

Lecture 21: Max-Flow II

Jessica Sorrell

November 11, 2025

601.433/633 Introduction to Algorithms

Slides by Mike Dinitz

Introduction

Last time:

- ▶ Max-Flow = Min-Cut
- ▶ Can compute max flow and min cut using Ford-Fulkerson: while residual graph has an $s \rightarrow t$ path, push flow along it.
 - ▶ Corollary: if all capacities integers, max-flow is integral
 - ▶ If max-flow has value F , time $O(F(m+n))$ (if all capacities integers)
 - ▶ Exponential time!

Today:

- ▶ Important setting where FF is enough: max bipartite matching
- ▶ Two ways of making FF faster: Edmonds-Karp

Integrality

Corollary

If all capacities are integers, then there is a max flow such that the flow through every edge is an integer

Integrality

Corollary

If all capacities are integers, then there is a max flow such that the flow through every edge is an integer

Proof.

Induction on iterations of the Ford-Fulkerson algorithm: initially true, stays true $\implies$ true at end. □

Running Time

Theorem

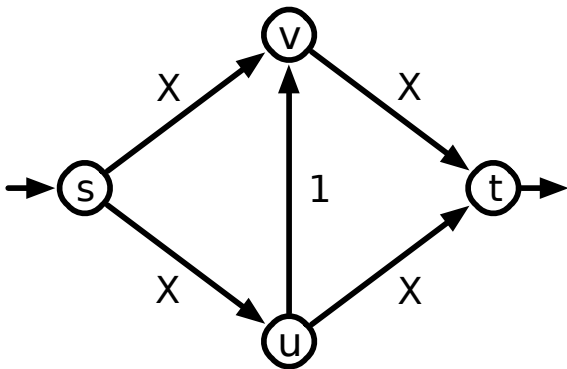
If all capacities are integers and the max flow value is $\mathbf{F}$, Ford-Fulkerson takes time at most $\mathbf{O(F(m + n))}$

Running Time

Theorem

If all capacities are integers and the max flow value is F , Ford-Fulkerson takes time at most $O(F(m + n))$

Finding path takes $O(m + n)$ time, increase flow by at least **1**



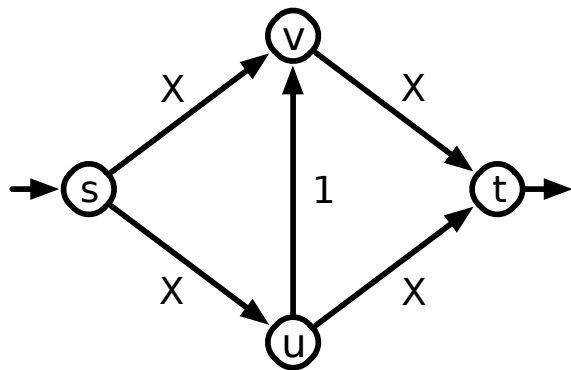
A bad example for the Ford-Fulkerson algorithm.

Running Time

Theorem

If all capacities are integers and the max flow value is F , Ford-Fulkerson takes time at most $O(F(m + n))$

Finding path takes $O(m + n)$ time, increase flow by at least 1



Running time $\geq \#$ iterations.

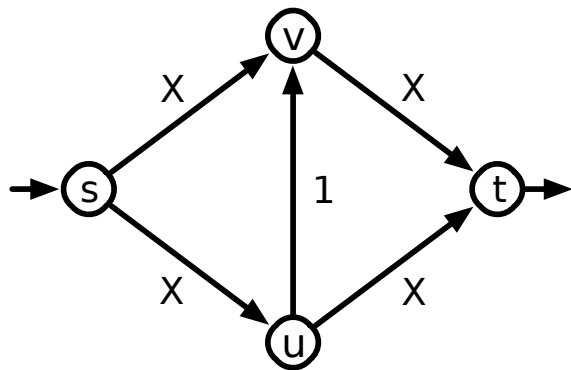
A bad example for the Ford-Fulkerson algorithm.

Running Time

Theorem

If all capacities are integers and the max flow value is F , Ford-Fulkerson takes time at most $O(F(m + n))$

Finding path takes $O(m + n)$ time, increase flow by at least 1



Running time $\geq \#$ iterations.

This example:

- ▶ Running time: $\Omega(x)$

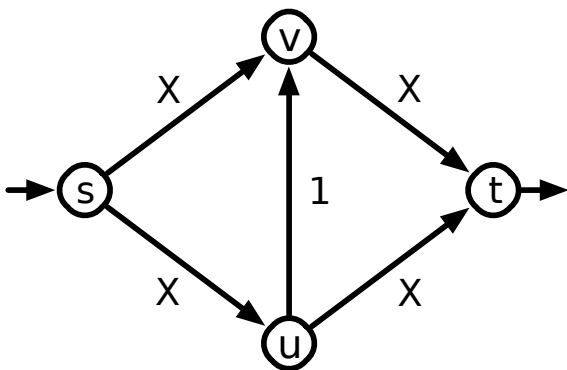
A bad example for the Ford-Fulkerson algorithm.

Running Time

Theorem

If all capacities are integers and the max flow value is F , Ford-Fulkerson takes time at most $O(F(m + n))$

Finding path takes $O(m + n)$ time, increase flow by at least 1



A bad example for the Ford-Fulkerson algorithm.

Running time $\geq \#$ iterations.

This example:

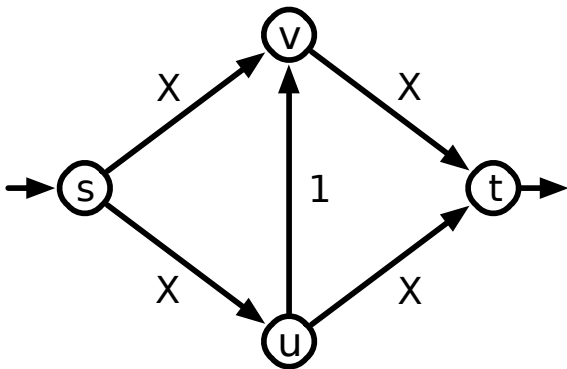
- ▶ Running time: $\Omega(x)$
- ▶ Input size $O(\log x) + O(1)$

Running Time

Theorem

If all capacities are integers and the max flow value is F , Ford-Fulkerson takes time at most $O(F(m + n))$

Finding path takes $O(m + n)$ time, increase flow by at least 1



A bad example for the Ford-Fulkerson algorithm.

Running time $\geq \#$ iterations.

This example:

- ▶ Running time: $\Omega(x)$
- ▶ Input size $O(\log x) + O(1)$

$\implies$ Exponential time!

Max Bipartite Matching

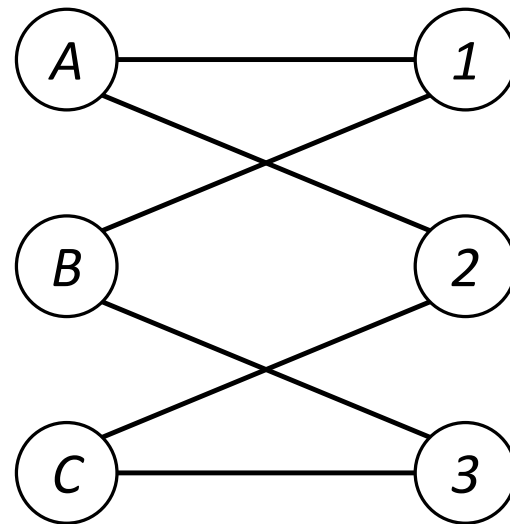
Setup

Definition

A graph $G = (V, E)$ is *bipartite* if V can be partitioned into two parts L, R such that every edge in E has one endpoint in L and one endpoint in R .

Definition

A *matching* is a subset $M \subseteq E$ such that $e \cap e' = \emptyset$ for all $e, e' \in M$ with $e \neq e'$ (no two edges share an endpoint)



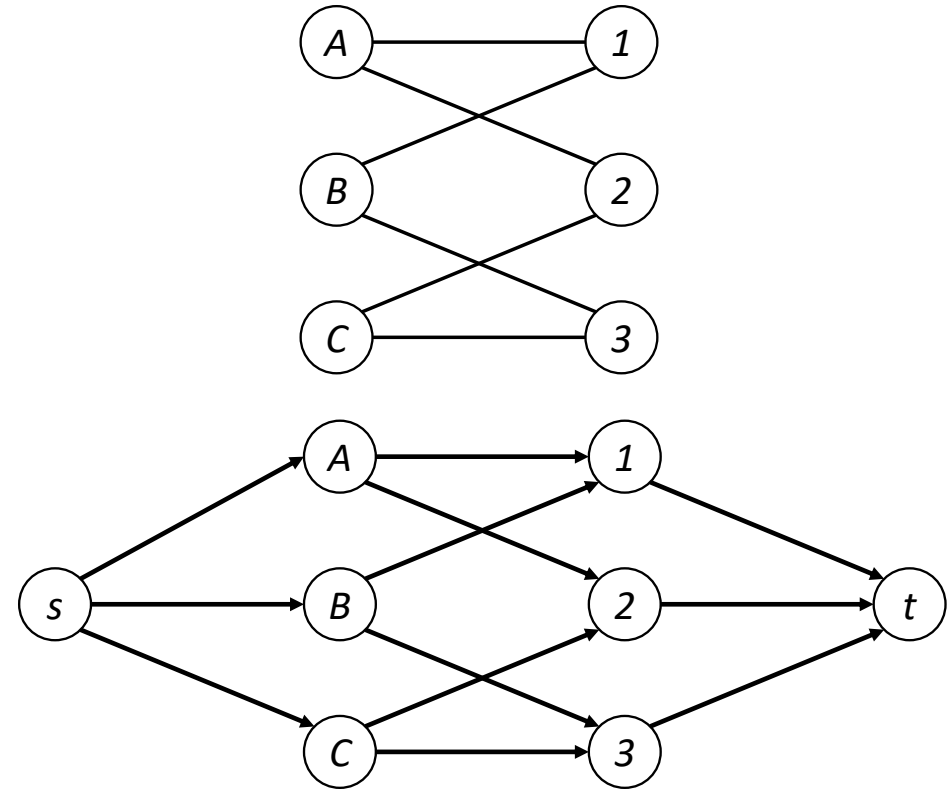
Bipartite Maximum Matching: Given bipartite graph $G = (V, E)$, find matching M maximizing $|M|$

- ▶ Extremely important problem, doesn't seem to have much to do with flow!

Algorithm

Give all edges capacity **1**
Direct all edges from **L** to **R**
Add source **s** and sink **t**
Add edges of capacity **1** from **s** to **L**
Add edges of capacity **1** from **R** to **t**

Run FF to get flow **f**
Return $\mathbf{M} = \{\mathbf{e} \in \mathbf{L} \times \mathbf{R} : \mathbf{f}(\mathbf{e}) > 0\}$



Correctness

Claim: M is a matching

Correctness

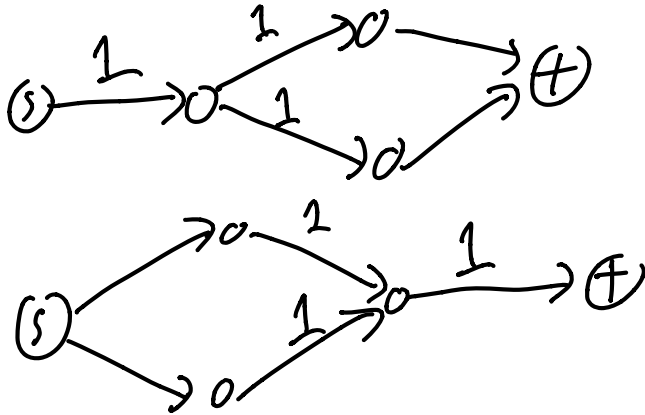
Claim: M is a matching

Proof: capacities in $\{0, 1\} \implies f(e) \in \{0, 1\}$
for all e (integrality)

Correctness

Claim: M is a matching

Proof: capacities in $\{0, 1\} \implies f(e) \in \{0, 1\}$
for all e (integrality)

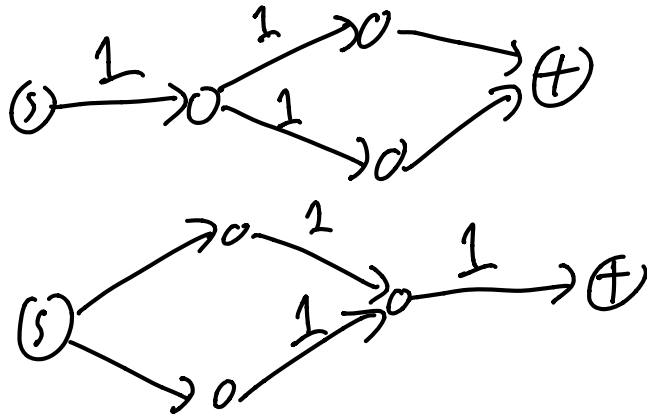


Correctness

Claim: M is a matching

Proof: capacities in $\{0, 1\} \implies f(e) \in \{0, 1\}$
for all e (integrality)

Claim: M is maximum matching



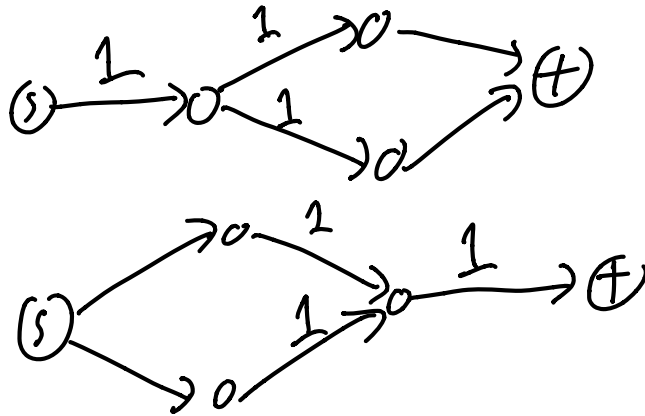
Correctness

Claim: M is a matching

Proof: capacities in $\{0, 1\} \implies f(e) \in \{0, 1\}$ for all e (integrality)

Claim: M is maximum matching

Proof: Suppose larger matching M'



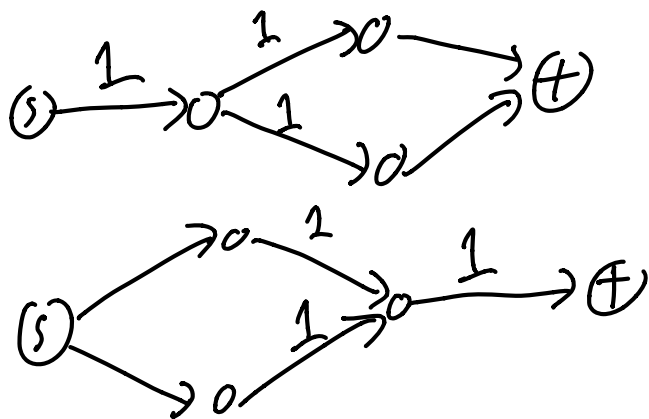
Correctness

Claim: M is a matching

Proof: capacities in $\{0, 1\} \implies f(e) \in \{0, 1\}$
for all e (integrality)

Claim: M is maximum matching

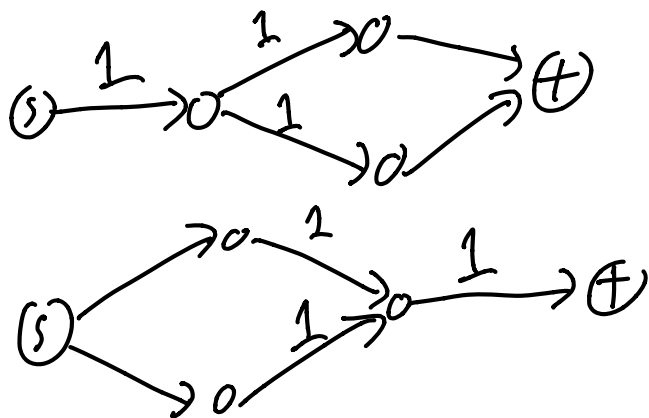
Proof: Suppose larger matching M'
Can send $|M'|$ flow using M' !



Correctness

Claim: M is a matching

Proof: capacities in $\{0, 1\} \implies f(e) \in \{0, 1\}$ for all e (integrality)



Claim: M is maximum matching

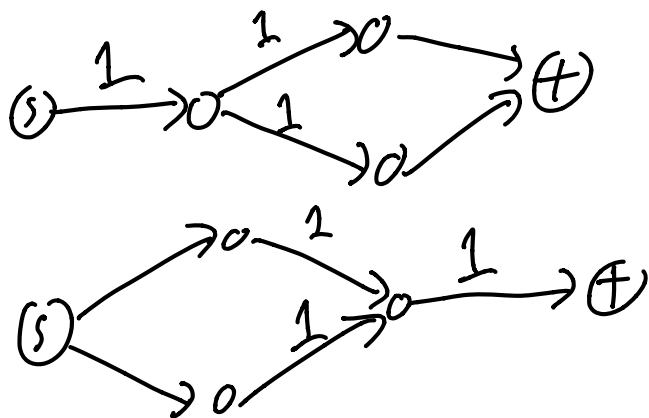
Proof: Suppose larger matching M'
Can send $|M'|$ flow using M' !

- ▶ $f'(s, u) = 1$ if u matched in M' , otherwise 0
- ▶ $f'(v, t) = 1$ if v matched in M' , otherwise 0
- ▶ $f'(u, v) = 1$ if $\{u, v\} \in M'$, otherwise 0

Correctness

Claim: M is a matching

Proof: capacities in $\{0, 1\} \implies f(e) \in \{0, 1\}$ for all e (integrality)



Claim: M is maximum matching

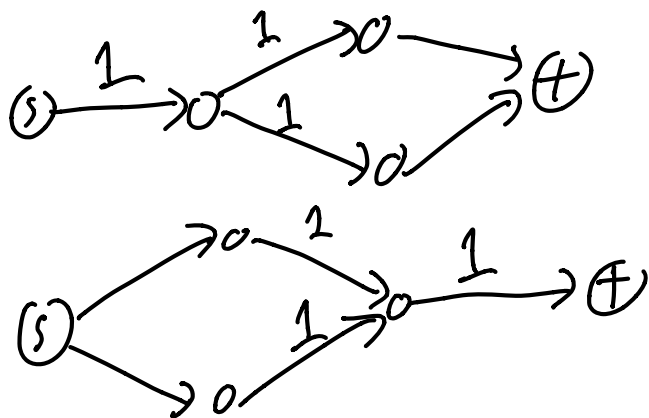
Proof: Suppose larger matching M'
Can send $|M'|$ flow using M' !

- ▶ $f'(s, u) = 1$ if u matched in M' , otherwise 0
- ▶ $f'(v, t) = 1$ if v matched in M' , otherwise 0
- ▶ $f'(u, v) = 1$ if $\{u, v\} \in M'$, otherwise 0
- ▶ $|f'| = |M'| > |M| = |f|$

Correctness

Claim: M is a matching

Proof: capacities in $\{0, 1\} \implies f(e) \in \{0, 1\}$ for all e (integrality)



Claim: M is maximum matching

Proof: Suppose larger matching M'
Can send $|M'|$ flow using M' !

- ▶ $f'(s, u) = 1$ if u matched in M' , otherwise 0
- ▶ $f'(v, t) = 1$ if v matched in M' , otherwise 0
- ▶ $f'(u, v) = 1$ if $\{u, v\} \in M'$, otherwise 0
- ▶ $|f'| = |M'| > |M| = |f|$
- ▶ Contradiction

Running Time

Running Time:

- ▶ $O(n + m)$ to make new graph
- ▶ $|f| = |M| \leq n/2$ iterations of FF

$\Rightarrow O(n(m + n)) = O(mn)$ time (assuming $m \geq \Omega(n)$)

Extensions

Many extensions:

- ▶ Max-weight bipartite matching
- ▶ Min-cost perfect matching
- ▶ Matchings in general graphs
- ▶ ...

Extensions

Many extensions:

- ▶ Max-weight bipartite matching
- ▶ Min-cost perfect matching
- ▶ Matchings in general graphs
- ▶ ...

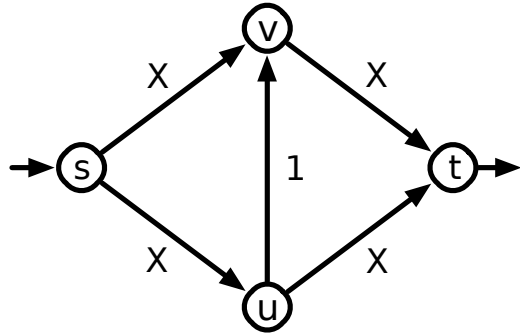
Still active area of study!

- ▶ Michael Dinitz, Sungjin Im, Thomas Lavastida, Benjamin Moseley, Sergei Vassilvitskii. *Faster Matchings via Learned Duals*. NeurIPS 2021.
- ▶ Michael Dinitz, George Li, Quanquan Liu, Felix Zhou. *Differentially Private Matchings*. Submitted, on arXiv.

Edmonds-Karp

Intuition

Bad example for Ford-Fulkerson:

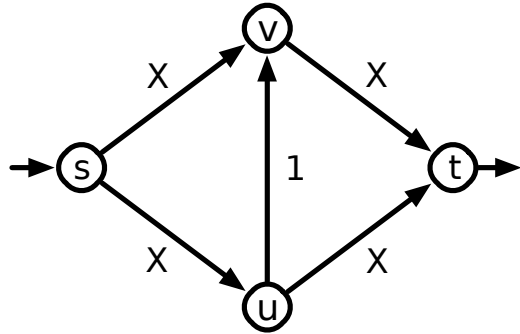


If Ford-Fulkerson chooses bad augmenting paths, super slow!

A bad example for the Ford-Fulkerson algorithm.

Intuition

Bad example for Ford-Fulkerson:



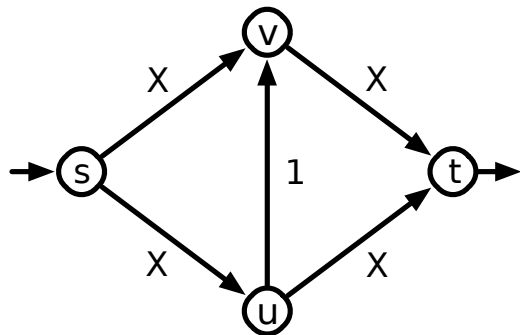
A bad example for the Ford-Fulkerson algorithm.

If Ford-Fulkerson chooses bad augmenting paths, super slow!

Obvious idea: Choose better paths!

Intuition

Bad example for Ford-Fulkerson:



A bad example for the Ford-Fulkerson algorithm.

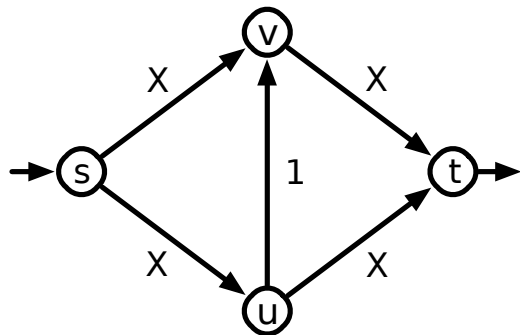
Obvious path to pick:

If Ford-Fulkerson chooses bad augmenting paths, super slow!

Obvious idea: Choose better paths!

Intuition

Bad example for Ford-Fulkerson:



A bad example for the Ford-Fulkerson algorithm.

Obvious path to pick:

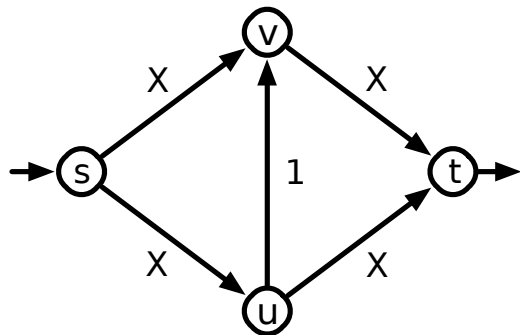
$$\arg \max_{\text{augmenting paths } P} \min_{e \in P} c_f(e) \quad (\text{"widest" augmenting path})$$

If Ford-Fulkerson chooses bad augmenting paths, super slow!

Obvious idea: Choose better paths!

Intuition

Bad example for Ford-Fulkerson:



If Ford-Fulkerson chooses bad augmenting paths, super slow!

Obvious idea: Choose better paths!

A bad example for the Ford-Fulkerson algorithm.

Obvious path to pick:

$$\arg \max_{\text{augmenting paths } P} \min_{e \in P} c_f(e) \quad (\text{"widest" augmenting path})$$

Less obvious path to pick:

$$\arg \min_{\text{augmenting paths } P} |P| \quad (\text{augmenting path with fewest edges})$$

Edmonds-Karp

Use Ford-Fulkerson, but pick *shortest* augmenting path (unweighted)

- ▶ Ignore capacities, just find augmenting path with fewest hops!
- ▶ Easy to compute with BFS in $O(m + n)$ time.
- ▶ Correct, since just FF with particular path choice.

Edmonds-Karp

Use Ford-Fulkerson, but pick *shortest* augmenting path (unweighted)

- ▶ Ignore capacities, just find augmenting path with fewest hops!
- ▶ Easy to compute with BFS in $O(m + n)$ time.
- ▶ Correct, since just FF with particular path choice.

Main question: how many iterations?

Edmonds-Karp

Use Ford-Fulkerson, but pick *shortest* augmenting path (unweighted)

- ▶ Ignore capacities, just find augmenting path with fewest hops!
- ▶ Easy to compute with BFS in $O(m + n)$ time.
- ▶ Correct, since just FF with particular path choice.

Main question: how many iterations?

Theorem

Edmonds-Karp has at most $O(mn)$ iterations, so at most $O(m^2n)$ running time (if $m \geq n$)

Proof (sketch) of Edmonds-Karp

Idea: prove that distance from s to t (unweighted) goes up by at least one every $\leq m$ iterations.

Proof (sketch) of Edmonds-Karp

Idea: prove that distance from s to t (unweighted) goes up by at least one every $\leq m$ iterations.

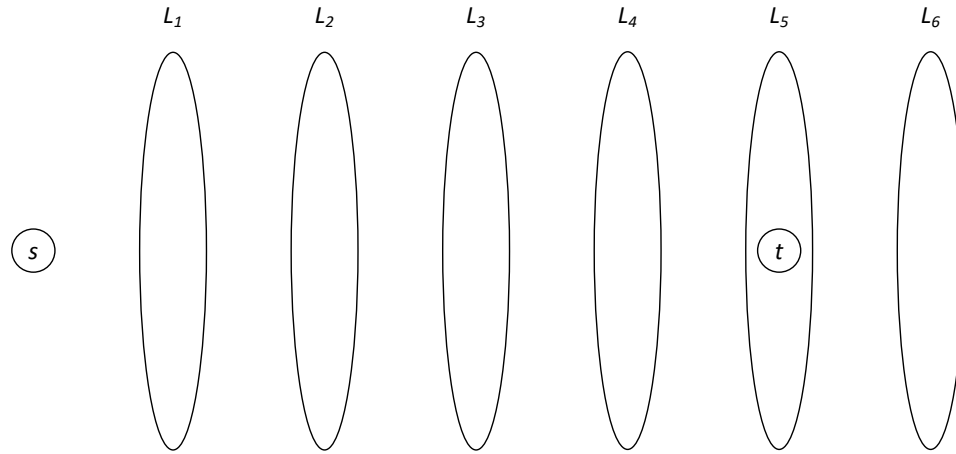
- ▶ Distance initially $\geq 1 \implies$ distance $> n$ after at most mn iterations
- ▶ Only distance larger than n is ∞ : no $s \rightarrow t$ path

$\implies$ Terminates after at most mn iterations.

Proof (sketch) of Edmonds-Karp (continued)

Suppose $s \rightarrow t$ distance is d .

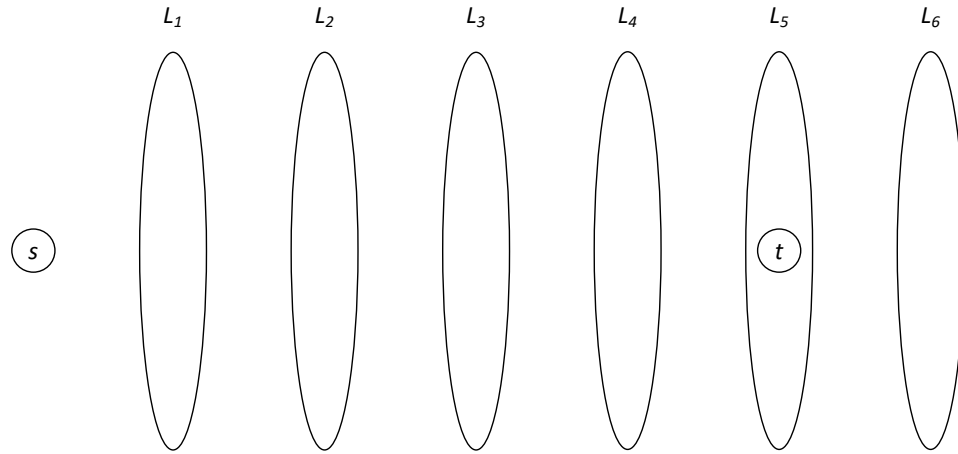
“Lay out” residual graph in levels by BFS (distance from s)



Proof (sketch) of Edmonds-Karp (continued)

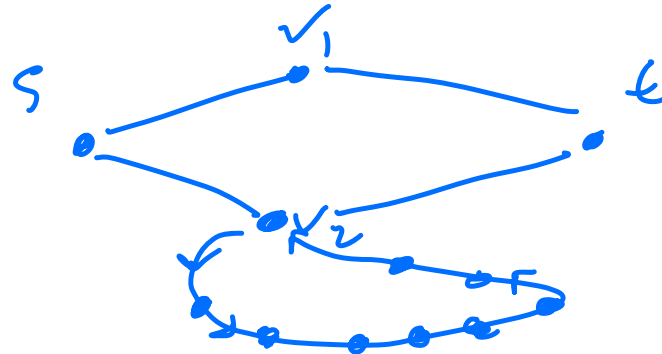
Suppose $s \rightarrow t$ distance is d .

“Lay out” residual graph in levels by BFS (distance from s)



Edge types:

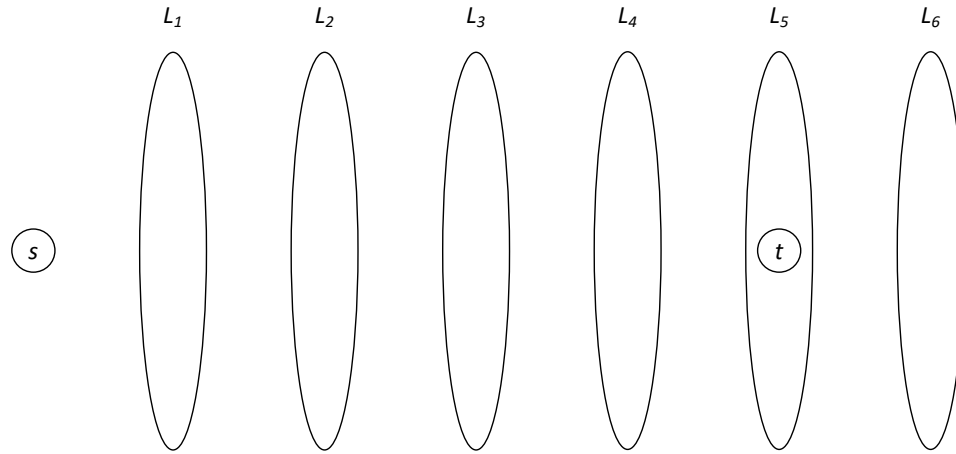
- ▶ Forward edges: **1** level
- ▶ Edges inside level
- ▶ Backwards edges



Proof (sketch) of Edmonds-Karp (continued)

Suppose $s \rightarrow t$ distance is d .

“Lay out” residual graph in levels by BFS (distance from s)



Edge types:

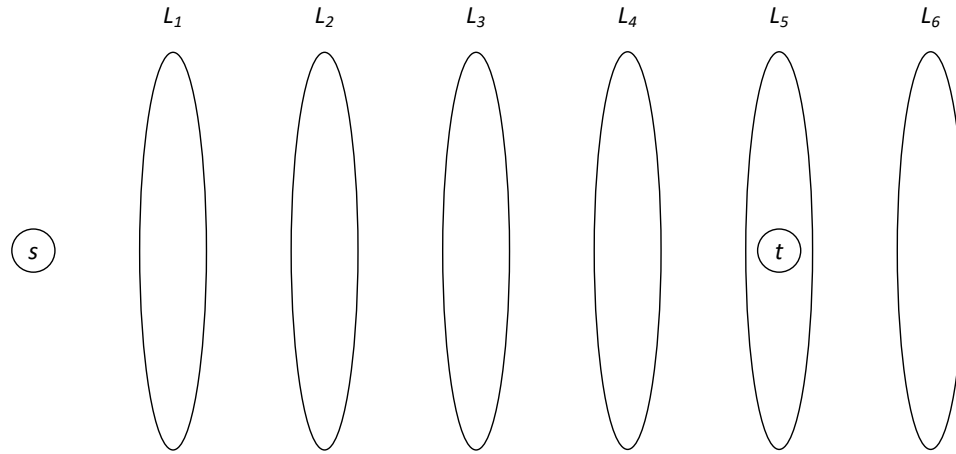
- ▶ Forward edges: **1** level
- ▶ Edges inside level
- ▶ Backwards edges

What happens when we choose a *shortest* augmenting path?

Proof (sketch) of Edmonds-Karp (continued)

Suppose $s \rightarrow t$ distance is d .

“Lay out” residual graph in levels by BFS (distance from s)



Edge types:

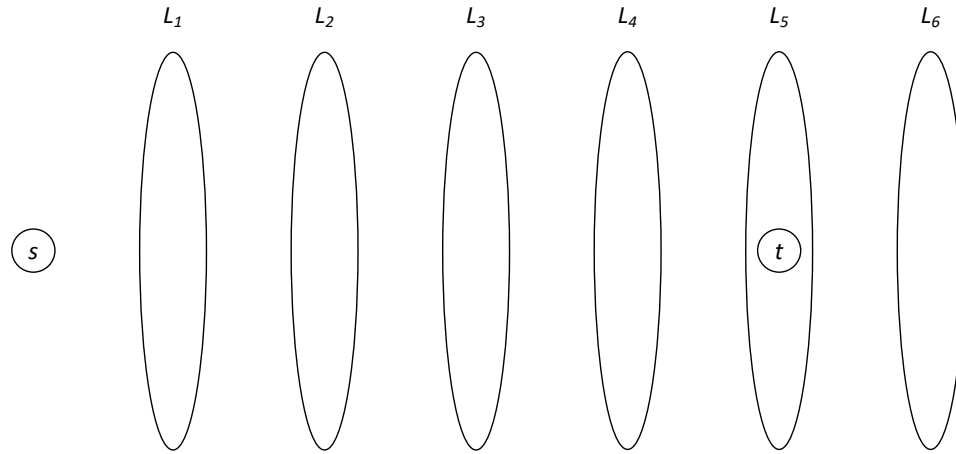
- ▶ Forward edges: **1** level
- ▶ Edges inside level
- ▶ Backwards edges

What happens when we choose a *shortest* augmenting path? Only uses forward edges!

Proof (sketch) of Edmonds-Karp (continued)

Suppose $s \rightarrow t$ distance is d .

“Lay out” residual graph in levels by BFS (distance from s)



Edge types:

- ▶ Forward edges: **1** level
- ▶ Edges inside level
- ▶ Backwards edges

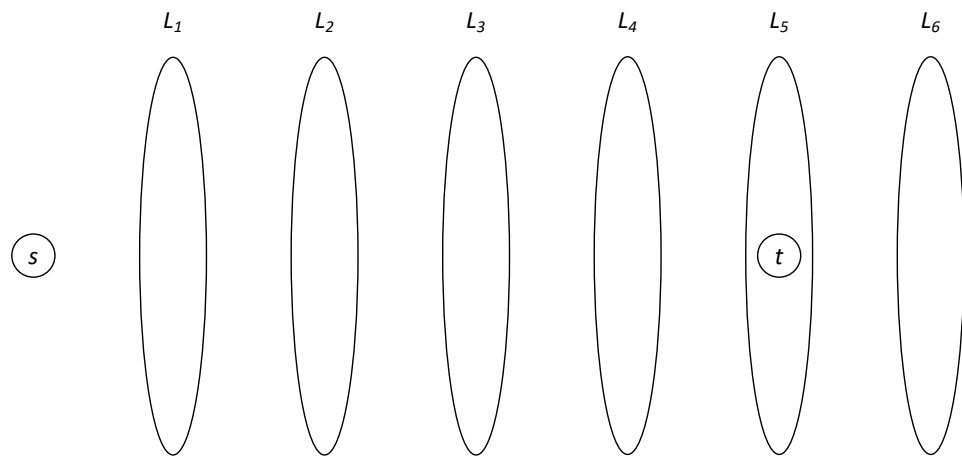
What happens when we choose a *shortest* augmenting path? Only uses forward edges!

- ▶ At least **1** forward edge gets removed, replaced with backwards edge.
- ▶ No backwards edges turned forward

Proof (sketch) of Edmonds-Karp (continued)

Suppose $s \rightarrow t$ distance is d .

“Lay out” residual graph in levels by BFS (distance from s)



Edge types:

- ▶ Forward edges: **1** level
- ▶ Edges inside level
- ▶ Backwards edges

What happens when we choose a *shortest* augmenting path? Only uses forward edges!

- ▶ At least **1** forward edge gets removed, replaced with backwards edge.
- ▶ No backwards edges turned forward

So after m iterations (same layout): no path using only forward edges $\implies$ distance larger than d !

Finishing Edmonds-Karp

So at most **mn** iterations. Each iteration unweighted shortest path: BFS, time **$O(m + n)$**

Finishing Edmonds-Karp

So at most mn iterations. Each iteration unweighted shortest path: BFS, time $O(m + n)$

Total time: $O(mn(m + n)) = O(m^2n)$. Independent of F !

Widest Path Algorithm

Algorithm: Ford-Fulkerson, always choose “widest” path.

- ▶ Correct, since FF. Running time?

Widest Path Algorithm

Algorithm: Ford-Fulkerson, always choose “widest” path.

- ▶ Correct, since FF. Running time?

Lemma

In any graph with max $s - t$ flow $\mathbf{F}$, there exists a path from s to t with capacity at least $\mathbf{F}/m$

Widest Path Algorithm

Algorithm: Ford-Fulkerson, always choose “widest” path.

- ▶ Correct, since FF. Running time?

Lemma

In any graph with max $s - t$ flow F , there exists a path from s to t with capacity at least F/m

Proof.

Let $X = \{e \in E : c(e) < F/m\}$.

Widest Path Algorithm

Algorithm: Ford-Fulkerson, always choose “widest” path.

- ▶ Correct, since FF. Running time?

Lemma

In any graph with max $s - t$ flow F , there exists a path from s to t with capacity at least F/m

Proof.

Let $X = \{e \in E : c(e) < F/m\}$.

If no $s \rightarrow t$ path in $G \setminus X$, then X an (edge) cut. Let S = nodes reachable from s in $G \setminus X$.

Widest Path Algorithm

Algorithm: Ford-Fulkerson, always choose “widest” path.

- ▶ Correct, since FF. Running time?

Lemma

In any graph with max $s - t$ flow F , there exists a path from s to t with capacity at least F/m

Proof.

Let $X = \{e \in E : c(e) < F/m\}$.

If no $s \rightarrow t$ path in $G \setminus X$, then X an (edge) cut. Let S = nodes reachable from s in $G \setminus X$.

$$\text{cap}(S, \bar{S}) \leq \text{cap}(X) = \sum_{e \in X} c(e) < m \cdot (F/m) = F$$

Widest Path Algorithm

Algorithm: Ford-Fulkerson, always choose “widest” path.

- ▶ Correct, since FF. Running time?

Lemma

In any graph with max $s - t$ flow F , there exists a path from s to t with capacity at least F/m

Proof.

Let $X = \{e \in E : c(e) < F/m\}$.

If no $s \rightarrow t$ path in $G \setminus X$, then X an (edge) cut. Let S = nodes reachable from s in $G \setminus X$.

$$\text{cap}(S, \bar{S}) \leq \text{cap}(X) = \sum_{e \in X} c(e) < m \cdot (F/m) = F$$

$$\implies \min(s, t) \text{ cut} \leq \text{cap}(S, \bar{S}) < F.$$

Widest Path Algorithm

Algorithm: Ford-Fulkerson, always choose “widest” path.

- ▶ Correct, since FF. Running time?

Lemma

In any graph with max $s - t$ flow F , there exists a path from s to t with capacity at least F/m

Proof.

Let $X = \{e \in E : c(e) < F/m\}$.

If no $s \rightarrow t$ path in $G \setminus X$, then X an (edge) cut. Let S = nodes reachable from s in $G \setminus X$.

$$\text{cap}(S, \bar{S}) \leq \text{cap}(X) = \sum_{e \in X} c(e) < m \cdot (F/m) = F$$

$\implies \min(s, t) \text{ cut} \leq \text{cap}(S, \bar{S}) < F$. Contradiction.

Widest Path Algorithm

Algorithm: Ford-Fulkerson, always choose “widest” path.

- ▶ Correct, since FF. Running time?

Lemma

In any graph with max $s - t$ flow F , there exists a path from s to t with capacity at least F/m

Proof.

Let $X = \{e \in E : c(e) < F/m\}$.

If no $s \rightarrow t$ path in $G \setminus X$, then X an (edge) cut. Let S = nodes reachable from s in $G \setminus X$.

$$\text{cap}(S, \bar{S}) \leq \text{cap}(X) = \sum_{e \in X} c(e) < m \cdot (F/m) = F$$

$\implies \min(s, t) \text{ cut} \leq \text{cap}(S, \bar{S}) < F$. Contradiction.

$\implies \exists s \rightarrow t$ path P in $G \setminus X$: every edge of P has capacity at least F/m



Widest Path Algorithm

Algorithm: Ford-Fulkerson, always choose “widest” path.

- ▶ Correct, since FF. Running time?

Lemma

In any graph with max $s - t$ flow F , there exists a path from s to t with capacity at least F/m

Proof.

Let $X = \{e \in E : c(e) < F/m\}$.

If no $s \rightarrow t$ path in $G \setminus X$, then X an (edge) cut. Let S = nodes reachable from s in $G \setminus X$.

$$\text{cap}(S, \bar{S}) \leq \text{cap}(X) = \sum_{e \in X} c(e) < m \cdot (F/m) = F$$

$\implies \min(s, t) \text{ cut} \leq \text{cap}(S, \bar{S}) < F$. Contradiction.

$\implies \exists s \rightarrow t$ path P in $G \setminus X$: every edge of P has capacity at least F/m



Does this imply at most m iterations?

Running Time II

Theorem

If F is the value of the maximum flow and all capacities are integers, # iterations is at most $O(m \log F)$

Running Time II

Theorem

If F is the value of the maximum flow and all capacities are integers, # iterations is at most $O(m \log F)$

How much flow remains to be sent after iteration i ?

Running Time II

Theorem

If F is the value of the maximum flow and all capacities are integers, # iterations is at most $O(m \log F)$

How much flow remains to be sent after iteration i ?

▶ $i = 0$:

Running Time II

Theorem

If F is the value of the maximum flow and all capacities are integers, # iterations is at most $O(m \log F)$

How much flow remains to be sent after iteration i ?

▶ $i = 0$: F

Running Time II

Theorem

If F is the value of the maximum flow and all capacities are integers, # iterations is at most $O(m \log F)$

How much flow remains to be sent after iteration i ?

- ▶ $i = 0$: F
- ▶ $i = 1$: Sent at least F/m , so at most $F - F/m = F(1 - 1/m)$ remaining

Running Time II

Theorem

If F is the value of the maximum flow and all capacities are integers, # iterations is at most $O(m \log F)$

How much flow remains to be sent after iteration i ?

- ▶ $i = 0$: F
- ▶ $i = 1$: Sent at least F/m , so at most $F - F/m = F(1 - 1/m)$ remaining
- ▶ $i = 2$: Sent at least R/m if R was remaining after iteration 1, so at most $R - R/m = R(1 - 1/m) \leq F(1 - 1/m)^2$ remaining

Running Time II

Theorem

If F is the value of the maximum flow and all capacities are integers, # iterations is at most $O(m \log F)$

How much flow remains to be sent after iteration i ?

- ▶ $i = 0$: F
- ▶ $i = 1$: Sent at least F/m , so at most $F - F/m = F(1 - 1/m)$ remaining
- ▶ $i = 2$: Sent at least R/m if R was remaining after iteration 1, so at most $R - R/m = R(1 - 1/m) \leq F(1 - 1/m)^2$ remaining

By induction: after iteration i , at most $F(1 - 1/m)^i$ flow remaining to be sent.

Running Time II

Theorem

If F is the value of the maximum flow and all capacities are integers, # iterations is at most $O(m \log F)$

How much flow remains to be sent after iteration i ?

- ▶ $i = 0$: F
- ▶ $i = 1$: Sent at least F/m , so at most $F - F/m = F(1 - 1/m)$ remaining
- ▶ $i = 2$: Sent at least R/m if R was remaining after iteration 1, so at most $R - R/m = R(1 - 1/m) \leq F(1 - 1/m)^2$ remaining

By induction: after iteration i , at most $F(1 - 1/m)^i$ flow remaining to be sent.

Super useful inequality: $1 + x \leq e^x$ for all $x \in \mathbb{R}$

Running Time II

Theorem

If F is the value of the maximum flow and all capacities are integers, # iterations is at most $O(m \log F)$

How much flow remains to be sent after iteration i ?

- ▶ $i = 0$: F
- ▶ $i = 1$: Sent at least F/m , so at most $F - F/m = F(1 - 1/m)$ remaining
- ▶ $i = 2$: Sent at least R/m if R was remaining after iteration 1, so at most $R - R/m = R(1 - 1/m) \leq F(1 - 1/m)^2$ remaining

By induction: after iteration i , at most $F(1 - 1/m)^i$ flow remaining to be sent.

Super useful inequality: $1 + x \leq e^x$ for all $x \in \mathbb{R}$

$\implies$ If $i > m \ln F$, amount remaining to be sent at most

$$F(1 - 1/m)^i < F(1 - 1/m)^{m \ln F} \leq F(e^{-1/m})^{m \ln F} = F \cdot e^{-\ln F} = 1$$

But all capacities integers, so must be finished!

Finishing up

Modified version of Dijkstra: find widest path in $O(m \log n)$ time

- ▶ Total time $O(m \log n \cdot m \log F) = O(m^2 \log n \log F)$
- ▶ Polynomial time!

Finishing up

Modified version of Dijkstra: find widest path in $O(m \log n)$ time

- ▶ Total time $O(m \log n \cdot m \log F) = O(m^2 \log n \log F)$
- ▶ Polynomial time!

We saw earlier how to get running time independent of F (a *strongly* polynomial-time algorithm running in $O(m^2 n)$).

Extensions

Many better algorithms for max-flow: *blocking flows* (Dinitz's algorithm (not that Dinitz)), *push-relabel* algorithms, etc.

- ▶ CLRS has a few of these.
- ▶ State of the art:
 - ▶ Strongly polynomial: $O(mn)$. Orlin [2013] & King, Rao, Tarjan [1994]
 - ▶ Weakly Polynomial: $O(m^{1+o(1)} \log U)$ (where U is maximum capacity). Chen, Kyng, Liu, Peng, Gutenberg and Sachdeva [2022]

Many other variants of flows, some of which are just $s - t$ max flow in disguise!

- ▶ Min-Cost Max-Flow: every edge also has a cost. Find minimum cost max-flow. Can be solved with just normal max flow!