Flooding Overcomes Small Covering Constraints

Christos Koufogiannakis and Neal E. Young
University of California, Riverside

Covering Integer Programs

$$\begin{aligned} \min \quad & c \cdot x \\ \text{subject to:} \quad & Ax \geq b \\ & x \leq u \\ & x \in Z_+^n \end{aligned}$$

Let δ be the maximum number of variables per constraint (i.e. δ = 2 for Vertex Cover).

Covering Integer Programs

$$\begin{aligned} \min \ x_1 + 0.5x_2 + 6x_3 : & \\ 2x_1 + x_2 &\geq 3 \\ x_2 + 4x_3 &\geq 5 \\ x_2 &\leq 2 \\ x_1, x_2, x_3 &\in Z_+ \end{aligned}$$

Let δ be the maximum number of variables per constraint (δ = 2 for this example).

Covering Integer Programs

$$\min \; x_1 + 0.5x_2 + 6x_3 :$$

$$2x_1 + x_2 \geq 3$$

$$x_2 + 4x_3 \geq 5$$

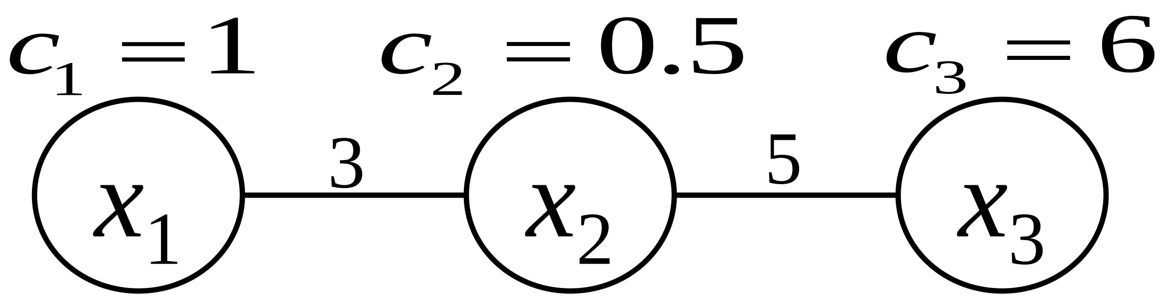
$$x_2 \leq 2$$

$$x_1, x_2, x_3 \in Z_+$$

Let δ be the maximum number of variables per constraint ($\delta = 2$ for this example).

Goal: Simple δ-approximation algorithm, applicable in the *sequential*

Covering Integer Programs



$$\min\ x_1 + 0.5x_2 + 6x_3 :$$

$$2x_1 + x_2 \geq 3$$

$$x_2 + 4x_3 \geq 5$$

$$x_2 \leq 2$$

$$x_1, x_2, x_3 \in Z_+$$

Let δ be the maximum number of variables per constraint (δ = 2 for this example).

Goal: Simple δ-approximation algorithm, applicable in the sequential, *distributed*

Covering Integer Programs

$$\min\ x_1 + 0.5x_2 + 6x_3 :$$

$$x_2 \le 2$$

$$x_1, x_2, x_3 \in Z_+$$

Let δ be the maximum number of variables per constraint (δ = 2 for this example).

Goal: Simple δ-approximation algorithm, applicable in the sequential, distributed and *online* setting.

Covering Integer Programs

$$\begin{aligned} \min \; & x_1 + 0.5x_2 + 6x_3 : \\ & 2x_1 + x_2 \geq 3 \\ & \\ & x_2 \leq 2 \\ & x_1, x_2, x_3 \in Z_+ \end{aligned}$$

Let δ be the maximum number of variables per constraint (δ = 2 for this example).

Goal: Simple δ-approximation algorithm, applicable in the sequential, distributed and *online* setting.

Covering Integer Programs

$$\min \ x_1 + 0.5x_2 + 6x_3 :$$

$$2x_1 + x_2 \geq 3$$

$$x_2 + 4x_3 \geq 5$$

$$x_2 \leq 2$$

$$x_1, x_2, x_3 \in Z_+$$

Let δ be the maximum number of variables per constraint (δ = 2 for this example).

Goal: Simple δ-approximation algorithm, applicable in the sequential, distributed and *online* setting.

Covering Integer Programs

$$
\begin{aligned}
\min \quad & x_1 + 0.5x_2 + 6x_3 : \\
& 2x_1 + x_2 \geq 3 \\
& x_2 + 4x_3 \geq 5 \\
& x_2 \leq 2 \\
& x_1, x_2, x_3 \in Z_+
\end{aligned}
$$

Previous sequential δ-approximation algorithms use the KC-inequalities and an ellipsoid-based method [Carr et al., 2000][Pritchard, 2009].

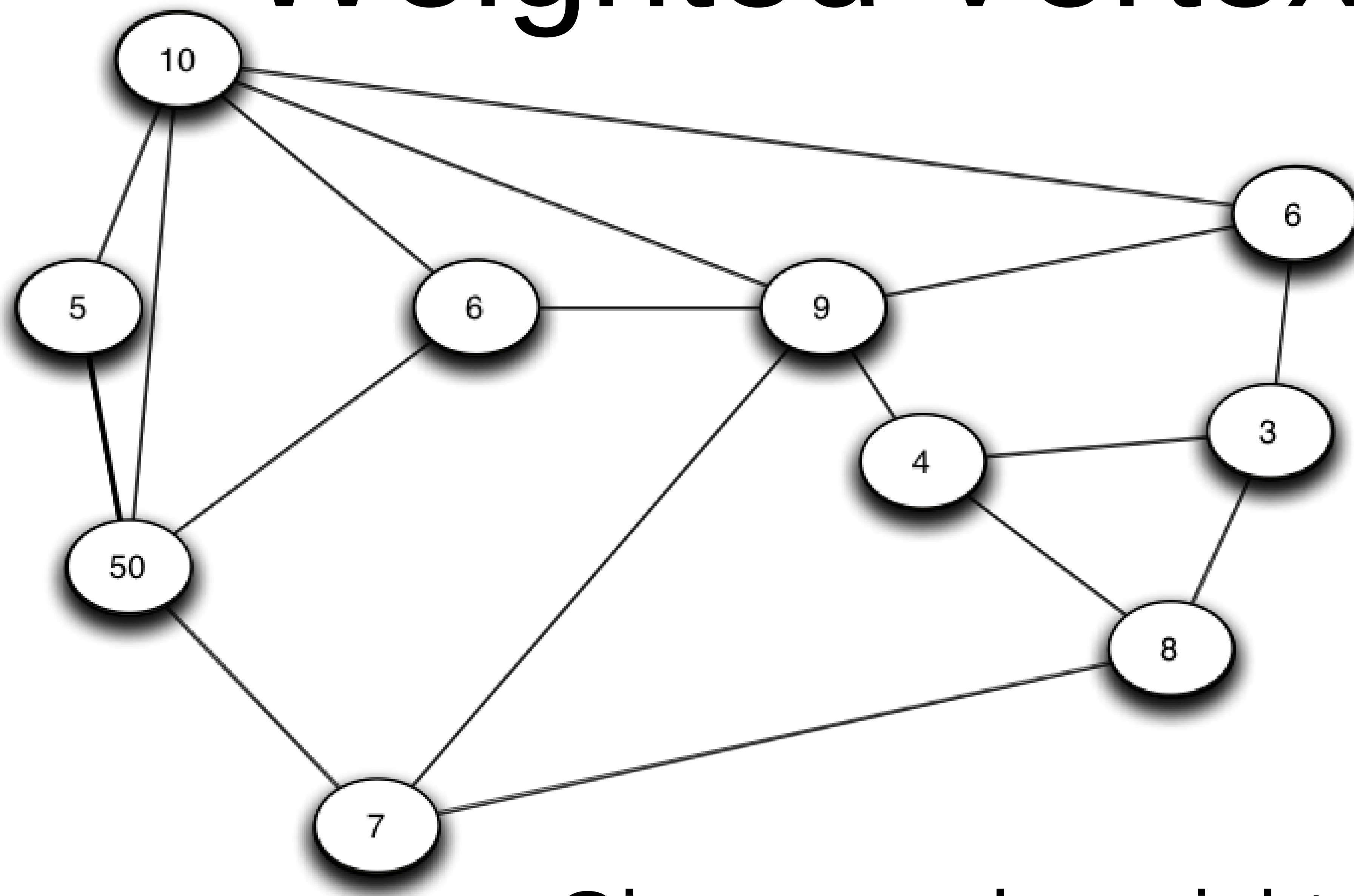
Covering Integer Programs

$$\begin{aligned} \min\ x_1 + 0.5x_2 + 6x_3 :& \\ 2x_1 + x_2 &\geq 3 \\ x_2 + 4x_3 &\geq 5 \\ x_2 &\leq 2 \\ x_1, x_2, x_3 &\in Z_+ \end{aligned}$$

Previous sequential δ-approximation algorithms use the KC-inequalities and an ellipsoid-based method [Carr et al., 2000][Pritchard, 2009].

Slow, not suitable for the distributed or online setting.

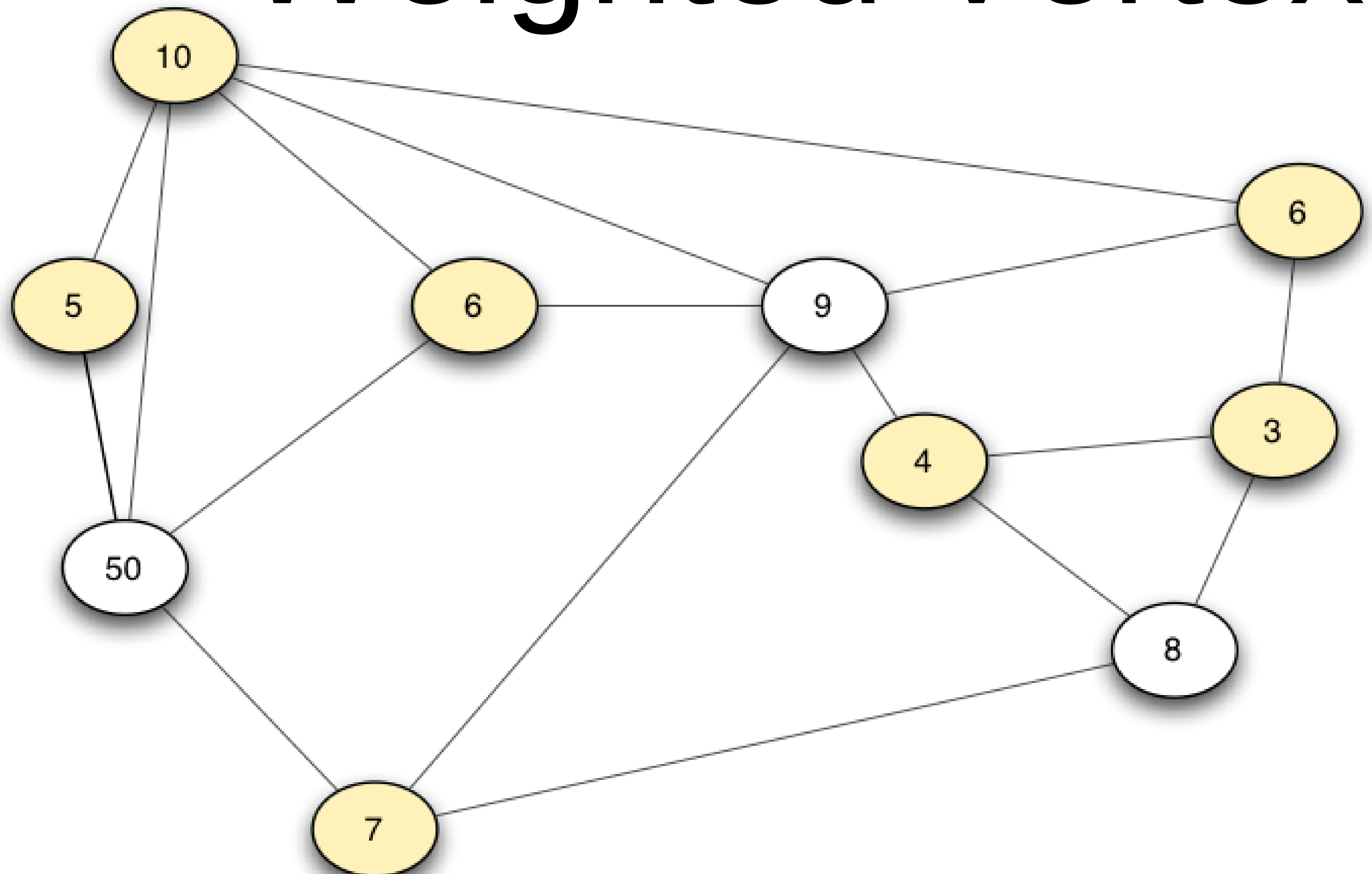
Weighted Vertex Cover



$$\min \; c \cdot x :$$
$$x_u + x_v \geq 1 \quad \forall (u,v)$$
$$x \in \{0,1\}^n$$

Given a node-weighted graph, find a minimum-weight subset of the nodes, touching all the edges.

Weighted Vertex Cover



$$\min \; c \cdot x :$$
$$x_u + x_v \geq 1 \quad \forall (u,v)$$
$$x \in \{0,1\}^n$$

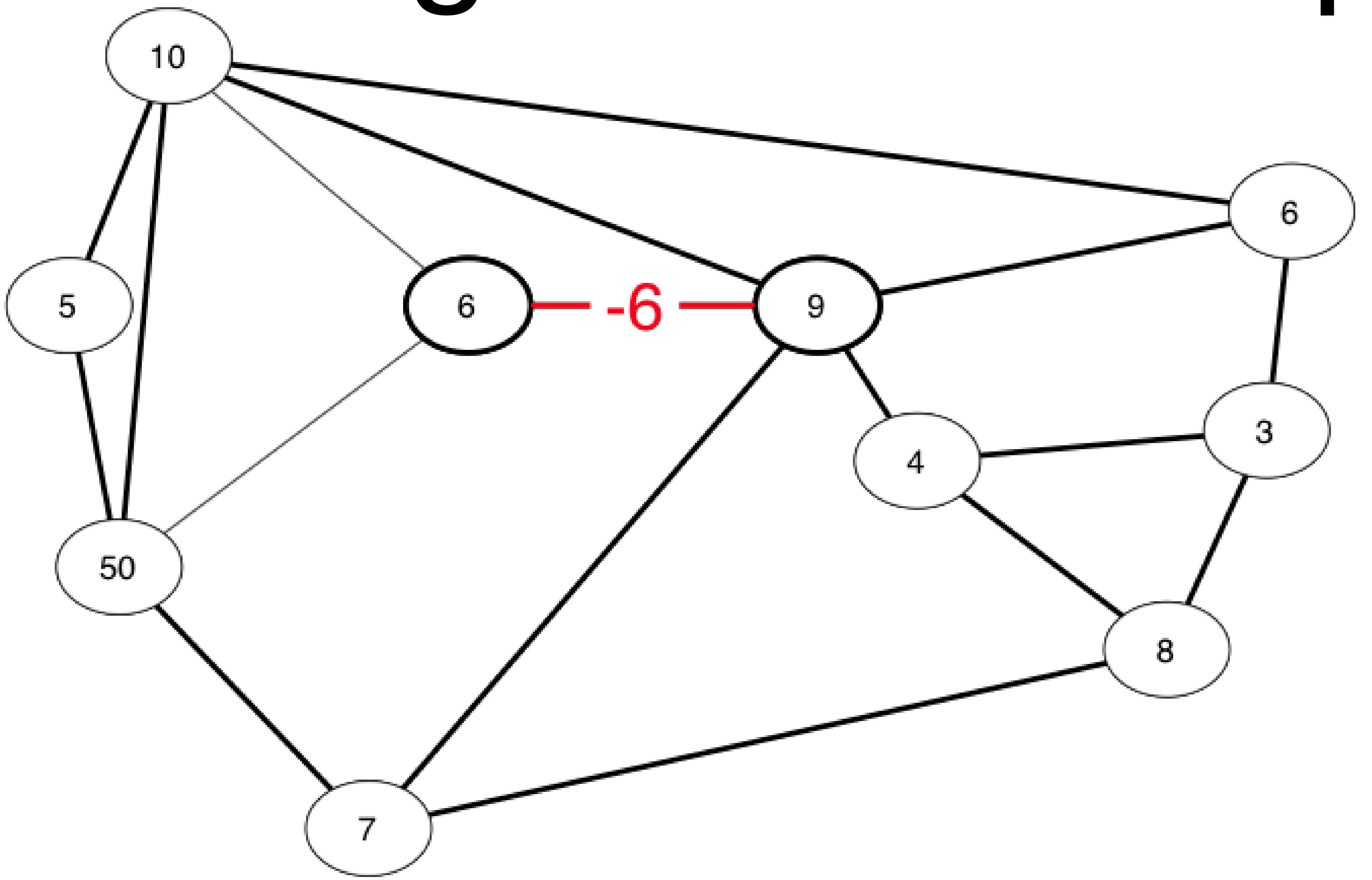
Given a node-weighted graph, find a minimum-weight subset of the nodes, touching all the edges.

A sequential and online 2-approximation algorithm for Weighted VC

- “Edge discount”: reduce edge endpoints’ costs equally.
- Do edge discounts until zero-cost nodes form a cover. Return the cover formed by the zero-cost nodes.

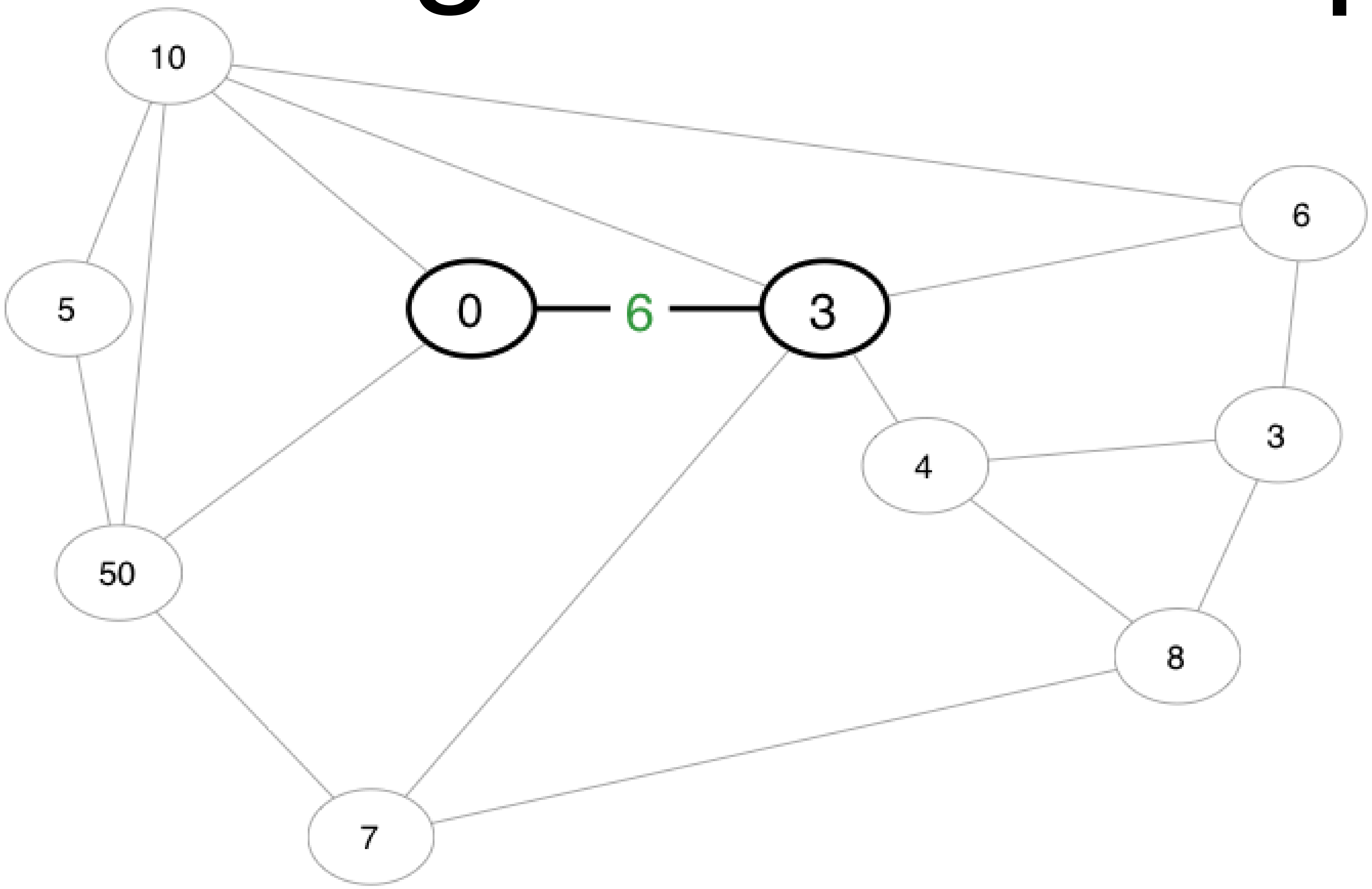
[Bar-Yehuda and Even, 1981]

edge discount operation



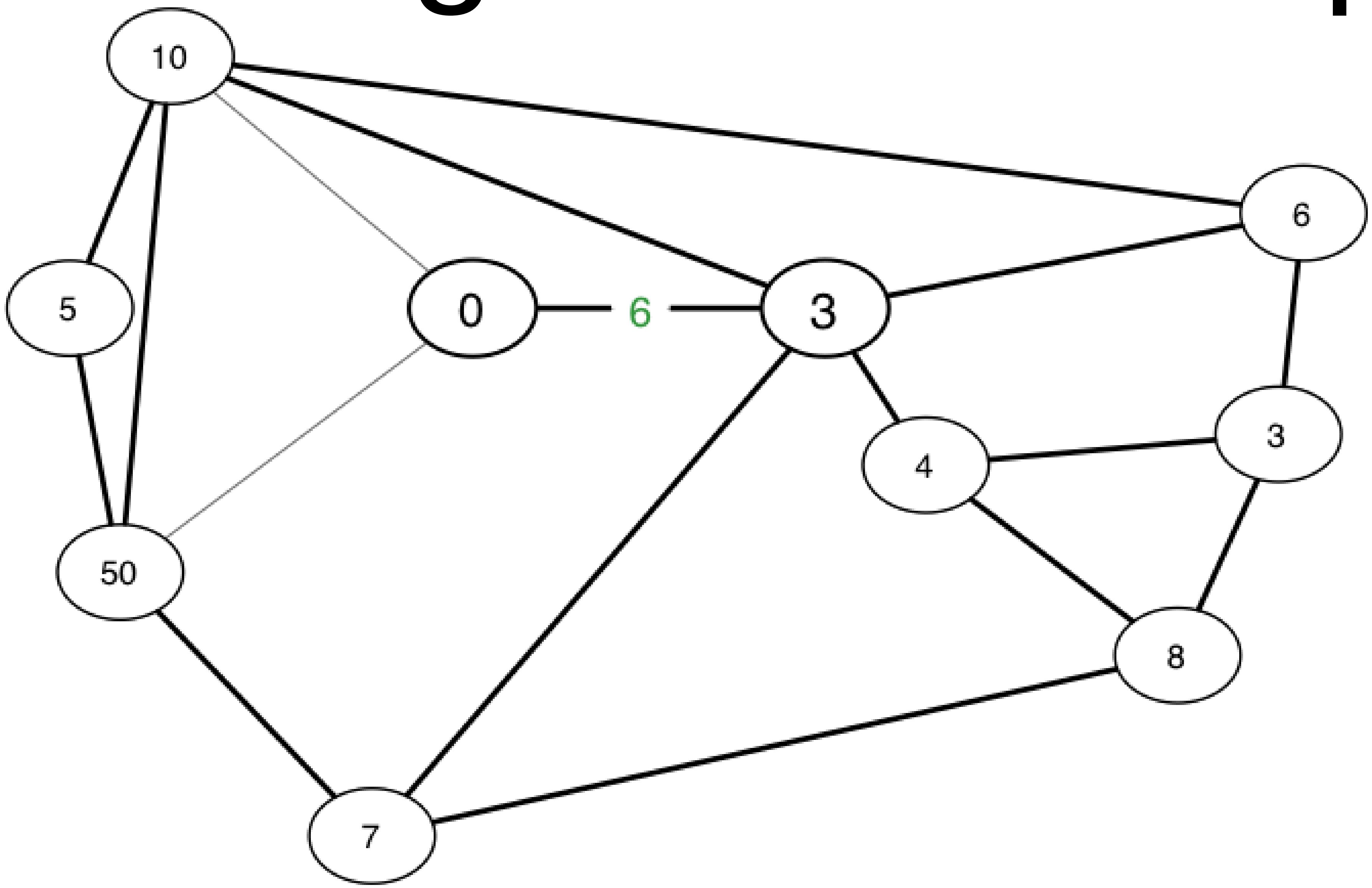
Reduce both endpoints' costs equally.

edge discount operation

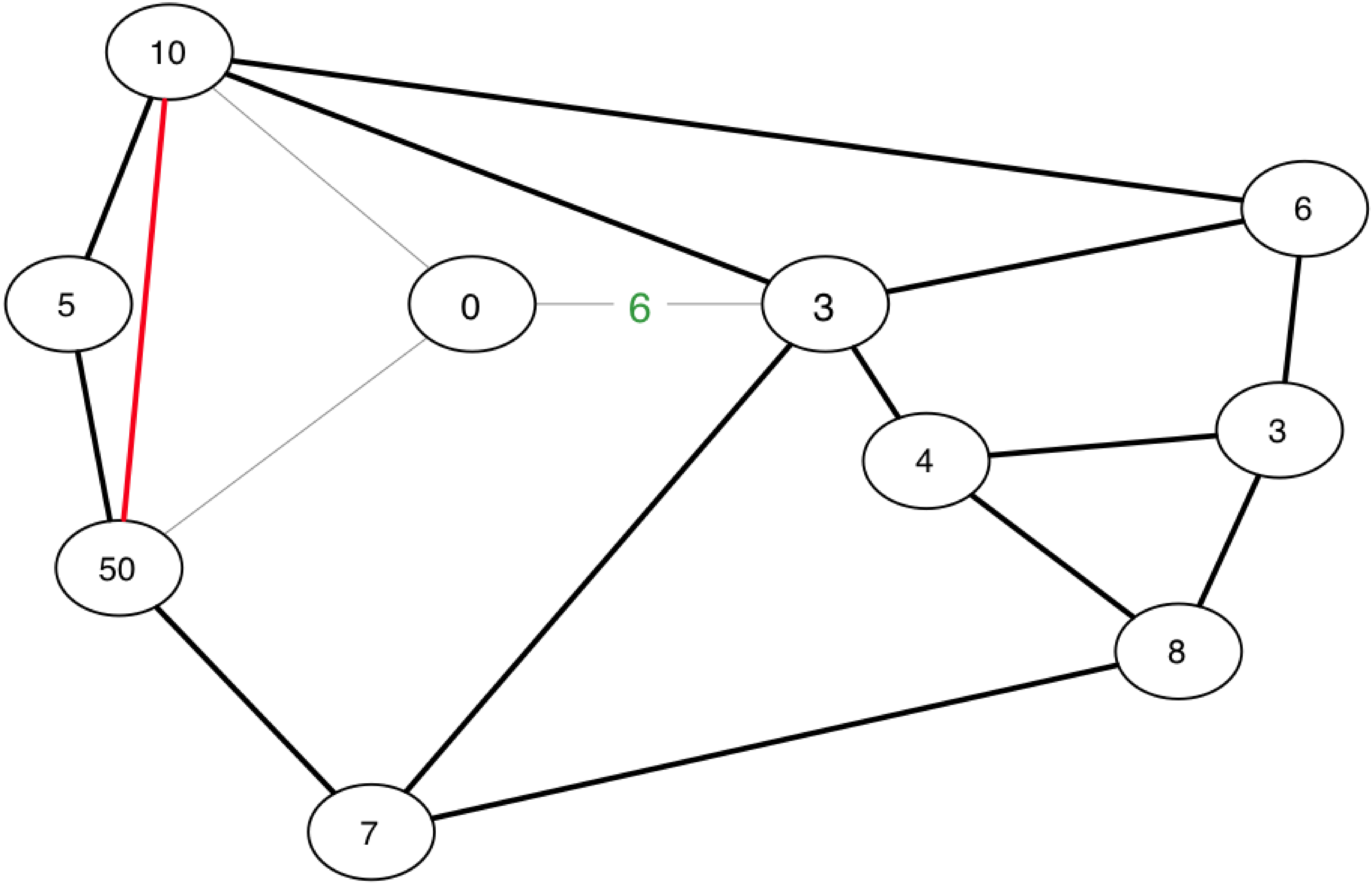


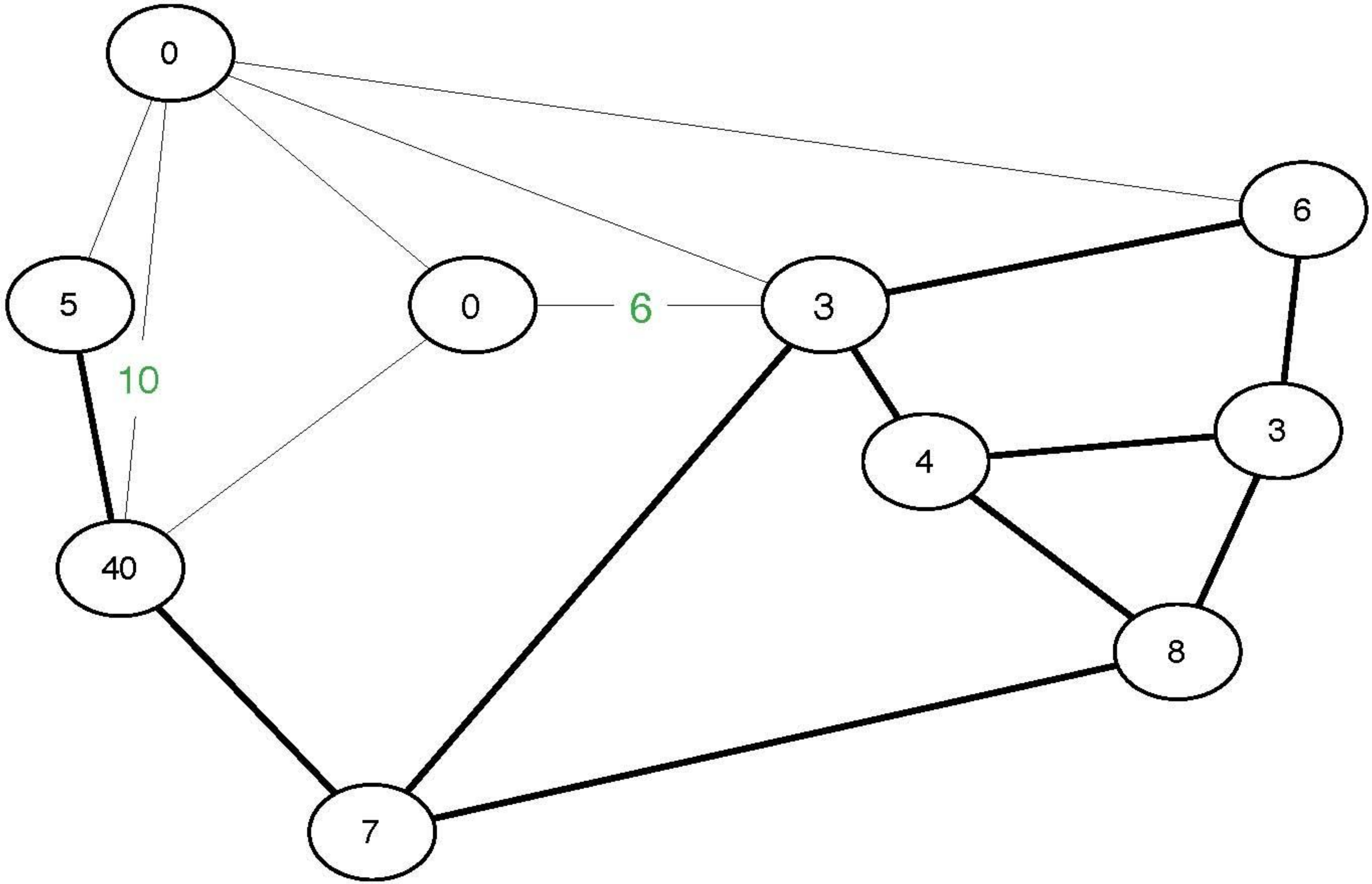
Reduce both endpoints' costs equally.

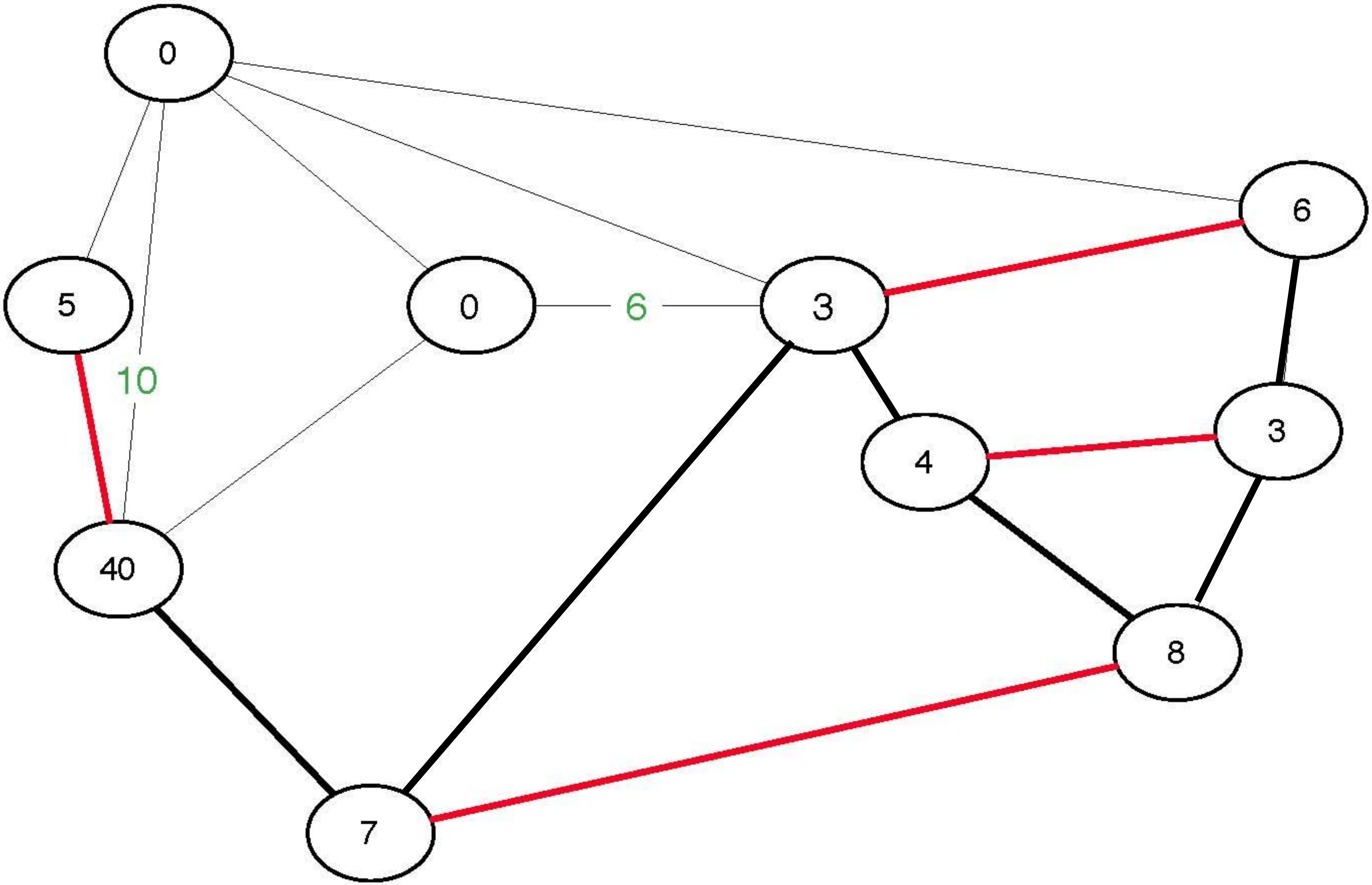
edge discount operation



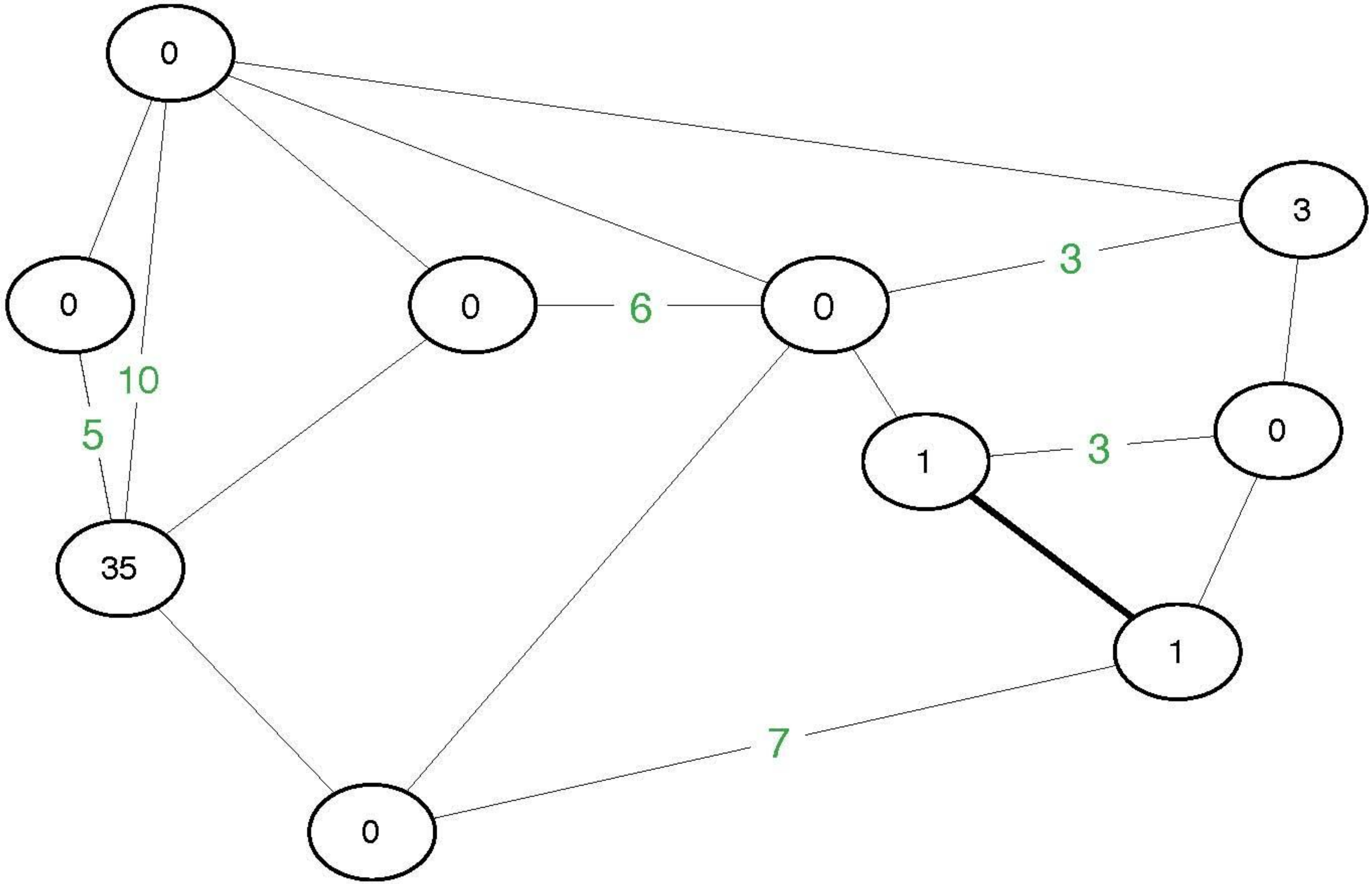
Reduce both endpoints' costs equally.

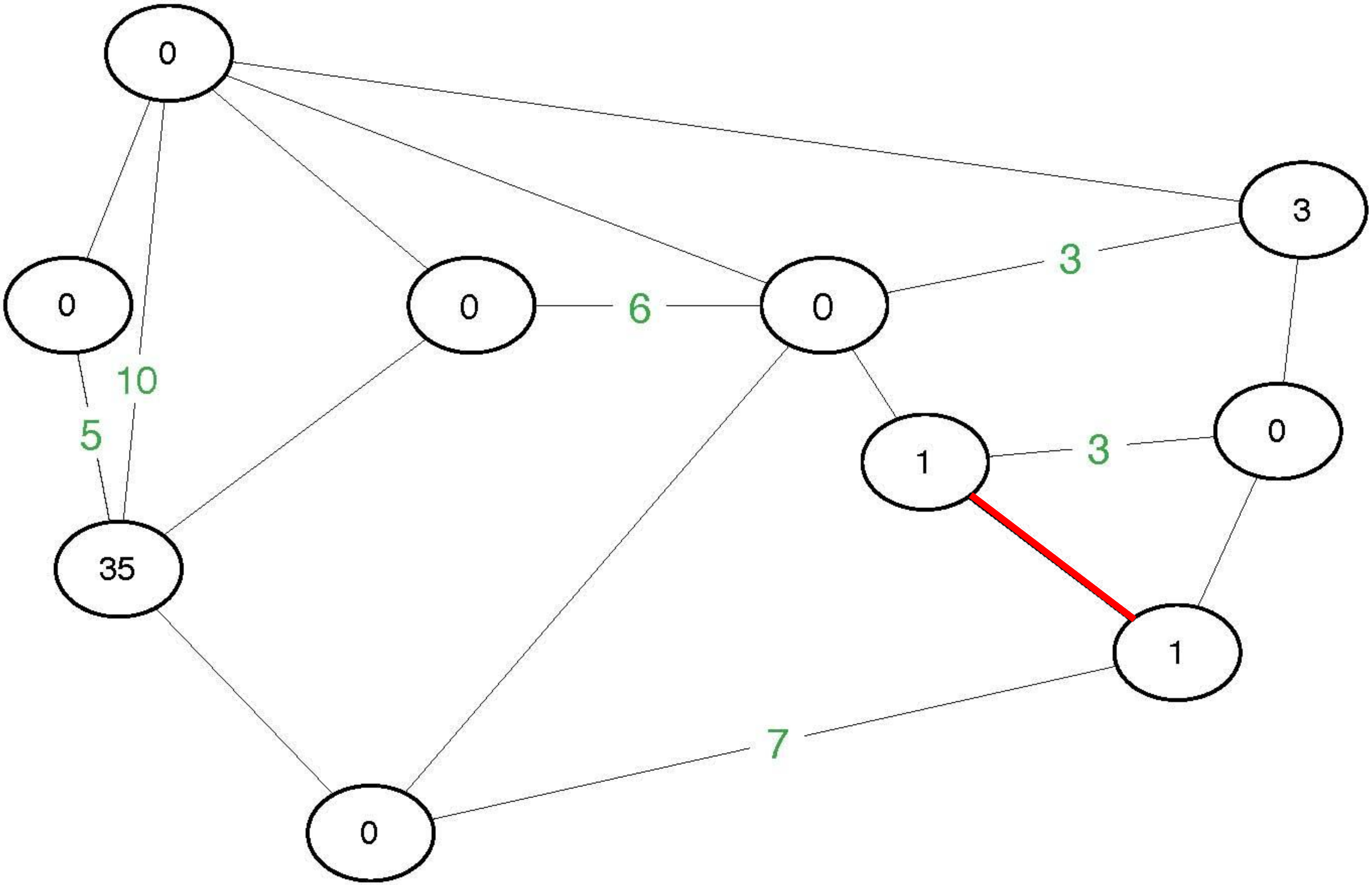

10
5
0
6
3
6
4
3
50
8
7

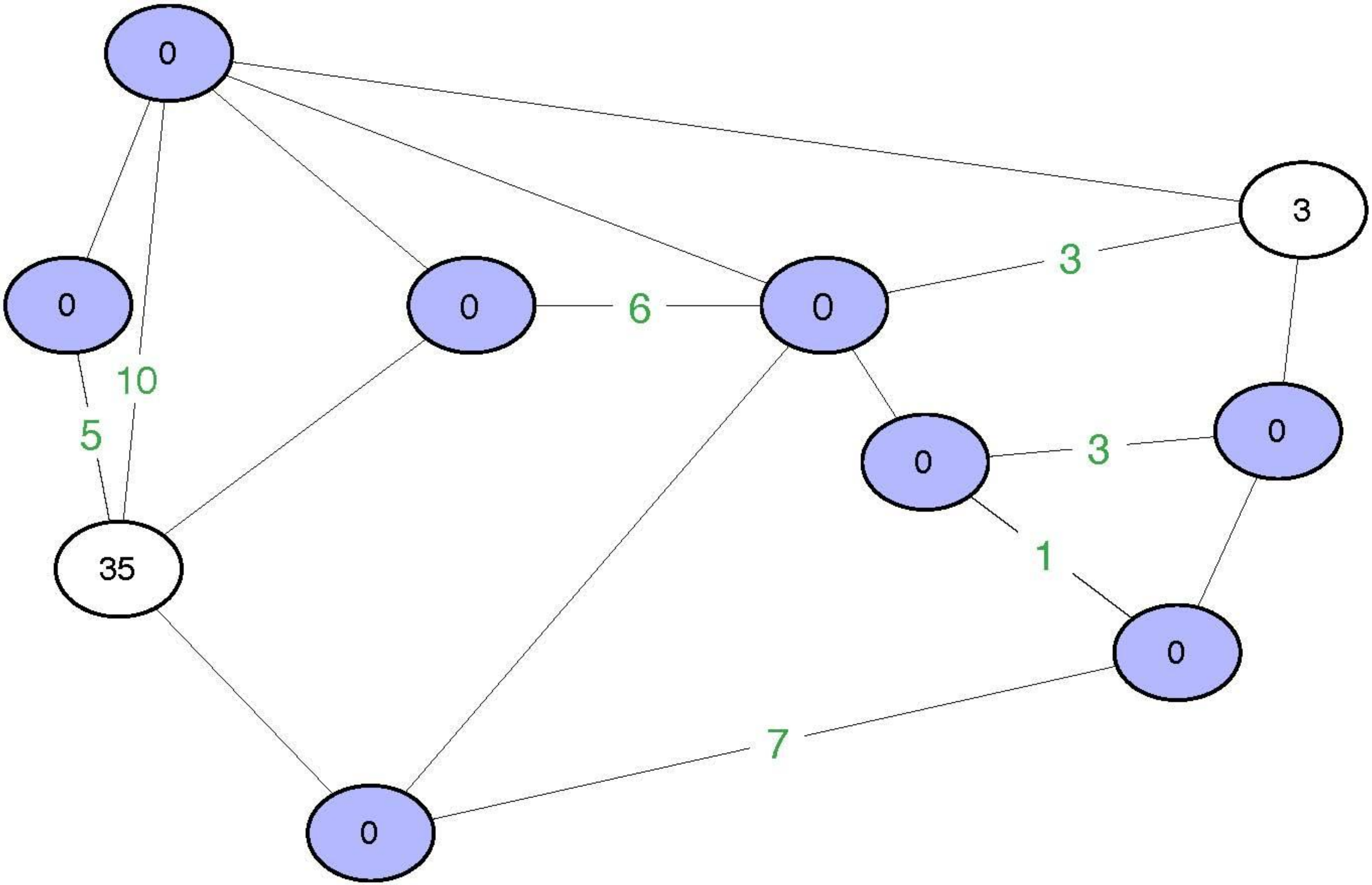

0
5
10
0
6
3
6
4
3
40
8
7

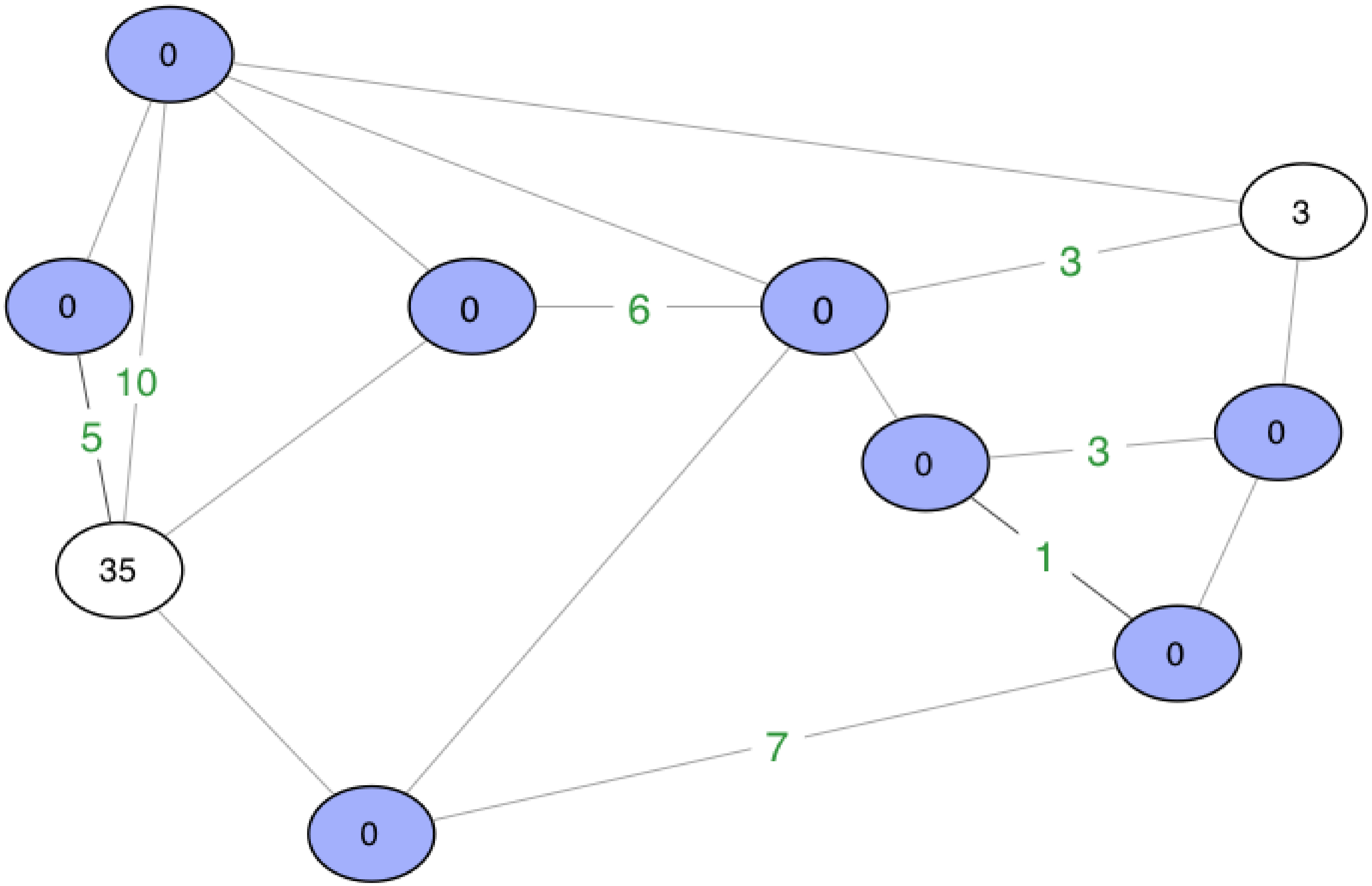


0
5
0
6
3
6
10
3
4
40
8
7


0
3
3
0
6
0
0
10
5
1
3
0
35
1
7
0


0
0
0
0
3
3
6
10
5
35
1
3
0
1
7
0


0
3
0
0
6
0
3
10
5
0
3
0
35
1
0
7
0



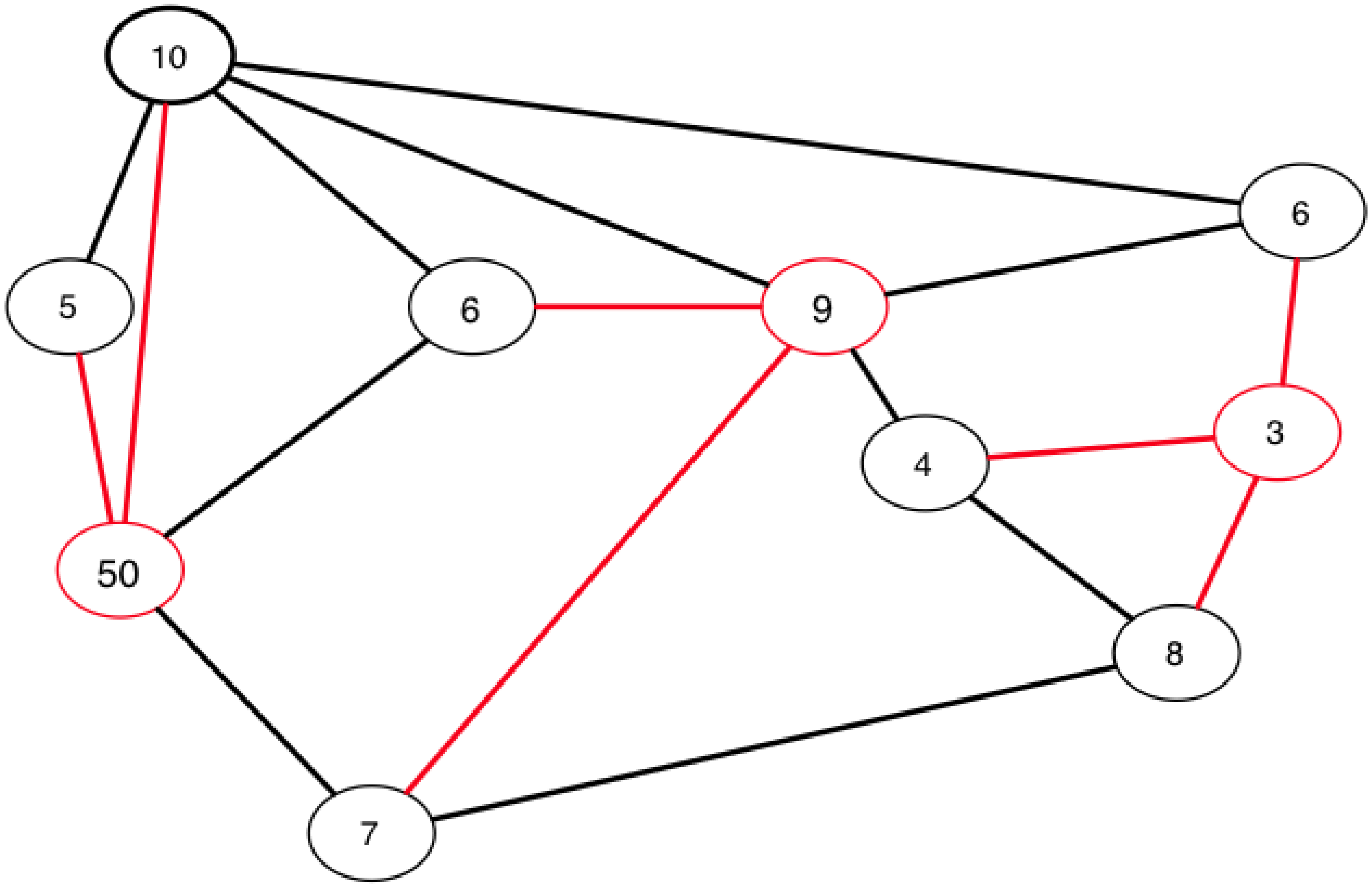
Theorem: $\mathrm{cost}(C) \leq 2\mathrm{OPT}$

Distributed Computation

- Proceed in rounds.
- In each round:
  - Each node exchanges O(1) messages with immediate neighbors,
  - then does some computation.
- **goal:** Finish in a poly-log number of rounds.

Distributed Algorithm

(PODC 2009)



Each round:

1. form independent rooted "stars"

2. coordinate discounts within stars

- Done when zero-cost vertices cover all edges.

goal: *Done after* $O(\log n)$ *rounds* (n = #nodes).

How to form stars

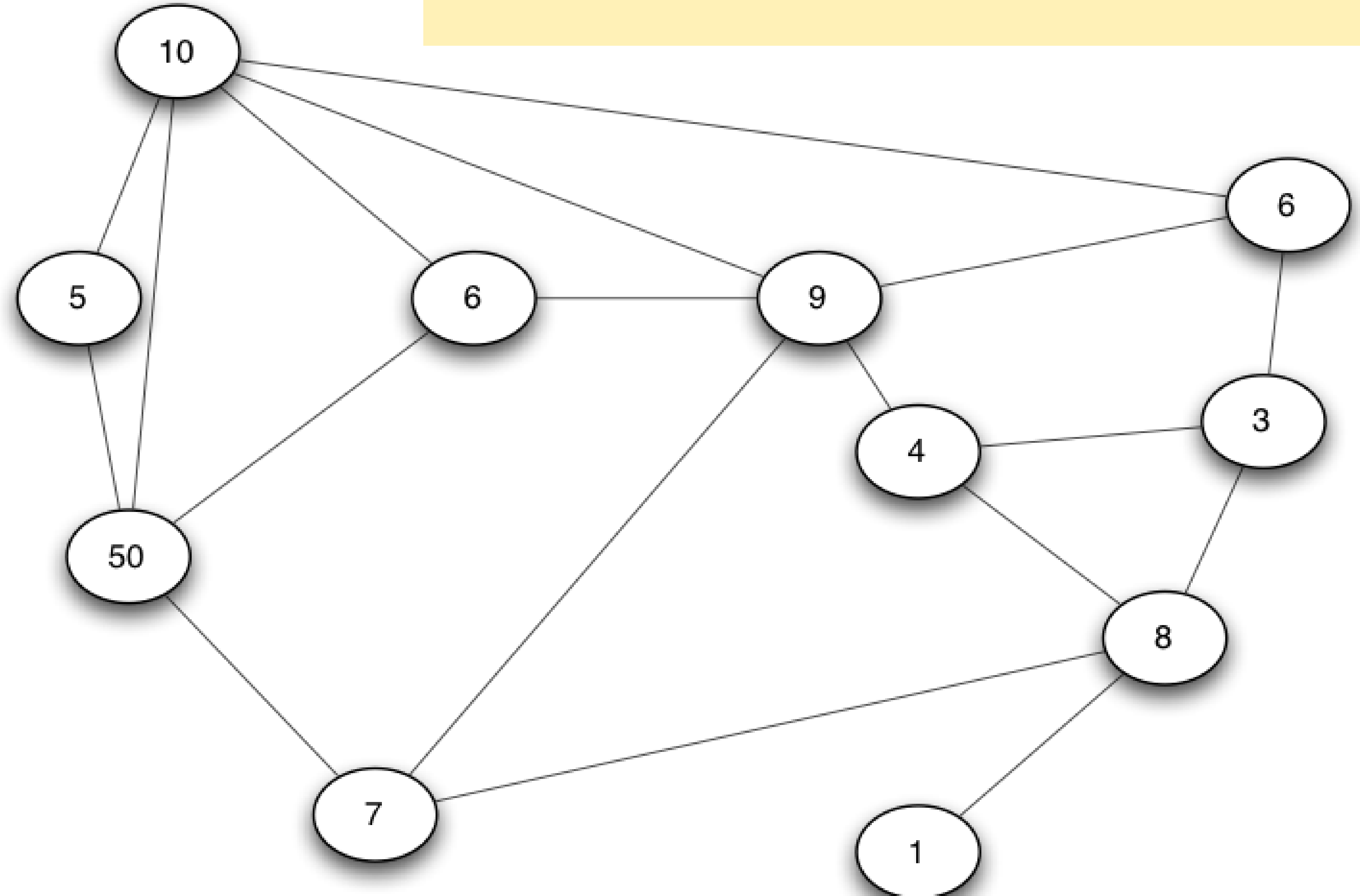
1. Each node randomly chooses to be "boy" or "girl" (just for this round)

2. For the round, use only edges from boys to higher-cost (or equal-cost) girls. (Pretend other edges don't exist.)†

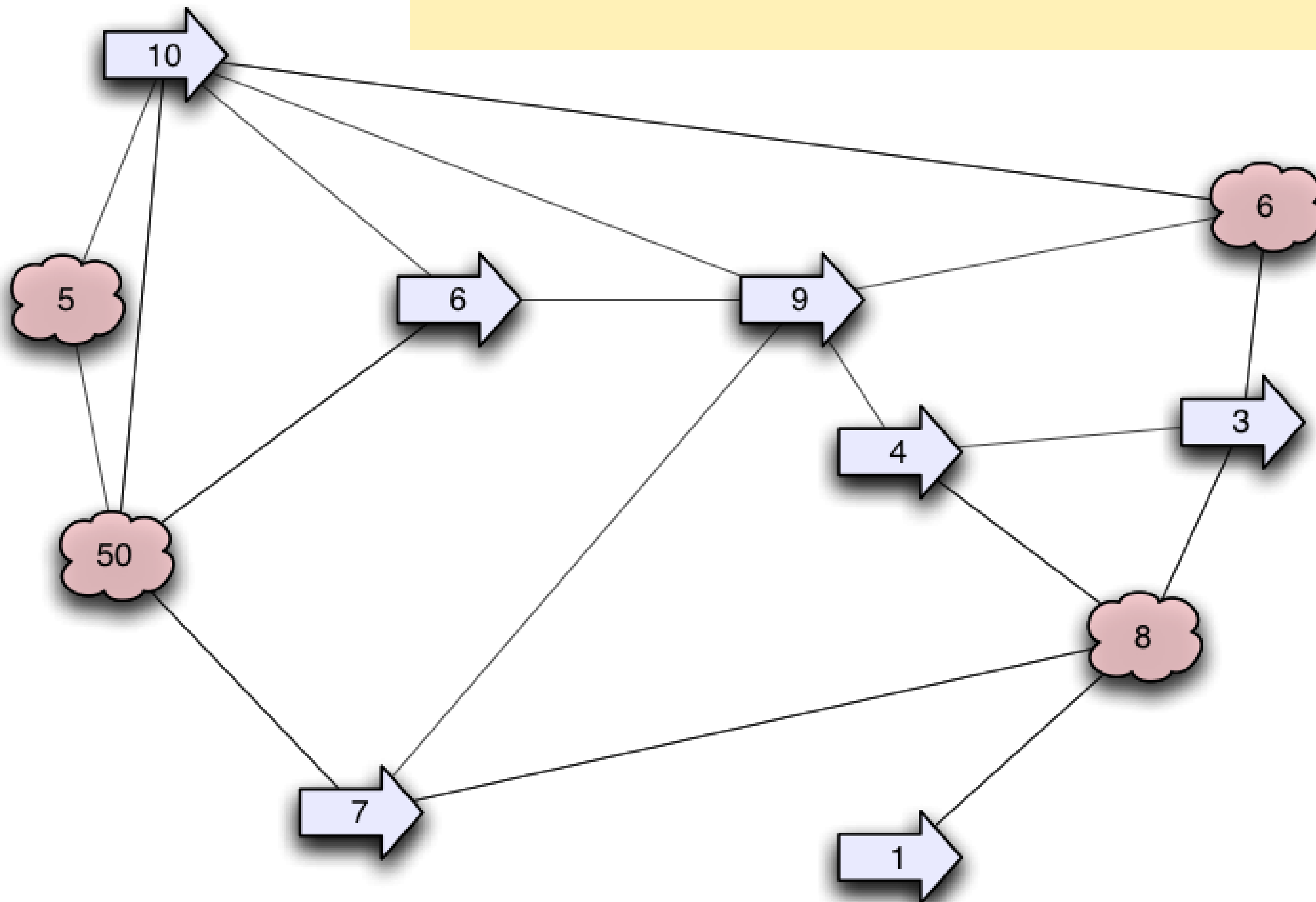
3. Each boy chooses a random neighbor (girl of ≥ cost).

† In each round, every edge has a one in four chance of being used. (... will be used if low-cost endpoint is boy, high-cost endpoint is girl)

- How to form stars:
    1. Each node randomly chooses to be “boy” or “girl”.
    2. Use only edges from boys to higher-cost (or equal-cost) girls. (Pretend other edges don’t exist.)
    3. Each boy chooses a random neighbor (girl of ≥ cost).

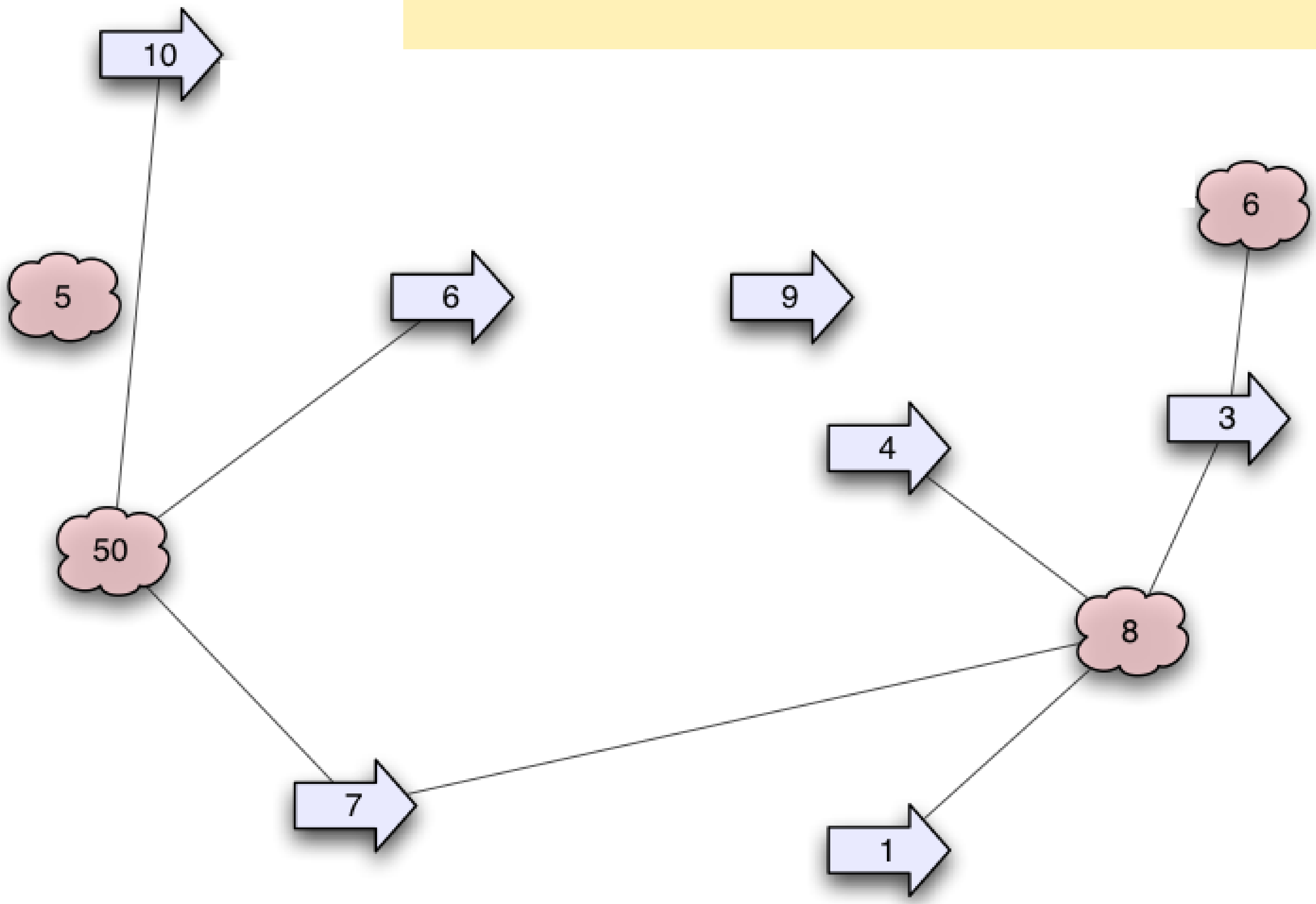


- How to form stars:
  1. **Each node randomly chooses to be “boy” or “girl”.**
  2. Use only edges from boys to higher-cost (or equal-cost) girls. (Pretend other edges don’t exist.)
  3. Each boy chooses a random neighbor (girl of ≥ cost).



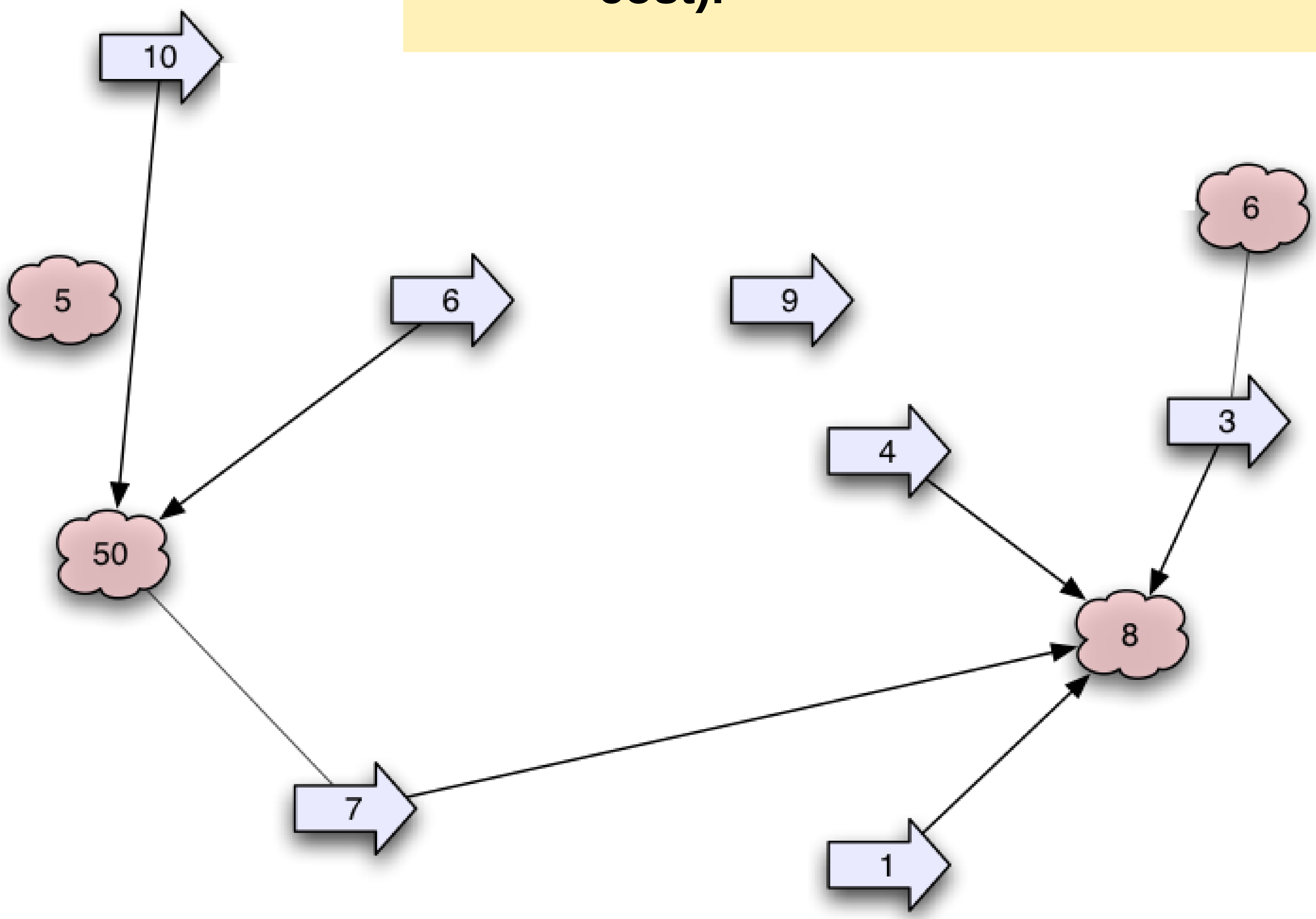
How to form stars:

1. Each node randomly chooses to be “boy” or “girl”.
2. **Use only edges from boys to higher-cost (or equal-cost) girls. (Pretend other edges don’t exist.)**
3. Each boy chooses a random neighbor (girl of ≥ cost).



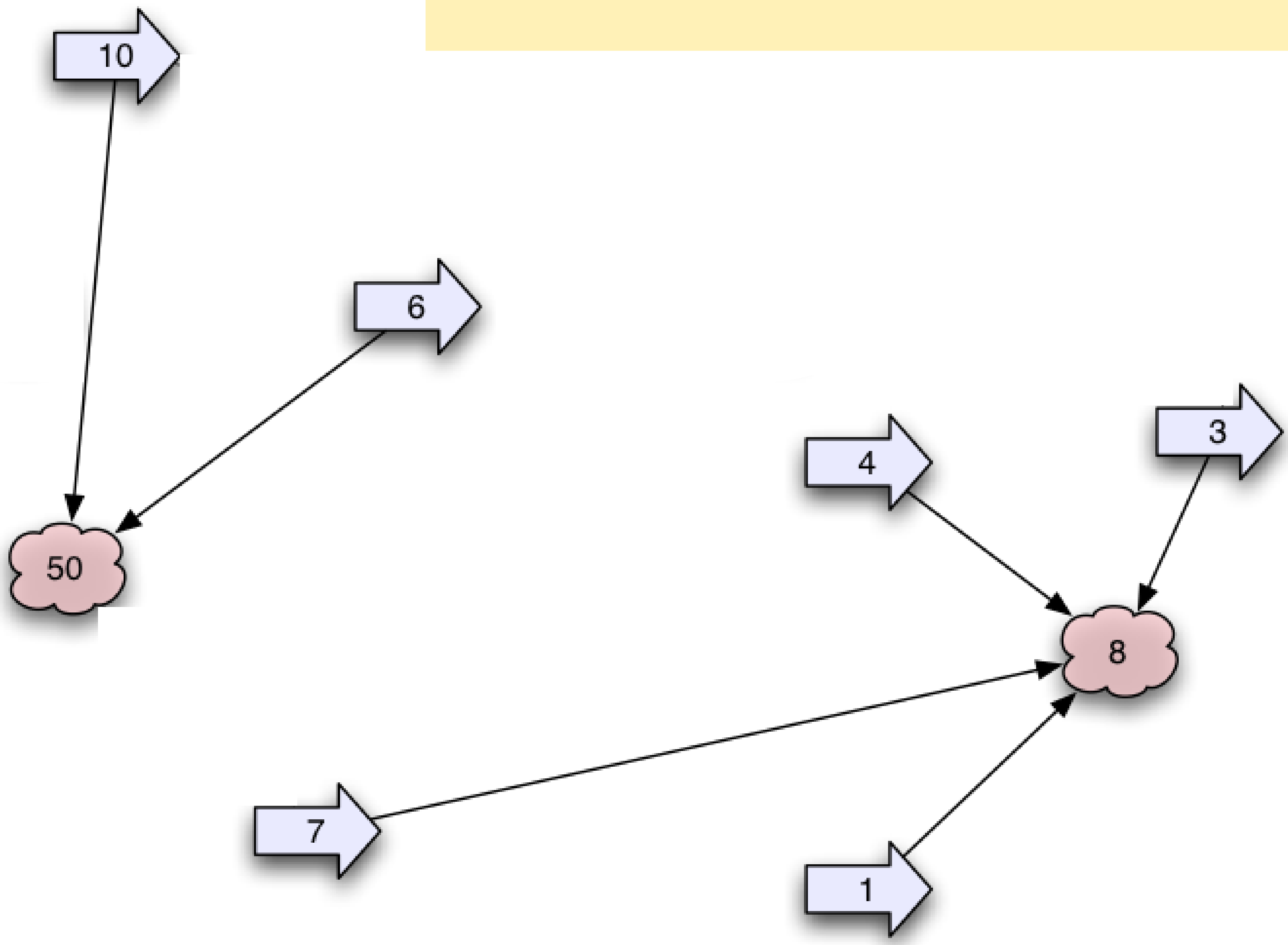
How to form stars:

1. Each node randomly chooses to be “boy” or “girl”.
2. Use only edges from boys to higher-cost (or equal-cost) girls. (Pretend other edges don’t exist.)
3. **Each boy chooses a random neighbor (girl of ≥ cost).**



How to form stars:

1. Each node randomly chooses to be “boy” or “girl”.
2. Use only edges from boys to higher-cost (or equal-cost) girls. (Pretend other edges don’t exist.)
3. Each boy chooses a random neighbor (girl of ≥ cost).



How girls allocate discounts

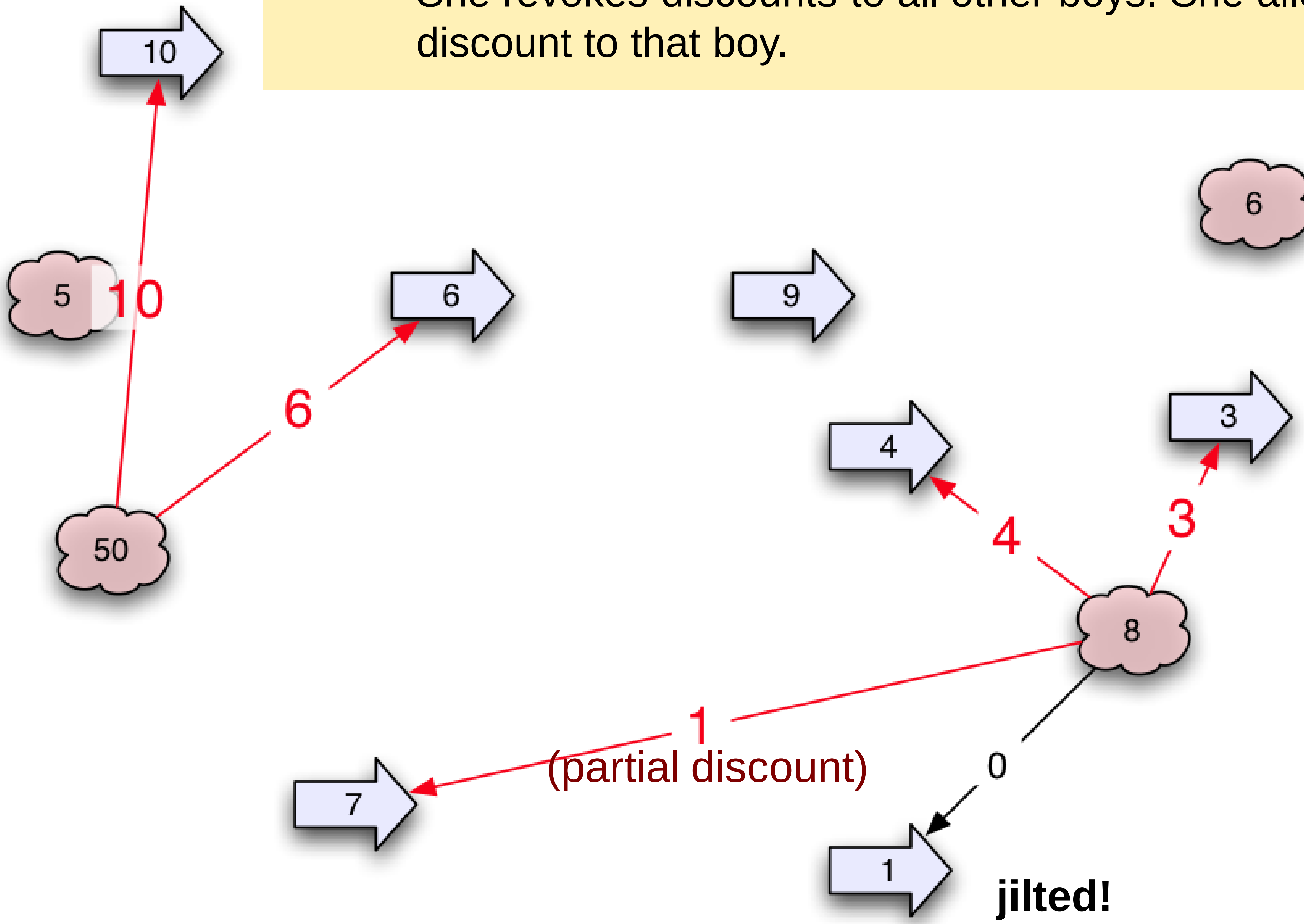
- Each girl allocates discounts greedily, in alphabetic order.
- If she *partially allocates* some boy's discount, then...
  - with probability 1/2:
    1. She revokes discounts to all other boys.
    2. She allocates full discount to that boy.

Some boys may be *jilted* (have *no chance* for discount).

- Each girl allocates discounts greedily, in alphabetical order.
- If she *partially allocates* some boy's discount, then...

  with probability 1/2:

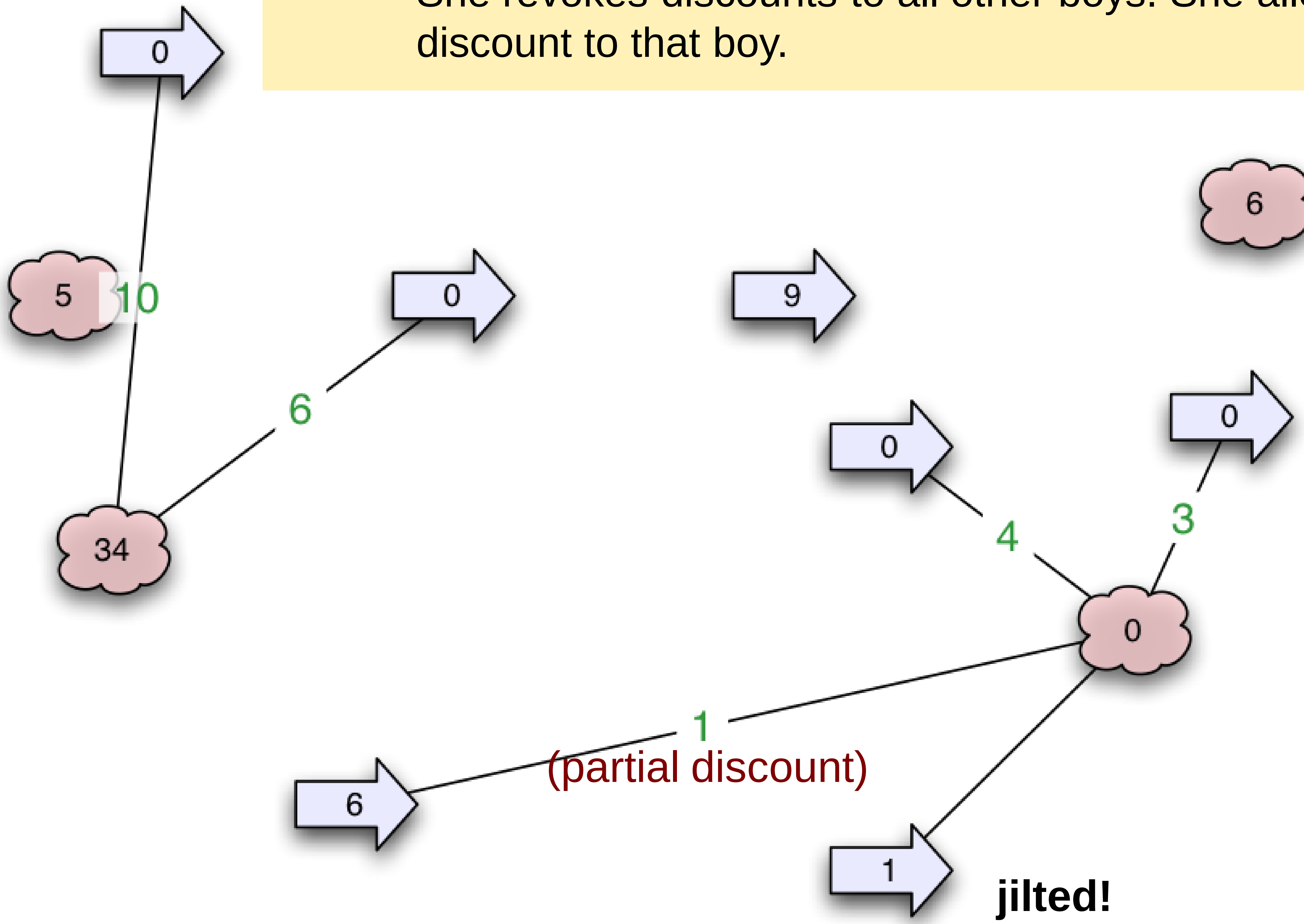
  She revokes discounts to all other boys. She allocates full discount to that boy.



- **Each girl allocates discounts greedily, in alphabetical order.**
- If she *partially allocates* some boy's discount, then...

  with probability 1/2:

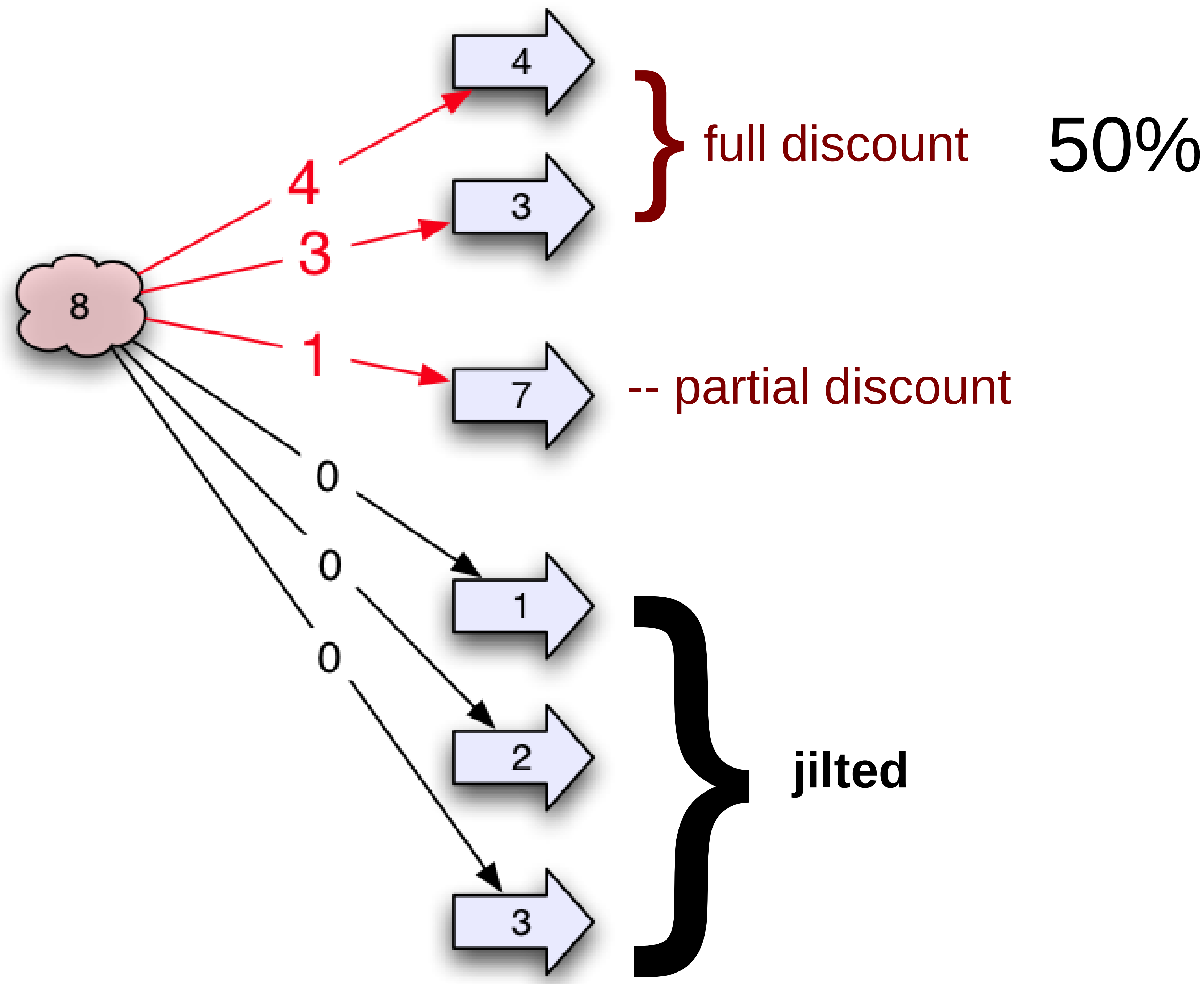
  She revokes discounts to all other boys. She allocates full discount to that boy.



- Each girl allocates discounts greedily, in alphabetical order.
- **If she *partially allocates* some boy's discount, then...**

with probability 1/2:

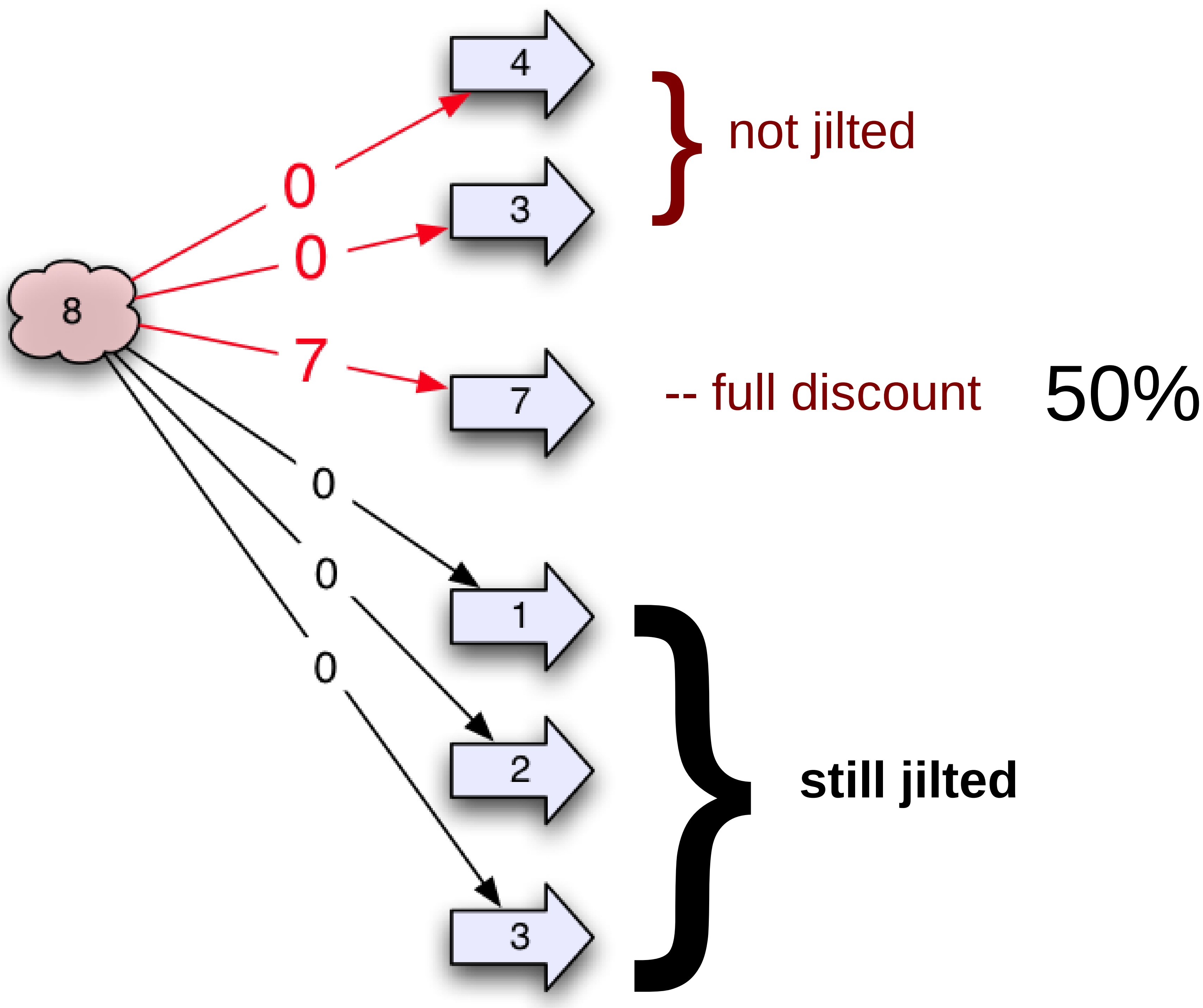
She revokes discounts to all other boys. She allocates full discount to that boy.



- Each girl allocates discounts greedily, in alphabetical order.
- **If she *partially allocates* some boy's discount, then...**

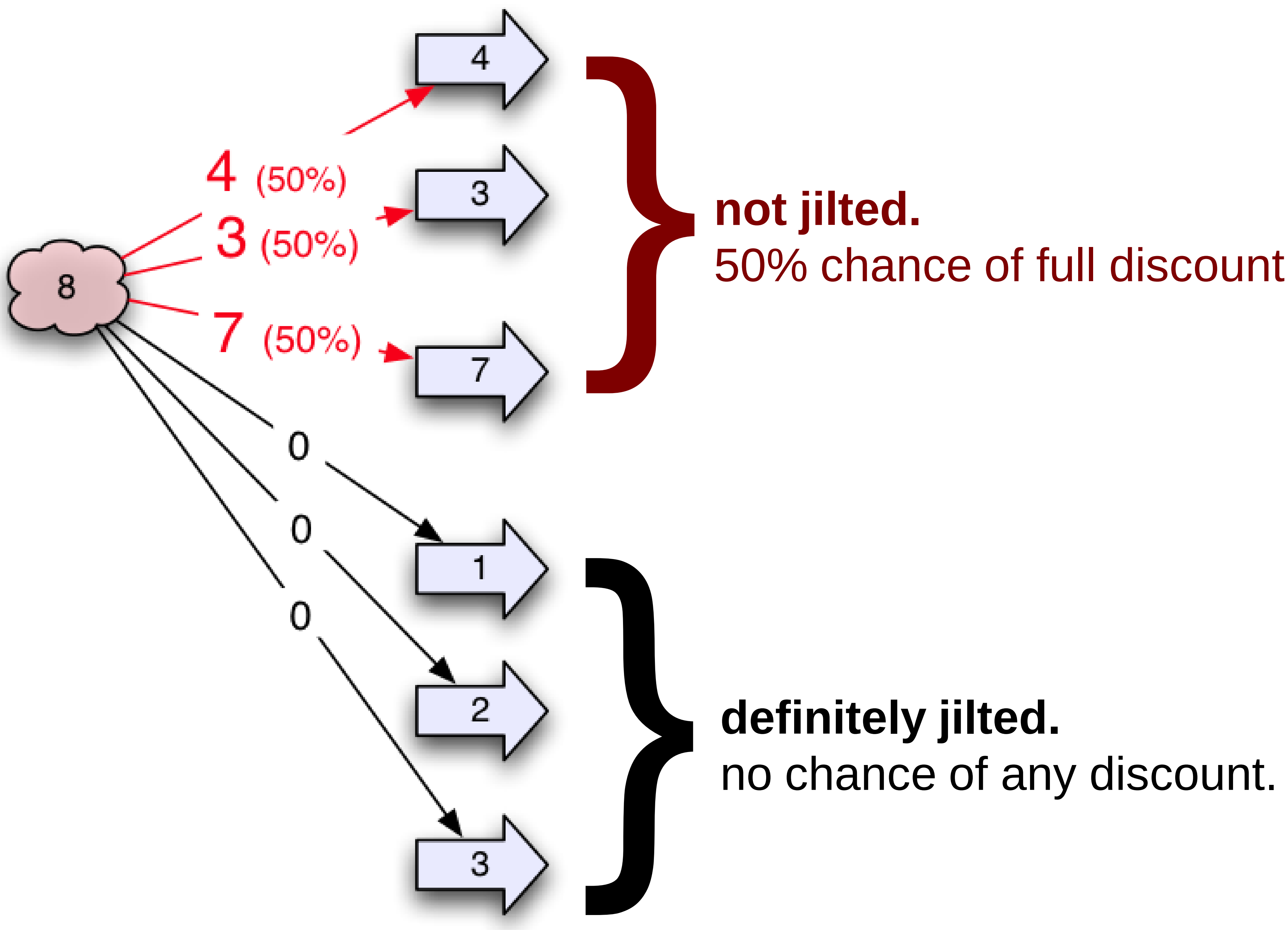
with probability 1/2:

She revokes discounts to all other boys. She allocates full discount to that boy.



each of girl's boys is either:

- ***jilted*** (girl gives no chance of any discount)
- not jilted (girl gives at least 50% chance of discount)



Analysis

- Guaranteed to return a 2-approximate solution, since it implements the edge-discount algorithm.
- What about running time?
- **Goal:** Show O(log n) rounds (w.h.p.).

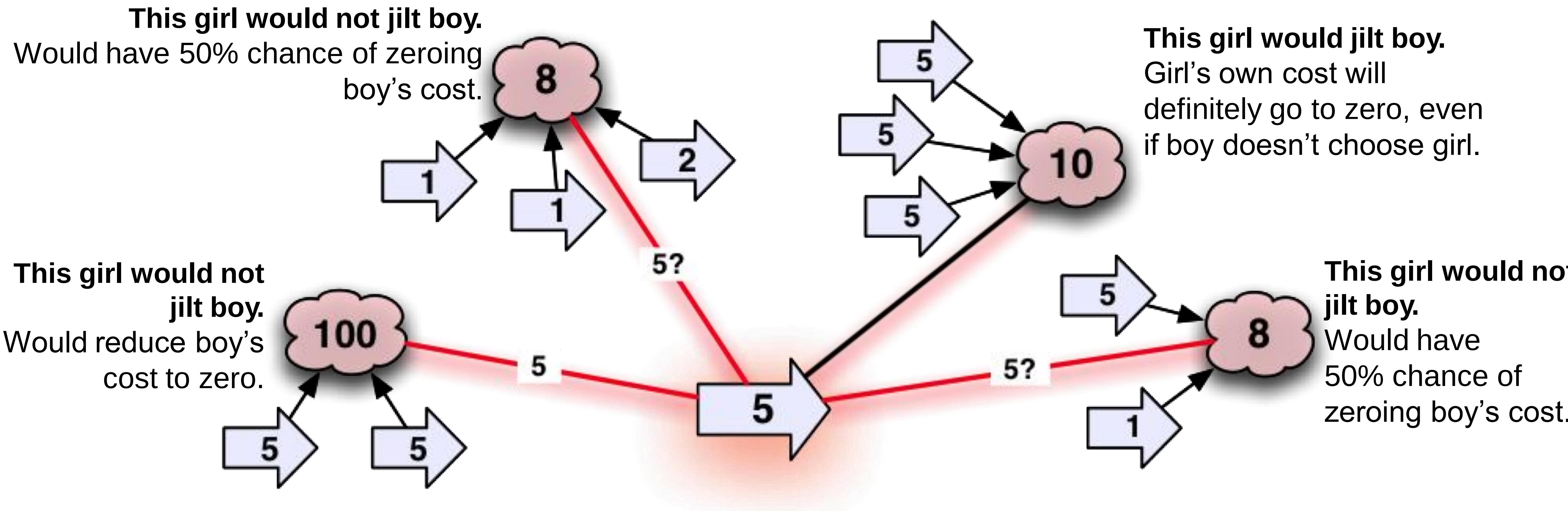
Analysis of number of rounds

- “Delete” edges when one endpoint’s cost becomes zero.

- **lemma:** *In each round, in expectation, a constant fraction of each boy’s active edges are deleted.*
- **proof:** (next)

corollary: *Number of rounds is* $O(\log n^2) = O(\log n)$ *in expectation and with high probability.*

lemma: *In each round, in expectation, a constant fraction of* ***each boy's*** *active edges are deleted.*



key observation:

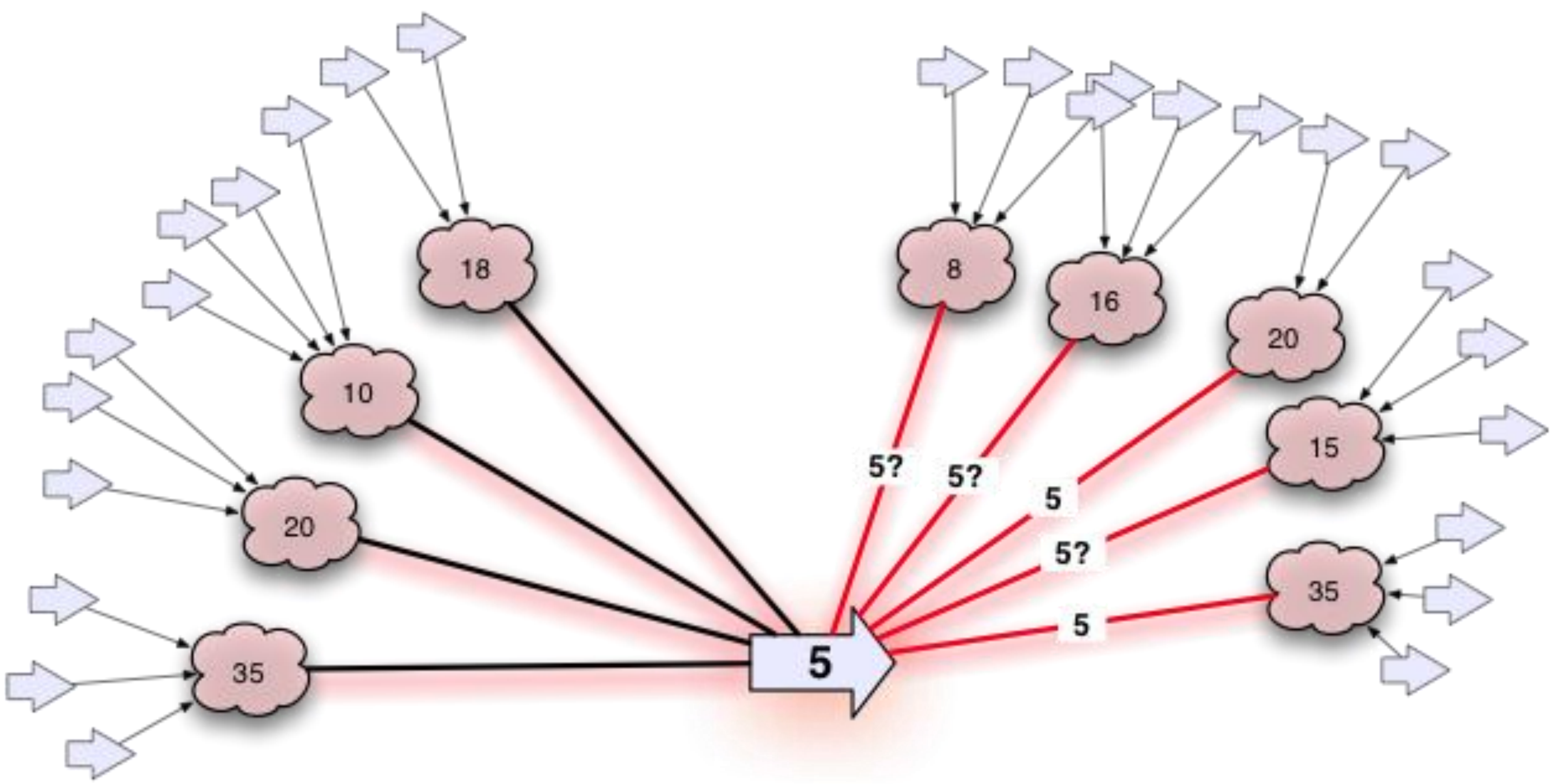
girl **would jilt** boy ⇒ her cost is going to zero regardless of what boy does.

girl **would not jilt** boy ⇒ if boy chooses her,
she has at least a 50% chance of zeroing boy's cost...

key observation:

girl **would jilt** boy ⇒ her cost is going to zero regardless of what boy does.

girl **would not jilt** boy ⇒ if boy chooses her,
she has at least a 50% chance of zeroing boy's cost...



- **case (i): At least half of boy's girls would jilt him.**

  ⇒ At least half of boy's edges will be deleted regardless of what boy does.

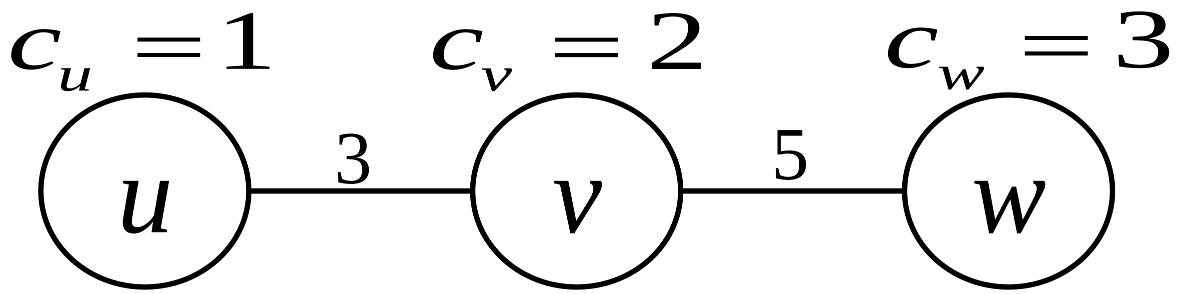
- **case (ii): At least half of boy's girls would not jilt him.**

  ⇒ Boy has at least a 50% chance of choosing a girl
  who has at least a 50% chance of zeroing his cost (deleting all his edges).

- There is a simple, fast, sequential, online and distributed 2-approximation algorithm for Weighted Vertex Cover!

More general Covering Problems

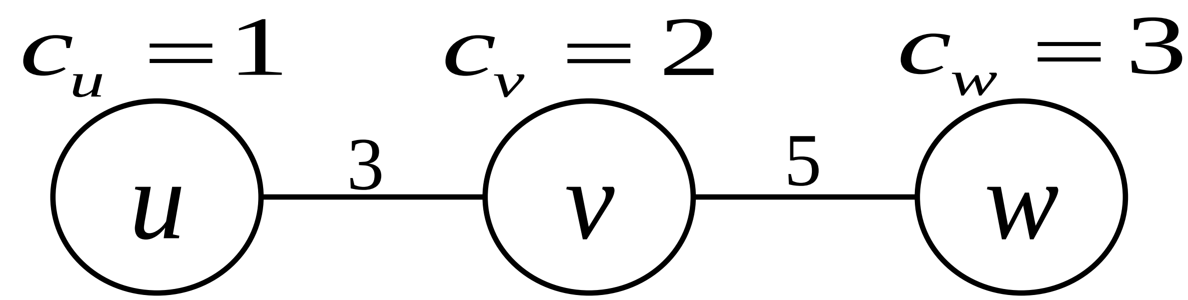
$$\min\ x_u + 2x_v + 3x_w :$$
$$x_u + x_v \geq 3$$
$$x_v + x_w \geq 4$$
$$x_u, x_v, x_w \in Z_+$$



Bar-Yehuda and Even's algorithm does not extend to non 0/1 problems.

More general Covering Problems

$$\min \; x_u + 2x_v + 3x_w :$$
$$x_u + x_v \geq 3$$
$$x_v + x_w \geq 4$$
$$x_u, x_v, x_w \in Z_+$$



Bar-Yehuda and Even's algorithm does not extend to non 0/1 problems.

Goal: Simple 2-approximation algorithm, applicable in the *sequential*, *distributed* and *online* setting.

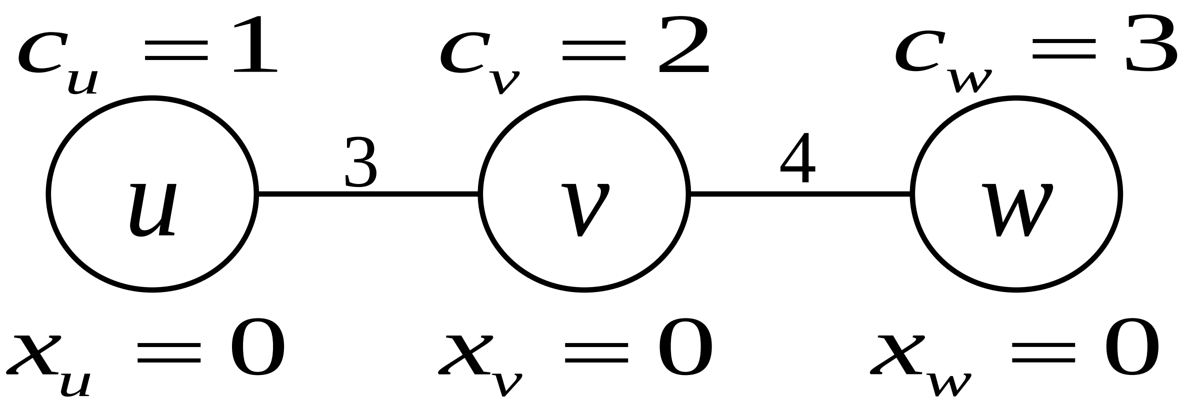
Flooding algorithm

1. Let $x \leftarrow 0$.
2. While $\exists$ edge (u,v) s.t. $\lfloor x_u \rfloor + \lfloor x_v \rfloor < b_{uv}$ do:
3. Raise x_u at rate $1/c_u$ and x_v at rate $1/c_v$ until $\lfloor x_u \rfloor + \lfloor x_v \rfloor \geq b_{uv}$.
4. Return $\lfloor x \rfloor$.

Flooding algorithm

1. Let $x \leftarrow 0$.
2. While $\exists$ edge (u,v) s.t. $\lfloor x_u \rfloor + \lfloor x_v \rfloor < b_{uv}$ do:
3. Raise x_u at rate $1/c_u$ and x_v at rate $1/c_v$ until $\lfloor x_u \rfloor + \lfloor x_v \rfloor \geq b_{uv}$.
4. Return $\lfloor x \rfloor$.

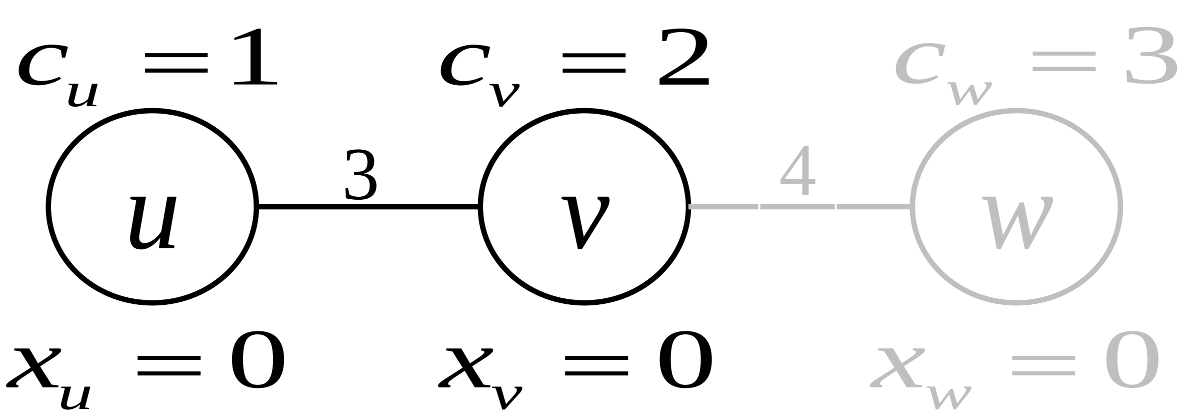
$$
\begin{aligned}
\min \ & x_u + 2x_v + 3x_w : \\
& x_u + x_v \geq 3 \\
& x_v + x_w \geq 4 \\
& x_u, x_v, x_w \in Z_+
\end{aligned}
$$



Flooding algorithm

1. Let $x \leftarrow 0$.
2. While $\exists$ edge (u,v) s.t. $\lfloor x_u \rfloor + \lfloor x_v \rfloor < b_{uv}$ do:
3. Raise x_u at rate $1/c_u$ and x_v at rate $1/c_v$ until $\lfloor x_u \rfloor + \lfloor x_v \rfloor \geq b_{uv}$.
4. Return $\lfloor x \rfloor$.

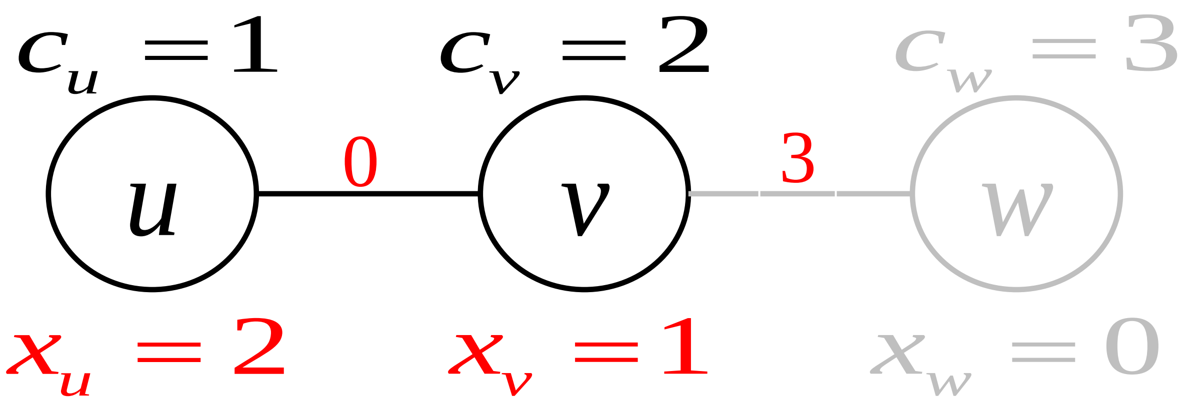
$$\begin{aligned} \min\ & x_u + 2x_v + 3x_w : \\ & x_u + x_v \geq 3 \\ & x_v + x_w \geq 4 \\ & x_u, x_v, x_w \in Z_+ \end{aligned}$$



Flooding algorithm

1. Let $x \leftarrow 0$.
2. While $\exists$ edge (u,v) s.t. $\lfloor x_u \rfloor + \lfloor x_v \rfloor < b_{uv}$ do:
3. Raise x_u at rate $1/c_u$ and x_v at rate $1/c_v$ until $\lfloor x_u \rfloor + \lfloor x_v \rfloor \geq b_{uv}$.
4. Return $\lfloor x \rfloor$.

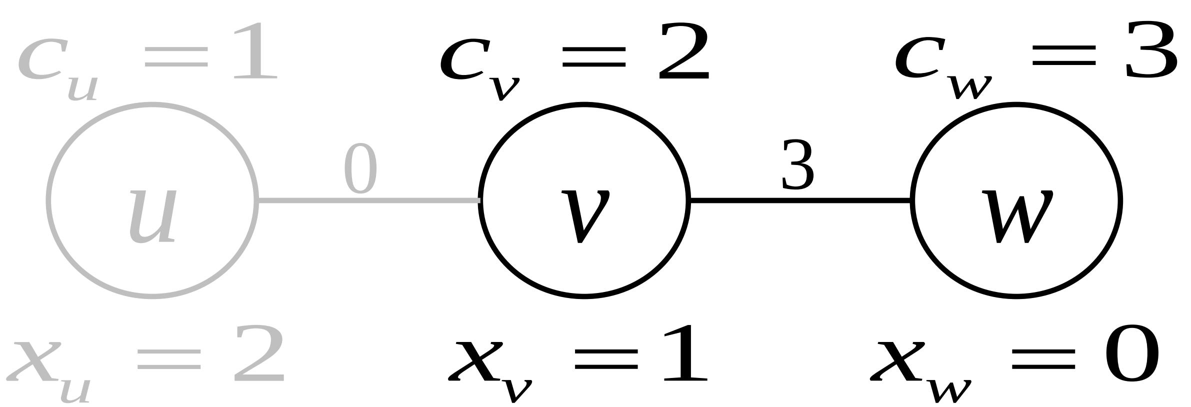
$$\min \; x_u + 2x_v + 3x_w :$$
$$x_u + x_v \geq 3$$
$$x_v + x_w \geq 4$$
$$x_u, x_v, x_w \in Z_+$$



Flooding algorithm

1. Let $x \leftarrow 0$.
2. While $\exists$ edge (u,v) s.t. $\lfloor x_u \rfloor + \lfloor x_v \rfloor < b_{uv}$ do:
3. Raise x_u at rate $1/c_u$ and x_v at rate $1/c_v$ until $\lfloor x_u \rfloor + \lfloor x_v \rfloor \geq b_{uv}$.
4. Return $\lfloor x \rfloor$.

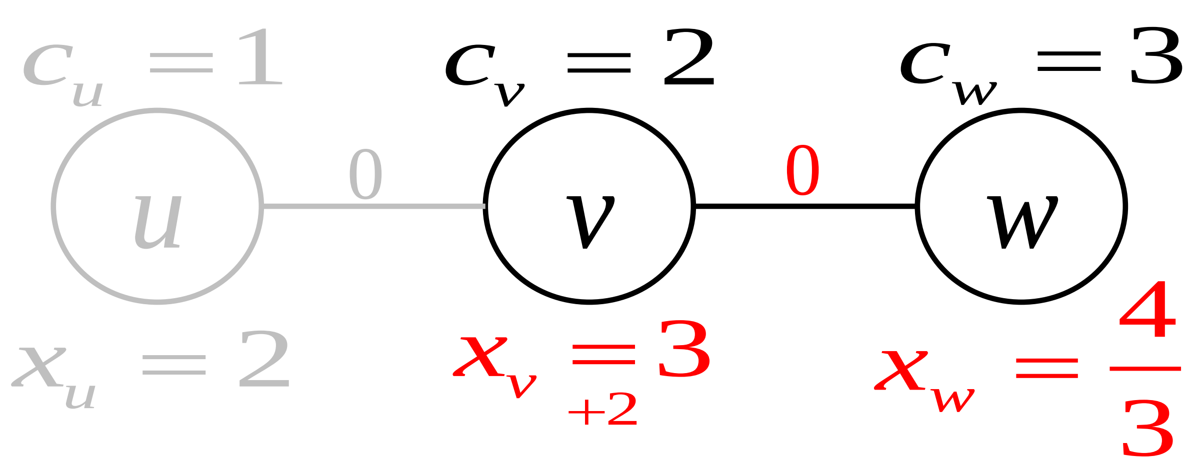
$$\min \; x_u + 2x_v + 3x_w :$$
$$x_u + x_v \geq 3$$
$$x_v + x_w \geq 4$$
$$x_u, x_v, x_w \in Z_+$$



Flooding algorithm

1. Let $x \leftarrow 0$.
2. While $\exists$ edge (u,v) s.t. $\lfloor x_u \rfloor + \lfloor x_v \rfloor < b_{uv}$ do:
3. Raise x_u at rate $1/c_u$ and x_v at rate $1/c_v$ until $\lfloor x_u \rfloor + \lfloor x_v \rfloor \geq b_{uv}$.
4. Return $\lfloor x \rfloor$.

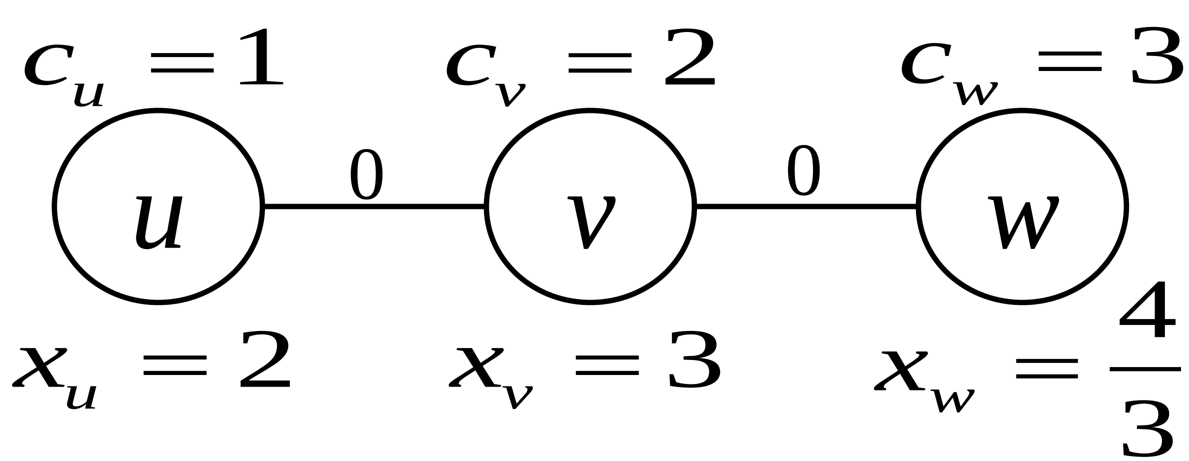
$$\min\ x_u + 2x_v + 3x_w :$$
$$x_u + x_v \geq 3$$
$$x_v + x_w \geq 4$$
$$x_u, x_v, x_w \in Z_+$$



Flooding algorithm

1. Let $x \leftarrow 0$.
2. While $\exists$ edge (u,v) s.t. $\lfloor x_u \rfloor + \lfloor x_v \rfloor < b_{uv}$ do:
3. Raise x_u at rate $1/c_u$ and x_v at rate $1/c_v$ until $\lfloor x_u \rfloor + \lfloor x_v \rfloor \geq b_{uv}$.
4. Return $\lfloor x \rfloor$.

$$\begin{aligned} \min\ & x_u + 2x_v + 3x_w : \\ & x_u + x_v \geq 3 \\ & x_v + x_w \geq 4 \\ & x_u, x_v, x_w \in Z_+ \end{aligned}$$



Return $x_u = 2, x_v = 3, x_w = \left\lfloor \frac{4}{3} \right\rfloor = 1$

Flooding algorithm

1. Let $x \leftarrow 0$.
2. While $\exists$ edge (u,v) s.t. $\lfloor x_u \rfloor + \lfloor x_v \rfloor < b_{uv}$ do:
3. Raise x_u at rate $1/c_u$ and x_v at rate $1/c_v$ until $\lfloor x_u \rfloor + \lfloor x_v \rfloor \geq b_{uv}$.
4. Return $\lfloor x \rfloor$.

Let x^* be any feasible solution.

Each step starts with a non-yet-satisfied constraint, so $x_u^* > x_u$ or $x_v^* > x_v$.

Let $residual_c(x)$ be the min cost to increase x to full feasibility.

The step increases the cost by 2β but it reduces $residual_c(x)$ by at least β $\Longrightarrow$ 2-approximation

Results (sequential)

- Covering Mixed Integer Programs: Nearly linear time δ-approximation algorithm. Improves over the previous ellipsoid-based (slow) algorithm.
- Non-metric Facility Location: Linear time δ-approximation. δ is the maximum number of facilities that might serve a customer.
- Covering problems with submodular cost function

Results (online)

- Online CMIP: δ-competitive algorithm.
- Paging, Weighted Caching, File Caching, Connection Caching: Generalize k-competitive algorithms i.e. Landlord, Harmonic, Greedy-Dual.
- Upgradable Caching: $(k+d)$-competitive algorithm, where k is the cache size and d is the number of upgradable parameters.

Results (distributed)

- Covering Mixed Integer Programs with 2 variables per constraint: 2-approximation in $O(\log |C|)$ rounds, where $|C|$ is the number of constraints. Also, 2-approximation in RNC.

⇒ 2-approximation for Weighted Vertex Cover in $O(\log n)$ rounds.

- Covering Problems with at most δ variables per constraint: δ-approximation in $O(\log^2 |C|)$ rounds.

thank you