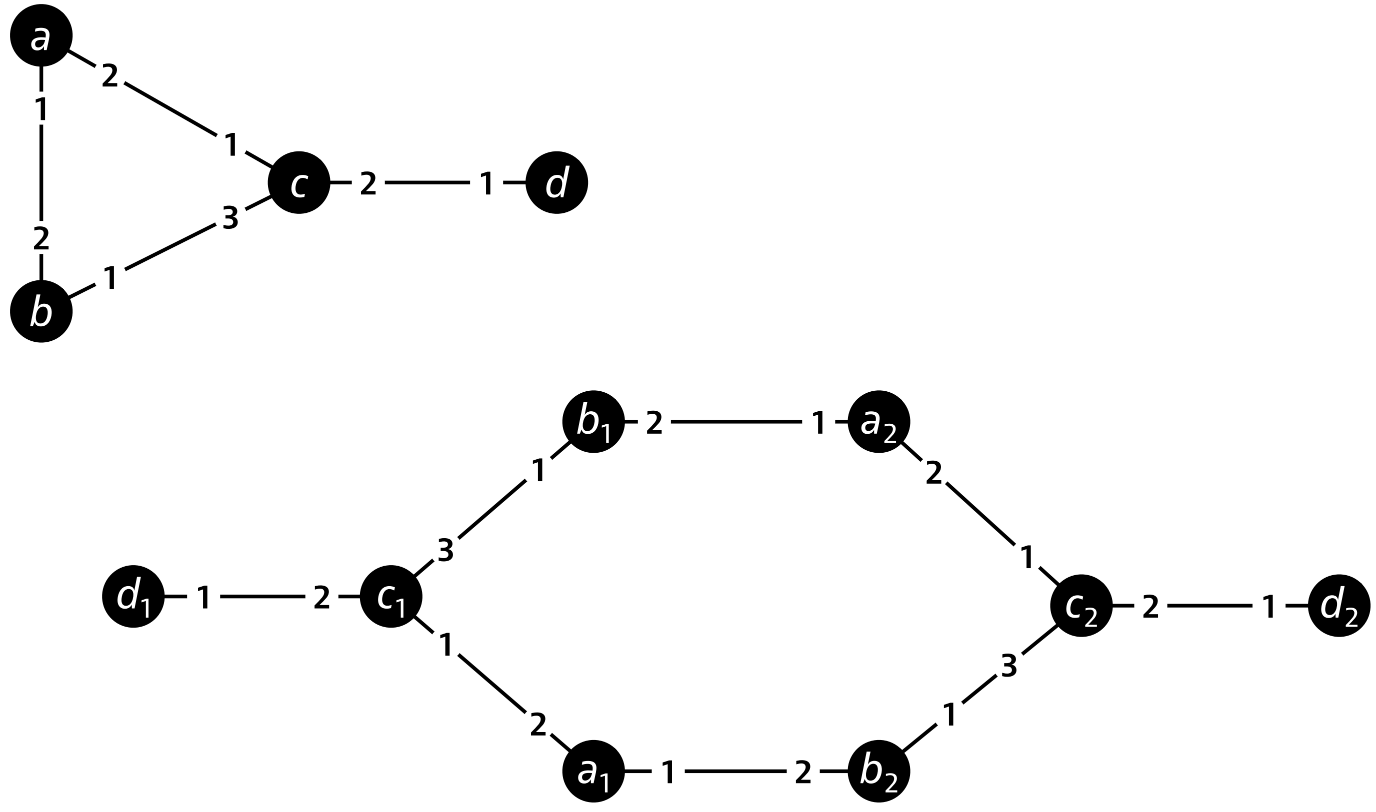
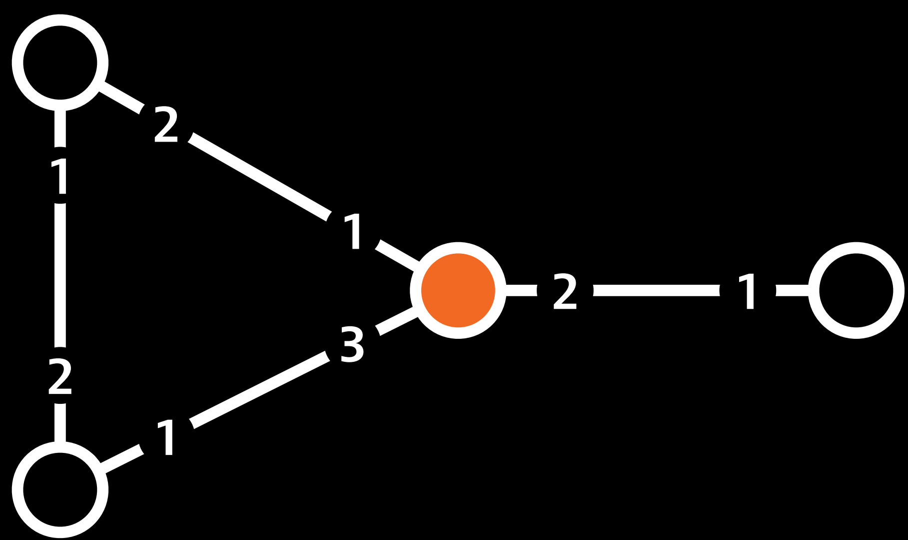
Lower Bounds:

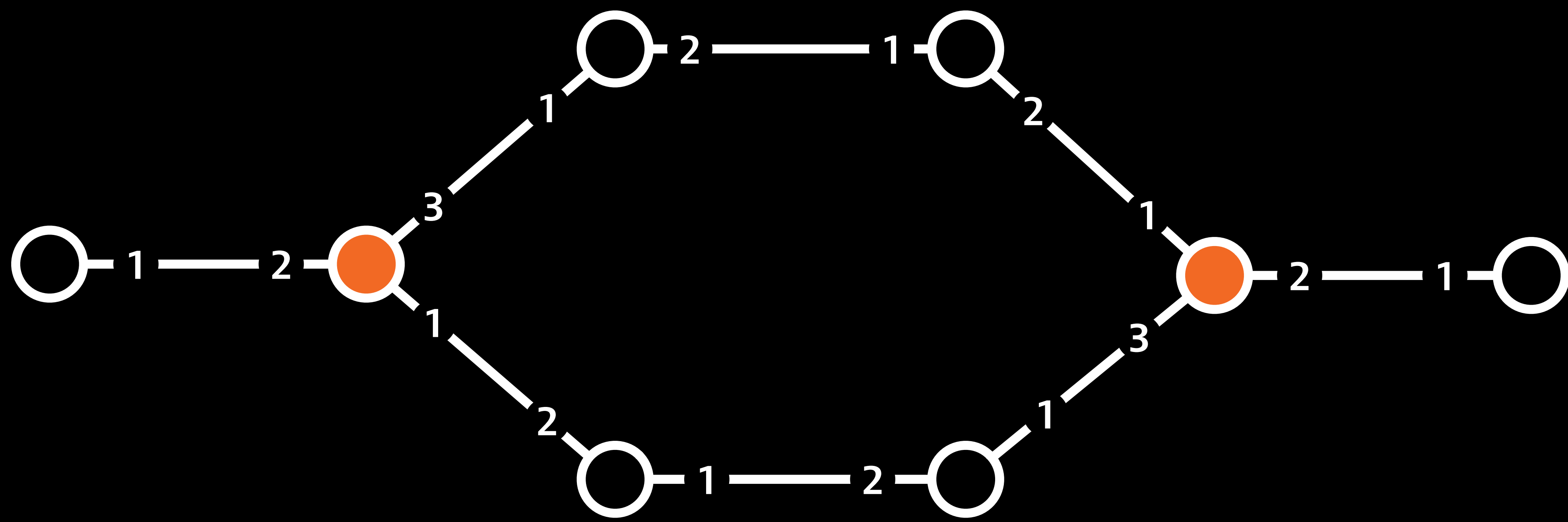
Covering Maps & Neighborhood graphs

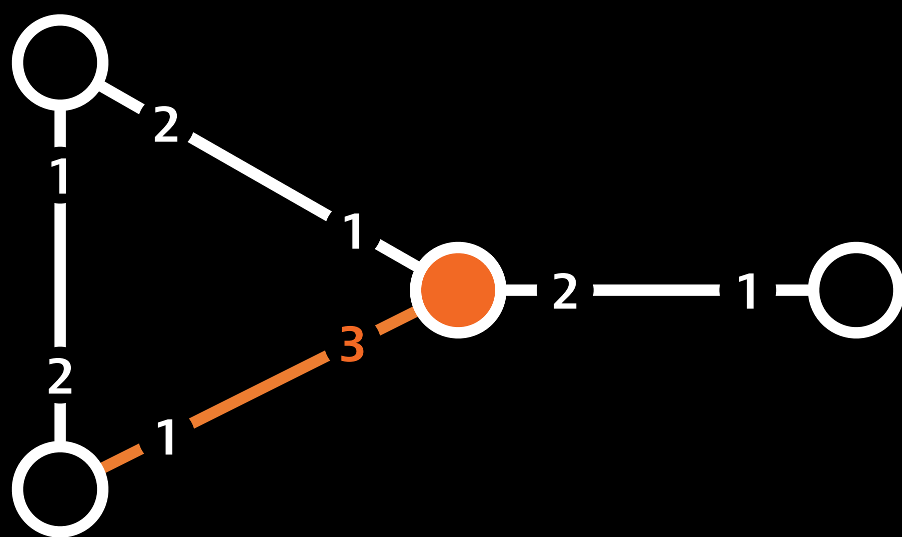


Port numbering model

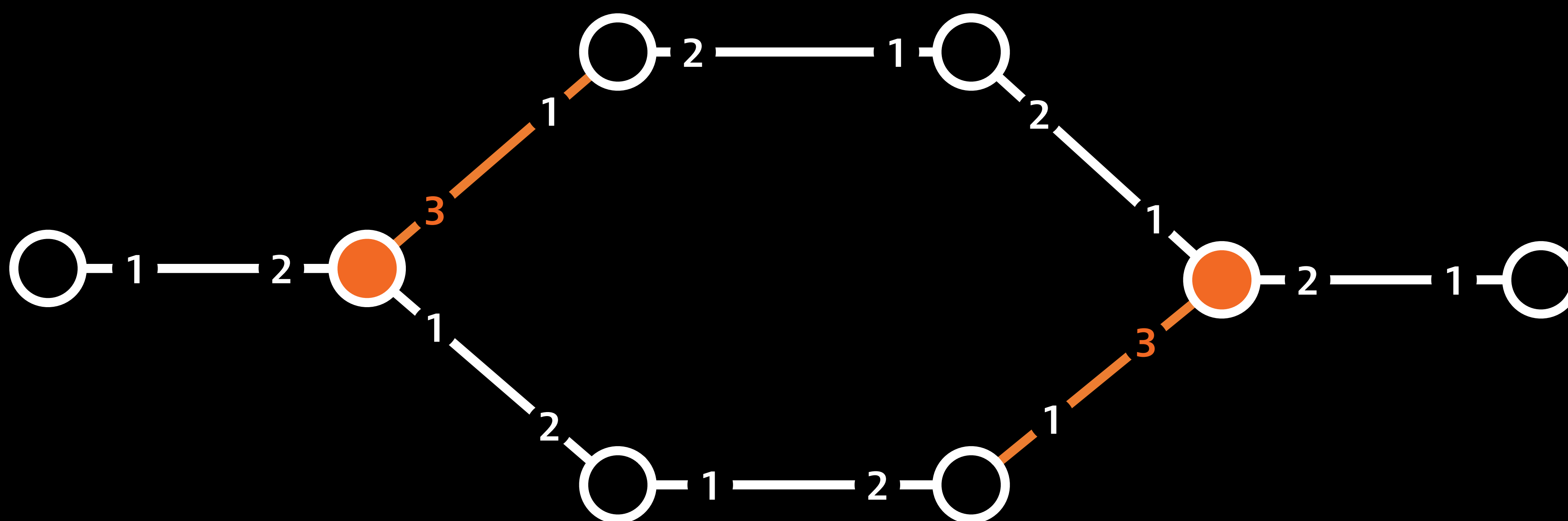


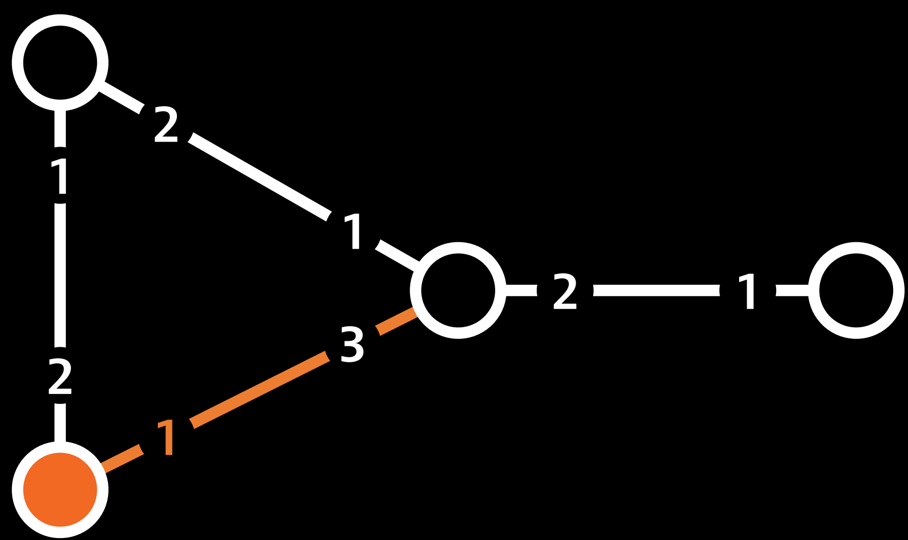
You are in a room
with three doors,
labeled 1, 2, and 3.
>





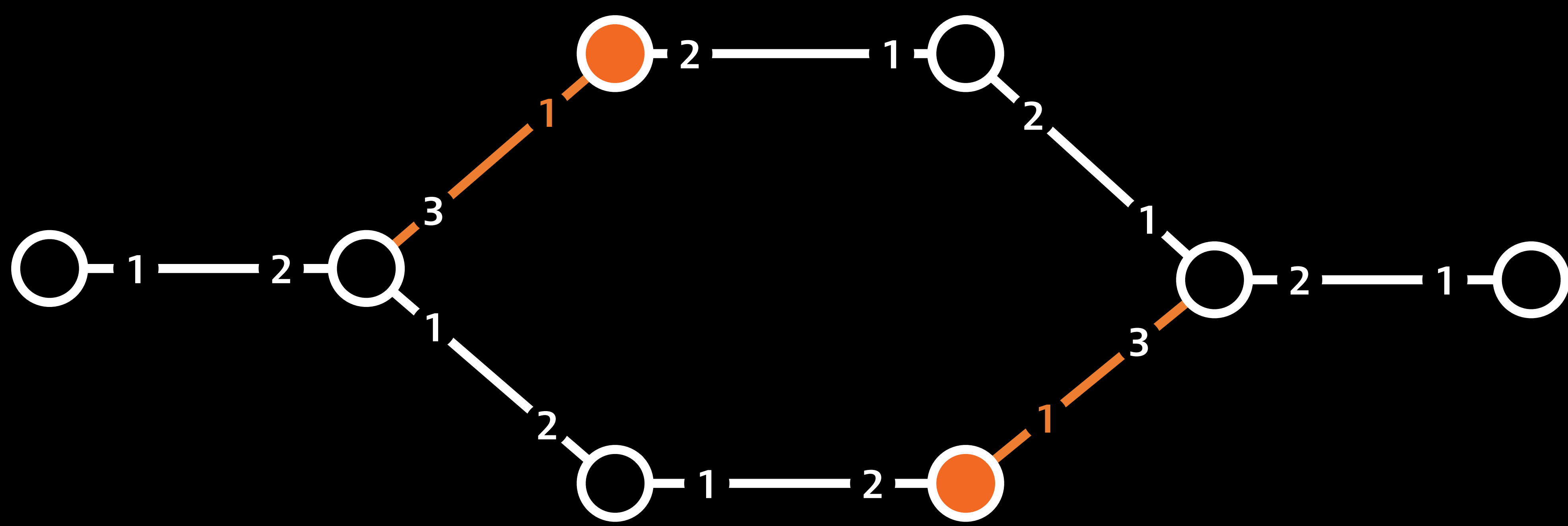
You are in a room
with three doors,
labeled 1, 2, and 3.
> open door 3

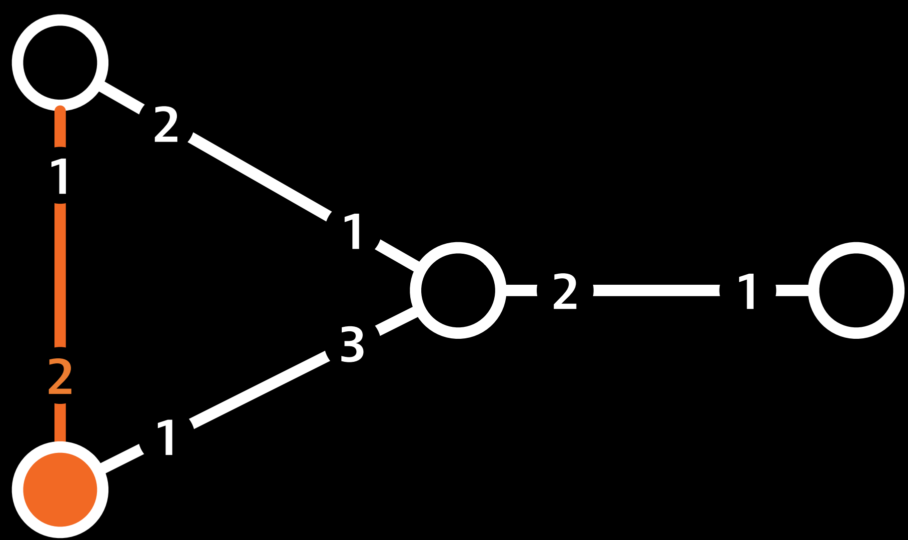




You enter a room with two doors, labeled 1 and 2. You just came in through doorway 1.

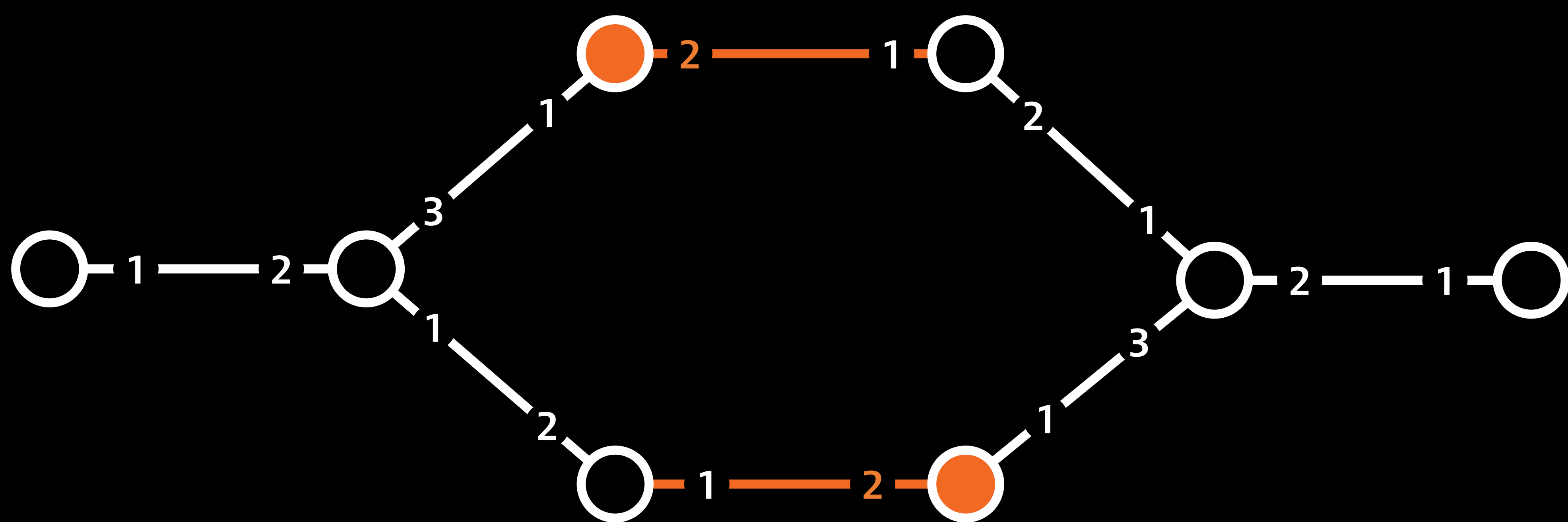
> _

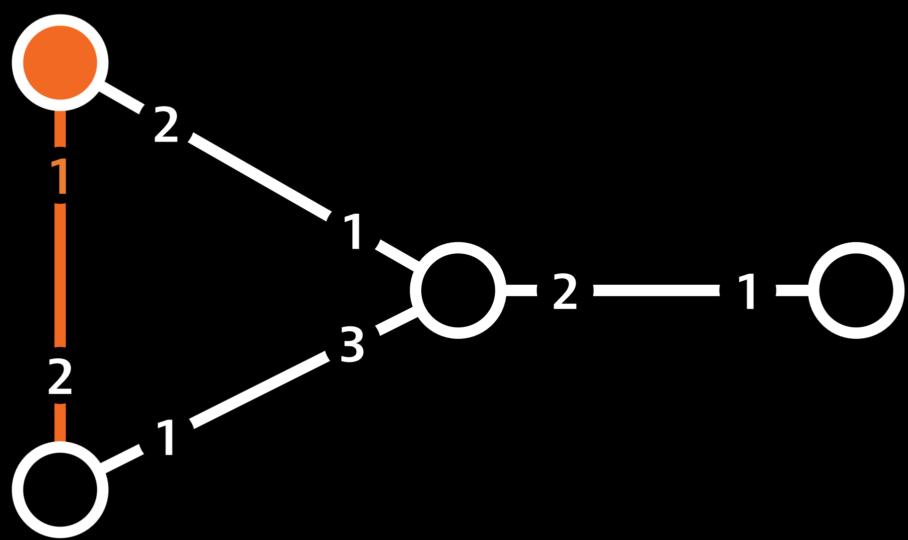




You enter a room with two doors, labeled 1 and 2. You just came in through doorway 1.

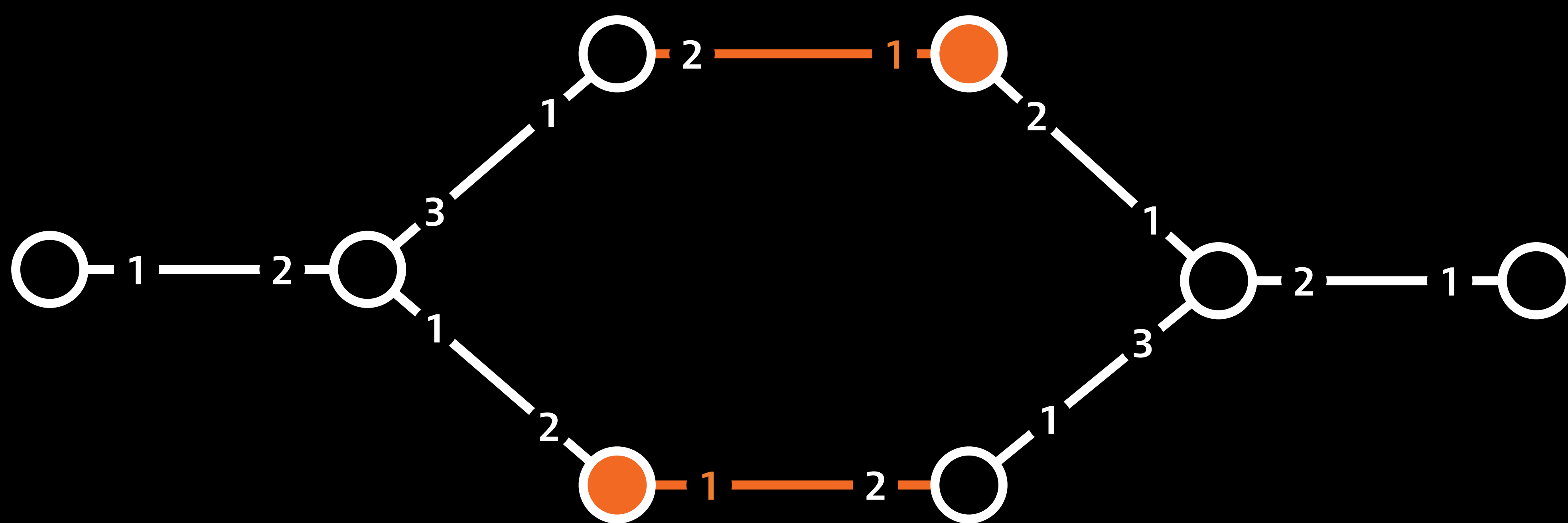
> open door 2

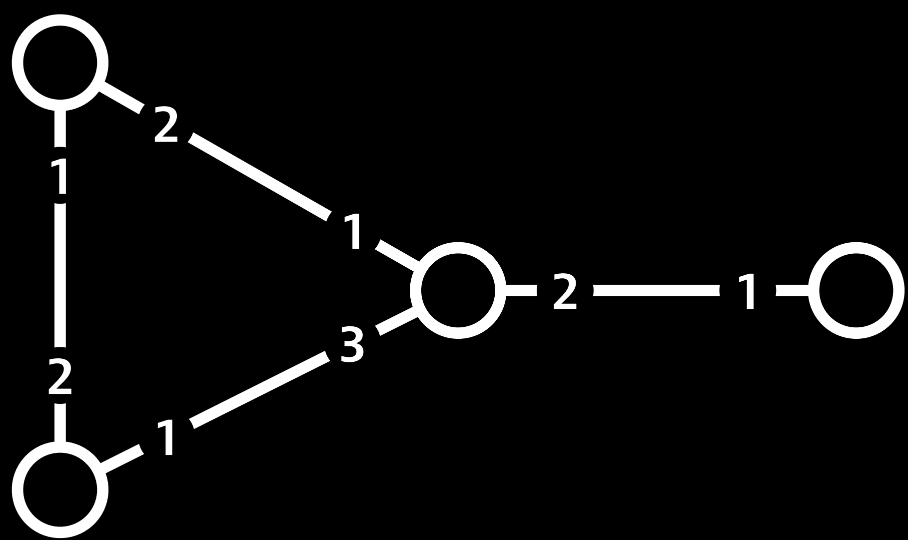


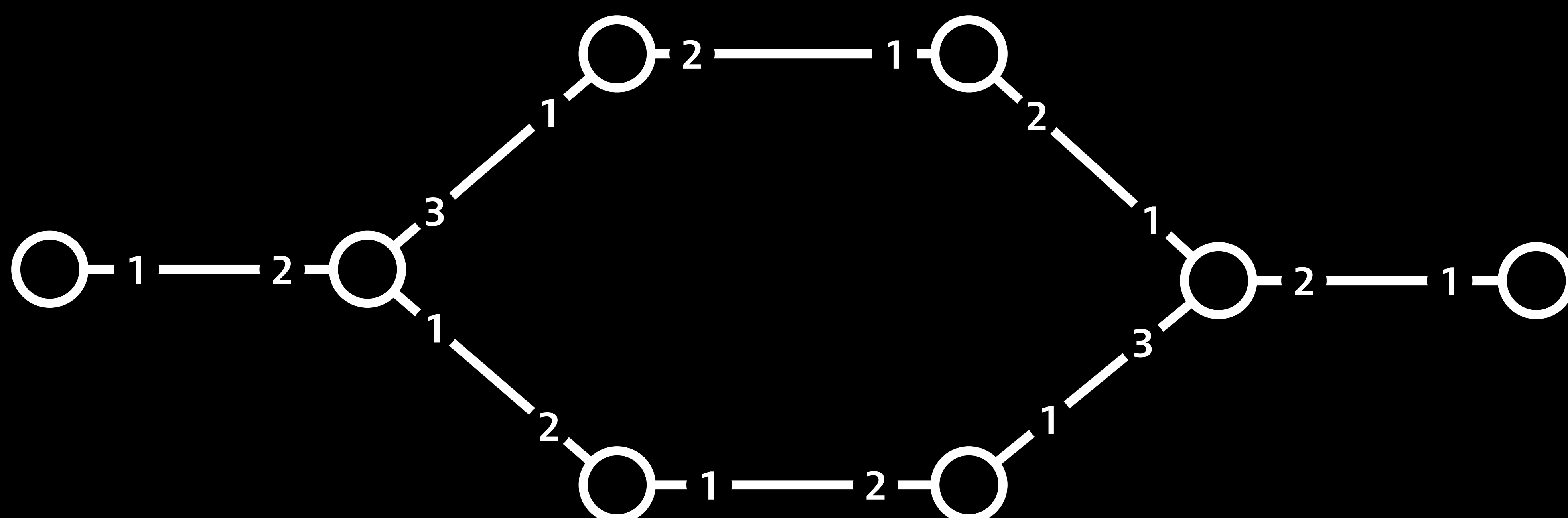


You enter a room with two doors, labeled 1 and 2. You just came in through doorway 1.

> _




2
1
1
2
3
1
2
1


1
2
3
1
2
1
2
1
1
2
1
2
1
3
2
1

High-level plan

- **Goal:**
  - show that problem X cannot be solved in the port-numbering model
- **General approach:**
  - construct port-numbered networks so that some nodes u, v, ... will always produce the ***same output***
  - show that if u, v, ... have the same output, then it is ***not a feasible solution*** for X

High-level plan

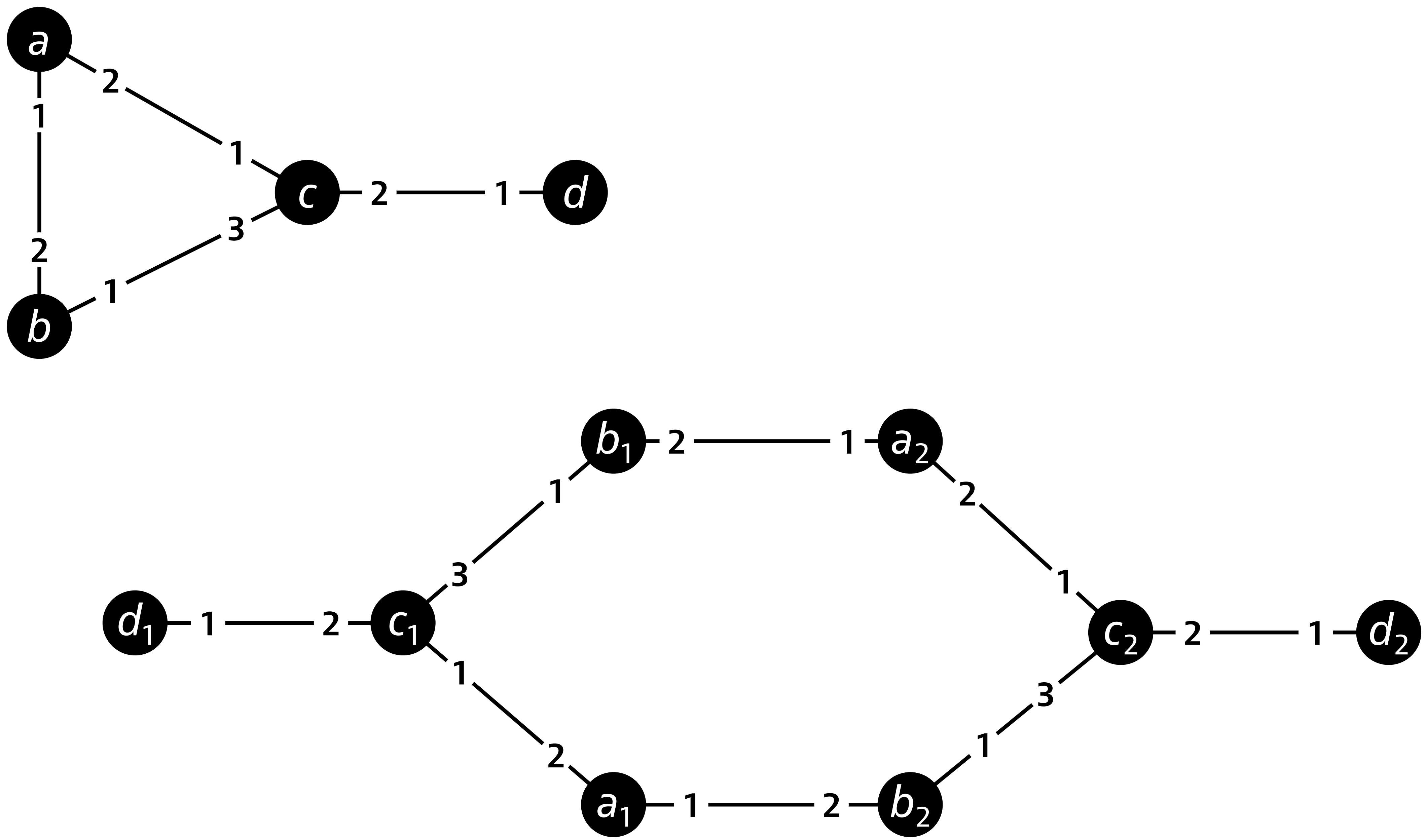
- **Goal:**
  - show that problem X cannot be so
    in the port-numbering model

Covering maps used here

- **General approach:**
  - construct port-numbered networks so that some nodes u, v, ... will always produce the ***same output***
  - show that if u, v, ... have the same output, then it is ***not a feasible solution*** for X

Covering map

- **Two port-numbered networks:**
  - $N = (V, P, \{p_v\}_{v \in V})$
  - $N' = (V', P', \{p'_v\}_{v \in V'})$
- **Surjection $f\colon V \longrightarrow V'$ that preserves:**
  - inputs
  - degrees
  - connections
  - port numbers


a
b
c
d
b_1
a_2
d_1
c_1
c_2
d_2
a_1
b_2

Covering map

- "Fools" any deterministic algorithm
- If $\boldsymbol{f}$ is a covering map from $\boldsymbol{N}$ to $\boldsymbol{N'}$, then:
  - v and $\mathrm{f}(v)$ have the same state *before* round 1
  - v and $\mathrm{f}(v)$ send the same messages in round 1
  - v and $\mathrm{f}(v)$ receive the same messages in round 1
  - v and $\mathrm{f}(v)$ have the same state *after* round 1

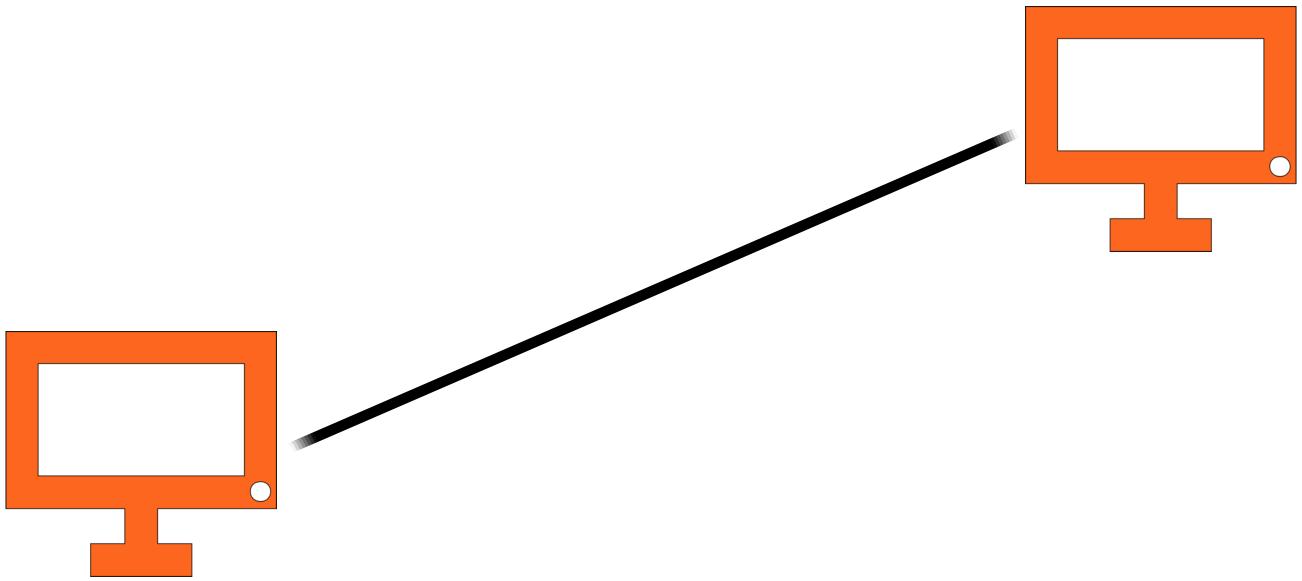
Covering map

- "Fools" any deterministic algorithm
- If $\boldsymbol{f}$ is a covering map from $\boldsymbol{N}$ to $\boldsymbol{N}'$, then:
    - v and $\mathrm{f}(v)$ have the same state *before* round T
    - v and $\mathrm{f}(v)$ send the same messages in round T
    - v and $\mathrm{f}(v)$ receive the same messages in round T
    - v and $\mathrm{f}(v)$ have the same state *after* round T

Common steps

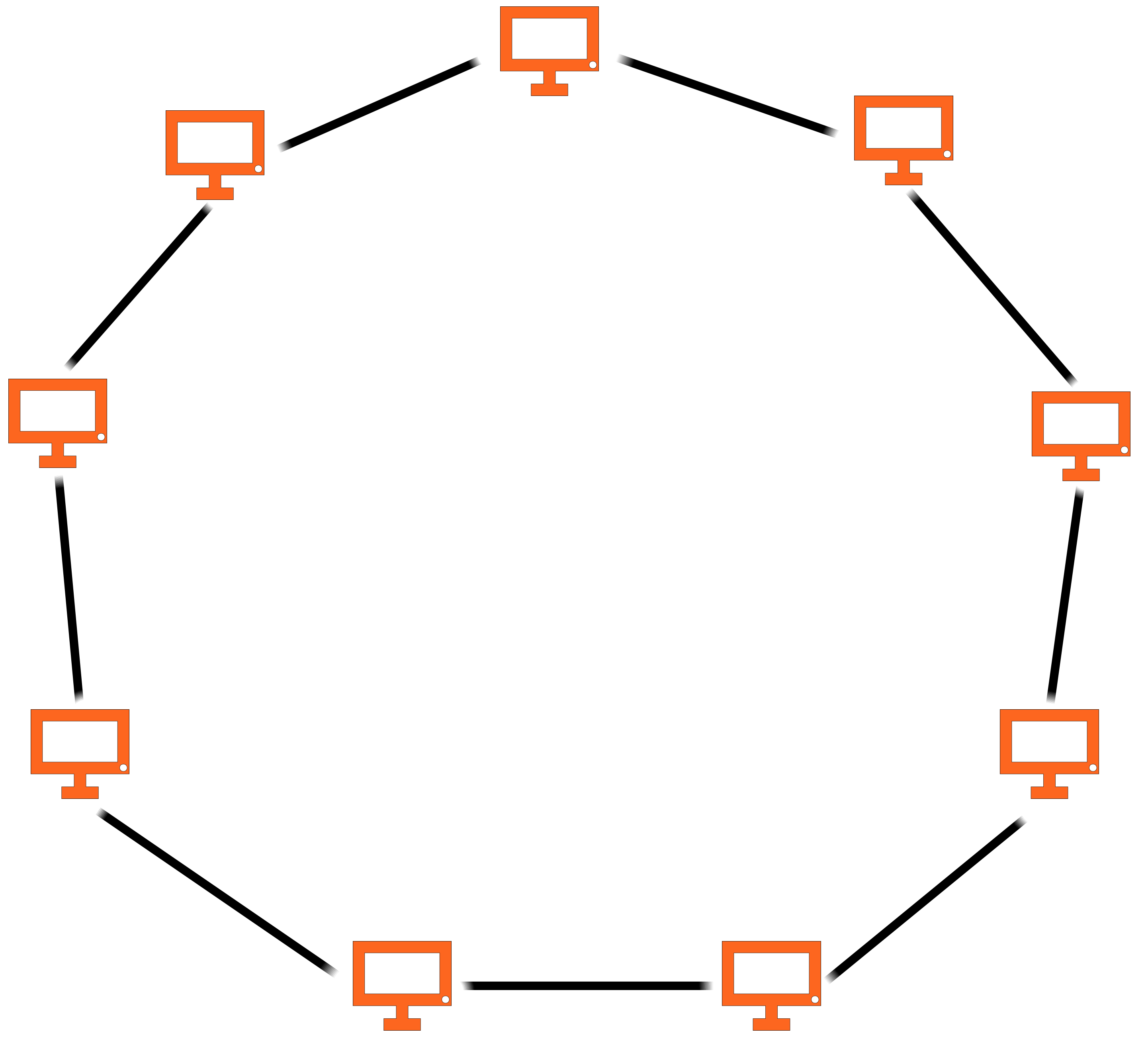
- Starting point: graph problem X
- Which graph $\boldsymbol{G}$ would be a “hard instance”?
- How to choose a port numbering $\boldsymbol{N}$ of $\boldsymbol{G}$?
- How to choose the other network $\boldsymbol{N'}$?
- How to construct mapping from $\boldsymbol{N}$ to $\boldsymbol{N'}$?

Example: 2-node path

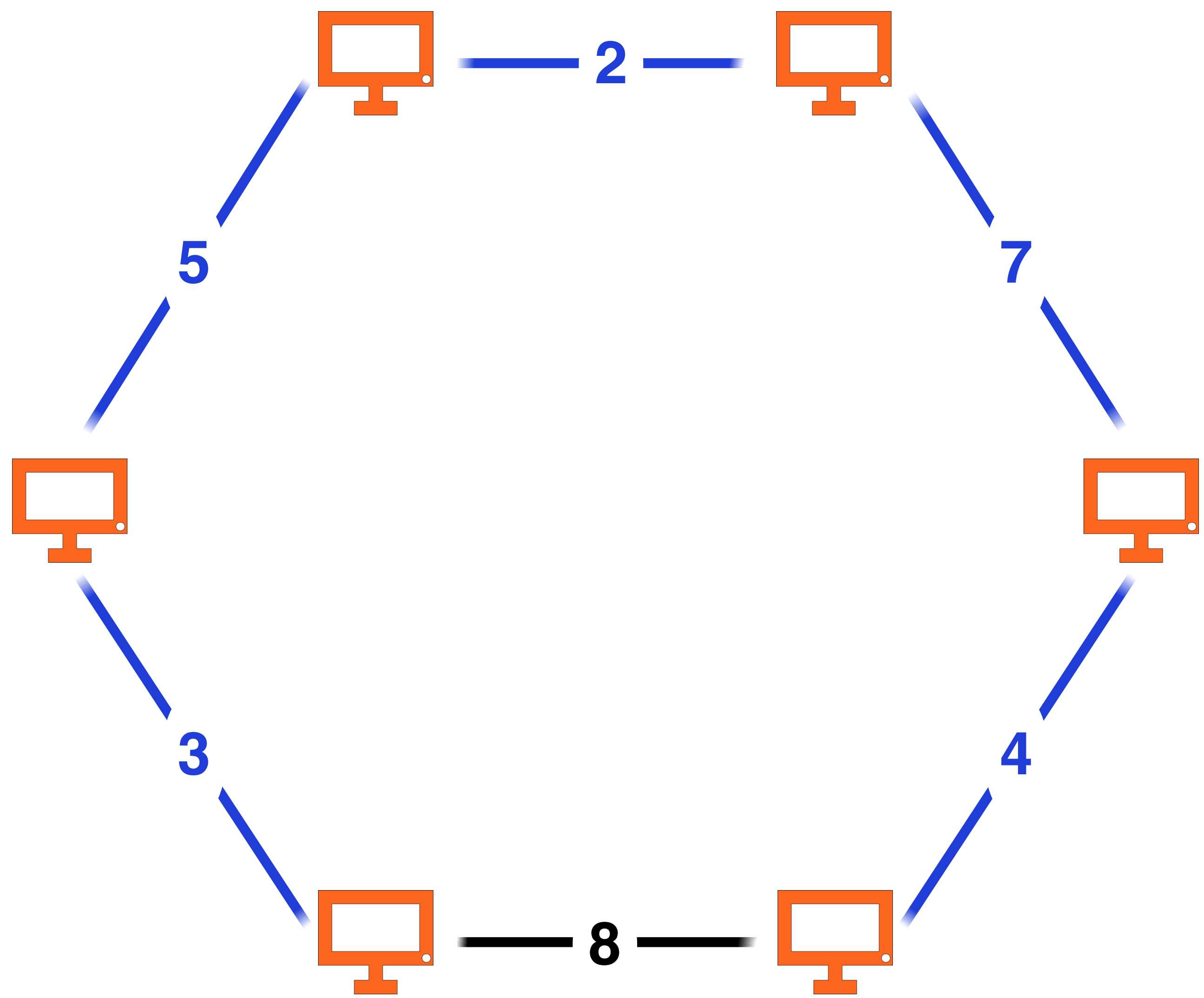


Example: 4-node path

Example: cycles



Example: spanning trees



Common setup

- *N is the network we care about*
  - simple port-numbered network
  - well-defined and interesting underlying graph
- *N' is something strange*
  - not necessarily a simple port-numbered network
  - running A in N' makes no sense
  - introduced only to analyze what happens when we run A in N

LOCAL model

What can you do in T rounds in the LOCAL model?

Best that we can do?

- No restrictions on message size

Best that we can do?

- No restrictions on message size
- No restrictions on local computation

Best that we can do?

- No restrictions on message size
- No restrictions on local computation
- Possible to do:
  in each round, tell each neighbor everything you know!

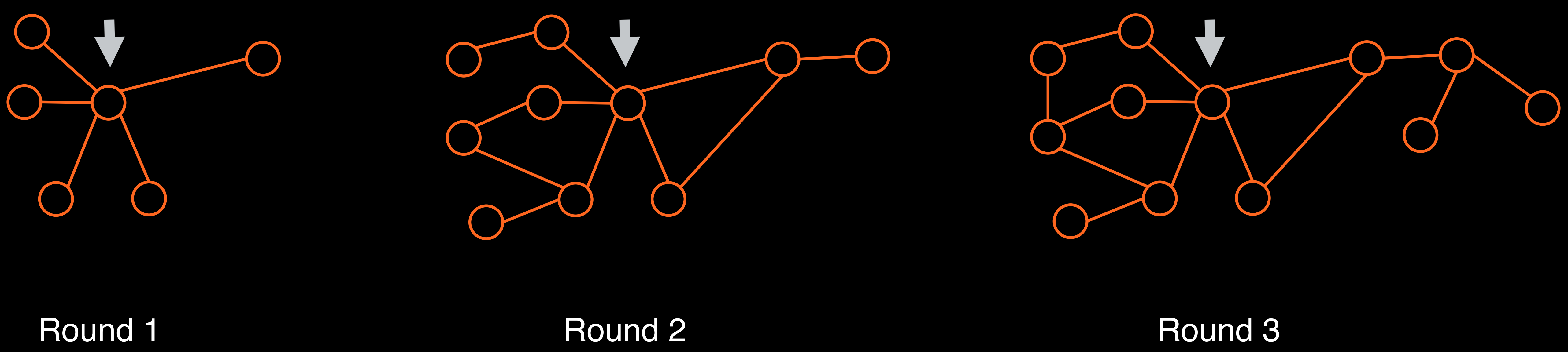
Best that we can do?

- No restrictions on message size
- No restrictions on local computation
- Possible to do, and best that you can do:
  in each round, tell each neighbor everything you know!

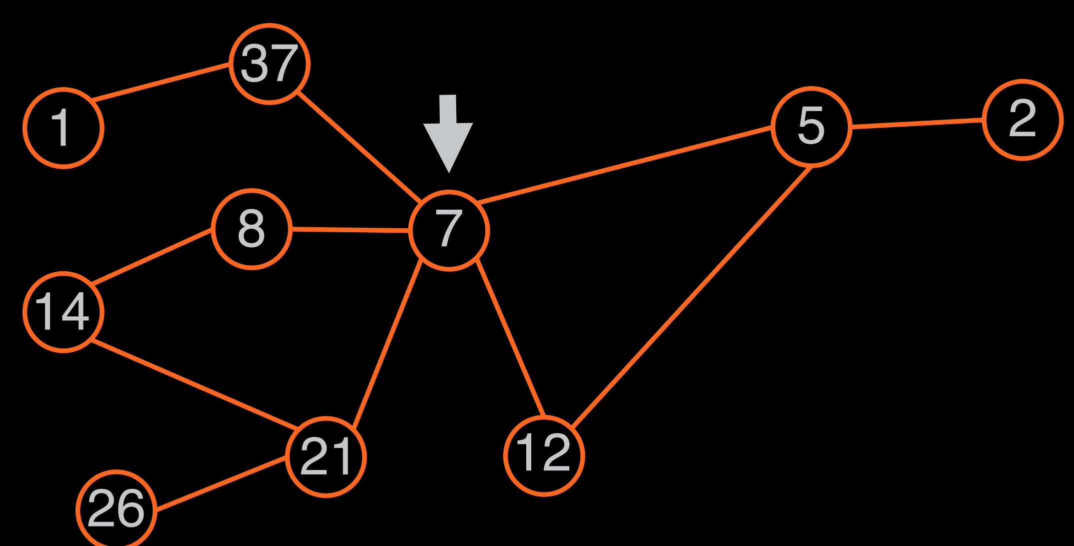
At best:

in T rounds, each node can

learn its radius-T neighborhood

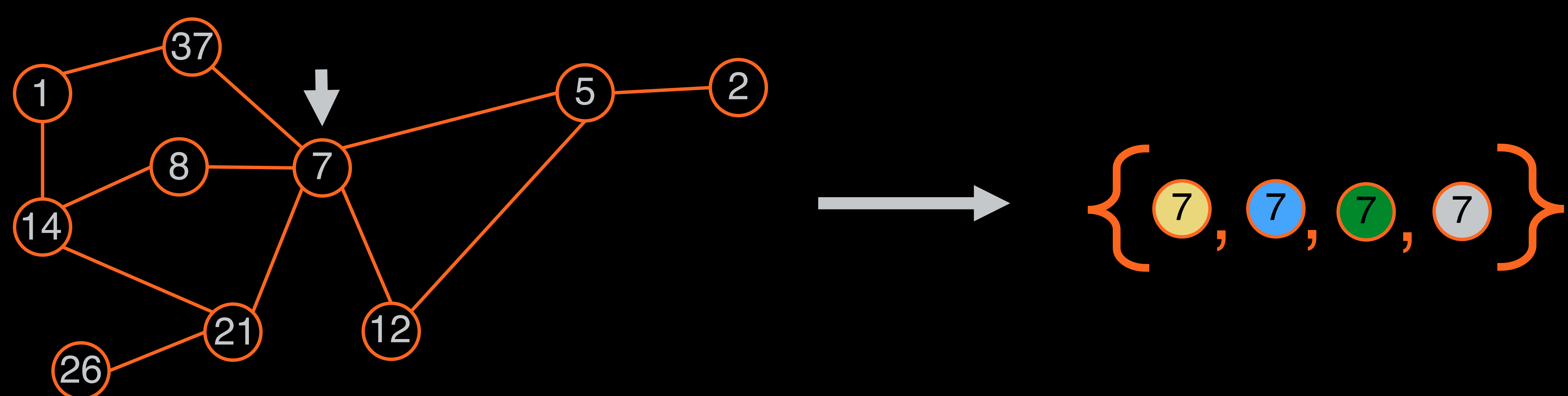


Your local output
is a function of your
local neighborhood



local neighborhood

T-round algorithm is just a mapping from radius-T neighborhoods to local outputs



$T = 2$

Running time =
number of communication rounds
until all nodes stop and produce
their local outputs

=

Locality =
how far do you need to see
in the graph to choose
your own part of the solution

New plan

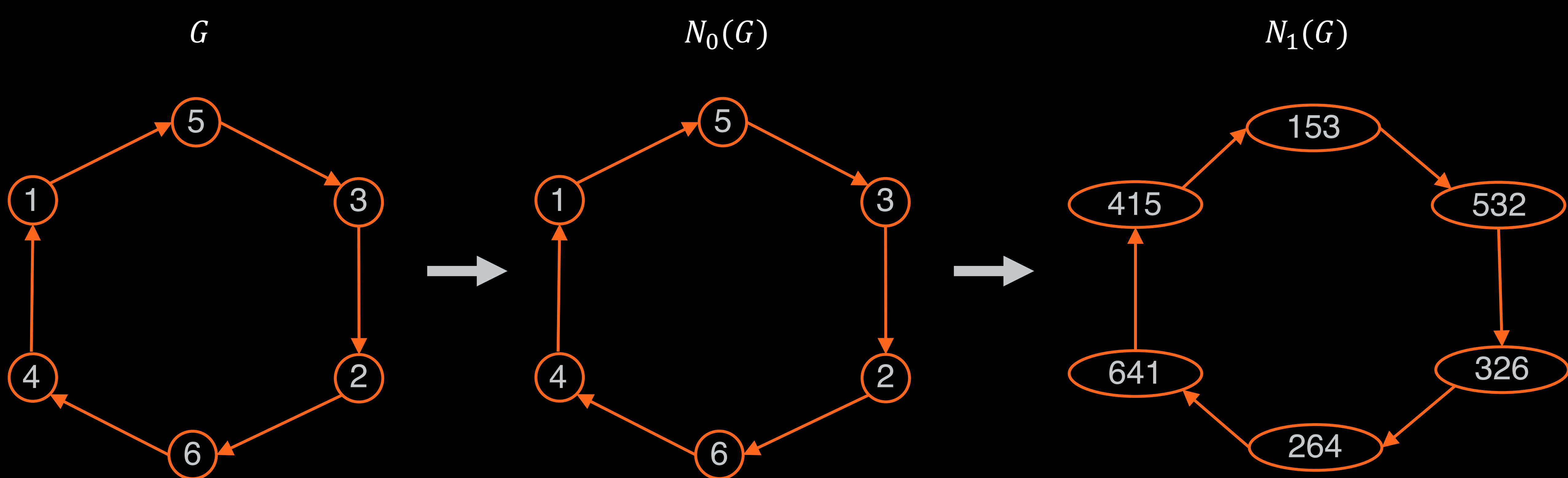
- Algorithm A runs in T rounds and solves problem X
  - A is a mapping from radius-T neighborhoods to local outputs
- Construct examples where such a mapping cannot solve X correctly
  - Problem X is not solvable in T rounds

Example: 2-coloring a path

What about 3-coloring a ~~path~~ directed cycle?

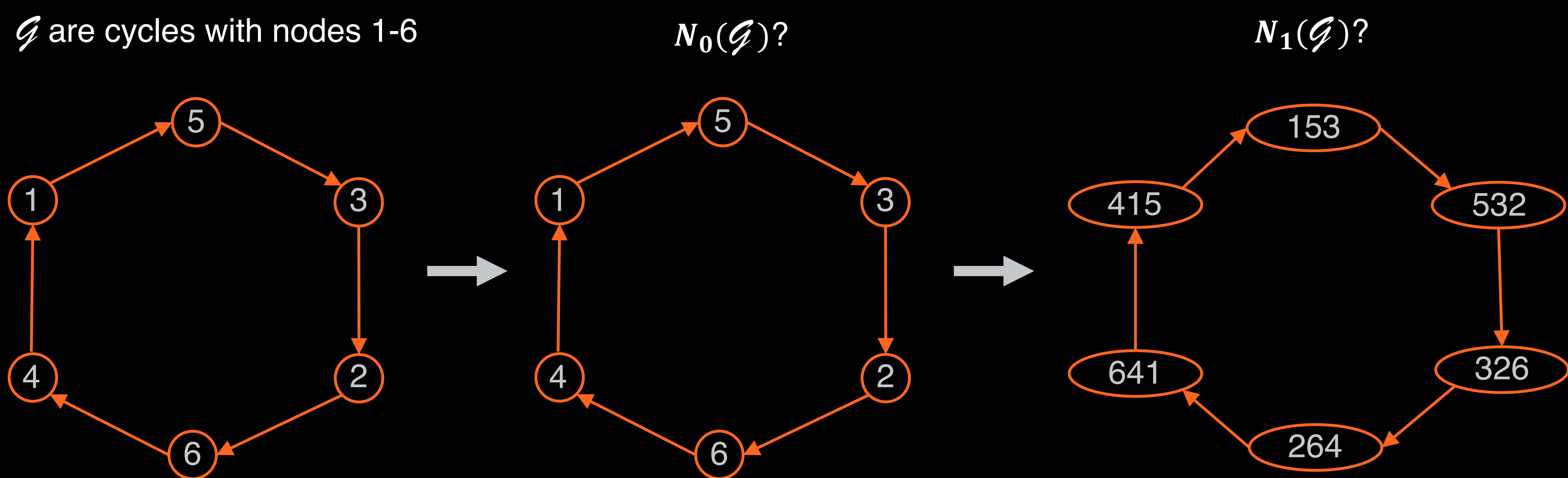
Neighborhood graph

The r-**neighborhood graph** $\boldsymbol{N_r(G)}$ consists of all r-hop views of G (for all nodes) which are connected iff they could originate from two adjacent nodes.



Neighborhood graph

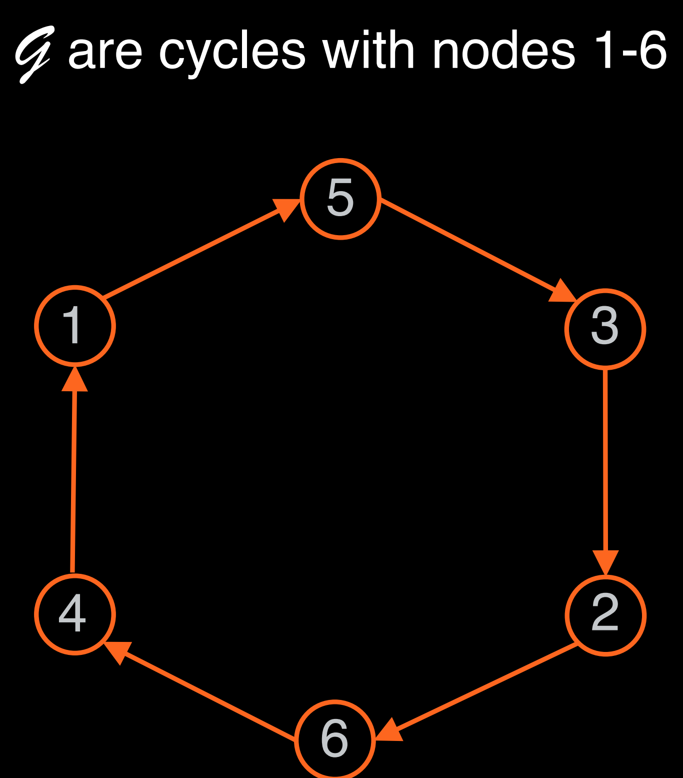
Let $\mathscr{G}$ be a graph class. The r**-neighborhood graph** $N_r(\mathscr{G})$ consists of all r-hop views of $\mathscr{G}$ (for all nodes) which are connected iff they could originate from two adjacent nodes.

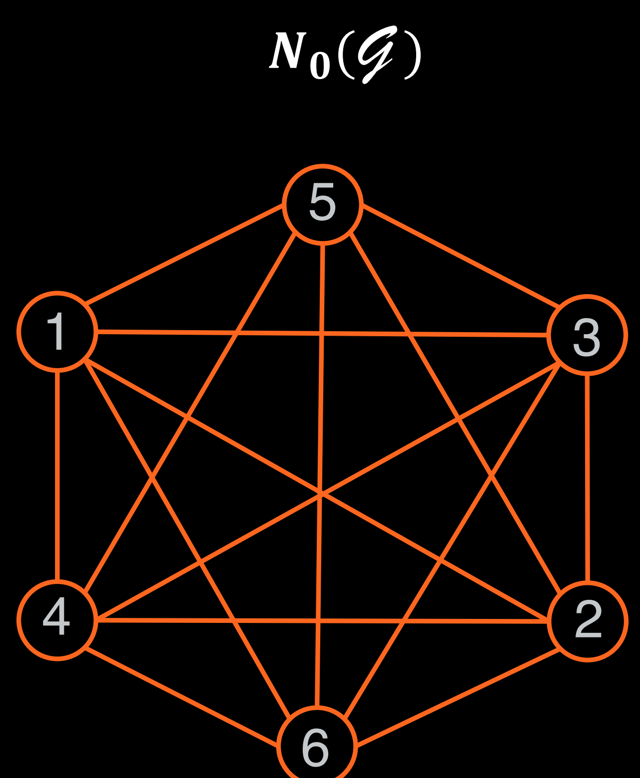


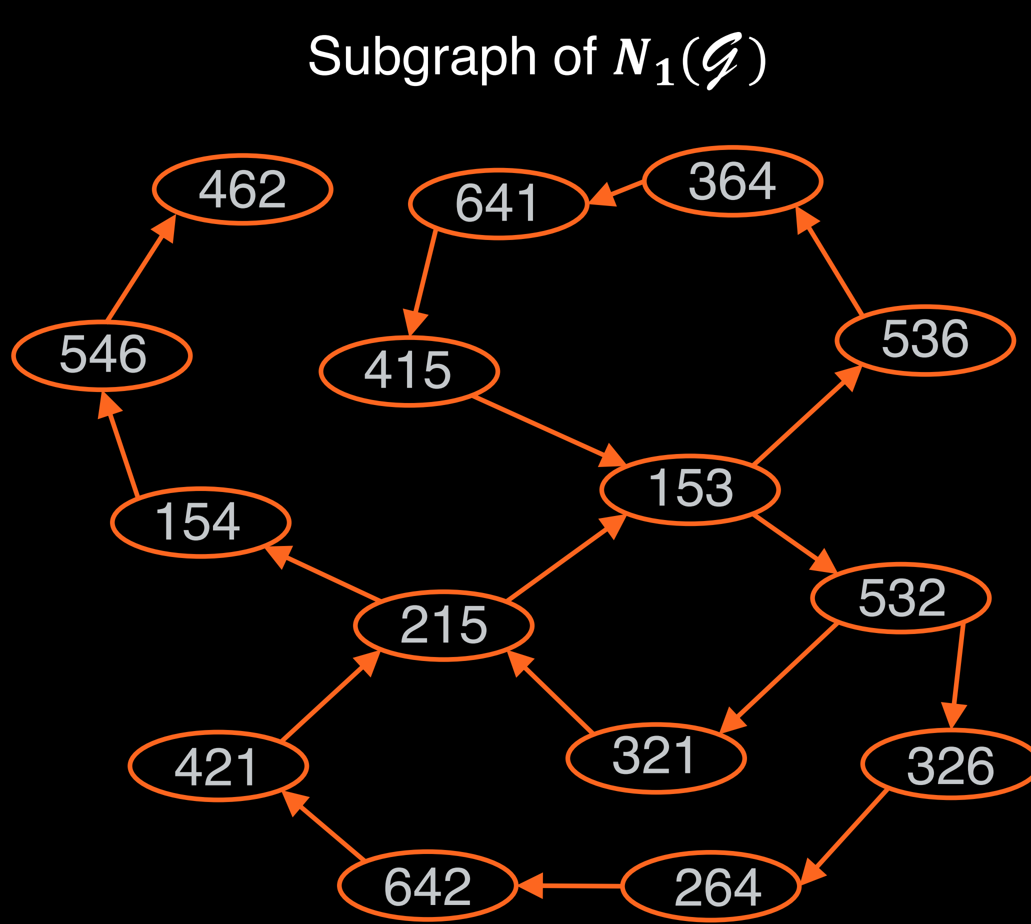
These are just subgraphs of the respective neighborhood graphs!

Neighborhood graph

Let $\mathcal{G}$ be a graph class. The r**-neighborhood graph** $N_r(\mathcal{G})$ consists of all r-hop views of $\mathcal{G}$ (for all nodes) which are connected iff they could originate from two adjacent nodes.

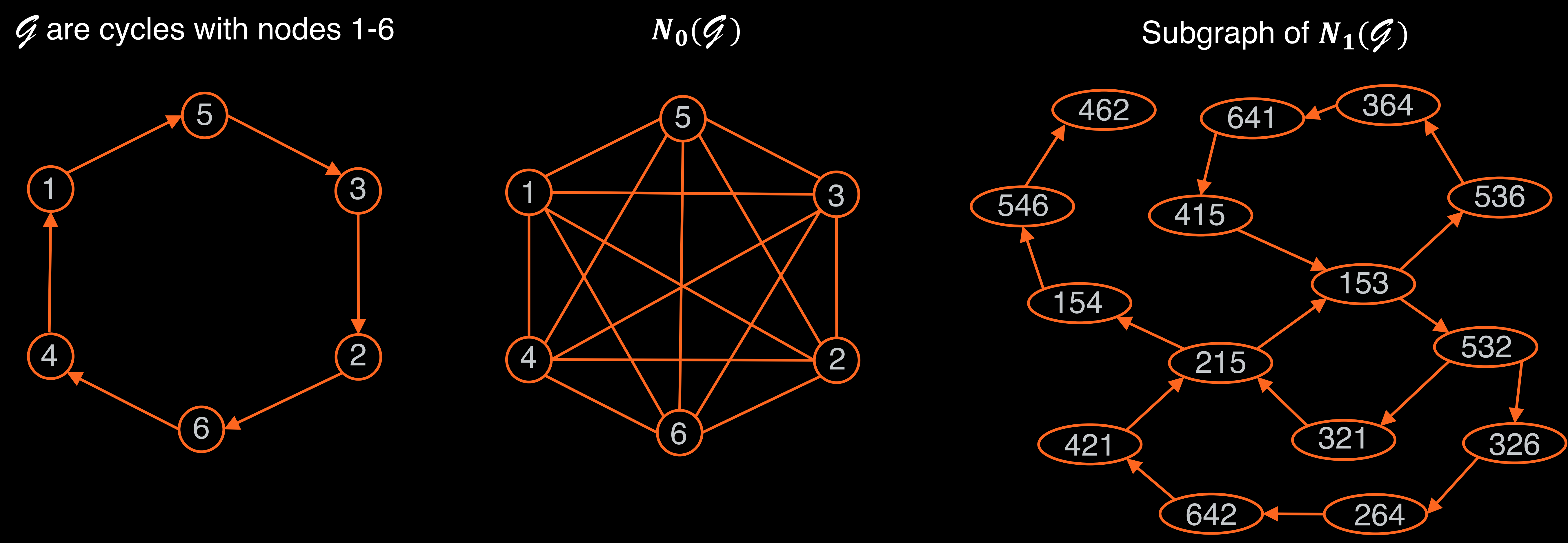






Neighborhood graph

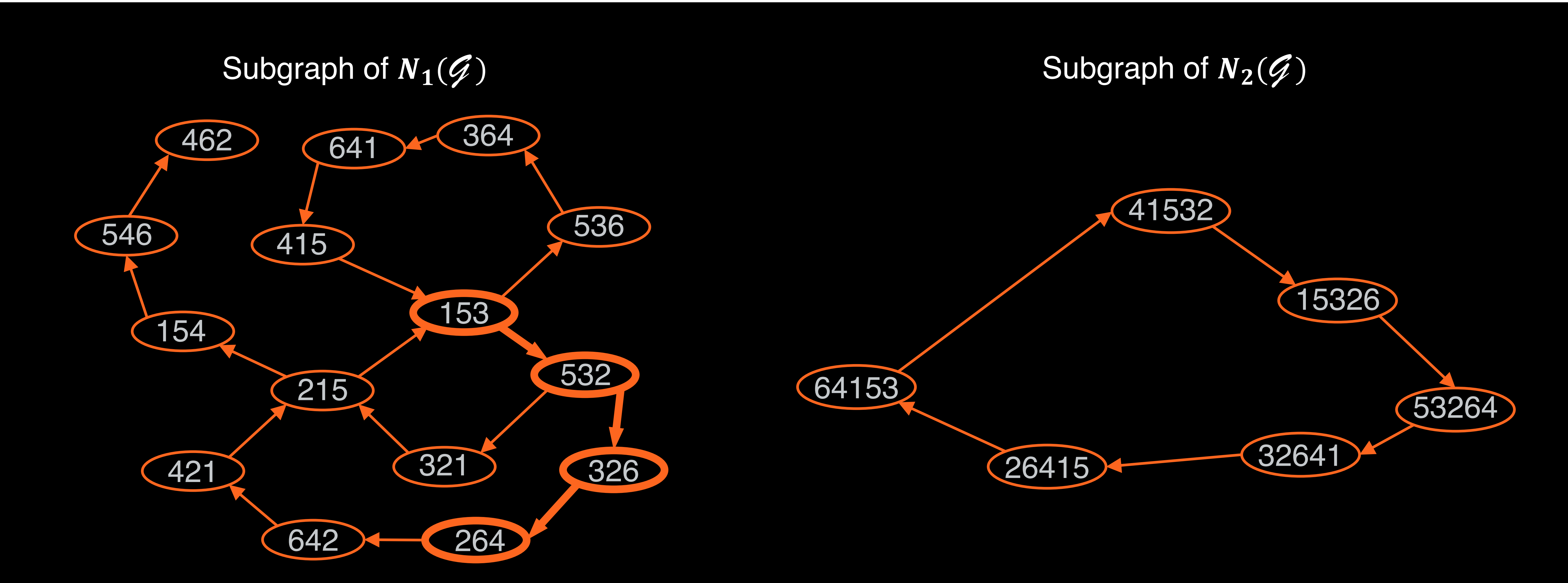
There is an r-round algorithm that colors graphs $\mathcal{G}$ with c colors iff the chromatic number of the neighborhood graph is $\chi(N_r(\mathcal{G})) \leq c$.



Example: 4-cycle

Neighborhood graph

Intuitively:
the chromatic number declines for larger r, in logarithmic factors “per hop”



Lower bound idea

- $\chi(N_r(\mathcal{G}))$ gives us a lower bound on the number of colors that any algorithm must use
- With each "hop", the chromatic number of $N_r(\mathcal{G})$ goes down by at most a log factor
- After r hops, the lower bound on the chromatic number is

$$\mathrm{c} > \chi(N_r(\mathcal{G})) \geq \log^{r-1} n > \log^* n$$

Lower bound idea

- $\chi(N_r(\mathcal{G}))$ gives us a lov
  number of colors that a

Here one needs to consider a subgraph of the neighborhood graph, see lecture notes

- With each "hop", the chromatic number of $N_r(\mathcal{G})$ goes down by at most a log factor

- After r hops, the lower bound on the chromatic number is

$$\mathrm{c} > \chi(N_r(\mathcal{G})) \geq \log^{r-1} n > \log^* n$$