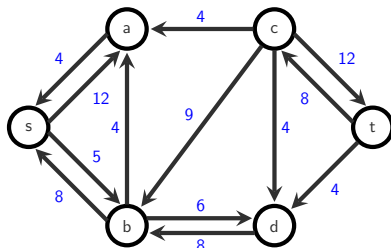
$h(v)$



$h(v)$ and $e(v)$



G_f and c_f



Currently selected vertex



Overflowing vertex



Edge selected for push



Edge available for push

No overflowing vertex:

Maximum flow found

Correctness

Why it even works?

$h(\cdot)$ stays height function

Path of k residual edges reduces h by at most k

Acyclic path has $\leq |V| - 1$ edges

$h(s) - h(t) = |V|$, therefore cut exists

Analysis

Order for choosing next operation?

- Any order (e.g. with stack): Goldberg-Tarjan algorithm, $\mathcal{O}(|V|^2 |E|)$.
- FIFO (with a queue): $\mathcal{O}(|V|^3)$.
- Highest label (with buckets): $\mathcal{O}(|V|^2 \sqrt{|E|})$.

Keep list of overflowing vertices in appropriate data structure and update accordingly after each operation!

Heuristics for the push-relabel algorithm

Two-phase algorithm

- In first phase, only push/relabel vertices with $h(v) < |V|$.
- Does not compute complete flow, but value of maximum flow at t .
- Remaining excess flow may be pushed back to s in second phase.

Initial labeling heuristic

- Compute initial heights as minimal distance to t by backwards BFS, computing $h(v) \leftarrow d_G(v, t)$.
- Avoids unnecessary initial relabeling operations.
- Can also compute labeling for second phase with $h(v) \leftarrow d_{G_f}(v, s) + |V|$.

Heuristics for the push-relabel algorithm

Gap heuristic

- After each relabeling, check if there is a height k with $0 < k < |V|$ such that there is no vertex v with $h(v) = k$ (keep a count array).
- If yes, all vertices u with $k < h(u) < |V|$ are disconnected from t in G_f and can be disregarded (set $h(u) \leftarrow |V|$).
- One of the most efficient heuristics, crucial for improving the performance.

Further reading

Several improved algorithms available:

- Orlin: Max flows in $\mathcal{O}(nm)$ time, or better, 2013: $\mathcal{O}(|V| |E|)$
- Sidford and Lee: Path-Finding Methods for Linear Programming, 2014: $\tilde{\mathcal{O}}(|E| \sqrt{|V|} \log^{\mathcal{O}(1)} U)$ (where $\tilde{\mathcal{O}}$ hides $\text{polylog}(|V|, |E|)$).

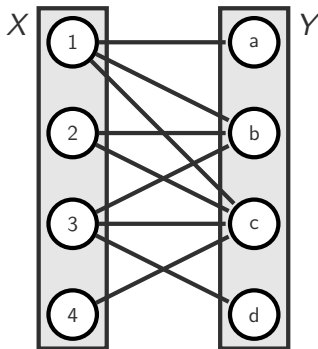
Additional literature on flow problems and algorithms:

- T. H. Cormen et al.: Introduction to Algorithms. MIT press, 2009.
- R. Ahuja, T. Magnanti and J. B. Orlin: Network Flows: Theory, Algorithms and Applications. Prentice Hall, 1993.
- B. Korte, J. Vygen: Combinatorial Optimization: Theory and Algorithms. Springer, 2012.

Application: Bipartite Matching

Definition (Bipartite Matching / Maximum Matching Problem)

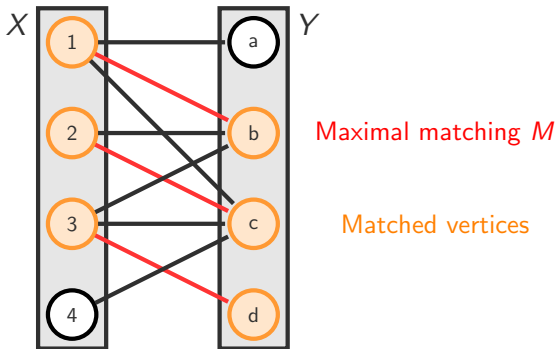
Given two disjoint sets of vertices X and Y and a set of edges $E \subseteq X \times Y$, a *matching* $M \subseteq E$ is a subset of edges such that each node of $X \cup Y$ appears in at most one edge of M . The *maximum matching problem* is finding a matching M such that $|M|$ is maximal.



Application: Bipartite Matching

Definition (Bipartite Matching / Maximum Matching Problem)

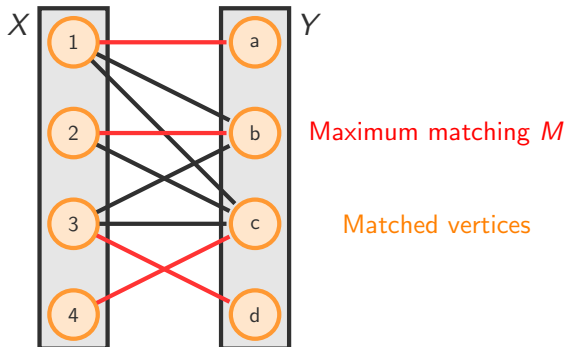
Given two disjoint sets of vertices X and Y and a set of edges $E \subseteq X \times Y$, a *matching* $M \subseteq E$ is a subset of edges such that each node of $X \cup Y$ appears in at most one edge of M . The *maximum matching problem* is finding a matching M such that $|M|$ is maximal.



Application: Bipartite Matching

Definition (Bipartite Matching / Maximum Matching Problem)

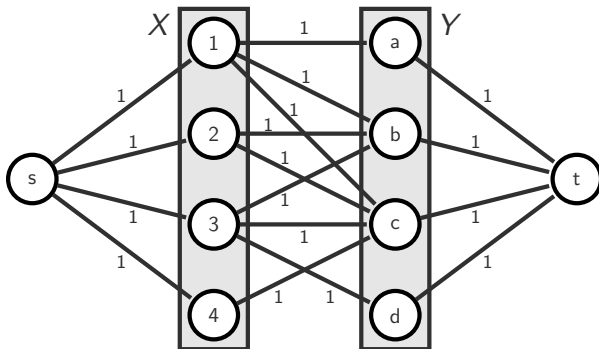
Given two disjoint sets of vertices X and Y and a set of edges $E \subseteq X \times Y$, a *matching* $M \subseteq E$ is a subset of edges such that each node of $X \cup Y$ appears in at most one edge of M . The *maximum matching problem* is finding a matching M such that $|M|$ is maximal.



Application: Bipartite Matching

Matching problem as a flow problem

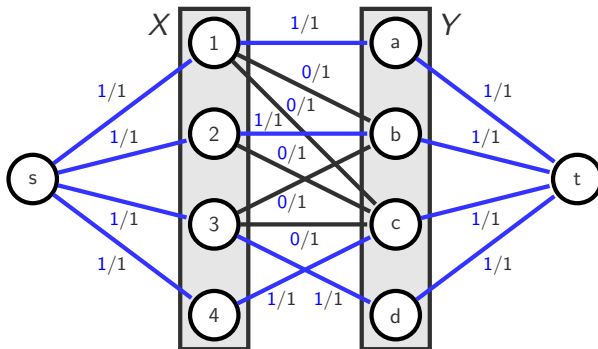
Given a bipartite matching problem, construct flow network $G = (V, E')$ with $V = \{s, t\} \cup X \cup Y$ and $E' = E \cup (\{s\} \times X) \cup (Y \times \{t\})$ and $c(e) = 1$ for all $e \in E'$. Then the value of the maximum flow is equal to the size of the maximum matching.



Application: Bipartite Matching

Matching problem as a flow problem

Given a bipartite matching problem, construct flow network $G = (V, E')$ with $V = \{s, t\} \cup X \cup Y$ and $E' = E \cup (\{s\} \times X) \cup (Y \times \{t\})$ and $c(e) = 1$ for all $e \in E'$. Then the value of the maximum flow is equal to the size of the maximum matching.



Notes on performance

General

- Better asymptotic complexity in general means nothing — check the constraints
- Constants matter

JIT (non-C++) languages

- We have Python (PyPy3) solutions to everything
- For JIT, many small functions are often easier to optimise than a single large one
- Data structure allocation also has a cost
- Recalculating can be cheaper than caching