

Maximum Flow

A flow network is a directed graph where:

Each edge (u, v) has a capacity $c(u, v) \geq 0$.

If (u, v) is not an edge, then $c(u, v) = 0$.

There is a source vertex s and a sink vertex t .

A flow $f(u, v)$ satisfies the following constraints:

for all (u, v) , $f(u, v) \leq c(u, v)$

for all (u, v) , $f(u, v) = -f(v, u)$

for all u except s and t , $\sum_{v \in V} f(u, v) = 0$

We want to maximize the flow F from s to t :

$$F = \sum_{v \in V} f(s, v) = \sum_{v \in V} f(v, t)$$

Algorithms for maximum flow can be used for
 liquids flowing through pipes,
 parts through assembly lines,
 current through electrical networks,
 information through communication networks,
 the maximum matching in a bipartite graph,
 and the minimum-size cut of a graph.

Ford-Fulkerson Method

Let $f(\cdot, \cdot)$ be a flow. Define residual capacity c_f

$$c_f(u, v) = c(u, v) - f(u, v)$$

Define residual edges E_f to be the edges with positive residual capacity.

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

FORD-FULKERSON(G, s, t, c)

for each edge (u, v) in G

do $f[u, v] \leftarrow 0$

$f[v, u] \leftarrow 0$

while ($P \leftarrow$ a path from s to t in E_f)

do $x \leftarrow \min\{c_f(u, v) : (u, v) \in P\}$

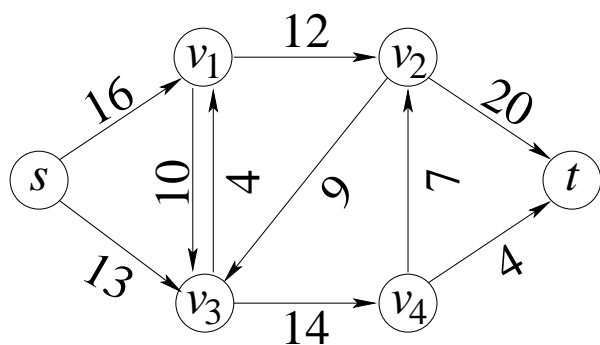
for each edge (u, v) in P

do $f[u, v] \leftarrow f[u, v] + x$

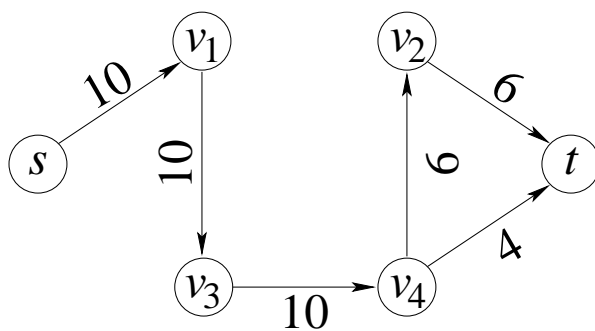
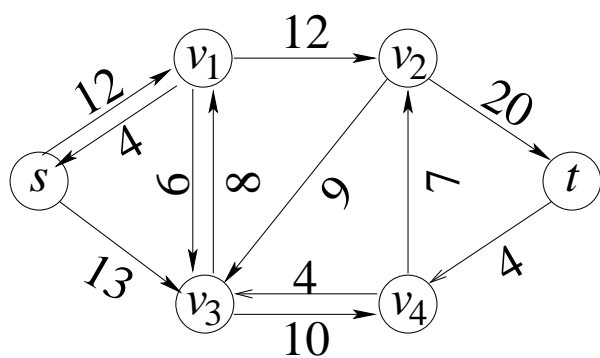
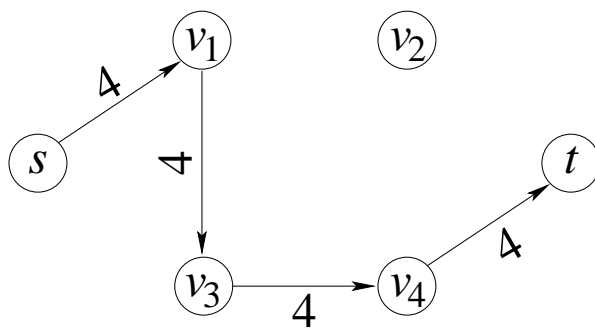
$f[v, u] \leftarrow -f[u, v]$

A maximum flow is found because every path from s to t is at full capacity.

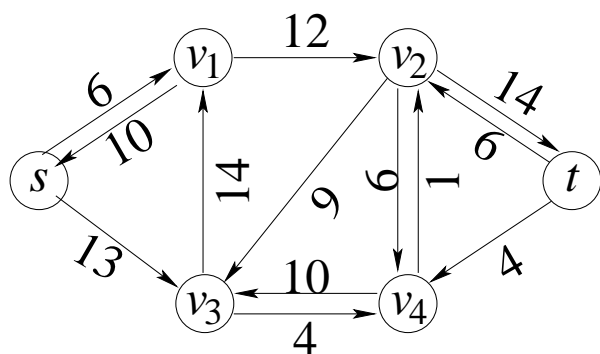
Residual Networks



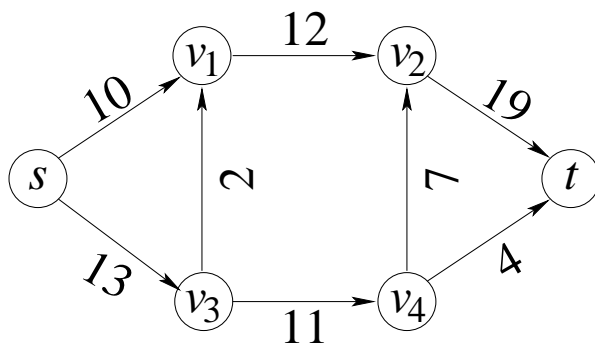
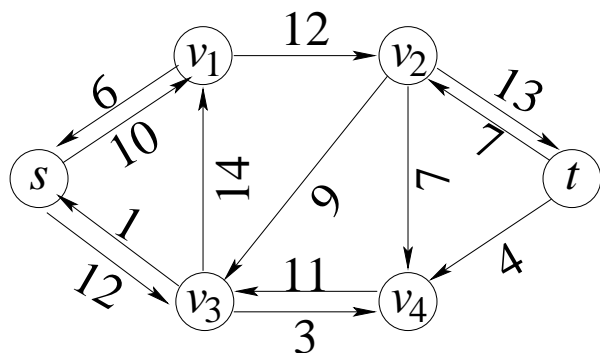
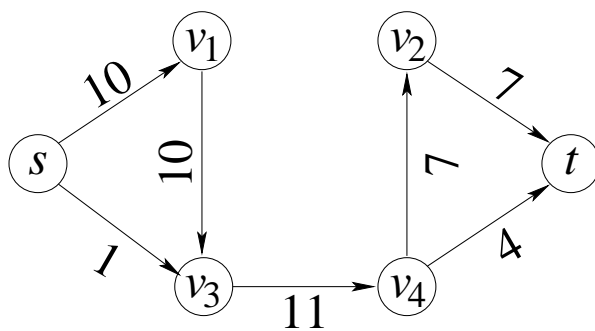
Flow Values



Residual Networks



Flow Values



Edmonds-Karp Algorithm

This algorithm uses the shortest augmenting path (found by BFS). Running time is $O(VE^2)$.

Lemma: Residual path lengths do not decrease.

Suppose the distance from s to v decreases.

Let v be the closest vertex where this happens.

Some vertex u becomes adjacent to v

(otherwise v could not be closer).

This implies (v, u) is on augmenting path,

but this means u is farther from s than v .

A contradiction, so distances do not decrease.

Lemma: Each edge is critical $O(V)$ times.

A $c_f(u, v)$ is minimum on each augmenting path.

This implies (u, v) removed from residual edges.

For (u, v) to reappear as a residual edge,

(v, u) must be on augmenting path.

Because distance to v cannot decrease,

distance to u must have increased.

This implies each edge can be minimal $O(V)$ times.

Push-Relabel Algorithms

The *excess flow* into vertex u is defined as

$$e(u) = \sum_{v \in V} f(v, u)$$

u is overflowing if $e(u) > 0$.

Flow goes from “higher” to “lower” vertices.

h is a *height function* if $h(s) = |V|$, $h(t) = 0$, and

$$(u, v) \in E_f \text{ implies } h(u) \leq h(v) + 1$$

$$h(u) > h(v) + 1 \text{ implies } (u, v) \notin E_f$$

INITIALIZE-PREFLOW(G, s)

```

for each vertex  $u$  in  $G$ 
    do  $h[u] \leftarrow e[u] \leftarrow 0$ 
for each edge  $(u, v)$  in  $G$ 
    do  $f[u, v] \leftarrow f[v, u] \leftarrow 0$ 
 $h[s] \leftarrow$  number of vertices
for each vertex  $u$  in  $Adj[s]$ 
    do  $f[s, u] \leftarrow c[s, u]$ 
         $f[u, s] \leftarrow -c[s, u]$ 
         $e[u] \leftarrow c[s, u]$ 
         $e[s] \leftarrow e[s] - c[s, u]$ 

```

PUSH(u, v)

▷ **Applies when:** $e(u) > 0$, $c_f(u, v) > 0$,
and $h[u] = h[v] + 1$

▷ **Action:** Increase flow from u to v

$d \leftarrow \min(e[u], c_f(u, v))$

$f[u, v] \leftarrow f[u, v] + d$

$f[v, u] \leftarrow -f[u, v]$

$e[u] \leftarrow e[u] - d$

$e[v] \leftarrow e[v] + d$

RELABEL(u)

▷ **Applies when:** $e(u) > 0$ and $h[u] \leq h[v]$
for all v such that $c_f(u, v) > 0$

▷ **Action:** Increase the height of u

$h[u] \leftarrow 1 + \min\{h[v] : (u, v) \in E_f\}$

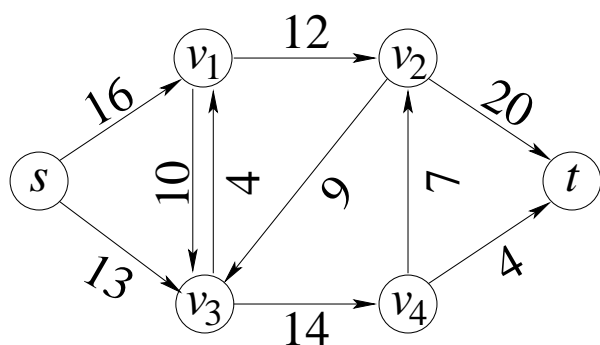
GENERIC-PUSH-RELABEL(G)

INITIALIZE-PREFLOW(G, s)

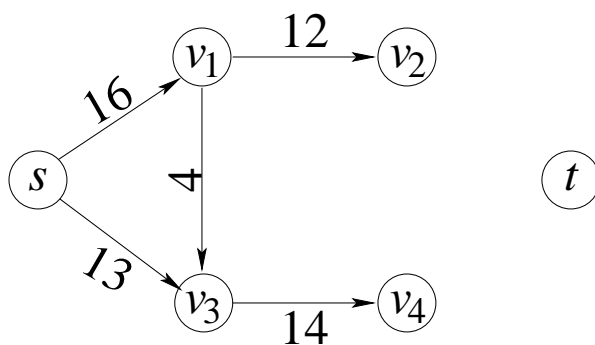
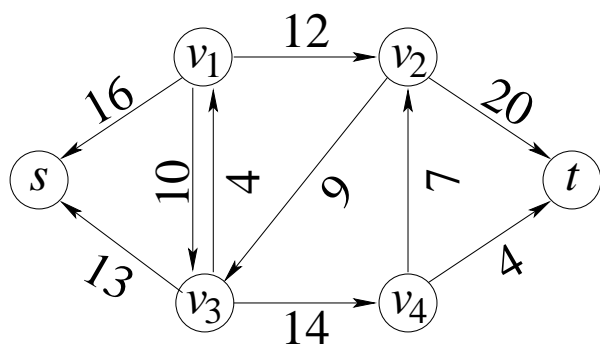
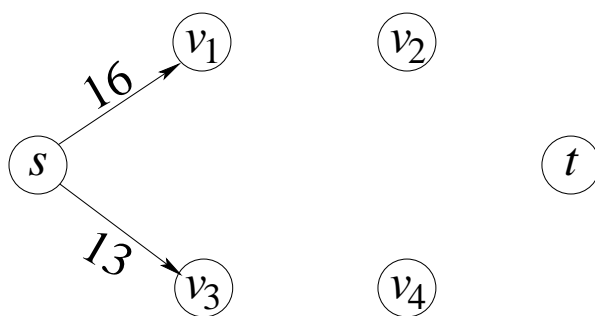
while there exists an applicable push or
relabel operation

do select an applicable push or relabel
operation and perform it

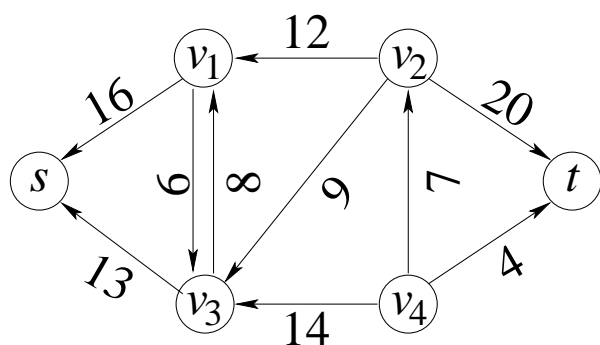
Residual Networks



Flow Values



Residual Networks



Flow Values

