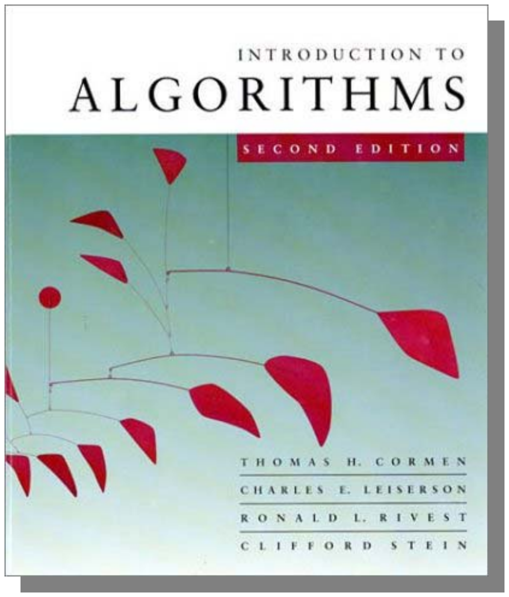
Introduction to Algorithms

6.046J/18.401J

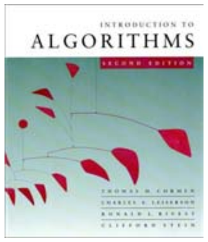


LECTURE 21

Network Flow II

- Max-flow, min-cut theorem
- Ford-Fulkerson algorithm and analysis
- Edmonds-Karp algorithm and analysis
- Best algorithms to date

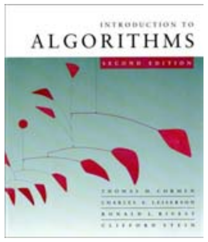
Prof. Charles E. Leiserson



Recall from Lecture 22

- ***Flow value:*** $|f| = f(s, V)$.
- ***Cut:*** Any partition (S, T) of V such that $s \in S$ and $t \in T$.
- **Lemma.** $|f| = f(S, T)$ for any cut (S, T).
- **Corollary.** $|f| \le c(S, T)$ for any cut (S, T).
- ***Residual graph:*** The graph $G_f = (V, E_f)$ with strictly positive ***residual capacities*** $c_f(u, v) = c(u, v) - f(u, v) > 0$.
- ***Augmenting path:*** Any path from s to t in G_f.
- ***Residual capacity*** of an augmenting path:

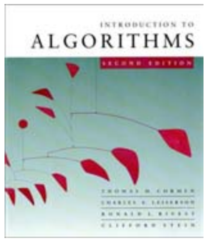
$$c_f(p) = \min_{(u,v)\in p} \{c_f(u,v)\}.$$



Max-flow, min-cut theorem

Theorem. The following are equivalent:

1. $|f| = c(S, T)$ for some cut (S, T).
2. f is a maximum flow.
3. f admits no augmenting paths.



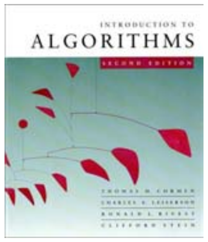
Max-flow, min-cut theorem

Theorem. The following are equivalent:

1. $|f| = c(S, T)$ for some cut (S, T).
2. f is a maximum flow.
3. f admits no augmenting paths.

Proof.

(*1*) ⇒ (*2*): Since $|f| \le c(S, T)$ for any cut (S, T) (by the corollary from Lecture 22), the assumption that $|f| = c(S, T)$ implies that f is a maximum flow.



Max-flow, min-cut theorem

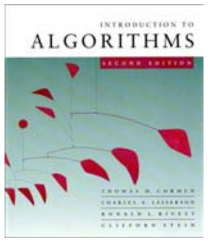
Theorem. The following are equivalent:

1. $|f| = c(S, T)$ for some cut (S, T).
2. f is a maximum flow.
3. f admits no augmenting paths.

Proof.

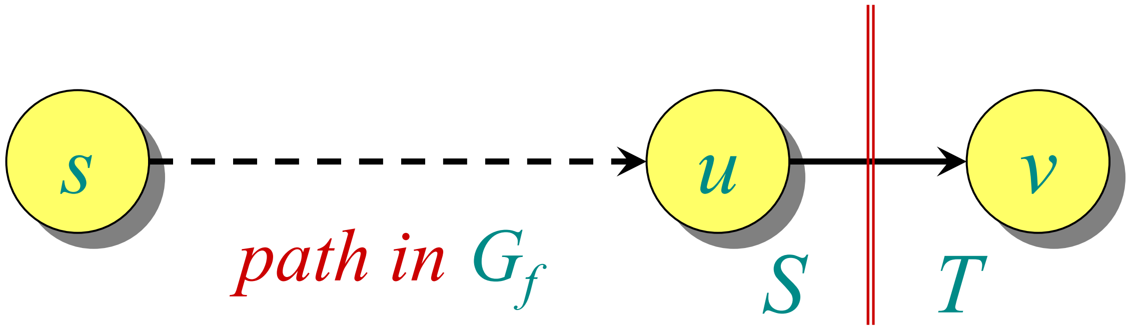
(*1*) $\Rightarrow$ (*2*): Since $|f| \le c(S, T)$ for any cut (S, T) (by the corollary from Lecture 22), the assumption that $|f| = c(S, T)$ implies that f is a maximum flow.

(*2*) $\Rightarrow$ (*3*): If there were an augmenting path, the flow value could be increased, contradicting the maximality of f.

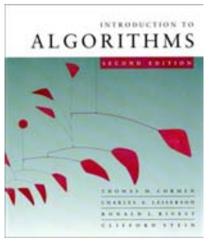


Proof (continued)

(*3*) ⇒ (*1*): Suppose that f admits no augmenting paths. Define $S = \{v \in V : \text{there exists a path in } G_f \text{ from } s \text{ to } v\}$, and let $T = V - S$. Observe that $s \in S$ and $t \in T$, and thus (S, T) is a cut. Consider any vertices $u \in S$ and $v \in T$.



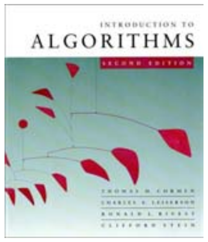
We must have $c_f(u, v) = 0$, since if $c_f(u, v) > 0$, then $v \in S$, not $v \in T$ as assumed. Thus, $f(u, v) = c(u, v)$, since $c_f(u, v) = c(u, v) - f(u, v)$. Summing over all $u \in S$ and $v \in T$ yields $f(S, T) = c(S, T)$, and since $|f| = f(S, T)$, the theorem follows. ◻



Ford-Fulkerson max-flow algorithm

Algorithm:

```
$f[u, v] \leftarrow 0$ for all $u, v \in V$
**while** an augmenting path $p$ in $G$ wrt $f$ exists
    **do** augment $f$ by $c_f(p)$
```

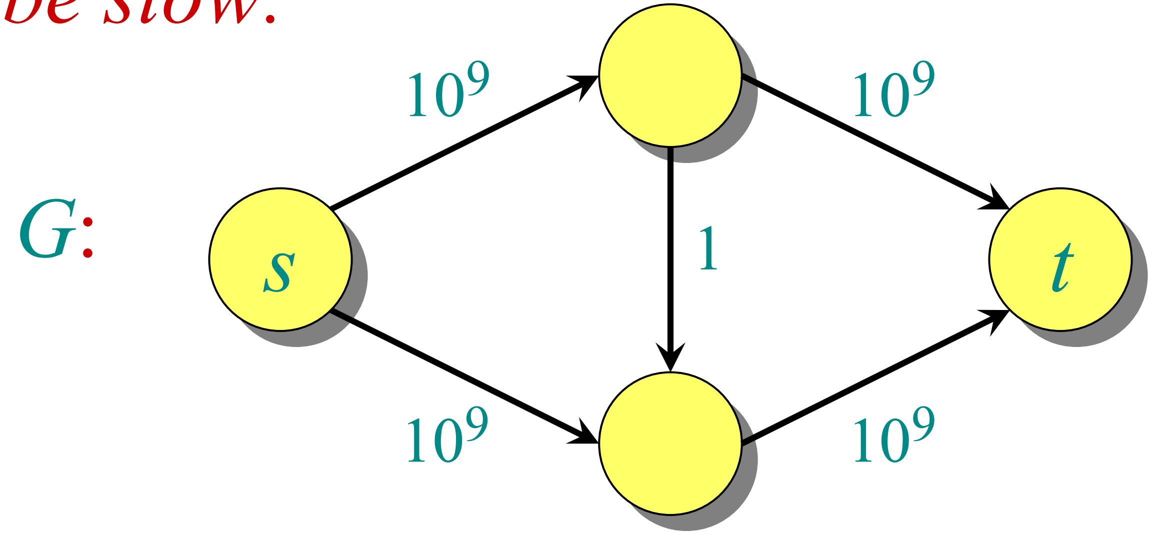


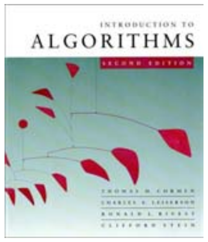
Ford-Fulkerson max-flow algorithm

Algorithm:

```
$f[u, v] \leftarrow 0$ for all $u, v \in V$
while an augmenting path $p$ in $G$ wrt $f$ exists
    do augment $f$ by $c_f(p)$
```

Can be slow:





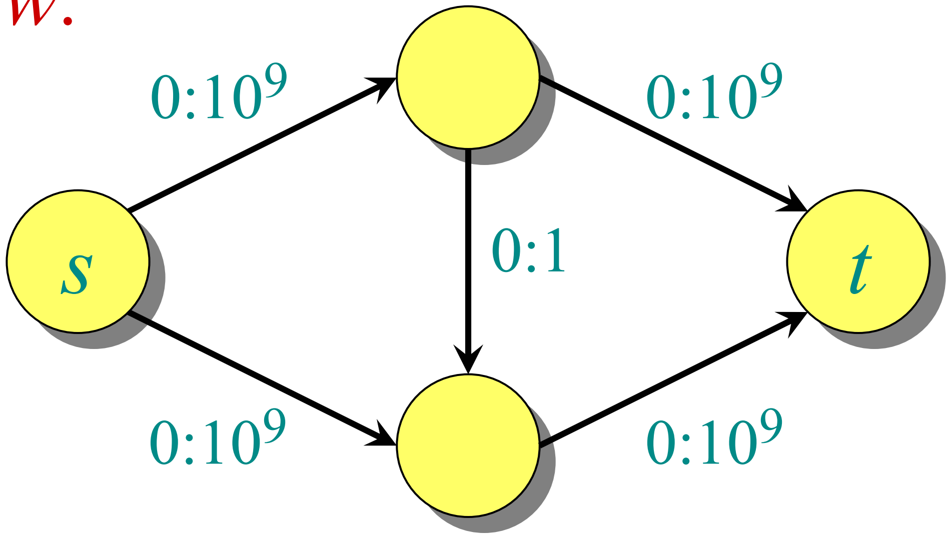
Ford-Fulkerson max-flow algorithm

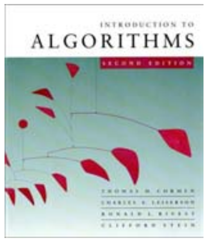
Algorithm:

```
$f[u, v] \leftarrow 0$ for all $u, v \in V$
while an augmenting path $p$ in $G$ wrt $f$ exists
    do augment $f$ by $c_f(p)$
```

Can be slow:

G:



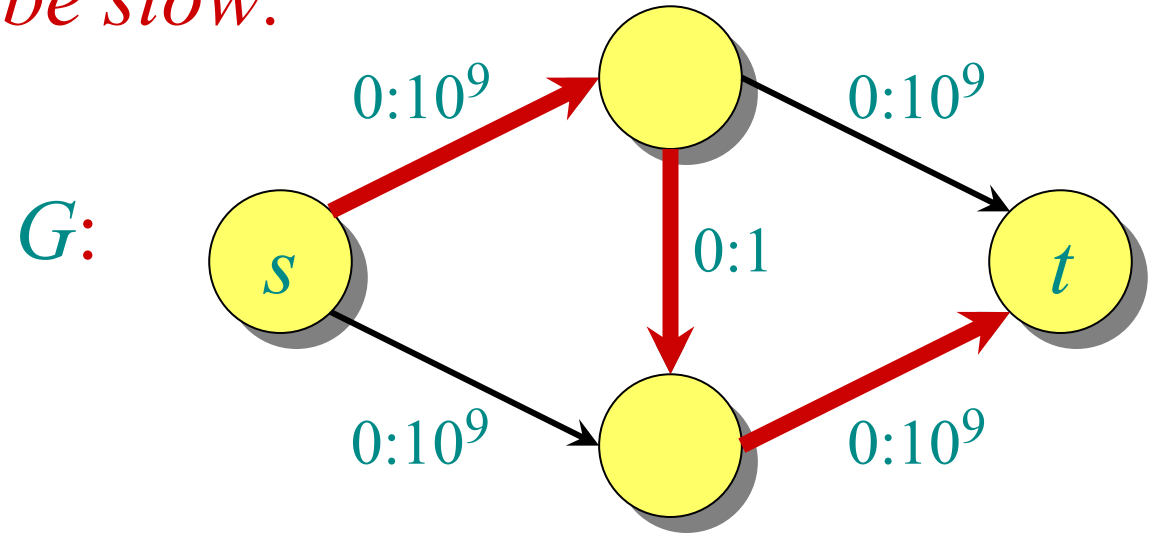


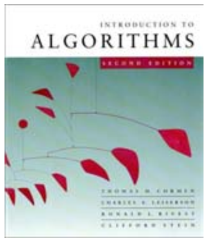
Ford-Fulkerson max-flow algorithm

Algorithm:

```
$f[u, v] \leftarrow 0$ for all $u, v \in V$
**while** an augmenting path $p$ in $G$ wrt $f$ exists
    **do** augment $f$ by $c_f(p)$
```

Can be slow:





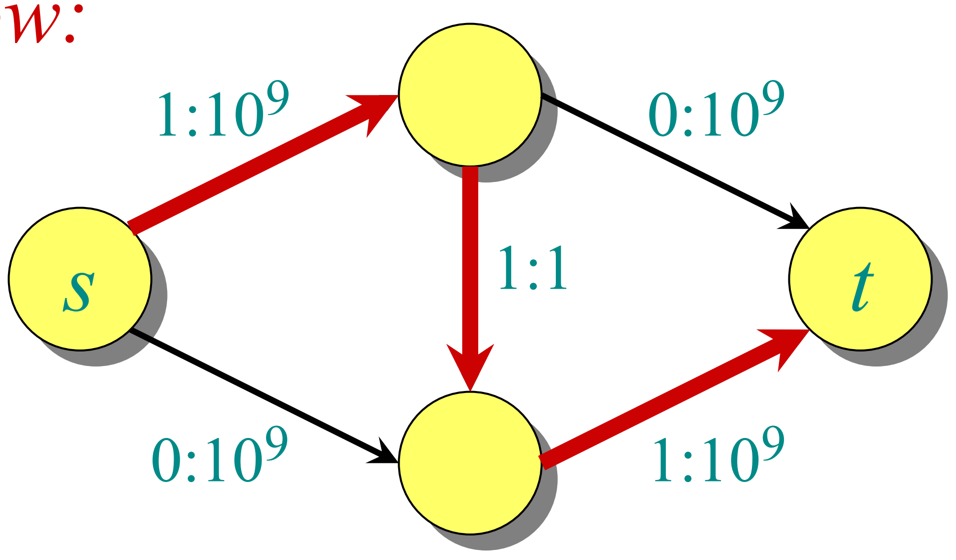
Ford-Fulkerson max-flow algorithm

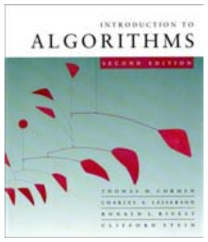
Algorithm:

```
$f[u, v] \leftarrow 0$ for all $u, v \in V$
while an augmenting path $p$ in $G$ wrt $f$ exists
    do augment $f$ by $c_f(p)$
```

Can be slow:

G:





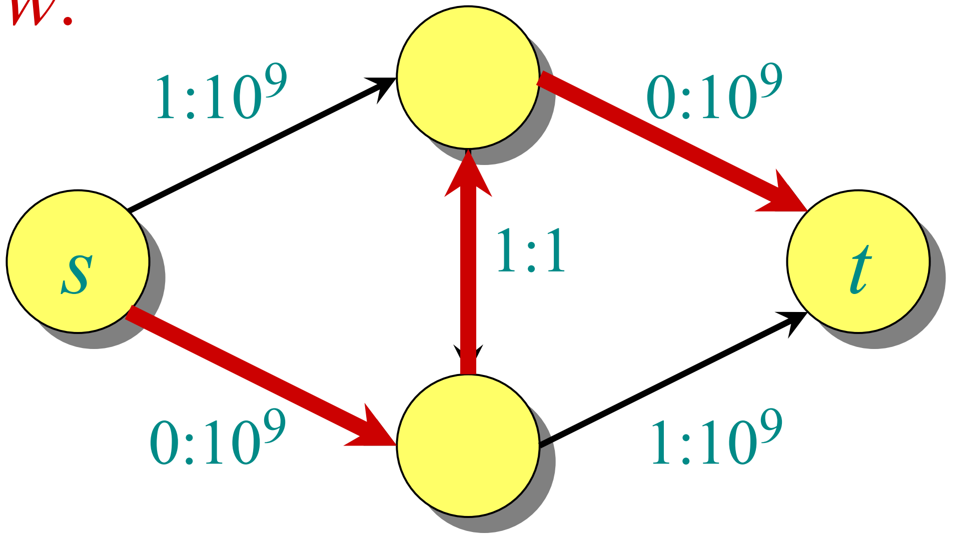
Ford-Fulkerson max-flow algorithm

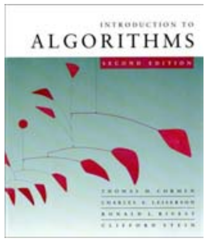
Algorithm:

```
$f[u, v] \leftarrow 0$ for all $u, v \in V$
while an augmenting path $p$ in $G$ wrt $f$ exists
    do augment $f$ by $c_f(p)$
```

Can be slow:

G:





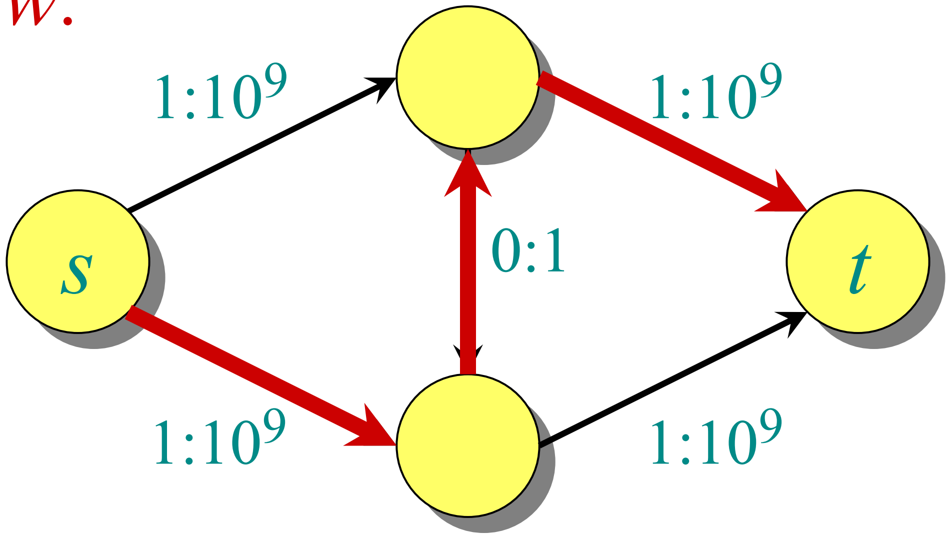
Ford-Fulkerson max-flow algorithm

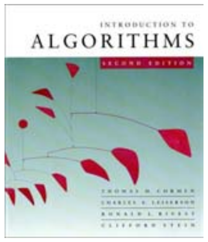
Algorithm:

```
$f[u, v] \leftarrow 0$ for all $u, v \in V$
while an augmenting path $p$ in $G$ wrt $f$ exists
    do augment $f$ by $c_f(p)$
```

Can be slow:

G:





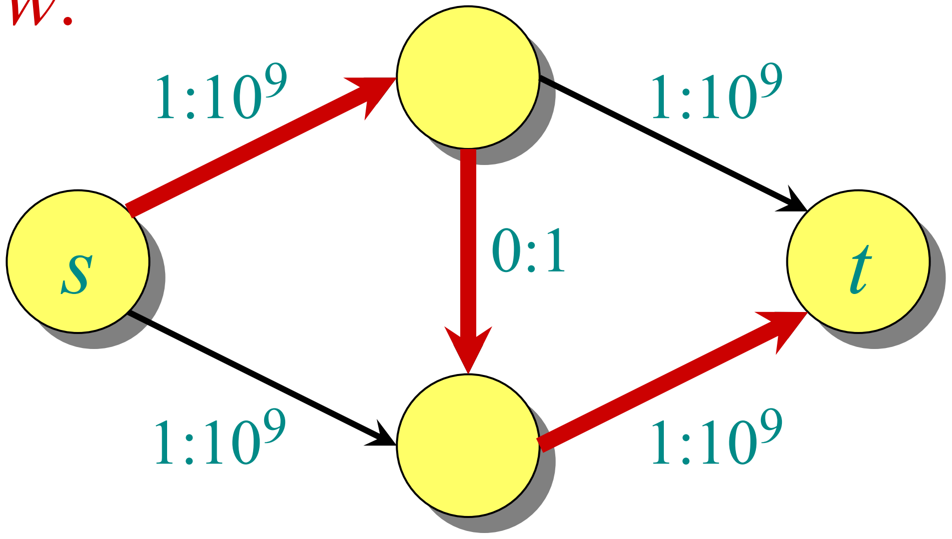
Ford-Fulkerson max-flow algorithm

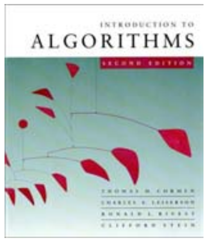
Algorithm:

```
$f[u, v] \leftarrow 0$ for all $u, v \in V$
while an augmenting path $p$ in $G$ wrt $f$ exists
    do augment $f$ by $c_f(p)$
```

Can be slow:

G:





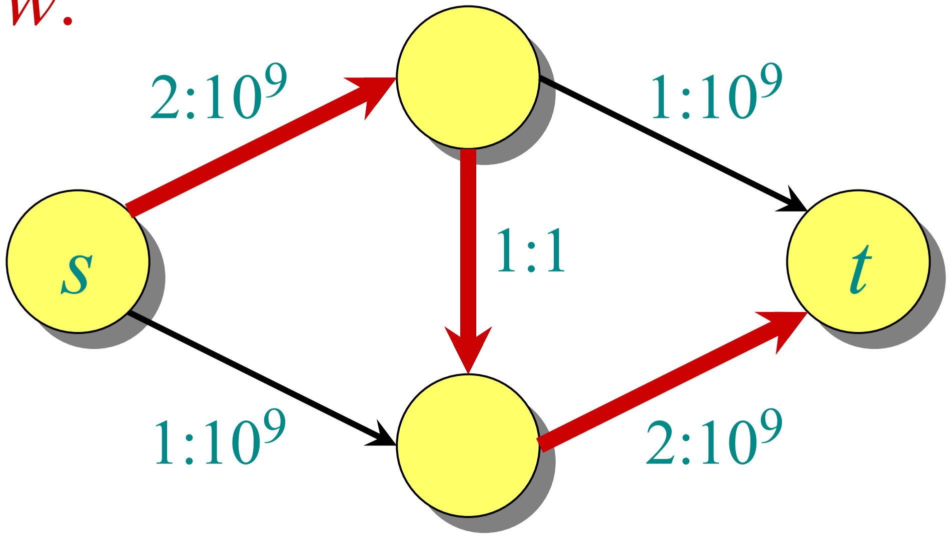
Ford-Fulkerson max-flow algorithm

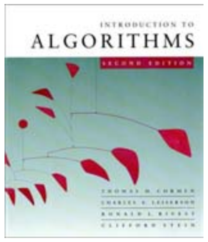
Algorithm:

```
$f[u, v] \leftarrow 0$ for all $u, v \in V$
**while** an augmenting path $p$ in $G$ wrt $f$ exists
    **do** augment $f$ by $c_f(p)$
```

Can be slow:

G:





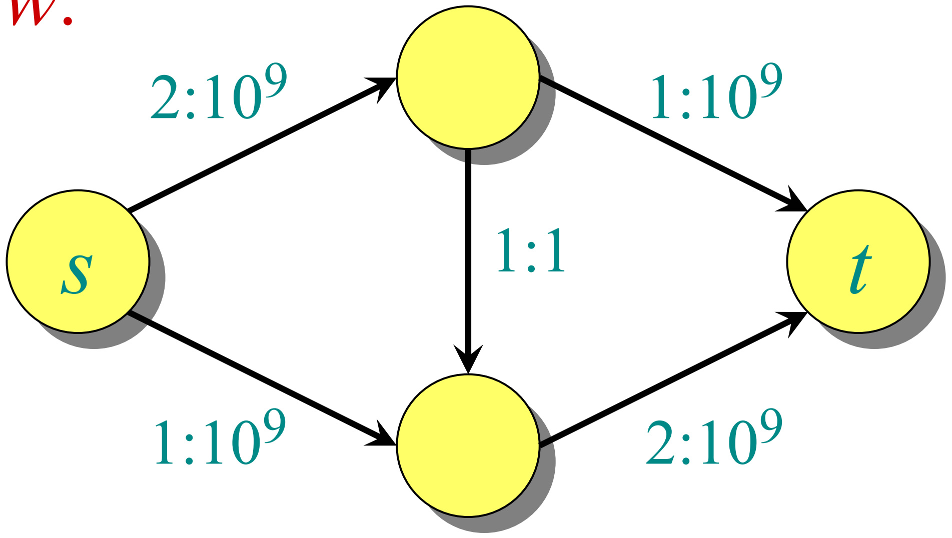
Ford-Fulkerson max-flow algorithm

Algorithm:

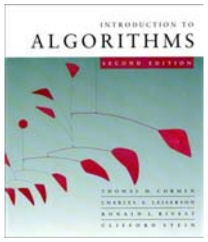
```
$f[u, v] \leftarrow 0$ for all $u, v \in V$
while an augmenting path $p$ in $G$ wrt $f$ exists
    do augment $f$ by $c_f(p)$
```

Can be slow:

G:



2 billion iterations on a graph with 4 vertices!

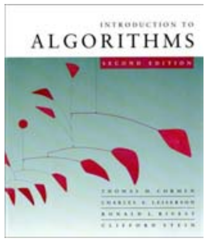


Edmonds-Karp algorithm

Edmonds and Karp noticed that many people's implementations of Ford-Fulkerson augment along a ***breadth-first augmenting path***: a shortest path in G_f from s to t where each edge has weight 1. These implementations would always run relatively fast.

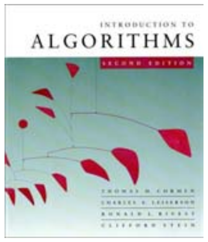
Since a breadth-first augmenting path can be found in $O(E)$ time, their analysis, which provided the first polynomial-time bound on maximum flow, focuses on bounding the number of flow augmentations.

(In independent work, Dinic also gave polynomial-time bounds.)



Monotonicity lemma

Lemma. Let $\delta(v) = \delta_f(s, v)$ be the breadth-first distance from s to v in G_f. During the Edmonds-Karp algorithm, $\delta(v)$ increases monotonically.

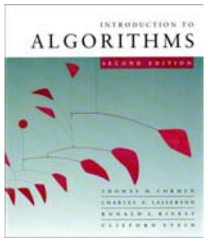


Monotonicity lemma

Lemma. Let $\delta(v) = \delta_f(s, v)$ be the breadth-first distance from s to v in G_f. During the Edmonds-Karp algorithm, $\delta(v)$ increases monotonically.

Proof. Suppose that augmenting a flow f on G produces a new flow f'. Let $\delta'(v) = \delta_{f'}(s, v)$. We'll show $\delta'(v) \geq \delta(v)$ by induction on $\delta'(v)$. For the base case, $\delta'(v) = 0$ implies $v = s$, and since $\delta(s) = 0$, we have $\delta'(v) \geq \delta(v)$.

For the inductive case, consider a breadth-first path $s \to \cdots \to u \to v$ in $G_{f'}$. We must have $\delta'(v) = \delta'(u) + 1$, since subpaths of shortest paths are shortest paths. Hence, we have $\delta'(u) \geq \delta(u)$ by induction, because $\delta'(v) > \delta'(u)$. Certainly, $(u, v) \in E_{f'}$.



Proof of Monotonicity Lemma — Case 1

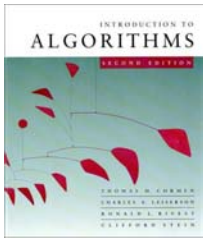
Consider two cases depending on whether $(u, v) \in E_f$.

Case 1: $(u, v) \in E_f$.

We have

$$\begin{aligned} \delta(v) &\le \delta(u) + 1 && \text{(triangle inequality)} \\ &\le \delta'(u) + 1 && \text{(induction)} \\ &= \delta'(v) && \text{(breadth-first path),} \end{aligned}$$

and thus monotonicity of $\delta(v)$ is established.



Proof of Monotonicity Lemma — Case 2

Case: $(u, v) \notin E_f$.

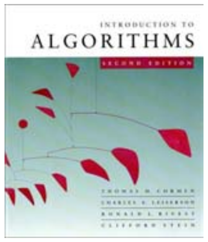
Since $(u, v) \in E_{f'}$, the augmenting path p that produced f' from f must have included (v, u). Moreover, p is a breadth-first path in G_f:

$$p = s \to \cdots \to v \to u \to \cdots \to t .$$

Thus, we have

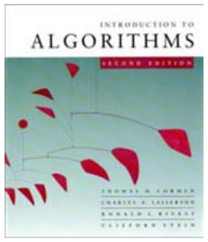
$$\begin{aligned}
\delta(v) &= \delta(u) - 1 && \text{(breadth-first path)} \\
&\le \delta'(u) - 1 && \text{(induction)} \\
&= \delta'(v) - 2 && \text{(breadth-first path)} \\
&< \delta'(v) ,
\end{aligned}$$

thereby establishing monotonicity for this case, too. □



Counting flow augmentations

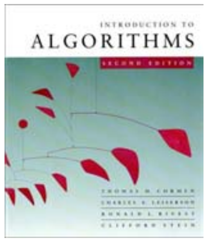
Theorem. The number of flow augmentations in the Edmonds-Karp algorithm (Ford-Fulkerson with breadth-first augmenting paths) is $O(VE)$.



Counting flow augmentations

Theorem. The number of flow augmentations in the Edmonds-Karp algorithm (Ford-Fulkerson with breadth-first augmenting paths) is $O(VE)$.

Proof. Let p be an augmenting path, and suppose that we have $c_f(u, v) = c_f(p)$ for edge $(u, v) \in p$. Then, we say that (u, v) is ***critical***, and it disappears from the residual graph after flow augmentation.



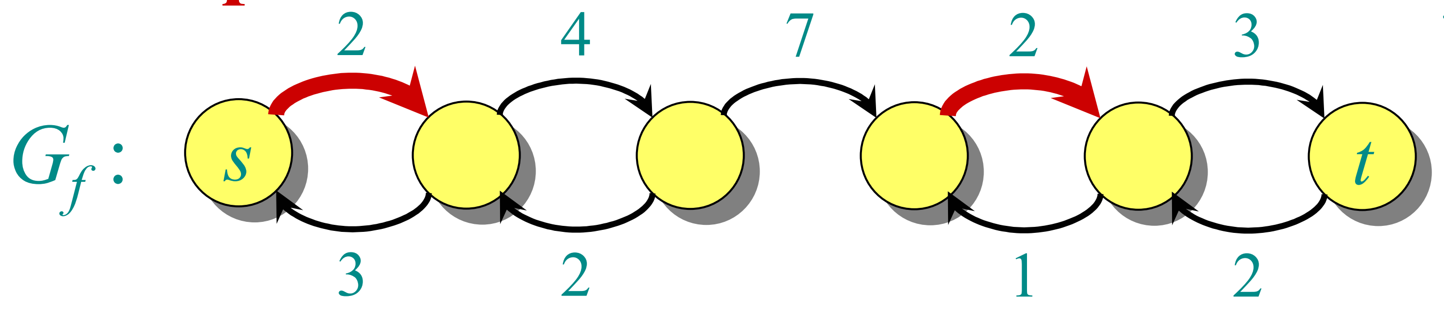
Counting flow augmentations

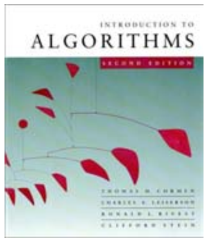
Theorem. The number of flow augmentations in the Edmonds-Karp algorithm (Ford-Fulkerson with breadth-first augmenting paths) is $O(VE)$.

Proof. Let p be an augmenting path, and suppose that we have $c_f(u, v) = c_f(p)$ for edge $(u, v) \in p$. Then, we say that (u, v) is ***critical***, and it disappears from the residual graph after flow augmentation.

Example:

$c_f(p) = 2$

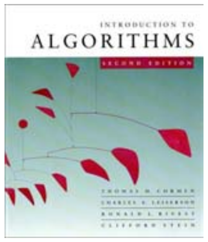




Counting flow augmentations (continued)

The first time an edge (u, v) is critical, we have $\delta(v) = \delta(u) + 1$, since p is a breadth-first path. We must wait until (v, u) is on an augmenting path before (u, v) can be critical again. Let δ' be the distance function when (v, u) is on an augmenting path. Then, we have

$$\begin{aligned} \delta'(u) &= \delta'(v) + 1 && \text{(breadth-first path)} \\ &\geq \delta(v) + 1 && \text{(monotonicity)} \\ &= \delta(u) + 2 && \text{(breadth-first path).} \end{aligned}$$

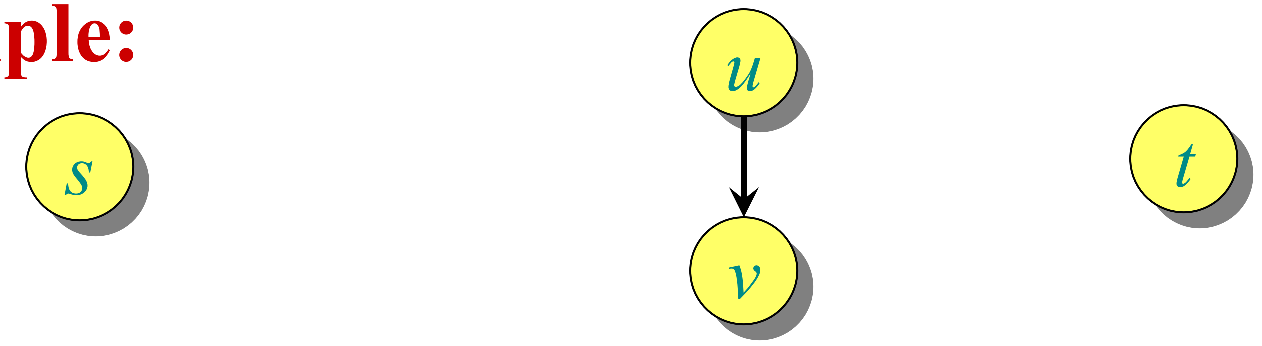


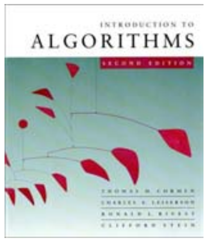
Counting flow augmentations (continued)

The first time an edge (u, v) is critical, we have $\delta(v) = \delta(u) + 1$, since p is a breadth-first path. We must wait until (v, u) is on an augmenting path before (u, v) can be critical again. Let δ' be the distance function when (v, u) is on an augmenting path. Then, we have

$$\begin{aligned}\delta'(u) &= \delta'(v) + 1 && \text{(breadth-first path)}\\ &\geq \delta(v) + 1 && \text{(monotonicity)}\\ &= \delta(u) + 2 && \text{(breadth-first path).}\end{aligned}$$

Example:



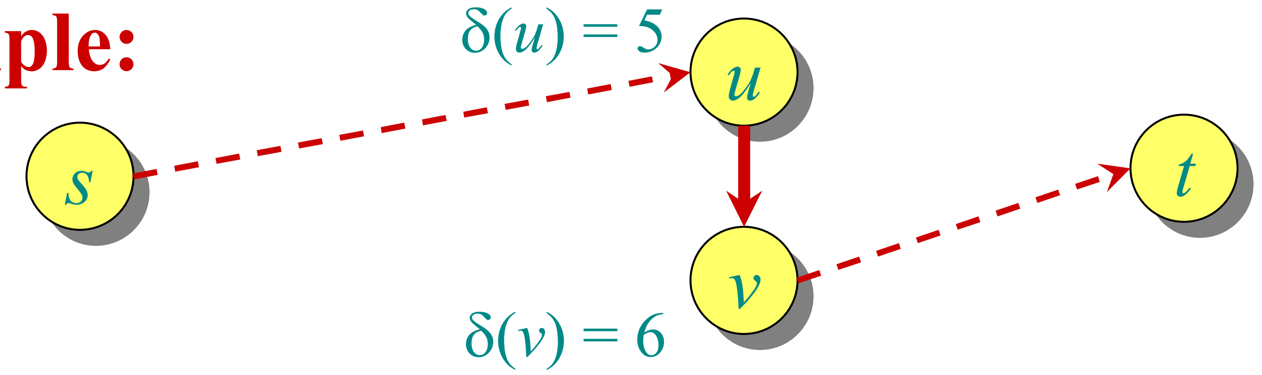


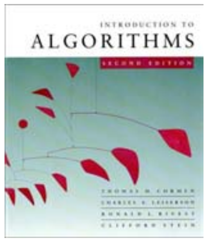
Counting flow augmentations (continued)

The first time an edge (u, v) is critical, we have $\delta(v) = \delta(u) + 1$, since p is a breadth-first path. We must wait until (v, u) is on an augmenting path before (u, v) can be critical again. Let δ' be the distance function when (v, u) is on an augmenting path. Then, we have

$$\begin{aligned} \delta'(u) &= \delta'(v) + 1 && \text{(breadth-first path)} \\ &\geq \delta(v) + 1 && \text{(monotonicity)} \\ &= \delta(u) + 2 && \text{(breadth-first path).} \end{aligned}$$

Example:



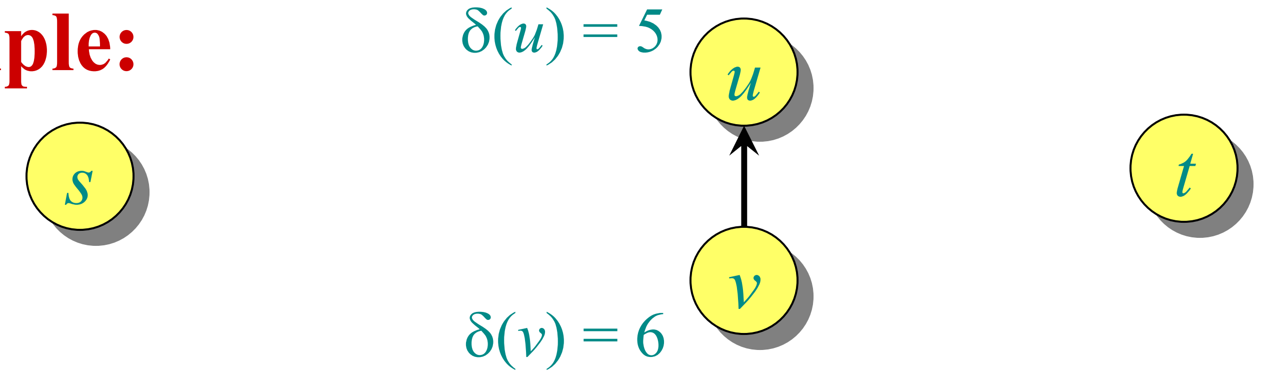


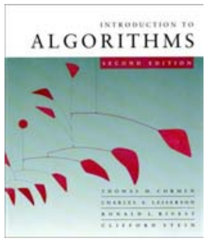
Counting flow augmentations (continued)

The first time an edge (u, v) is critical, we have $\delta(v) = \delta(u) + 1$, since p is a breadth-first path. We must wait until (v, u) is on an augmenting path before (u, v) can be critical again. Let δ' be the distance function when (v, u) is on an augmenting path. Then, we have

$$\begin{aligned} \delta'(u) &= \delta'(v) + 1 && \text{(breadth-first path)} \\ &\ge \delta(v) + 1 && \text{(monotonicity)} \\ &= \delta(u) + 2 && \text{(breadth-first path).} \end{aligned}$$

Example:



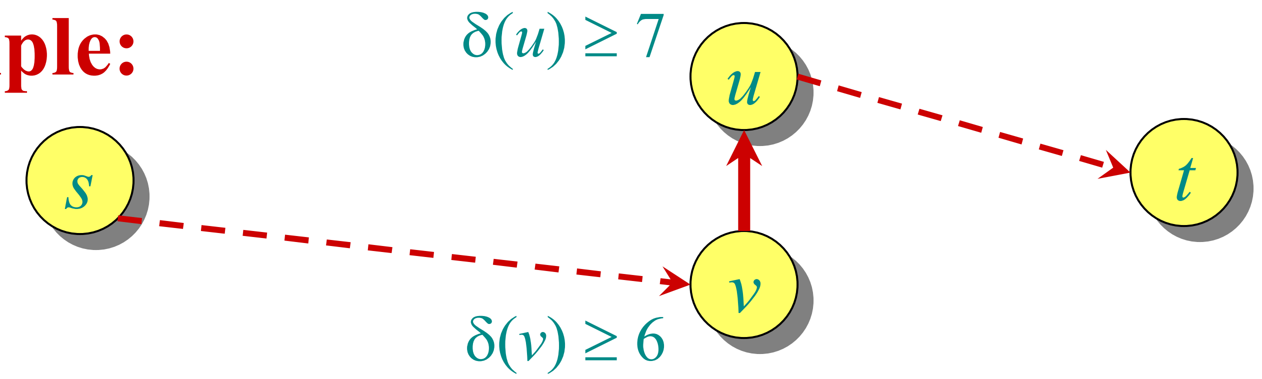


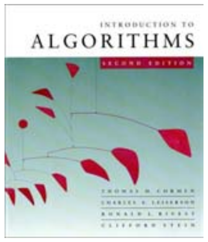
Counting flow augmentations (continued)

The first time an edge (u, v) is critical, we have $\delta(v) = \delta(u) + 1$, since p is a breadth-first path. We must wait until (v, u) is on an augmenting path before (u, v) can be critical again. Let δ' be the distance function when (v, u) is on an augmenting path. Then, we have

$$\begin{aligned} \delta'(u) &= \delta'(v) + 1 && \text{(breadth-first path)} \\ &\geq \delta(v) + 1 && \text{(monotonicity)} \\ &= \delta(u) + 2 && \text{(breadth-first path).} \end{aligned}$$

Example:



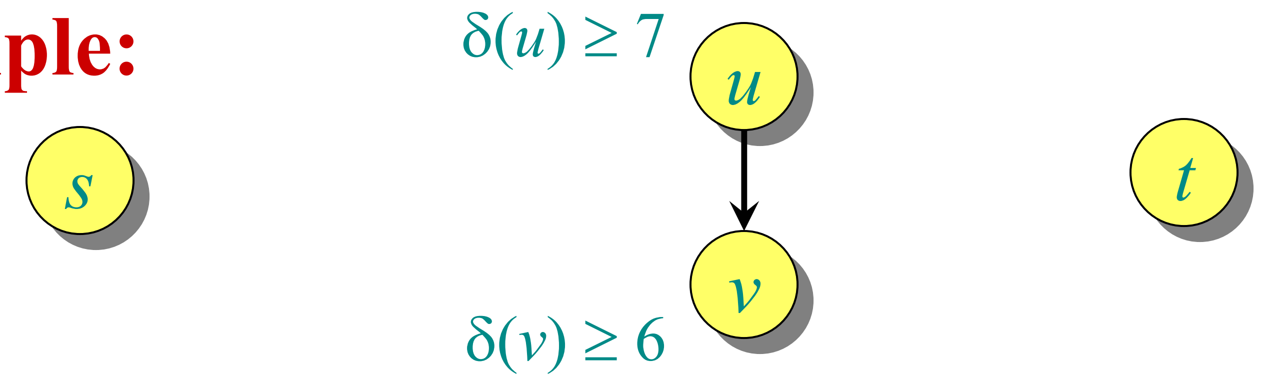


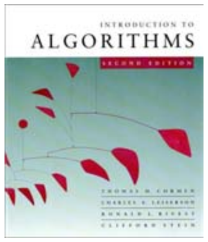
Counting flow augmentations (continued)

The first time an edge (u, v) is critical, we have $\delta(v) = \delta(u) + 1$, since p is a breadth-first path. We must wait until (v, u) is on an augmenting path before (u, v) can be critical again. Let δ' be the distance function when (v, u) is on an augmenting path. Then, we have

$$\begin{aligned} \delta'(u) &= \delta'(v) + 1 && \text{(breadth-first path)} \\ &\geq \delta(v) + 1 && \text{(monotonicity)} \\ &= \delta(u) + 2 && \text{(breadth-first path).} \end{aligned}$$

Example:



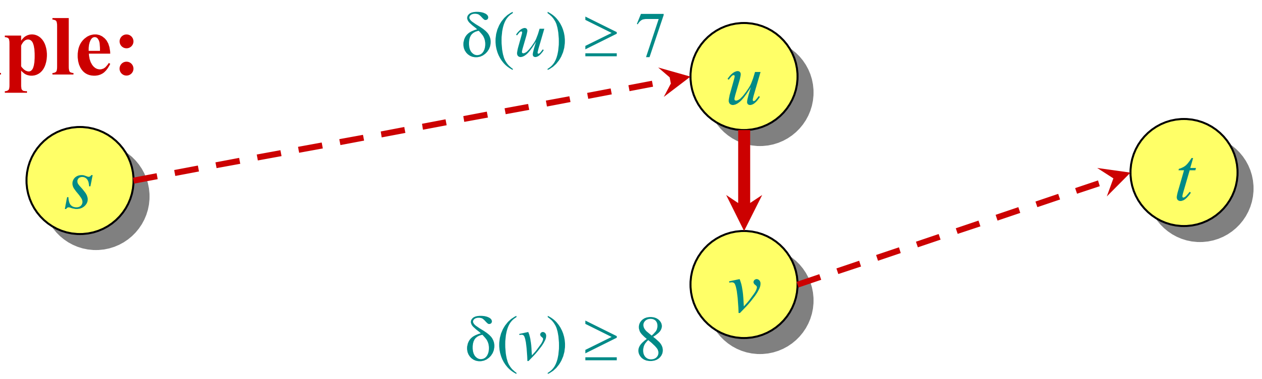


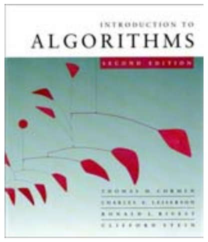
Counting flow augmentations (continued)

The first time an edge (u, v) is critical, we have $\delta(v) = \delta(u) + 1$, since p is a breadth-first path. We must wait until (v, u) is on an augmenting path before (u, v) can be critical again. Let δ' be the distance function when (v, u) is on an augmenting path. Then, we have

$$\begin{aligned} \delta'(u) &= \delta'(v) + 1 && \text{(breadth-first path)} \\ &\geq \delta(v) + 1 && \text{(monotonicity)} \\ &= \delta(u) + 2 && \text{(breadth-first path).} \end{aligned}$$

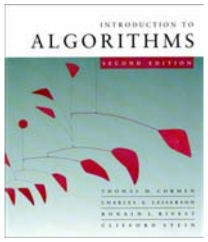
Example:





Running time of Edmonds-Karp

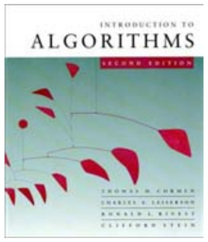
Distances start out nonnegative, never decrease, and are at most $|V| – 1$ until the vertex becomes unreachable. Thus, (u, v) occurs as a critical edge $O(V)$ times, because $\delta(v)$ increases by at least 2 between occurrences. Since the residual graph contains $O(E)$ edges, the number of flow augmentations is $O(V E)$. ◻



Running time of Edmonds-Karp

Distances start out nonnegative, never decrease, and are at most $|V| - 1$ until the vertex becomes unreachable. Thus, (u, v) occurs as a critical edge $O(V)$ times, because $\delta(v)$ increases by at least 2 between occurrences. Since the residual graph contains $O(E)$ edges, the number of flow augmentations is $O(V E)$. □

Corollary. The Edmonds-Karp maximum-flow algorithm runs in $O(V E^2)$ time.

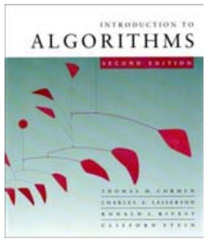


Running time of Edmonds-Karp

Distances start out nonnegative, never decrease, and are at most $|V| - 1$ until the vertex becomes unreachable. Thus, (u, v) occurs as a critical edge $O(V)$ times, because $\delta(v)$ increases by at least 2 between occurrences. Since the residual graph contains $O(E)$ edges, the number of flow augmentations is $O(V E)$. □

Corollary. The Edmonds-Karp maximum-flow algorithm runs in $O(V E^2)$ time.

Proof. Breadth-first search runs in $O(E)$ time, and all other bookkeeping is $O(V)$ per augmentation. □



Best to date

- The asymptotically fastest algorithm to date for maximum flow, due to King, Rao, and Tarjan, runs in $O(VE \log_{E/(V \lg V)} V)$ time.
- If we allow running times as a function of edge weights, the fastest algorithm for maximum flow, due to Goldberg and Rao, runs in time
$$O\left(\min\{V^{2/3}, E^{1/2}\} \cdot E \lg (V^2/E + 2) \cdot \lg C\right),$$
where C is the maximum capacity of any edge in the graph.