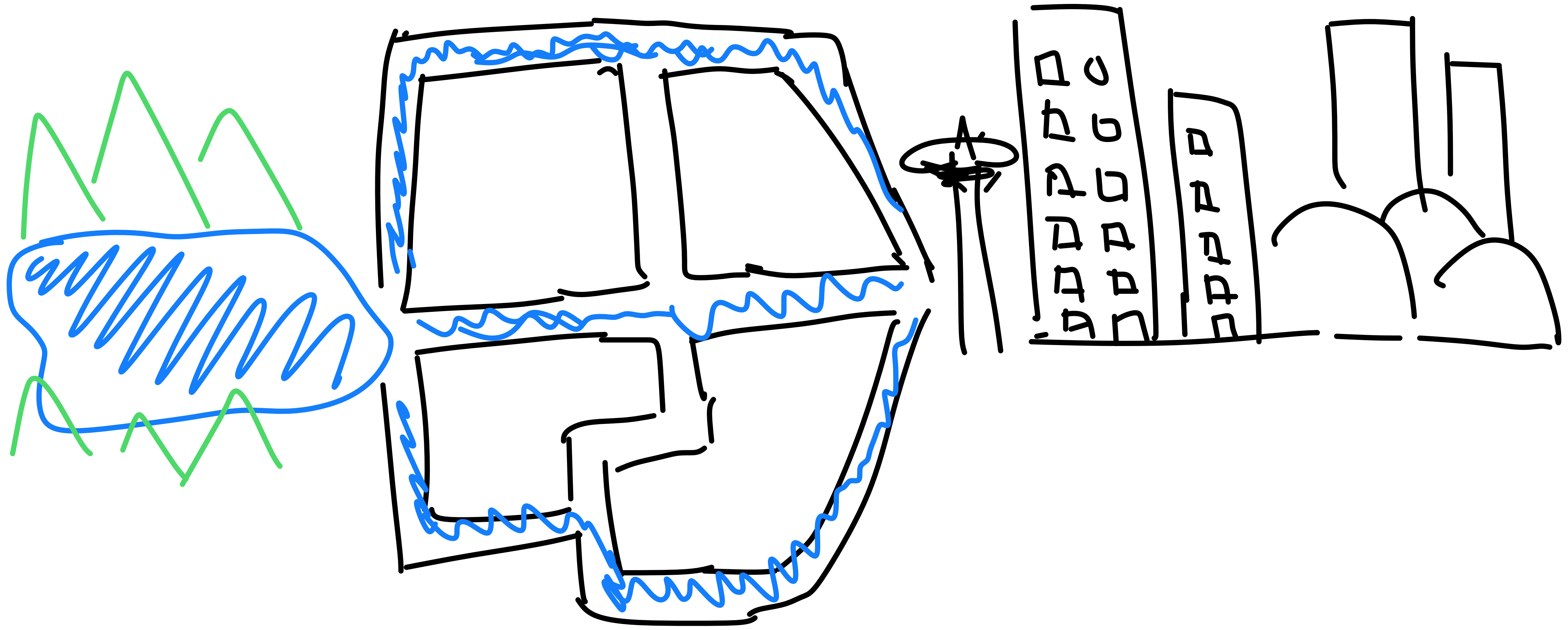
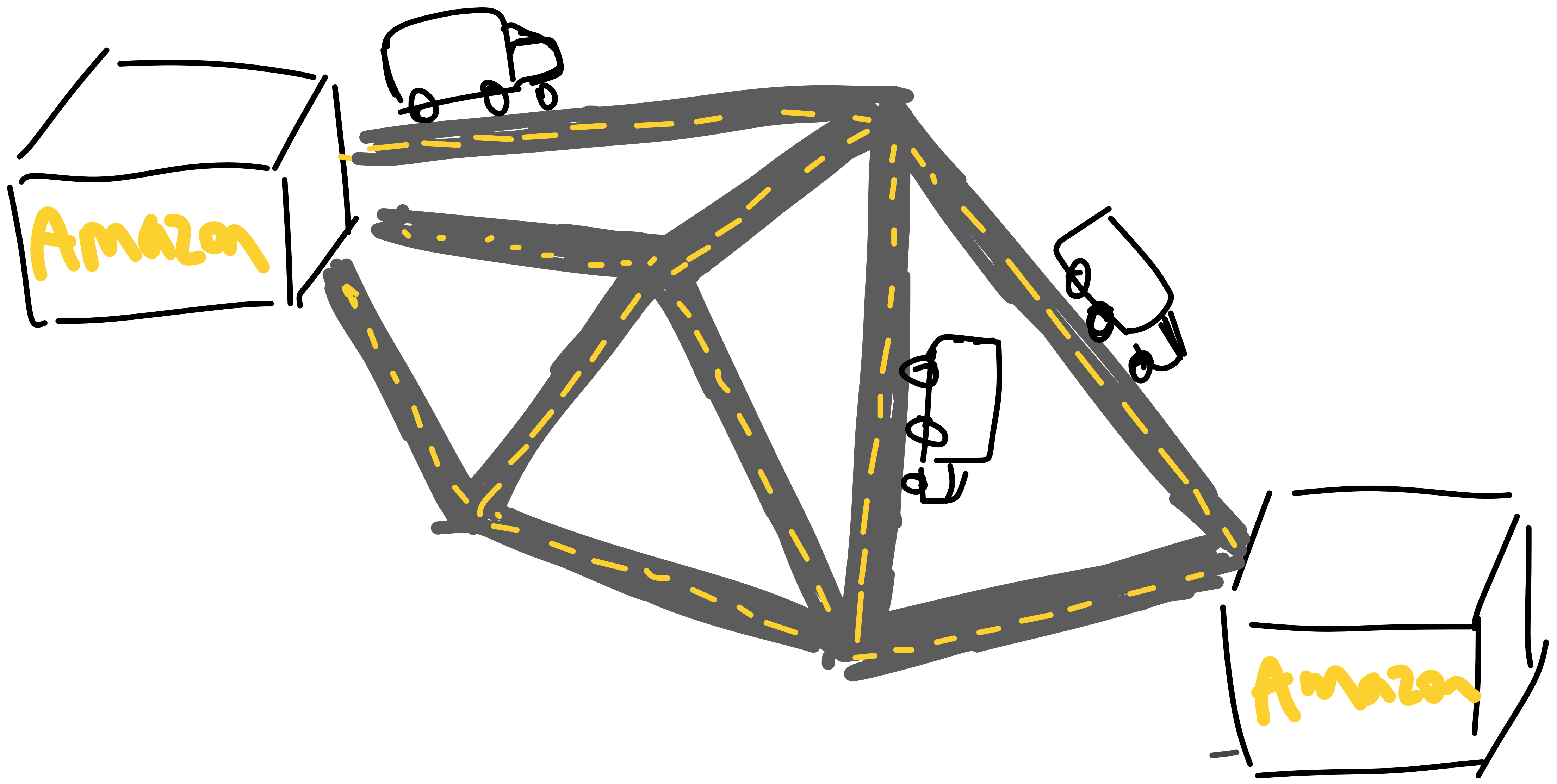
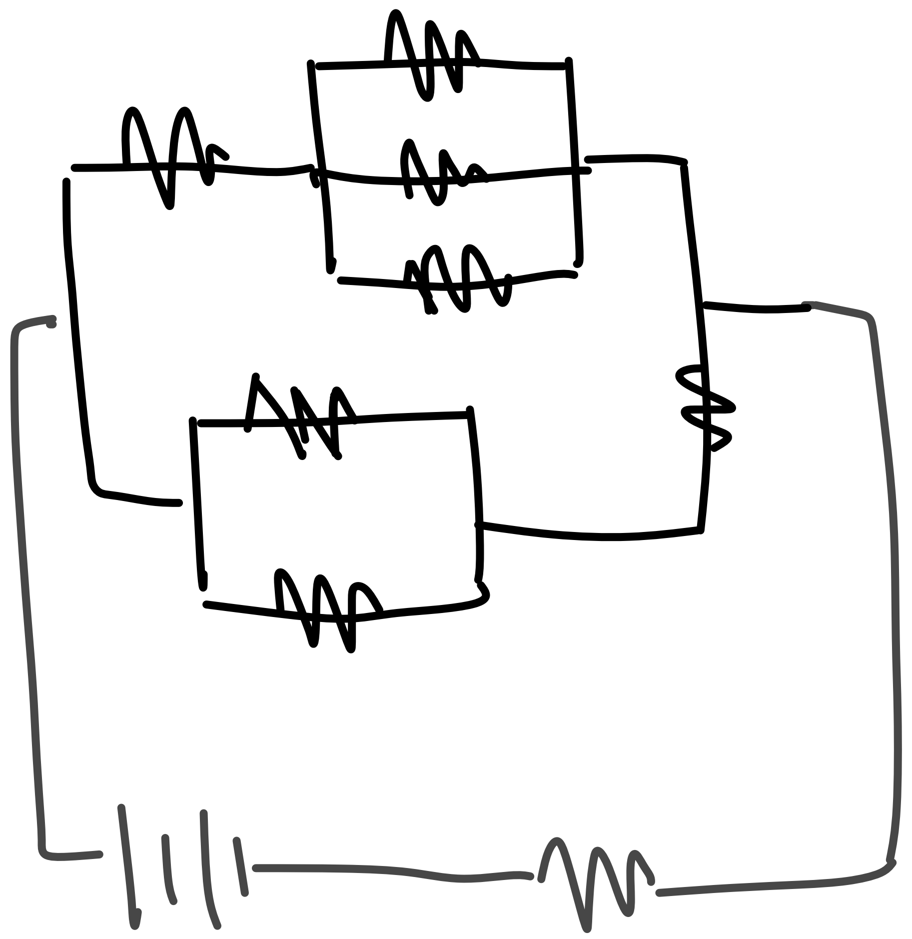
# Lecture 18: max flow



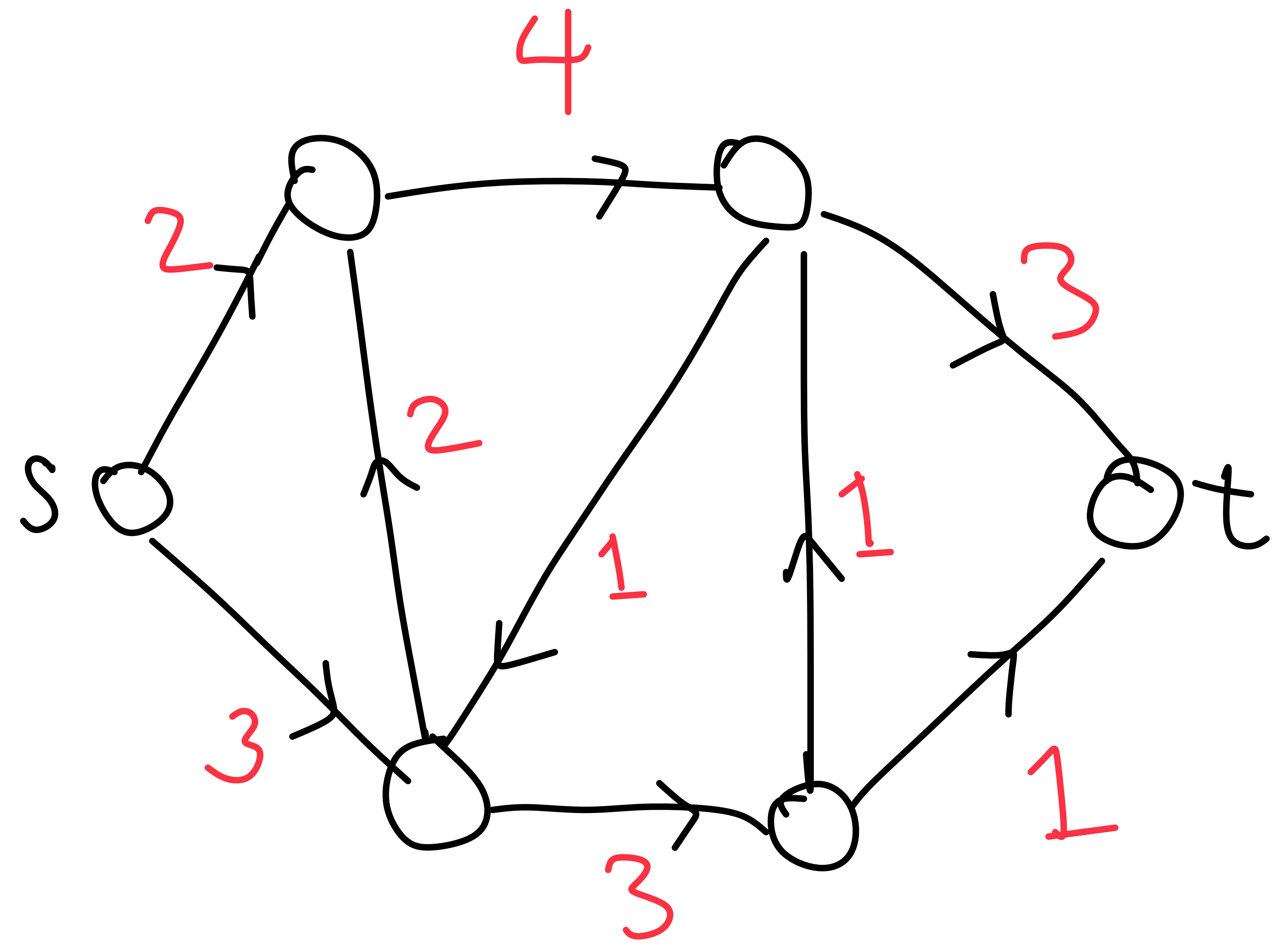


# Problem setup

Input:

- **directed** graph $G = (V, E)$ with special vertices $s$ (source) and $t$ (sink)
- edge capacities $\boldsymbol{c} = (c_e : e \in E) \geq \mathbf{0}$



# Problem setup

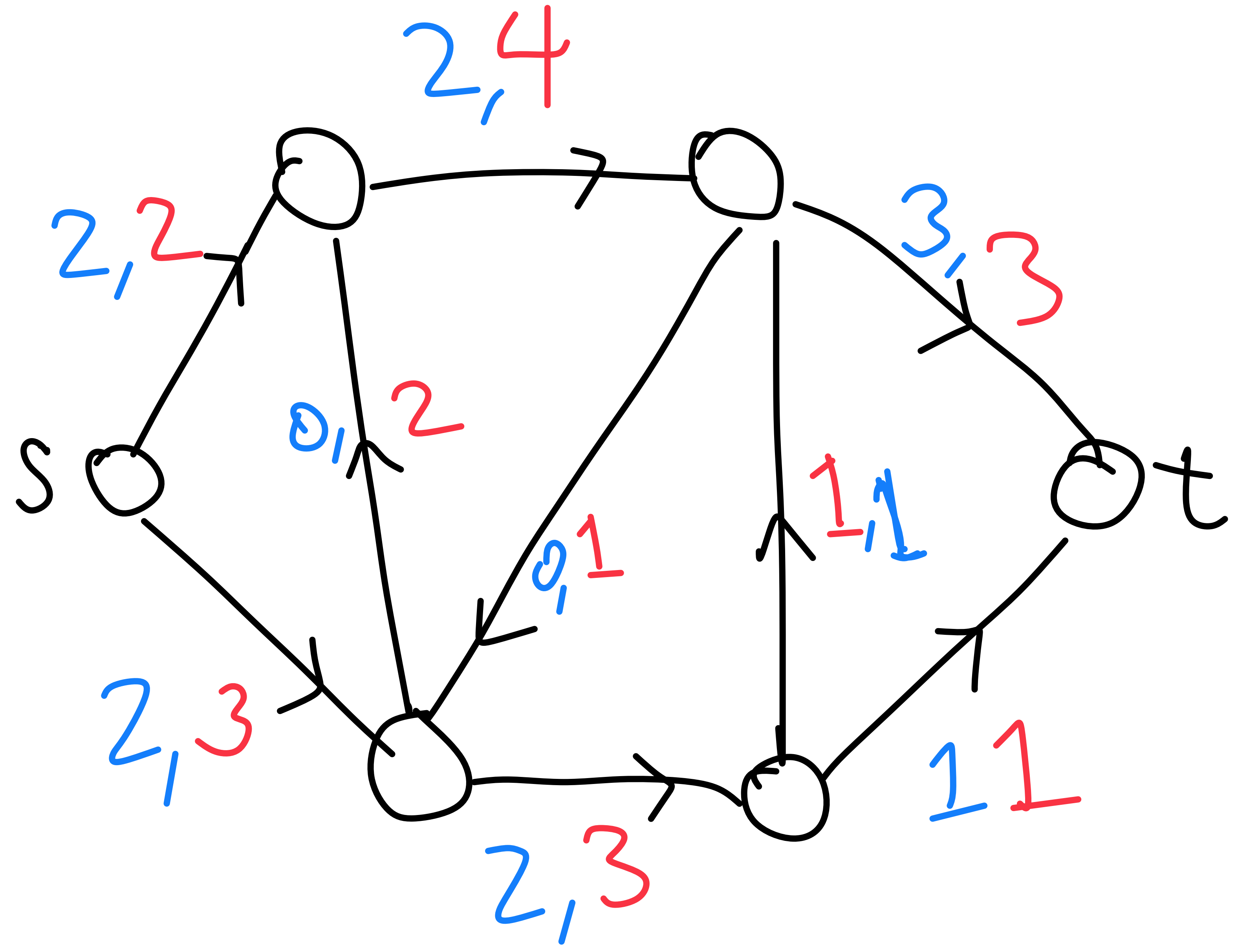
Input:

- **directed** graph $G = (V, E)$ with special vertices $s$ (source) and $t$ (sink)
- edge capacities $\boldsymbol{c} = (c_e : e \in E) \geq \mathbf{0}$

Output:

- *flow* $\boldsymbol{f} = (f_e : e \in E)$, i.e. satisfies
  - $0 \leq f_e \leq c_e$ for each edge $e$
  - $f^{in}(v) = f^{out}(v)$ for each vertex $v \neq s, t$
- maximize the *value* of the flow, i.e.

$$v(\boldsymbol{f}) := f^{out}(s) = f^{in}(t)$$



# Brief history of max flow

| year | authors | run-time bound |
|---|---|---|
| 1954 | Harris and Ross | first introduced to model Soviet railway flow |
| 1955 | Ford and Fulkerson | O(nmU) |
| 1970 | Dinitz, Edmond and Karp | $O(nm^2)$ |
| 1983 | Sleater and Tarjan | O(nm log n) |
| 1986 | Goldberg and Tarjan | $O(nm \log (n^2/m))$ |
| 1987 | Ahuja and Orlin | $O(nm + n^2 \log U)$ |
| 1997 | Goldberg and Rao | $O(m^{3/2} \log(n^2/m) \log U)$<br>$O(n^{2/3} m \log(n^2/m) \log U)$ |
| 2012 | Orlin, King et al. | $O(nm)$ |
| 2014 | Lee and Sidford | $O(m\sqrt{n} \log^{O(1)} U \log^{O(1)} n)$ |
| ⋮ | ⋮ | ⋮ |

m edges

n vertices

max edge capacity U

# Minimum Cost Flows, MDPs, and $\ell_1$-Regression in Nearly Linear Time for Dense Instances

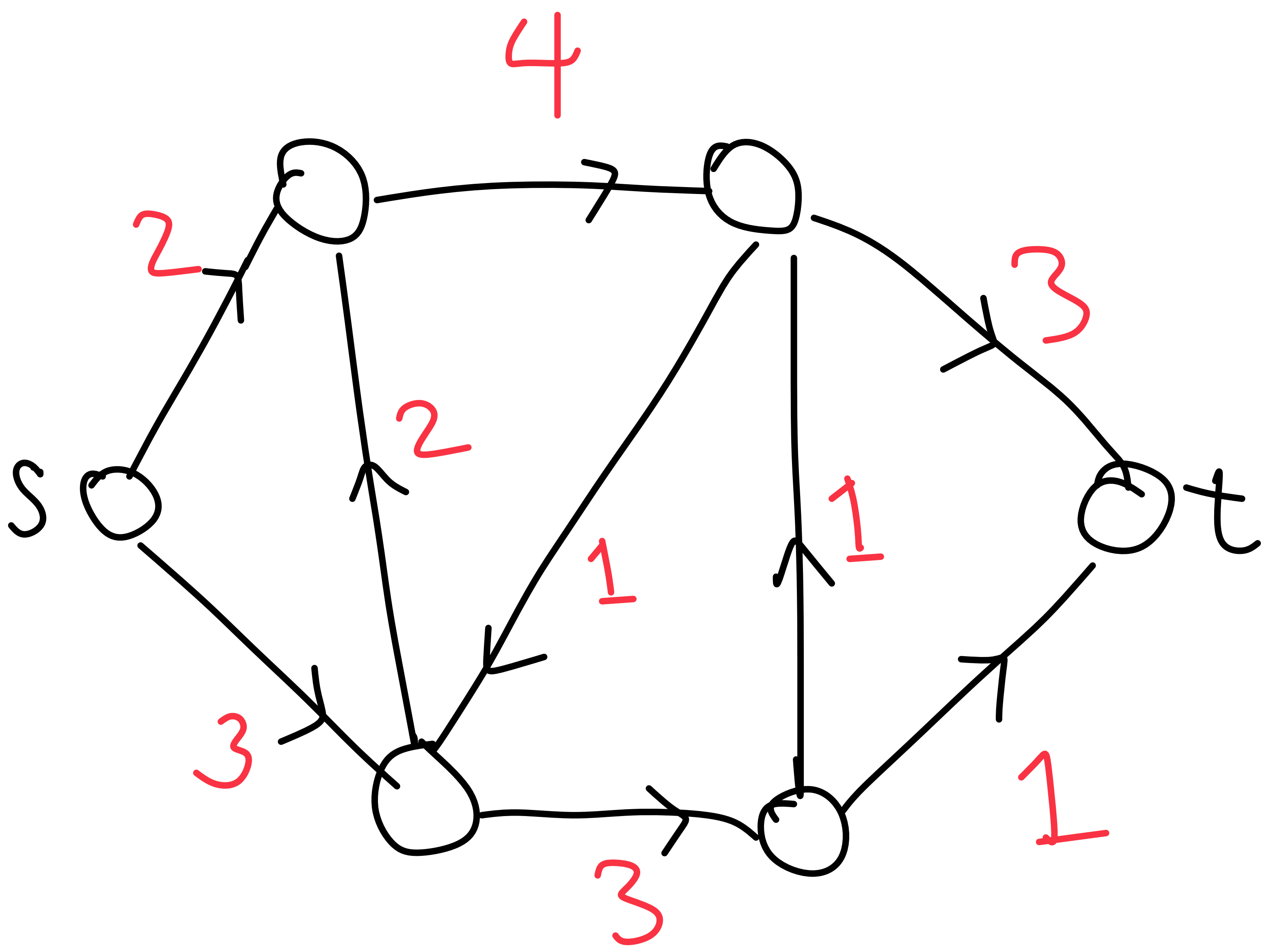
Jan van den Brand* Yin Tat Lee† Yang P. Liu‡ Thatchaphol Saranurak§
Aaron Sidford¶ Zhao Song‖ Di Wang**

August 24, 2021

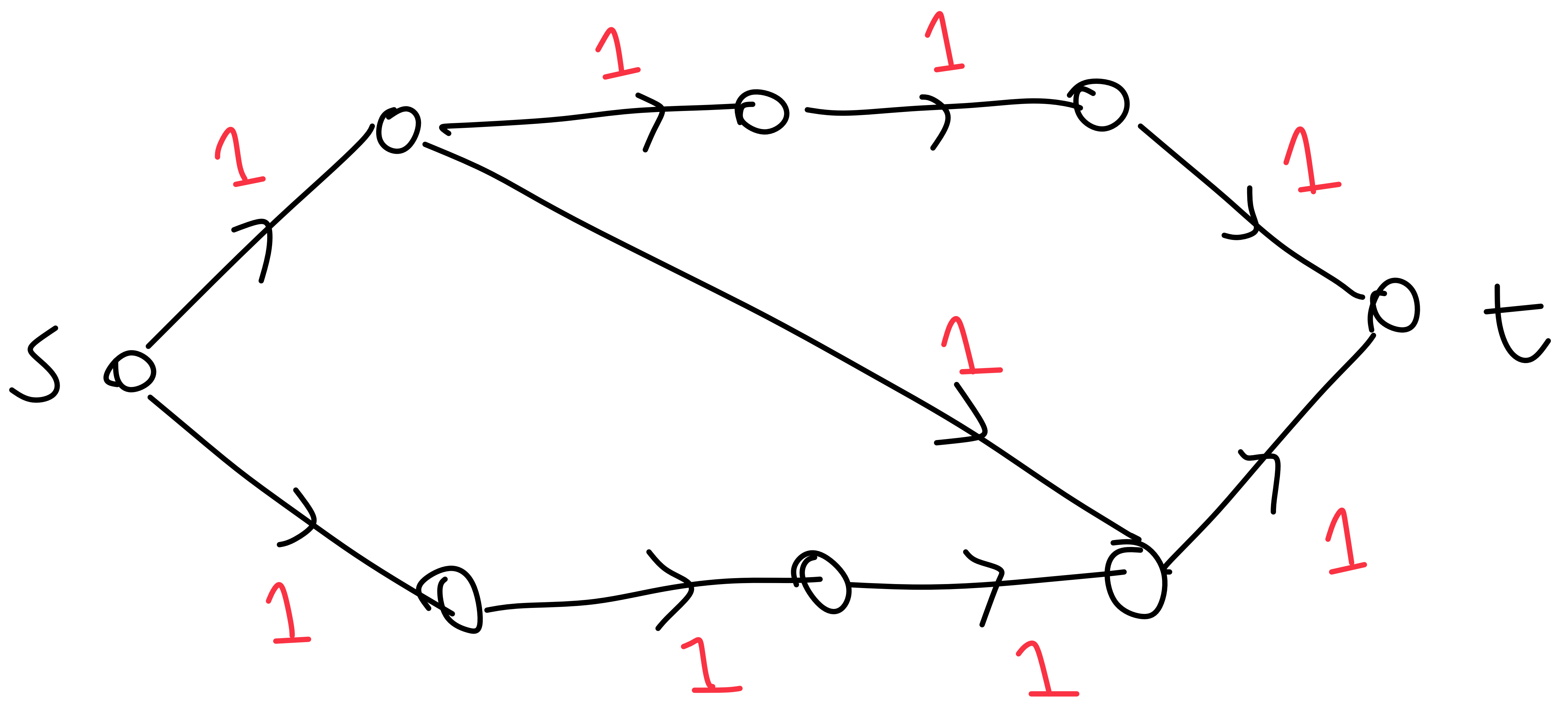
**Abstract**

In this paper we provide new randomized algorithms with improved runtimes for solving linear programs with two-sided constraints. In the special case of the minimum cost flow problem on $n$-vertex $m$-edge graphs with integer polynomially-bounded costs and capacities we obtain a randomized method which solves the problem in $\widetilde{O}(m+n^{1.5})$ time. This improves upon the previous best runtime of $\widetilde{O}(m\sqrt{n})$ [LS14] and, in the special case of unit-capacity maximum flow, improves upon the previous best runtimes of $m^{4/3+o(1)}$ [LS20a, Kat20] and $\widetilde{O}(m\sqrt{n})$ [LS14] for sufficiently dense graphs.

# Ideas for an approach?



# Greedy pitfalls



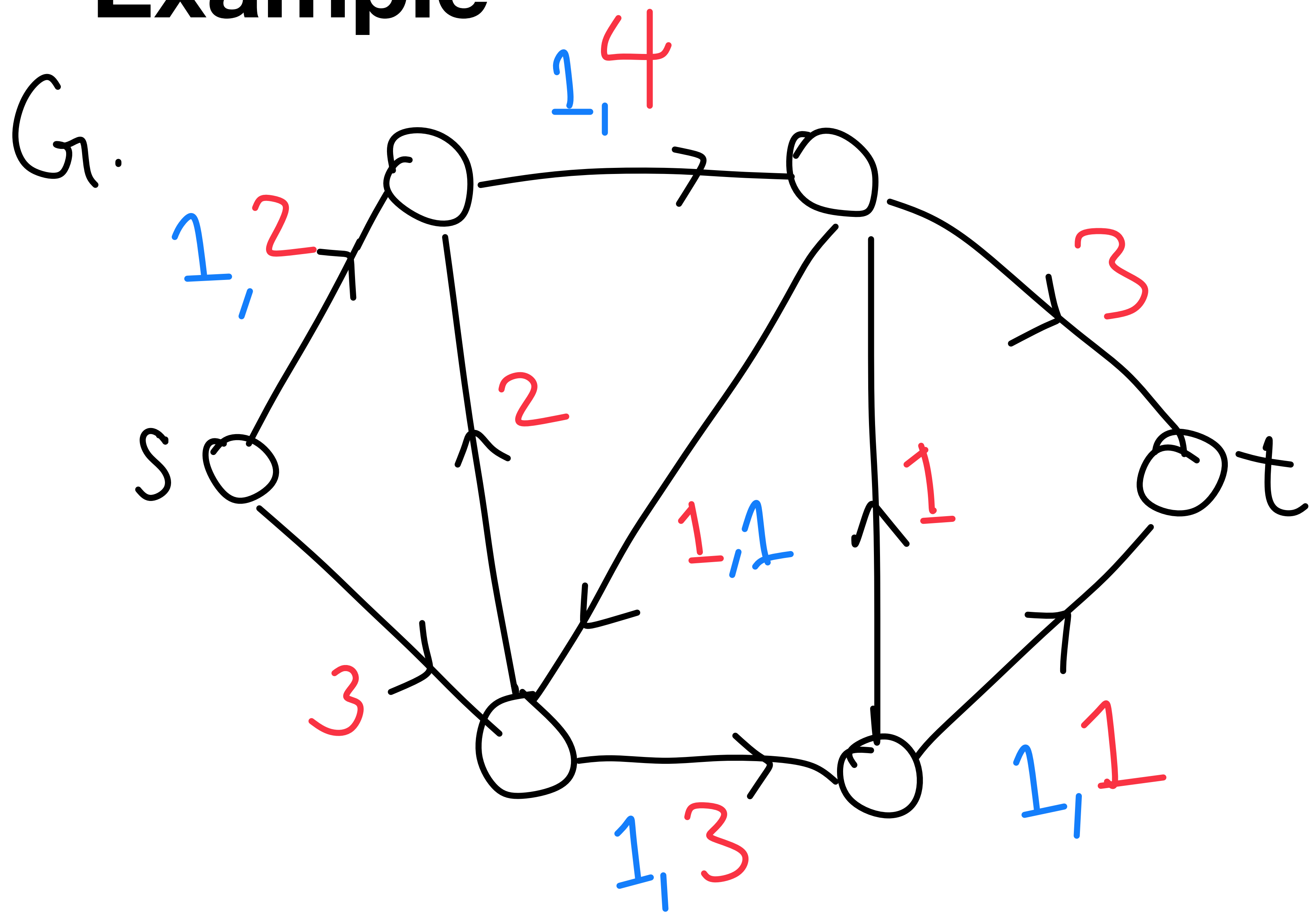
# Residual graph

- denote by $G_f$, depends on $G$ and $f$
- same set of vertices
- for every edge $e = (u, v)$ in $G$ with flow $f_e$, add
  - edge $(u, v)$ with capacity $c_e - f_e$, if $c_e > f_e$
  - edge $(v, u)$ with capacity $f_e$, if $f_e > 0$

# Residual graph

- denote by $G_f$, depends on $G$ and $f$
- same set of vertices
- for every edge $e = (u, v)$ in $G$ with flow $f_e$, add
  - edge $(u, v)$ with capacity $c_e - f_e$, if $c_e > f_e$
  - edge $(v, u)$ with capacity $f_e$, if $f_e > 0$

**Example**



# Augmenting path

**Definition:**

An *augmenting path* is an $s - t$ path in $G_f$.

# Augmenting path

**Definition:**

An *augmenting path* is an $s - t$ path in $G_f$.

We can send flow in $G$ along the augmenting path. This gives an updated flow.

Repeat this process? Until when?

# Ford-Fulkerson algorithm

start with $\boldsymbol{f} = \mathbf{0}$

**while true do**:

construct residual graph $G_f$

find an *augmenting path* $P$ *in* $G_f$, if none exists, **break**

update $f$ by sending as much flow as possible along $P$ in $G$

**return** $f$

# Ford-Fulkerson algorithm
## (more precise)

start with $\boldsymbol{f} = \mathbf{0}$

**while true do**:

  construct residual graph $G_{\boldsymbol{f}}$ with capacities $\boldsymbol{c}'$

  find an augmenting path $P$ in $G_{\boldsymbol{f}}$

    if none exists, **break**

  $\Delta = \min\{c'_e : e \in P\}$

  **for each** $e \in P$:

    **if** $e$ is a forward edge in $G$, set $f_e = f_e + \Delta$

    **else**, set $f_e = f_e - \Delta$

**return** $\boldsymbol{f}$

# Ford-Fulkerson algorithm example

start with $\boldsymbol{f} = \mathbf{0}$

**while true do**:

construct residual graph $G_{\boldsymbol{f}}$ with capacities $\boldsymbol{c}'$

find an augmenting path $P$ in $G_{\boldsymbol{f}}$

if none exists, **break**

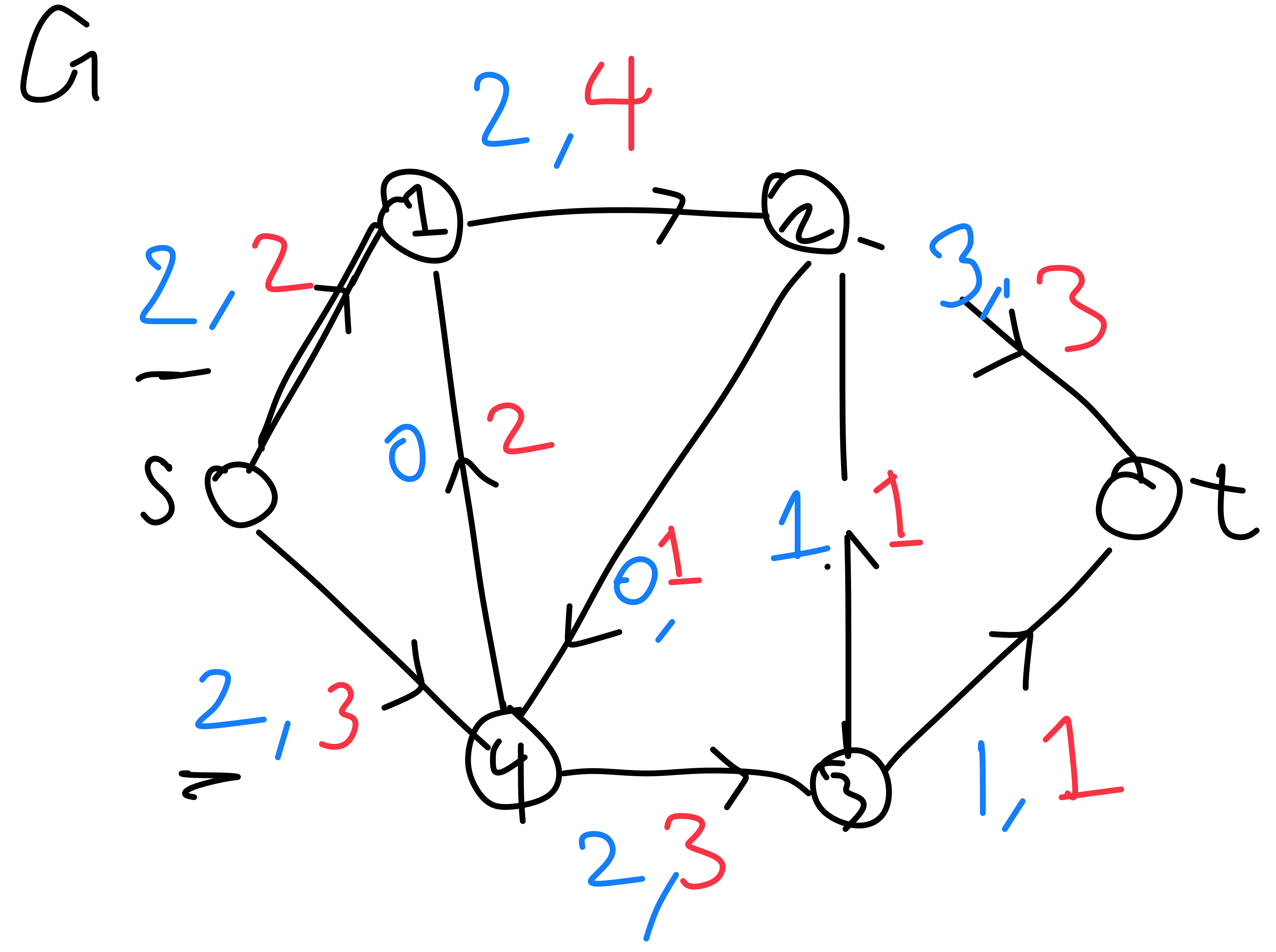
$\Delta = \min\{c'_e : e \in P\}$

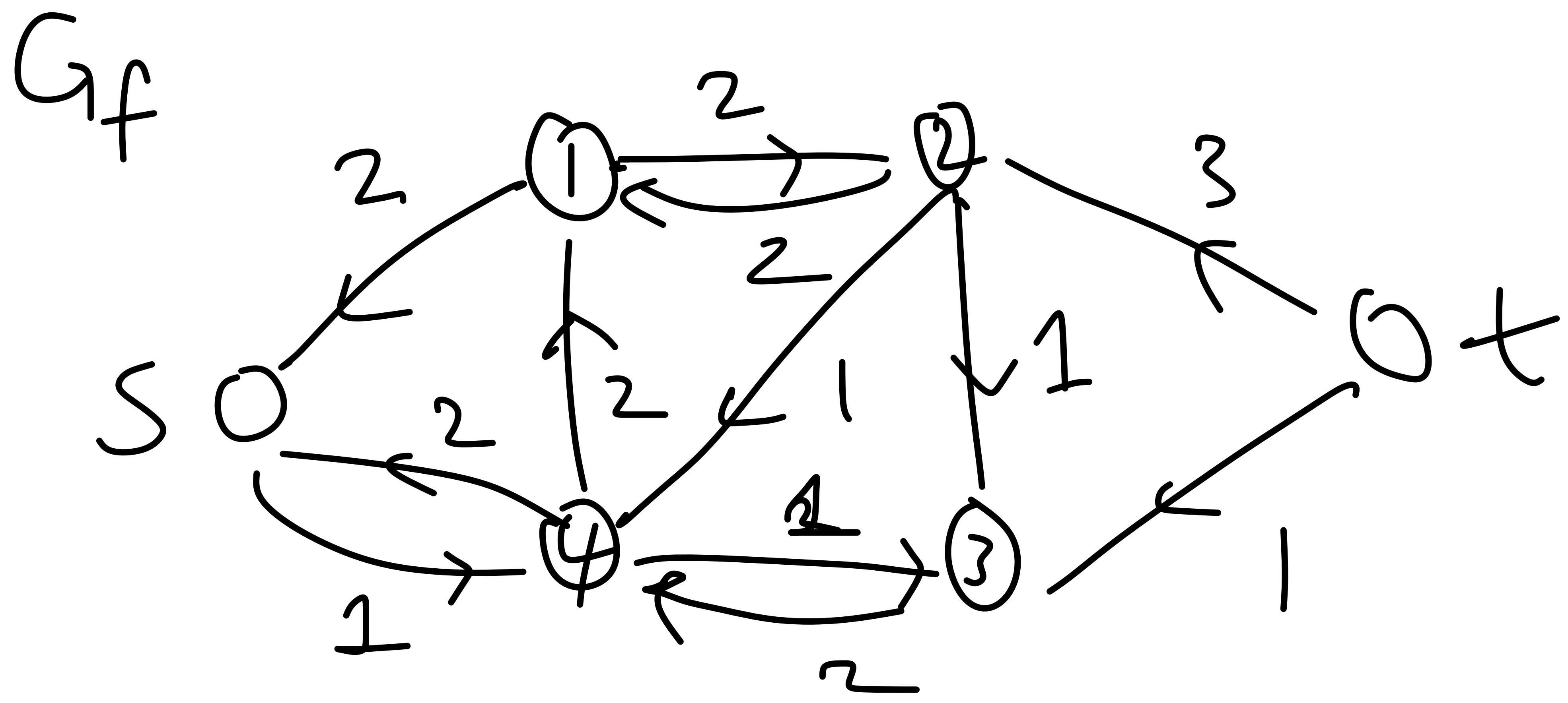
**for each** $e \in P$:

**if** $e$ is a forward edge in $G$, set $f_e = f_e + \Delta$

**else**, set $f_e = f_e - \Delta$

**return** $\boldsymbol{f}$



# Analysis of Ford-Fulkerson

- Termination
- Run-time
- Correctness

# Termination

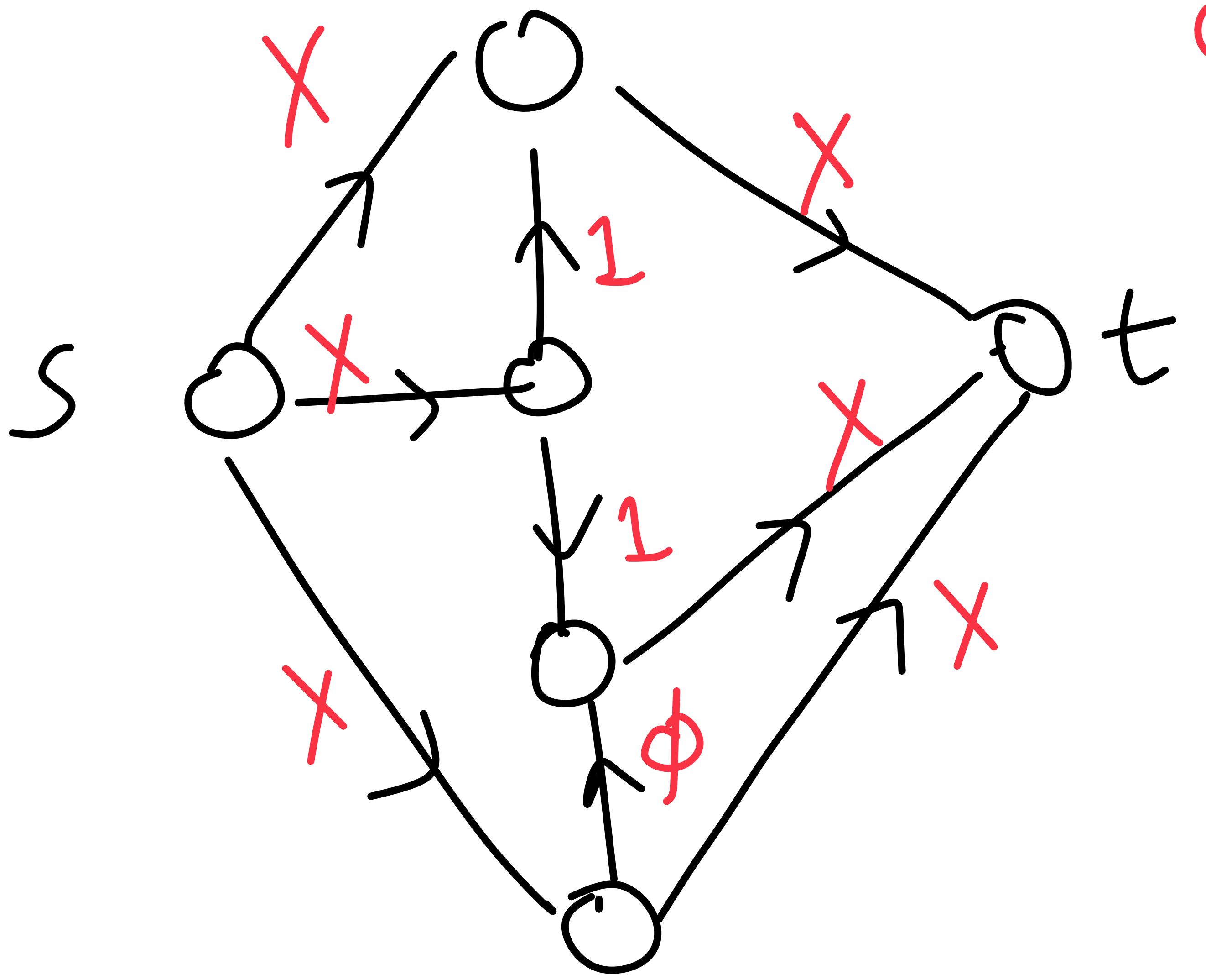
- Stop when there’s no augmenting path in the residual graph

# Termination

- Stop when there's no augmenting path in the residual graph
- Does this always happen? (seems natural)

# Termination

- Stop when there's no augmenting path in the residual graph
- Does this always happen? (seems natural) NO!!!




$\phi = \frac{\sqrt{5}-1}{2}$ (root of $\phi^2 + \phi - 1 = 0$)

there is a sequence of aug. paths with values $1, \phi, \phi, \phi^2, \phi^2, \phi^3, \ldots$

Total value $=$

$$1 + 2\sum_{i=1}^{\infty} \phi^i = 1 + \frac{2}{1-\phi} < 7.$$

max flow: $2X+1$.

# Run-time

## Assuming positive integer capacities

- Why do integer capacities help?

# Ford-Fulkerson run-time

## Assuming positive integer capacities

start with $f = \mathbf{0}$

**while true do**:

construct residual graph $G_f$ with capacities $\boldsymbol{c}'$

find an augmenting path $P$ in $G_f$

if none exists, **break**

$\Delta = \min\{c'_e : e \in P\}$

**for each** $e \in P$:

**if** $e$ is a forward edge in $G$, set $f_e = f_e + \Delta$

**else**, set $f_e = f_e - \Delta$

- $f$ is integral throughout the algorithm
- each loop, value of flow increases by integer $\Delta$

# Ford-Fulkerson run-time

## Assuming positive integer capacities

start with $\boldsymbol{f} = \mathbf{0}$

**while true do**:

construct residual graph $G_f$ with capacities $\boldsymbol{c}'$

find an augmenting path $P$ in $G_f$

if none exists, **break**

$\Delta = \min\{c'_e : e \in P\}$

**for each** $e \in P$:

**if** $e$ is a forward edge in $G$, set $f_e = f_e + \Delta$

**else**, set $f_e = f_e - \Delta$

- $\boldsymbol{f}$ is integral throughout the algorithm
- each loop, value of flow increases by integer $\Delta$
- can only loop OPT times

# Ford-Fulkerson run-time

## Assuming positive integer capacities

start with $f = \mathbf{0}$

**while true do**:

construct residual graph $G_f$ with capacities $\boldsymbol{c}'$

find an augmenting path $P$ in $G_f$

if none exists, **break**

$\Delta = \min\{c'_e : e \in P\}$

**for each** $e \in P$:

**if** $e$ is a forward edge in $G$, set $f_e = f_e + \Delta$

**else**, set $f_e = f_e - \Delta$

- $f$ is integral throughout the algorithm
- each loop, value of flow increases by integer $\Delta$
- can only loop OPT times
- each loop takes O(m) time

# Ford-Fulkerson run-time

## Assuming positive integer capacities

start with $f = \mathbf{0}$

**while true do**:

construct residual graph $G_f$ with capacities $c'$

find an augmenting path $P$ in $G_f$

if none exists, **break**

$\Delta = \min\{c'_e : e \in P\}$

**for each** $e \in P$:

**if** $e$ is a forward edge in $G$, set $f_e = f_e + \Delta$

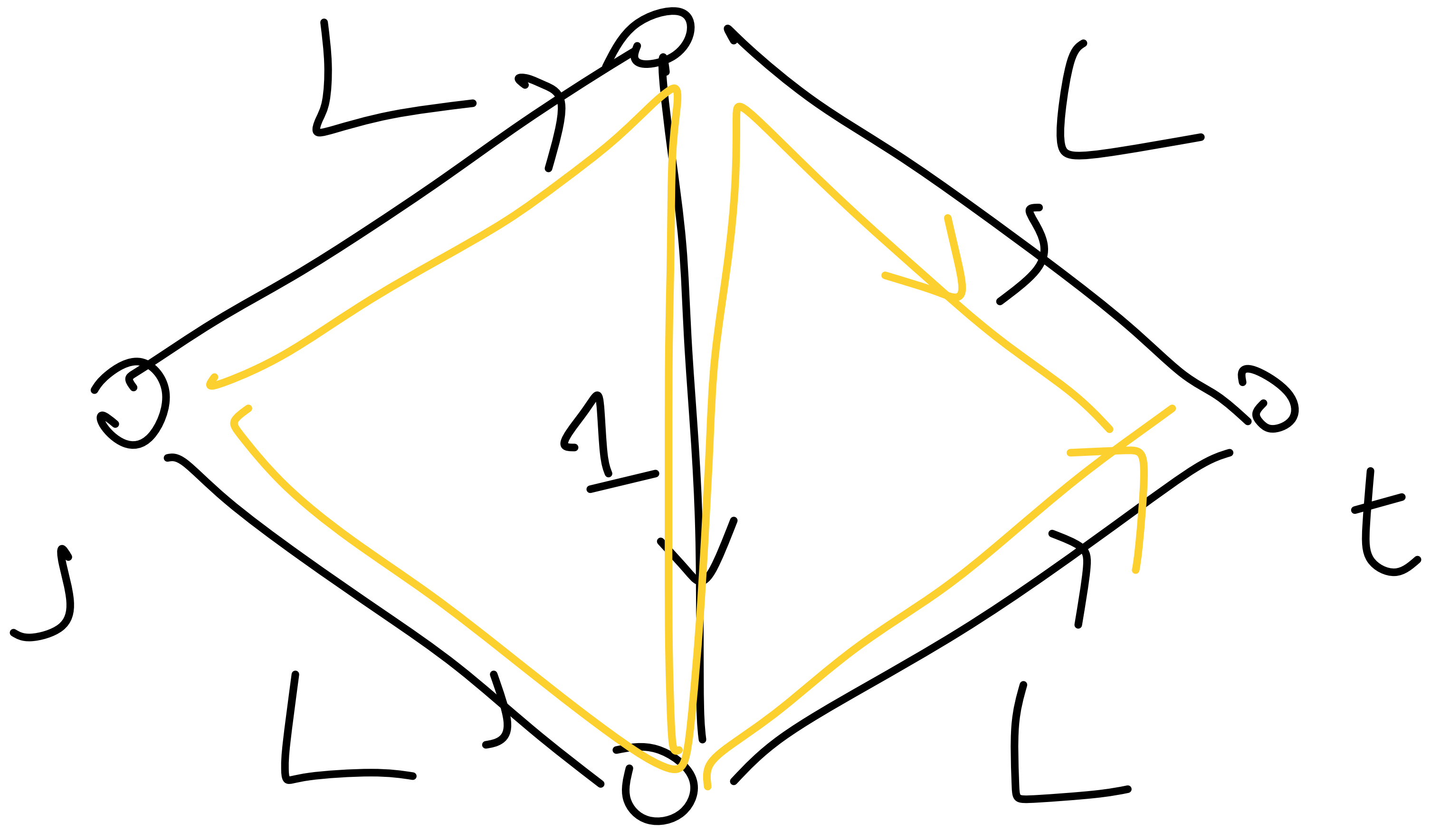
**else**, set $f_e = f_e - \Delta$

- $f$ is integral throughout the algorithm
- each loop, value of flow increases by integer $\Delta$
- can only loop OPT times
- each loop takes O(m) time
- total time: $O(m \cdot OPT)$

# Run-time

## Assuming positive integer capacities

- $O(m \cdot OPT)$ is… not very good. Can be exponential (in the size of input)




$O(m \cdot OPT)$ time $= O(m \cdot L)$

encode input with C bits + log L bits.

input size $= O(C + \log L)$

# Correctness

- Result is a valid flow
- The flow value is maximal

# Valid flow

start with $\boldsymbol{f} = \mathbf{0}$
**while true do**:
construct residual graph $G_{\boldsymbol{f}}$ with capacities $\boldsymbol{c}'$
find an augmenting path $P$ in $G_{\boldsymbol{f}}$
if none exists, **break**
$\Delta = \min\{c'_e : e \in P\}$
**for each** $e \in P$:
**if** $e$ is a forward edge in $G$, set $f_e = f_e + \Delta$
**else**, set $f_e = f_e - \Delta$
**return** $\boldsymbol{f}$

**Claim:**

Flow is valid at the end of each loop.

# Valid flow

start with $\boldsymbol{f} = \mathbf{0}$
**while true do**:
construct residual graph $G_{\boldsymbol{f}}$ with capacities $\boldsymbol{c}'$
find an augmenting path $P$ in $G_{\boldsymbol{f}}$
if none exists, **break**
$\Delta = \min\{c'_e : e \in P\}$
**for each** $e \in P$:
**if** $e$ is a forward edge in $G$, set $f_e = f_e + \Delta$
**else**, set $f_e = f_e - \Delta$
**return** $\boldsymbol{f}$

**Claim:**

Flow is valid at the end of each loop.

**Proof:**

# Optimality

**Definitions:**

An $s - t$ cut of the graph $G = (V, E)$ is a partition of $V$ into 2 sets $A, B$ so that $s \in A, t \in B$ .

The capacity of the cut is $c(A, B) = \sum_{e=(u,v), u \in A, v \in B} c_e$.

# Optimality

**Definitions:**

Let $f$ be any flow. For a subset of vertices $A \subseteq V$, define

$$f^{in}(A) = \sum_{e=(u,v), u \notin A, v \in A} f_e, \text{ and } f^{out}(A) = \sum_{e=(u,v), u \in A, v \notin A} f_e = f^{in}(V \backslash A)$$

For any cut $(A, B)$, define the *net flow across the cut* as

$$f(A, B) = f^{out}(A) - f^{in}(A).$$

# Optimality

**Lemma 1:**

For *any* flow $f$ and *any* cut $(A, B)$, we have

$$v(f) = f(A, B) := f^{out}(A) - f^{in}(A).$$

**Lemma 2:**
The net flow across the cut cannot exceed the capacity of the cut, i.e.

$$f^{(out)}(A) - f^{(in)}(A) \leq c(A, B).$$

**Corollary:**

$$v(f) \leq c(A, B).$$

# Optimality

**Lemma 1:**

For *any* flow $f$ and *any* cut $(A, B)$, we have

$$v(f) = f(A, B) := f^{out}(A) - f^{in}(A).$$

**Proof:**

# Optimality

**Lemma 1:**

For *any* flow $f$ and *any* cut $(A, B)$, we have

$$v(\boldsymbol{f}) = f(A, B) := f^{out}(A) - f^{in}(A).$$

**Lemma 2:**

The net flow across the cut cannot exceed the capacity of the cut, i.e.

$$f^{(out)}(A) - f^{(in)}(A) \leq c(A, B).$$

**Proof:**

# Optimality

**Lemma 1:**

For *any* flow $f$ and *any* cut $(A, B)$, we have

$$v(f) = f(A, B) := f^{out}(A) - f^{in}(A).$$

**Lemma 2:**

The net flow across the cut cannot exceed the capacity of the cut, i.e.

$$f^{(out)}(A) - f^{(in)}(A) \leq c(A, B).$$

**Corollary:**

$$v(f) \leq c(A, B).$$

# Optimality

**Corollary:**

For *any* flow $f$ and *any* cut $(A, B)$,

$$v(f) \leq c(A, B).$$

In particular,

$$\max_{f} v(f) \leq \min_{(A,B)} c(A, B).$$

# Optimality

**Corollary:**

For *any* flow $f$ and *any* cut $(A, B)$,

$$v(f) \leq c(A, B).$$

In particular, $$\max_f v(f) \leq \min_{(A,B)} c(A, B).$$

Goal: To show our solution $f$ is optimal, find a cut $(A, B)$ where $v(f) = c(A, B)$.

# Optimality

**Lemma 3:**

Let $f$ be the flow returned by Ford-Fulkerson. Let $A$ be the set of vertices reachable from $s$ in $G_f$, and let $B = V \backslash A$ . Then $v(f) = c(A, B)$.

**Proof:**

# Optimality

**Lemma 3:**

Let $f$ be the flow returned by Ford-Fulkerson. Let $A$ be the set of vertices reachable from $s$ in $G_f$, and let $B = V \backslash A$. Then $v(f) = c(A, B)$.

**Corollary:**

Ford-Fulkerson is correct, and max flow = min cut.