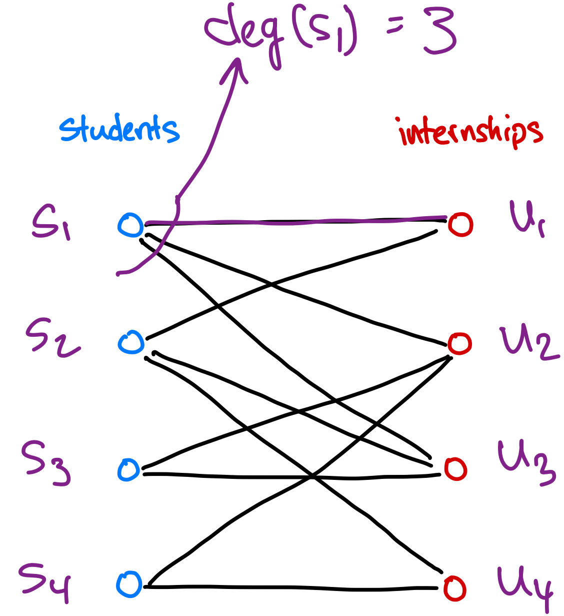
Lecture 03: Port Ordering and Locality

Outline

1. Recap of Distributed Stable Matchings
2. Port Ordering (PO) Model
3. Stable Matchings in PO Model

Port-Ordering (PO) Model



$V = \{s_1, \ldots, s_4, u_1, \ldots, u_4\}$

$E = \{(s_1, u_1), (s_1, u_2), \ldots$
$(s_4, u_4)\}$

Network

Modeled as a graph $G = (V, E)$

- V = set of nodes a.k.a. vertices
- E = set of edges
  - each edge is a pair of nodes
  - edges = connections between nodes

Variation nodes may have distinct roles

- e.g. students & internships

Terminology vertices $u, v \in V$ are neighbors if (u,v) is an edge in E

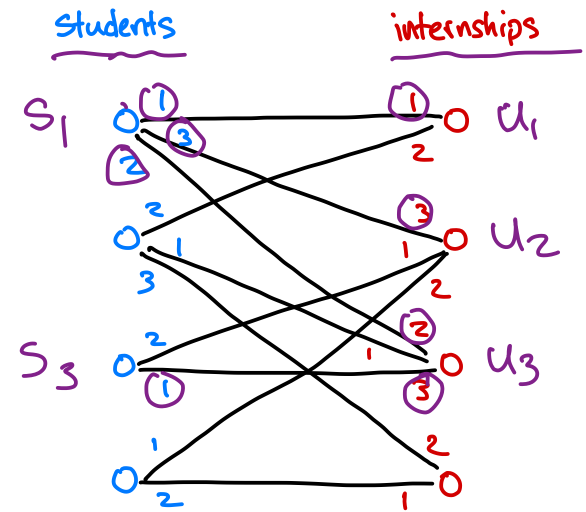
- u and v are incident to (u,v)

The degree of v, denoted $\deg(v)$ is

$$\deg(v) = \#\text{ of neighbors of } v$$
$$= \#\text{ incident edges}$$

Port - ordering

Each vertex v has an ordering of its neighbors : 1, 2, 3, ..., deg(v)



Crucial Vertices do not have "unique identifiers"

- "anonymous network"
- nodes distinguish neighbors only by port number

Note: Stable matching instances ⟹ PO Networks

students ⟿ blue nodes
internships ⟿ red nodes

blue nodes only have red neighbors and vice versa → graph is bipartite

preferences ⟿ port ordering

→ top ranked partner gets port 1, etc.

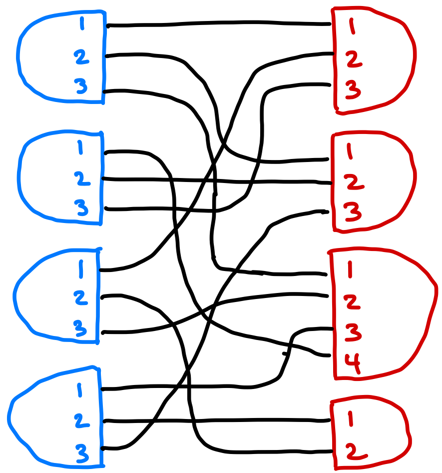
SM Instance

	1	2	3	4
Anna:	a	b	c	
Beck:	c	b	a	
Cameron:	a	d	c	
Daniel:	c	d	b	

	1	2	3	4
a:	A	C	B	
b:	A	B	D	
c:	A	C	D	B
d:	D	C		

$\Downarrow$

PO Network



Computation in PO model

Execution proceeds in $\boxed{\text{synchronous rounds}}$

Each round, each node does:

1. receive messages sent by neighbors in prev. rounds
2. perform local computations

   → no restrictions on complexity
3. sends messages to neighboring nodes,

   or halts and produces output

More formally a PO protocol consists of

Gale - Shapley PO Protocol ?

Sets:

- Inputs = allowable local inputs → pref lists, role
- States = allowable local states → round #, cur match, prev. msgs received
- Outputs ⊆ States halting states w/ outputs → port label of match
- Messages = allowable messages → "apply", "reject", "halt"

Functions (d = degree of node)

- init : Inputs → States

  determine initial state from initial input

- send : States → Messages^d

  determine the d messages to send to neighbors from current state

- receive : States × Messages^d → States

  determine how to update state from received messages

Gale Shapley as a PO protocol

Procedure for students:

Initialize:

cur = 1

$\boxed{\text{each round } i, \text{ do}}$

if i = 1, send "apply" to cur

if received "reject" from cur in round i-1

if cur = my degree, return $\boxed{\perp}$ and halt

else, cur ← cur+1, send "apply" to cur

return cur and halt

Procedure for internships

Initialize:

cur = $\boxed{\infty}$ no app received

$\boxed{\text{each round } i, \text{ do}}$

set $\boxed{\text{reject}}$ ← ∅

for each $\boxed{j}$ from which received "apply" in round i-1

if $\boxed{j < cur}$ rec'd app from preferred applicant

add cur to reject

set cur ← j

else

add j to reject

for each j in reject

send "reject" to j

return cur

Does this work?

Problem: when does loop terminate?

Issue with Gale Shapley in PO model:

TERMINATION

When is the procedure complete?

Previously: if there are $\boxed{m}$ total acceptable partners (i.e. edges in graph), then GS returns after $\leq m$ iterations.

- Note m iterations of GS $= 2m$ rounds of PO model

[Could we just detect when $2m$ rounds have elapsed and halt/return then?]

→ can we detect earlier termination?

Gale Shapley Algorithm

While some student is active

1. All active students apply to next most favored internship
2. Each internship defers best app so far, rejects others

Return set of deferred pairs

Eg. 1000 total applications

⇒ wait 2000 rounds, and matches are finalized.

The Story so far

1. Gale-Shapley algorithm can be implemented in the PO Model* ⟵ how to detect termination?

2. GS will eventually find a stable matching....

3. ... but nodes do not detect when the process should terminate

One option. Tell each node the value of n (= # of agents) or m (= # of applications)

- How does this solve the problem?
- What are issues w/ this solution?

We will see:

1. Termination cannot be detected in the PO model w/out some global information

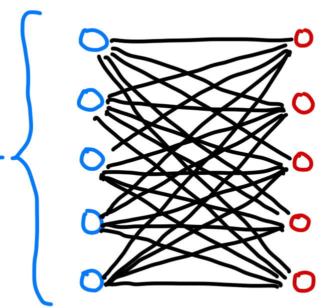
2. Termination can be detected if nodes additionally have unique identifiers

PO model + unique identities = "LOCAL model"

3. No distributed algorithm can find a stable matching (much) faster than GS on all instances

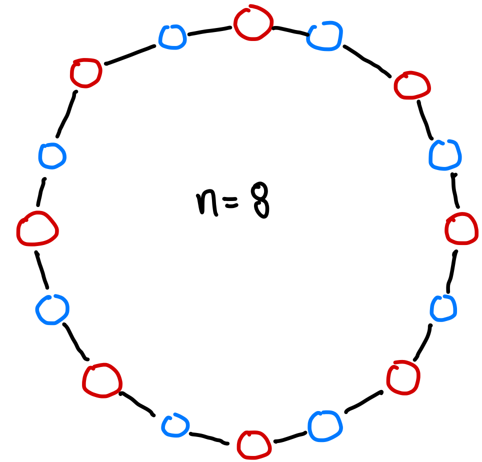
→ still much room for improvement in "small diameter" networks

→ still unknown: how fast can a distributed solution be for complete preferences

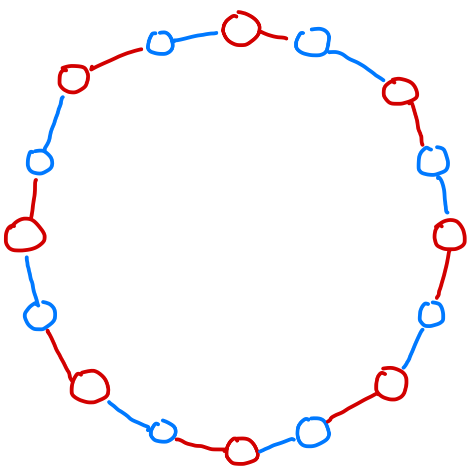


An Illustrative Family of SM Instances

1. n students, n internships
2. each agent has only 2 acceptable partners
3. the graph of acceptable partners forms a cycle



Some preferences



- students prefer blue neighbor to red neighbor
- internships prefer red neighbor to blue neighbor

Questions.

1. Which SM does GS algorithm find?
2. What other SMs are there?

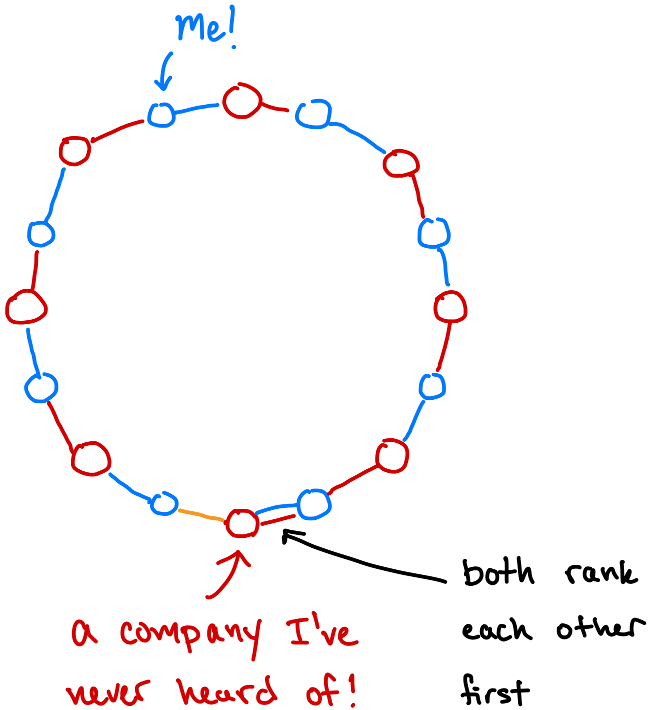
Recall

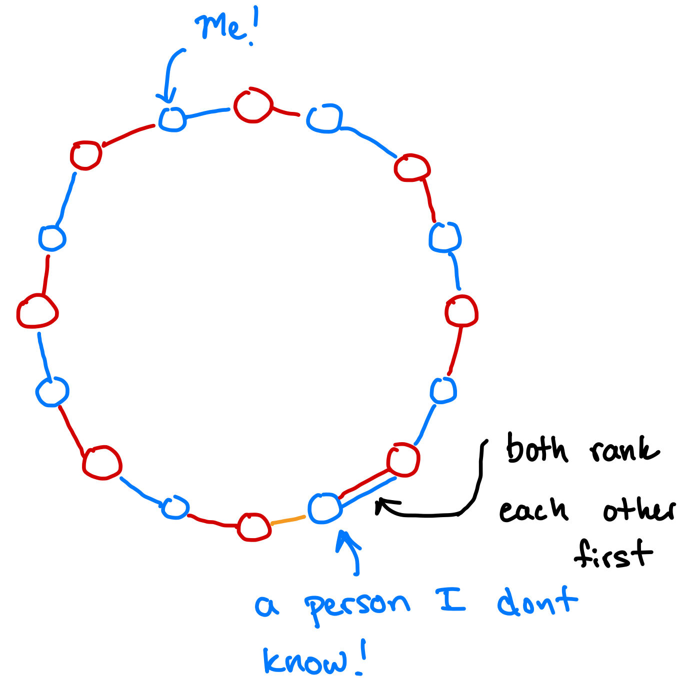
- can sometimes have multiple possible stable matchings
- here's how to find 2 (if possible):
  - run GS alg w/ students applying to internships
  - run GS alg. w/ internships applying to students

Fact. If the SM instance has multiple stable matchings then procedures above will give different stable matchings

$\Rightarrow$ if procedures above give same SM, then it is the unique SM for the instance.

Slight Modifications





What happens now?

The Moral.

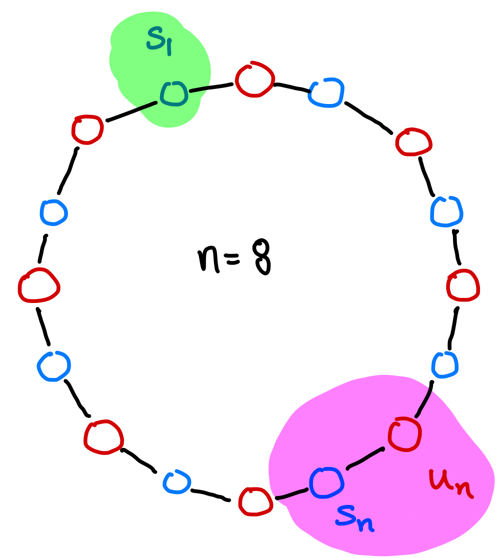
My (correct) output depends on input (prefs) of some company I've never heard of and some person I don't know!

- correct local output requires knowledge about nodes that are <u>far away</u> in the network

"A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable."

— Leslie Lamport

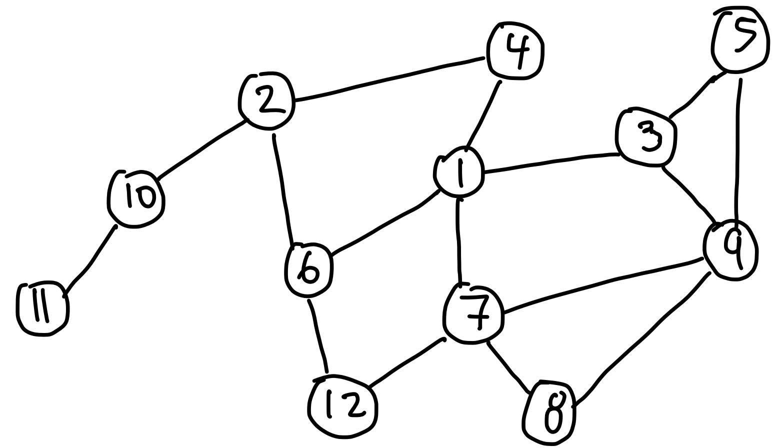
<u>Obs</u>. Correct output of S_1 depends on input of S_n and U_n, regardless of what algorithm is used to find SM



<u>Next</u>. We will use this observation to show that <u>no distributed algorithm</u> (in PO or LOCAL model) can find SMs much faster than GS on some instances

A Bit More Graph Theory

- $G = (V, E)$ a graph
- edges labelled w/ port ordering
- vertices have local inputs
  (possibly unique identifiers)

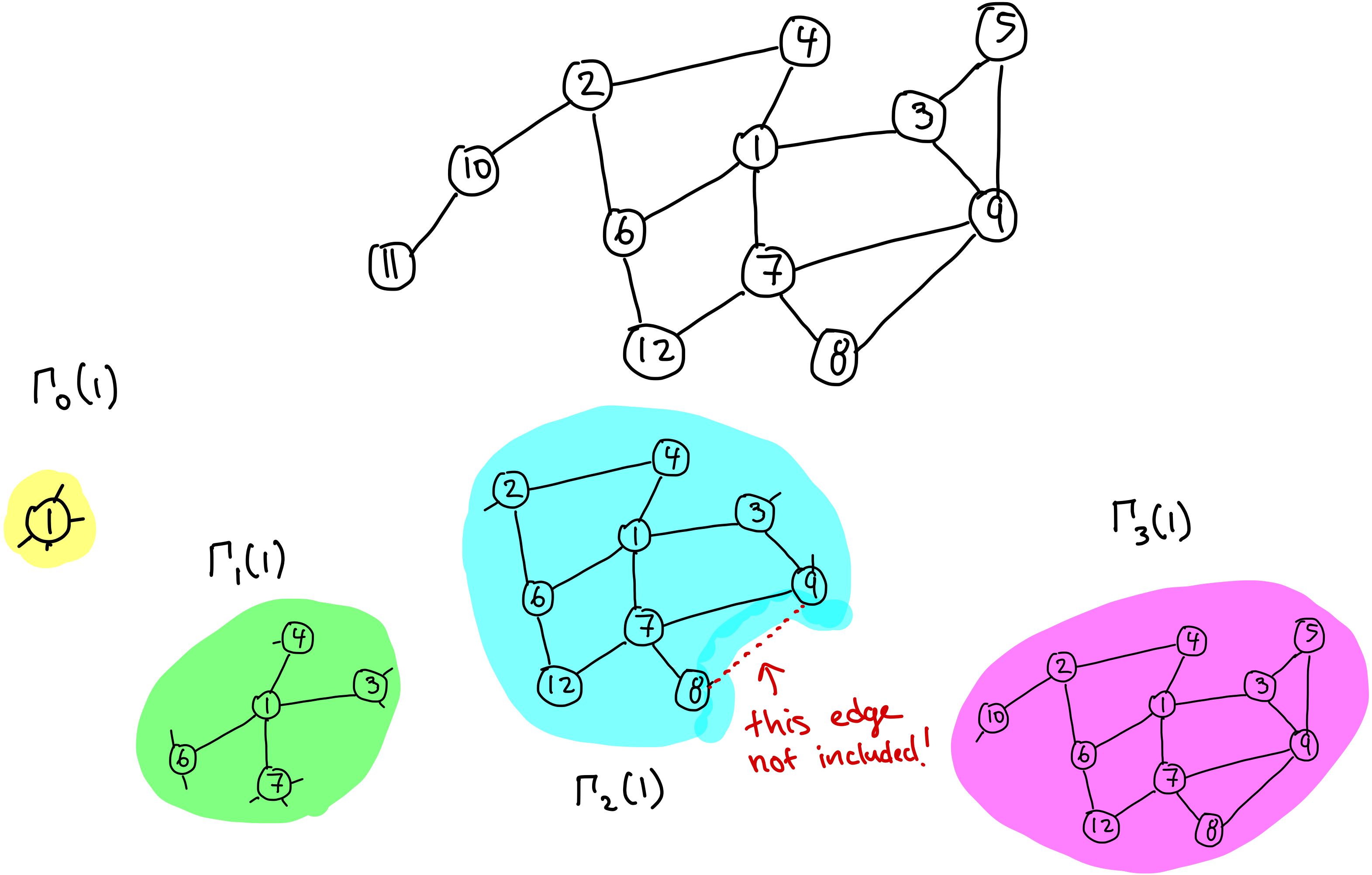


Def. Given vertex v, distance d the d-neighborhood of v is the subgraph of G consisting of all vertices within distance d of v and edges w/ one endpoint within distance $d-1$ of v

$$\Gamma_d(v) = (V', E')$$

V' = vertices within distance d of v

E' = edges w/ one or more endpoints within dist $d-1$ of v


Γ₀(1)
Γ₁(1)
Γ₂(1)
Γ₃(1)
this edge not included!

Connecting Neighborhoods and Protocols.

Goal. In any distributed protocol, messages sent in round r are determined by distance $r-1$ neighborhoods.

- Intuitively: data can only travel one "hop" per round
  - $\Rightarrow$ takes $r-1$ rounds to travel $r-1$ hops
  - $\Rightarrow$ by round r, can only have gotten info. from nodes within dist $r-1$

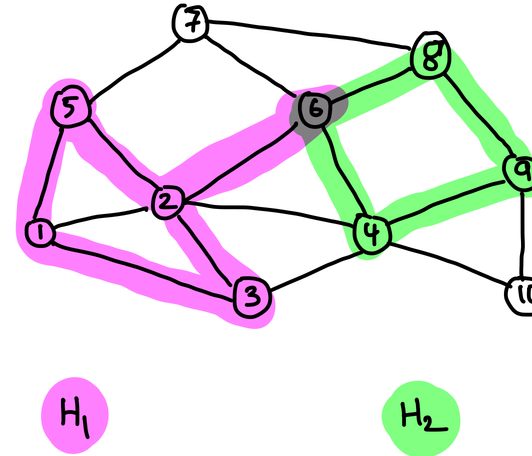
- Formally: prove by induction that each v's state in round r is determined by the initial states of nodes in $\Gamma_{r-1}(v)$
  - msgs sent in round r are determined by state in round r.

Union of Subgraphs

Suppose H_1, H_2 are sub-graphs of a graph $G = (V, E)$. Then the <u>union</u> of H_1 and H_2, denoted $H_1 \cup H_2$, consists of

(1) all vertices in H_1 or H_2

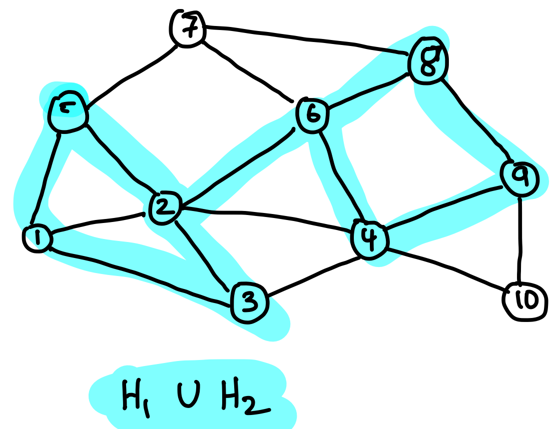
(2) all edges in H_1 or H_2



We can also form unions of many subgraphs:

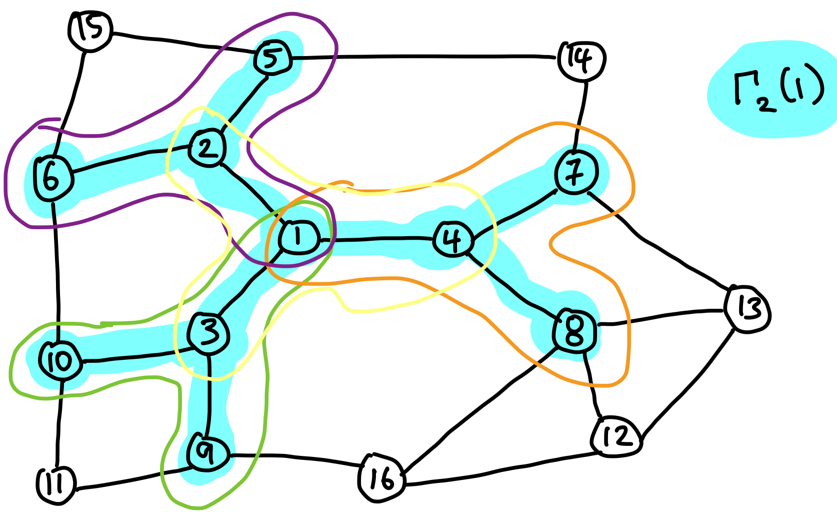
$$H_1 \cup H_2 \cup H_3 \cup \cdots \cup H_k = \bigcup_{i=1}^{k} H_i$$

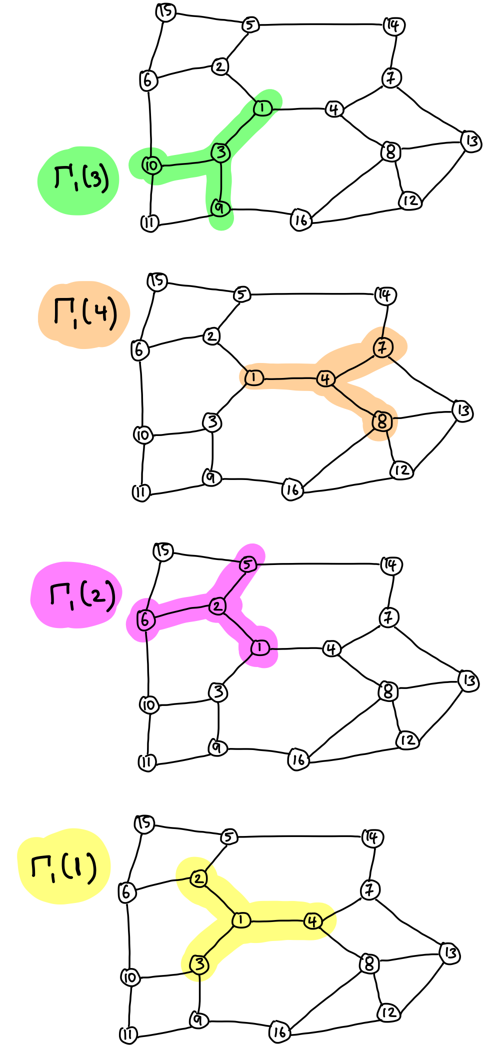
= vertices and edges in <u>any</u> of the subgraphs



Neighborhood Covering Lemma. Let $G=(V,E)$ be a graph and $v \in V$ a vertex in G. For any distance $d \geq 2$, the distance d neighborhood of v is the union of v's neighbors distance $d-1$ neighborhoods. Symbolically:

$$\Gamma_d(v) = \bigcup_{w \in \Gamma_1(v)} \Gamma_{d-1}(v)$$

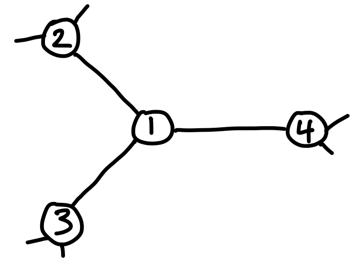


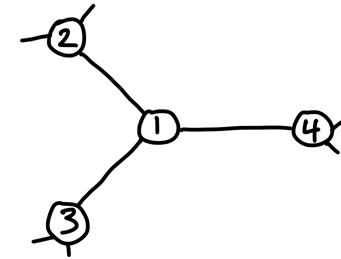


For Now. Take neighborhood covering lemma on faith

→ see notes for a proof

Want to show: state of each vertex v is determined by the initial state of vertices in $\Gamma_{r-1}(v)$. Formally:





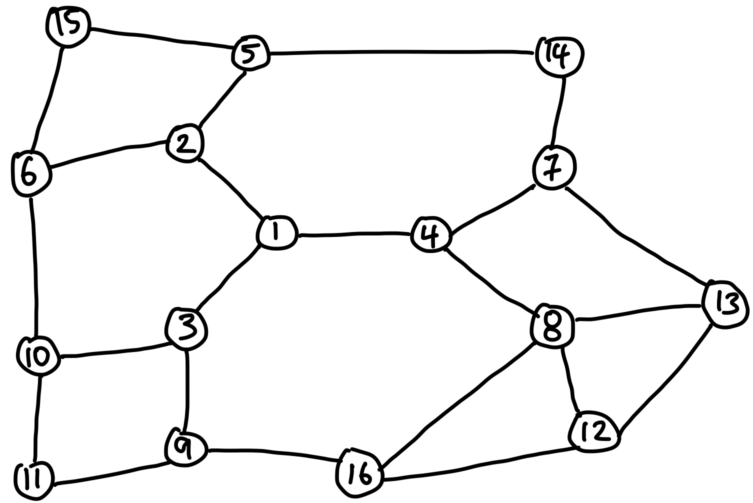
Locality Lemma. Suppose Π is a PO protocol, $G = (V, E)$ is a graph together w/ local inputs for each vertex v. Then for every round r, and vertex v, the state of v in round r is determined by $\Gamma_{r-1}(v)$.

Proof idea. The state of v in round 1 is determined only by v's local input

⇒ msgs sent in round 1 are functions of local inputs

State in round 2 is function of v's local input and msgs received in round 1

⇒ msgs sent by v in round 2 are functions of inputs of distance 1 neighborhoods


15
5
14
2
6
7
1
4
13
3
8
10
12
9
16
11