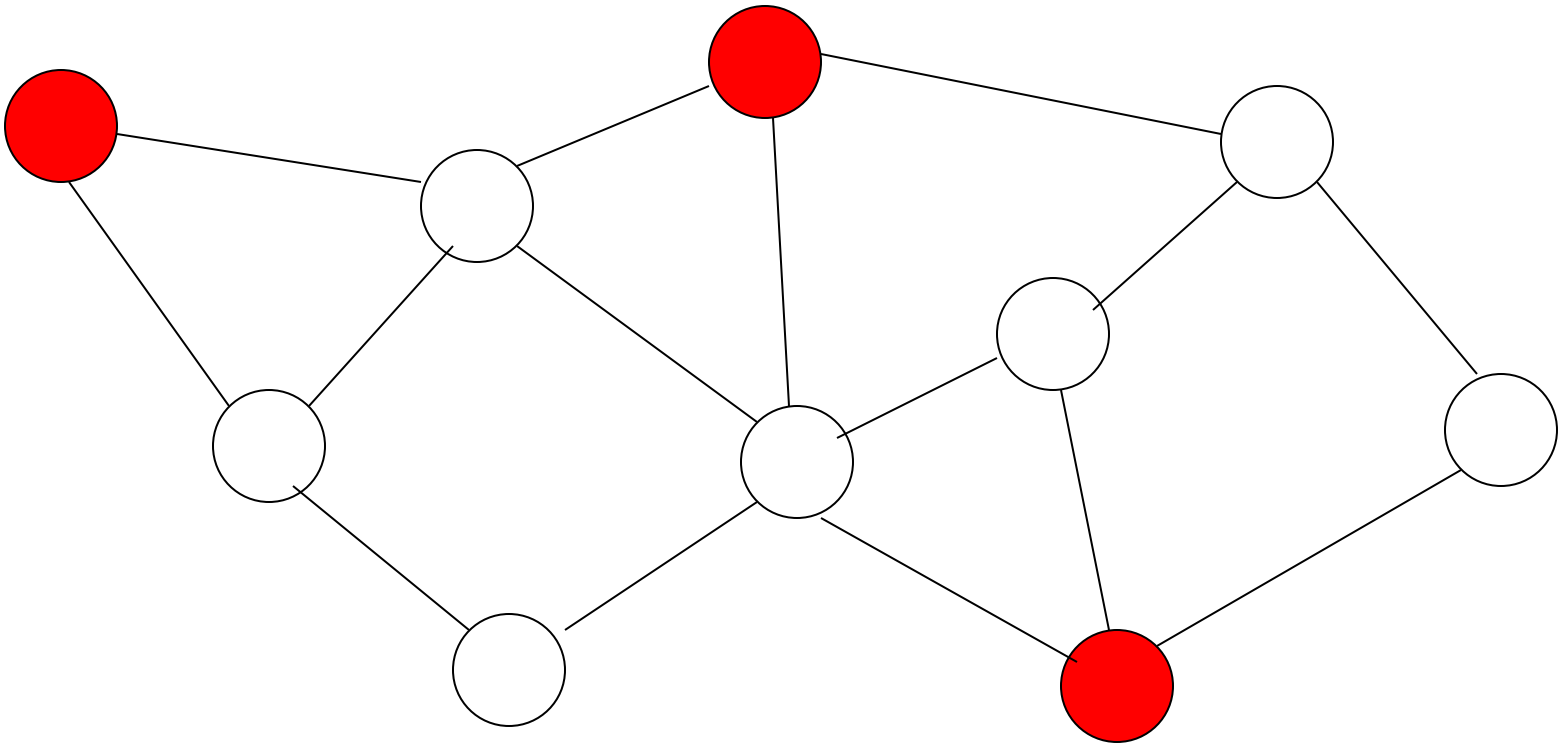
Maximal Independent Set

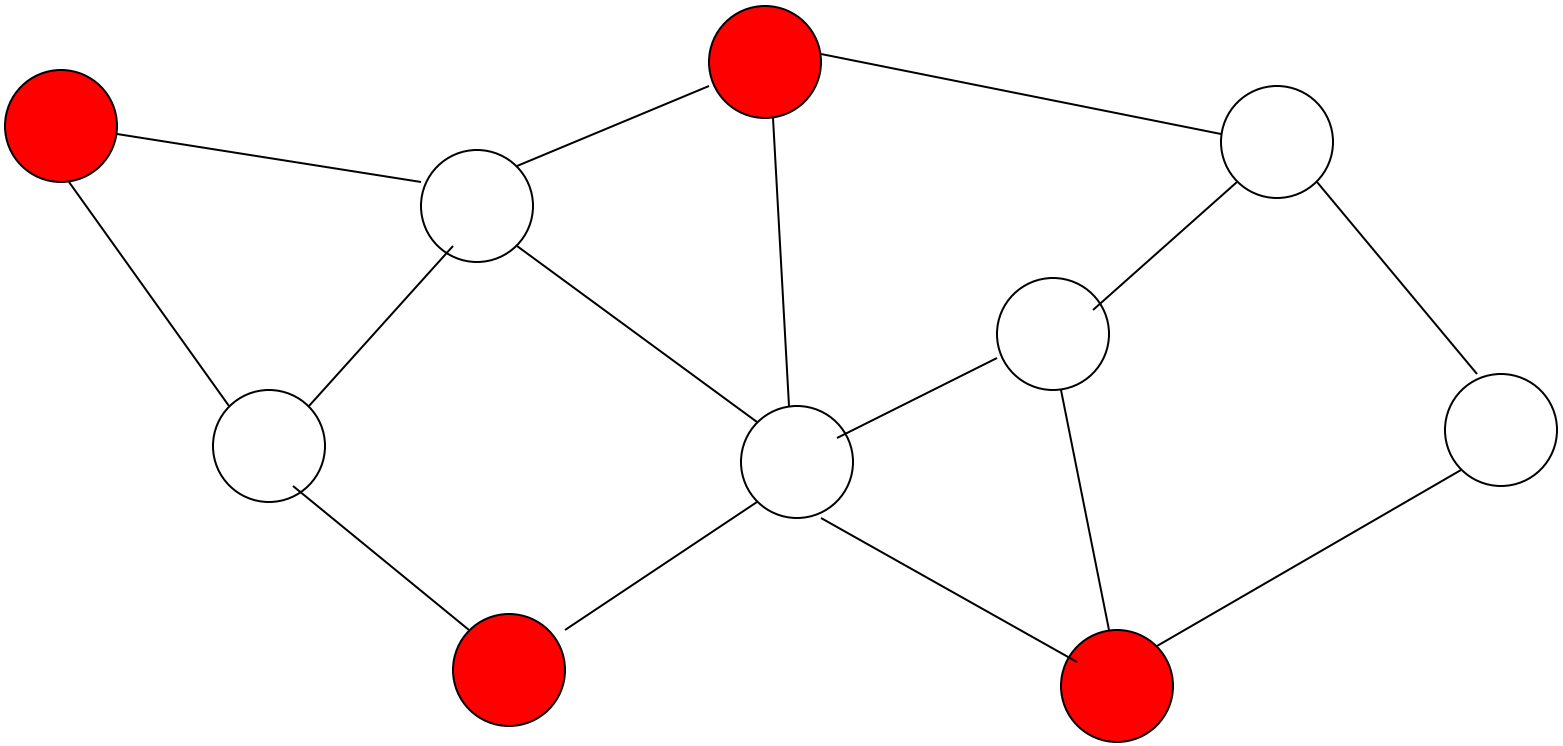
Independent (or stable) Set (IS):

In a graph G=(V,E), |V|=n, |E|=m, any set of nodes that are not adjacent



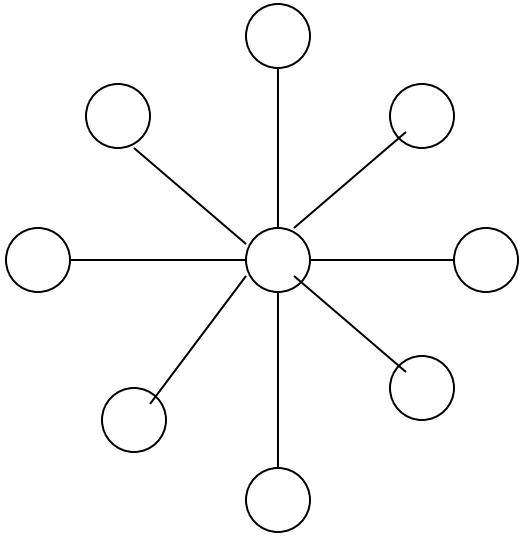
Maximal Independent Set (MIS):

An independent set that is no subset of any other independent set

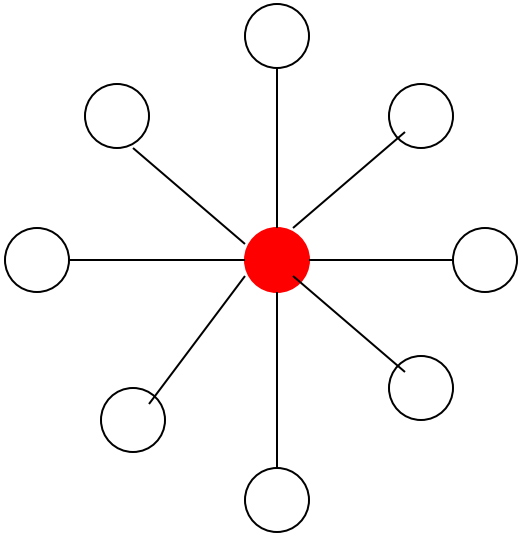


Size of Maximal Independent Sets

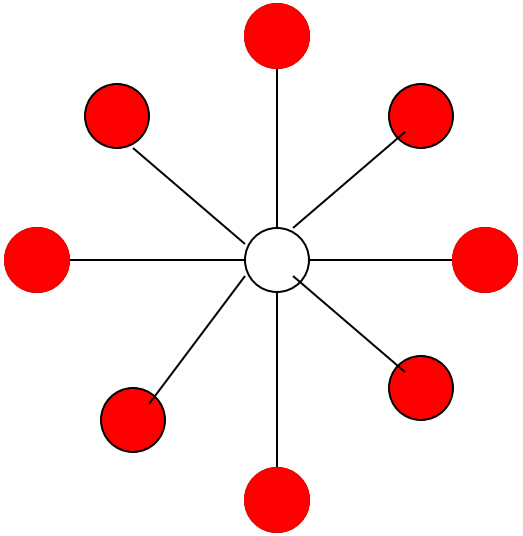
A graph G…



…a MIS of minimum size…



…a MIS of maximum size (a.k.a. maximum independent set)



Remark 1: The ratio between the size of a maximum MIS and a minimum MIS is unbounded (i.e., $O(n)$)!

Remark 2: Depending on the application, we might be interested in finding a MIS of either small or large size, but unfortunately finding a minimum/maximum MIS is an NP-hard problem since deciding whether a graph has a MIS of size k is NP-complete

Remark 3: On the other hand, a MIS can be found in polynomial time

Applications in DS: network topology control

- In a network graph consisting of nodes representing processors, a MIS defines a set of processors which can operate in parallel without interference
- For instance, in wireless ad hoc networks, to avoid interferences, a conflict graph (based on the overlapping in the transmission ranges) is built, and a MIS of such a graph defines a partition of the nodes enabling interference-free communication, where messages are broadcasted by the nodes in the MIS to their neighbors (in such an application, to reduce the congestion at the MIS nodes, one should find a MIS of maximum size, but as said before this is known to be NP-hard)

Applications in DS: network monitoring

- A MIS is also a Dominating Set (DS) of the graph (the converse in not true, unless the DS is independent), namely every node in G is at distance at most 1 from at least one node in the MIS (otherwise the MIS could be enlarged, against the assumption of maximality!)

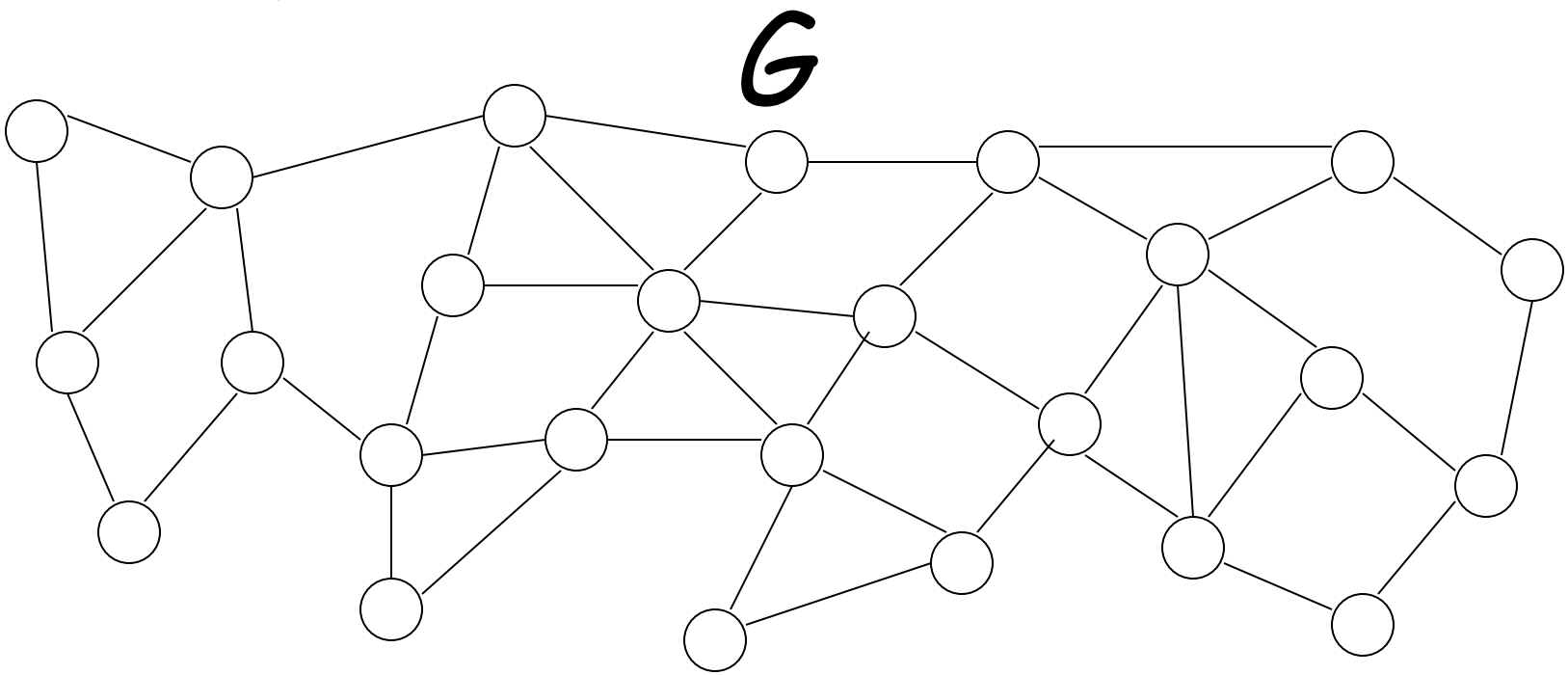
$\Rightarrow$In a network graph G consisting of nodes representing processors, a MIS defines a set of processors which can monitor the correct functioning of all the nodes in G: each node in the MIS will ping continuously its neighbors (in such an application, one should find a MIS of minimum size, to minimize the number of sentinels, but as said before this is known to be NP-hard)

Question: Exhibit a graph G s.t. the ratio between a Minimum MIS and a Minimum Dominating Set is $\Theta(n)$, where n is the number of vertices of G.

A sequential algorithm to find a MIS

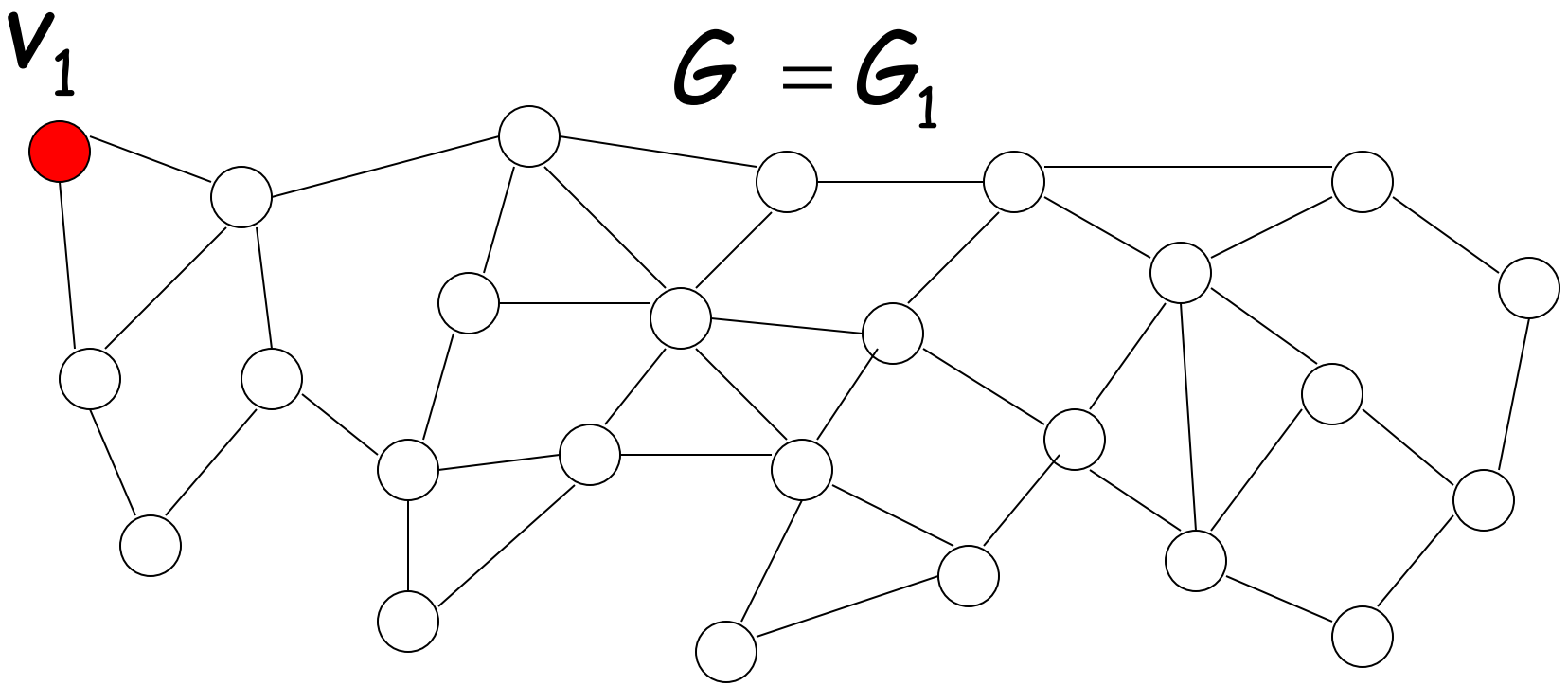
Suppose that I will hold the final MIS, and assume that we are not interested in any respect to the final size of the MIS

Initially $I = \emptyset$



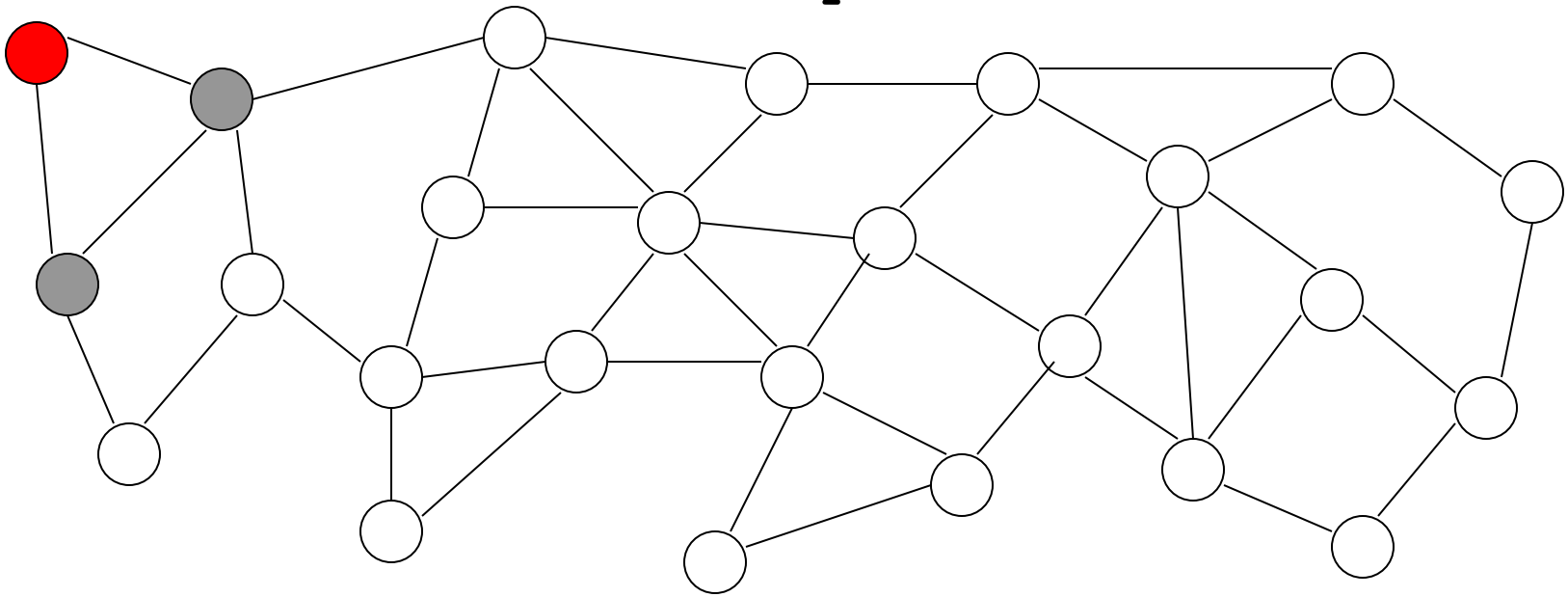
Phase 1:

Pick an arbitrary node v_1 and add it to I



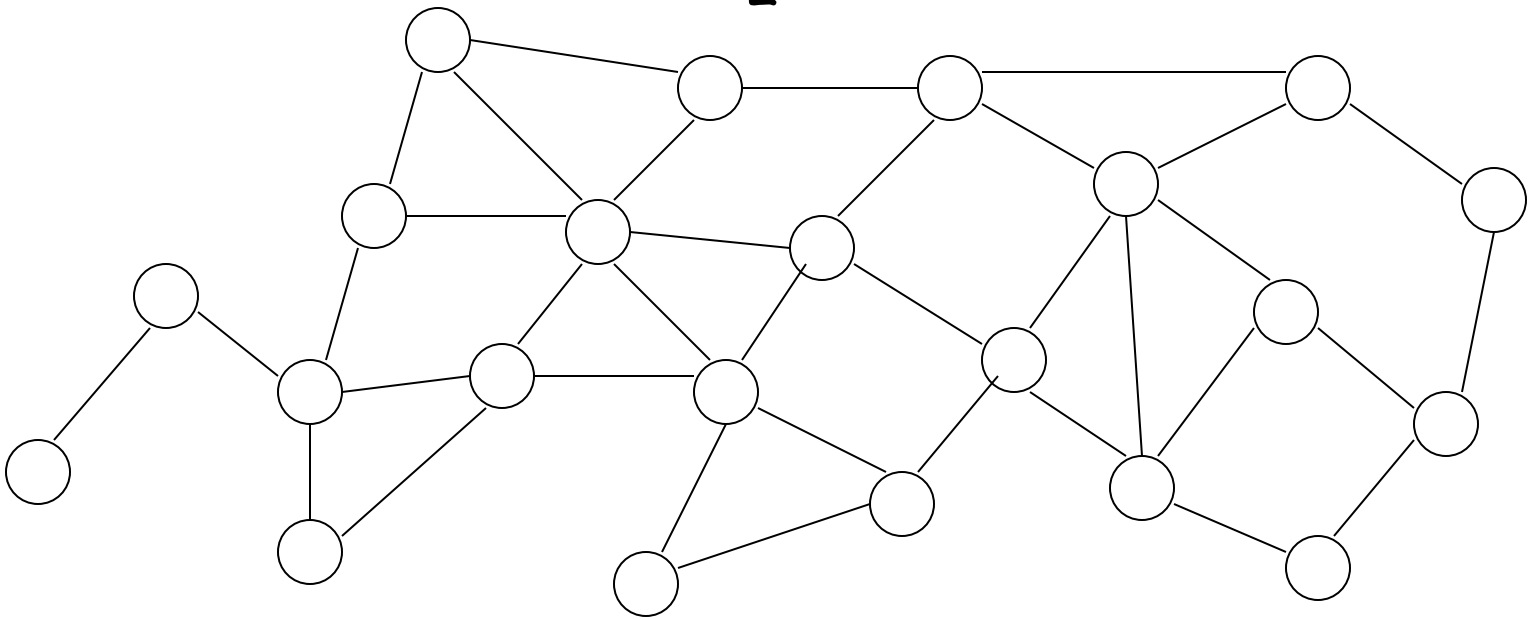
Remove v_1 and neighbors $N(v_1)$

G_1



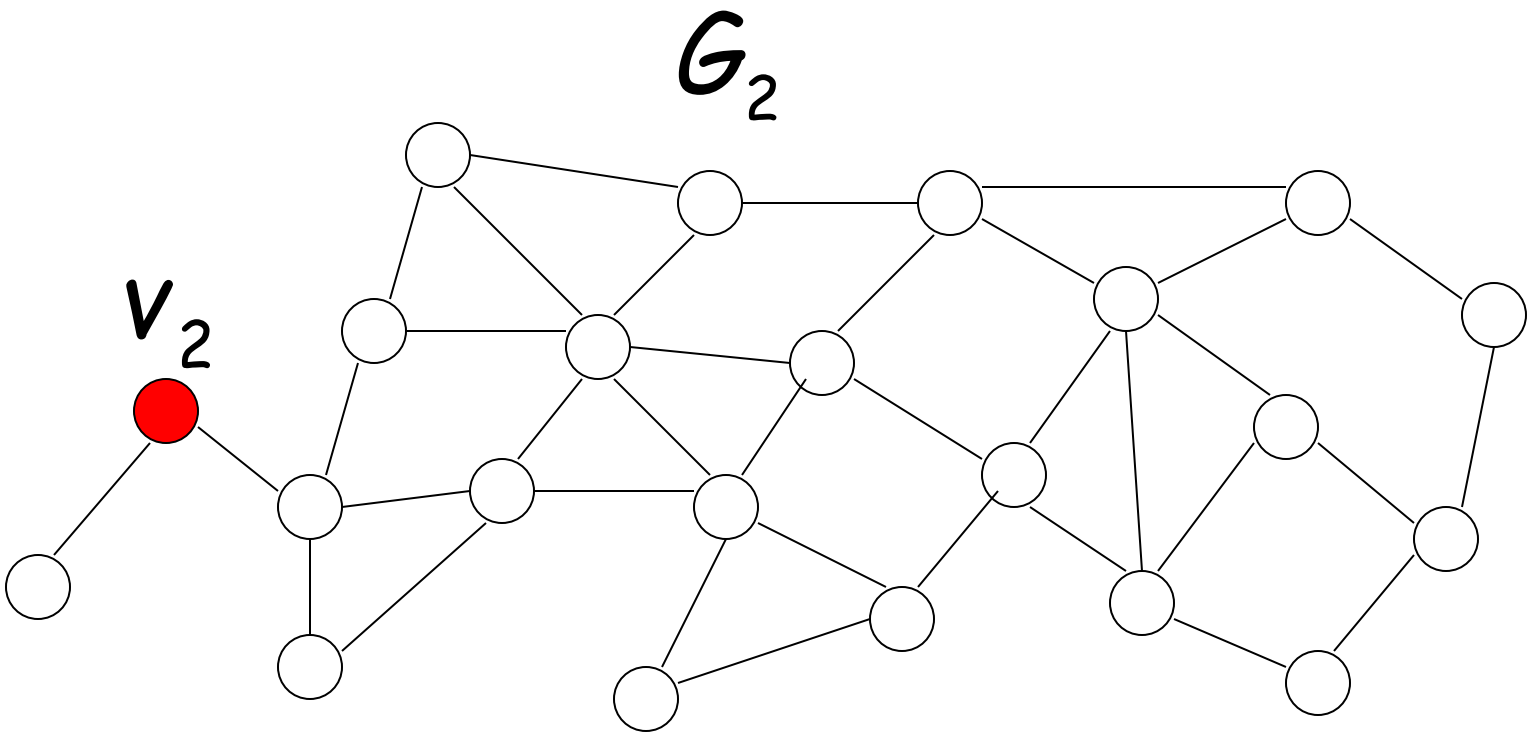
Remove v_1 and neighbors $N(v_1)$

G_2

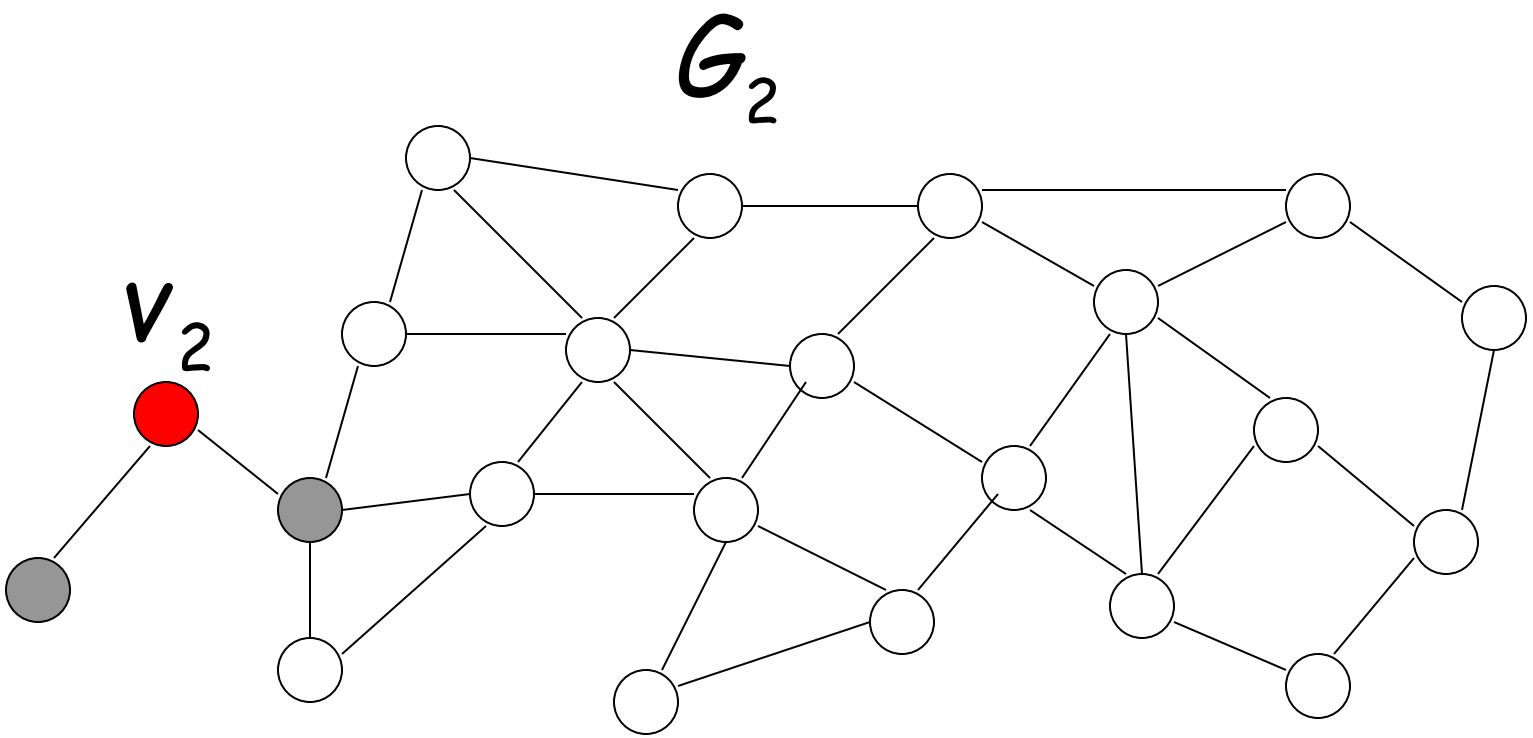


Phase 2:

Pick a node v_2 and add it to I

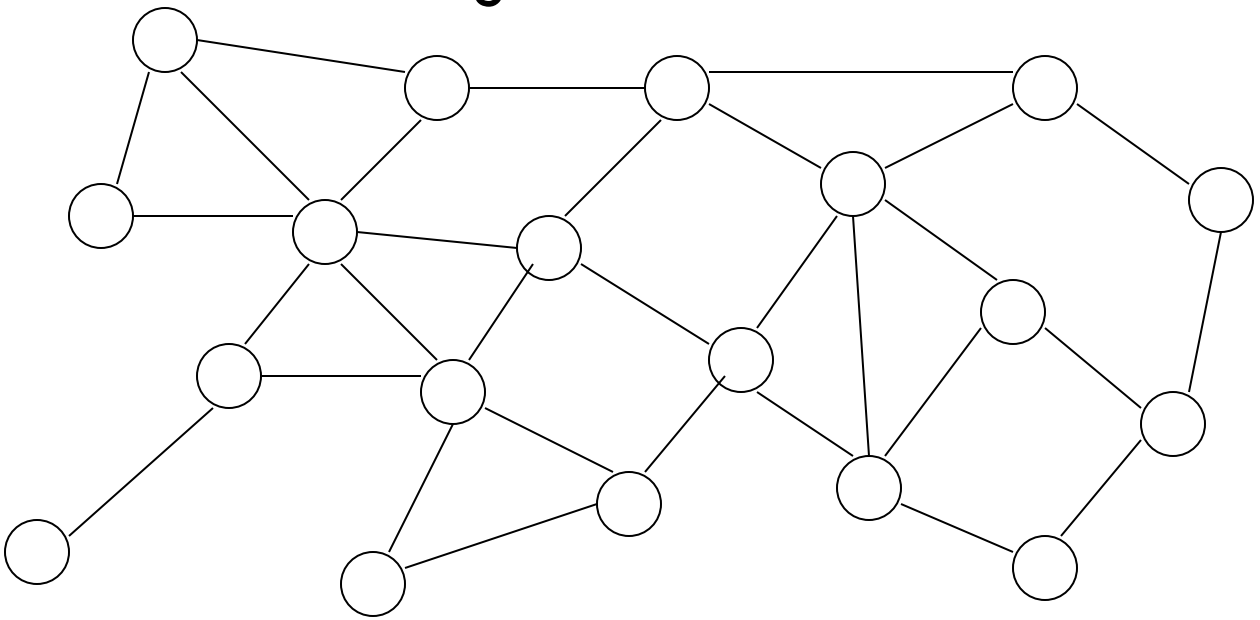


Remove v_2 and neighbors $N(v_2)$



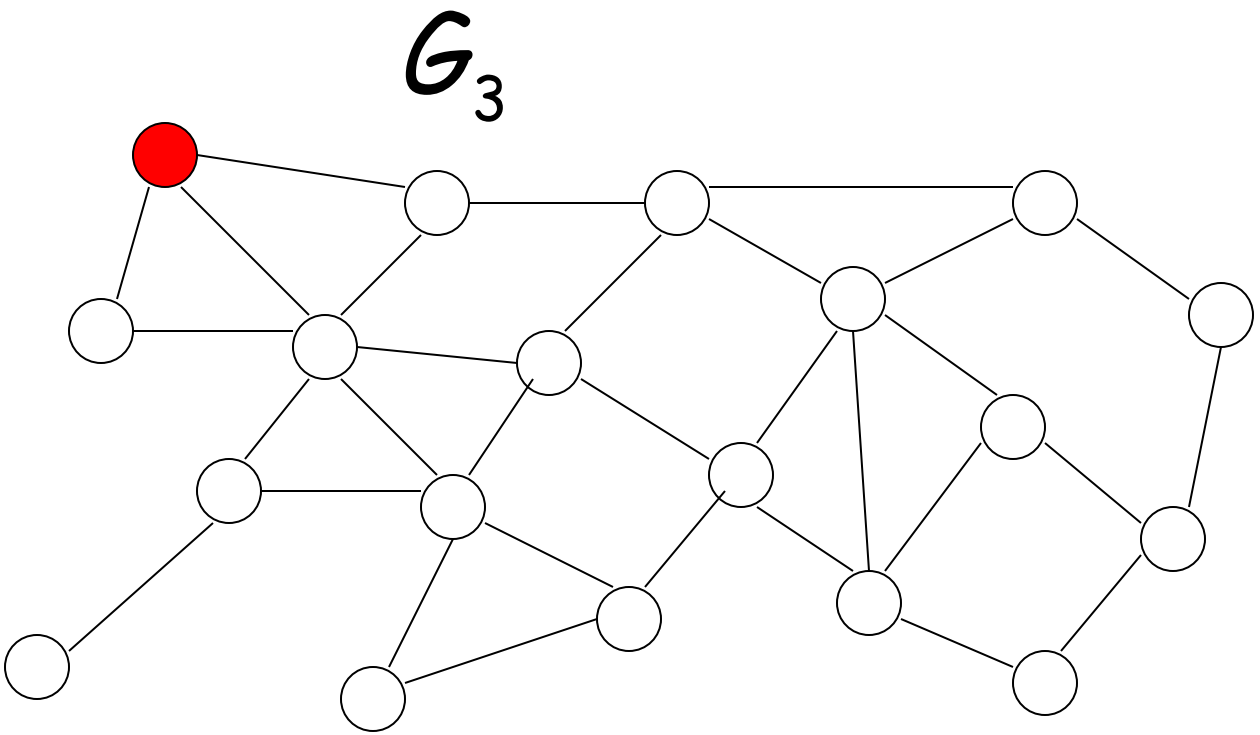
Remove v_2 and neighbors $N(v_2)$

G_3



Phases 3,4,5,…:

Repeat until all nodes are removed



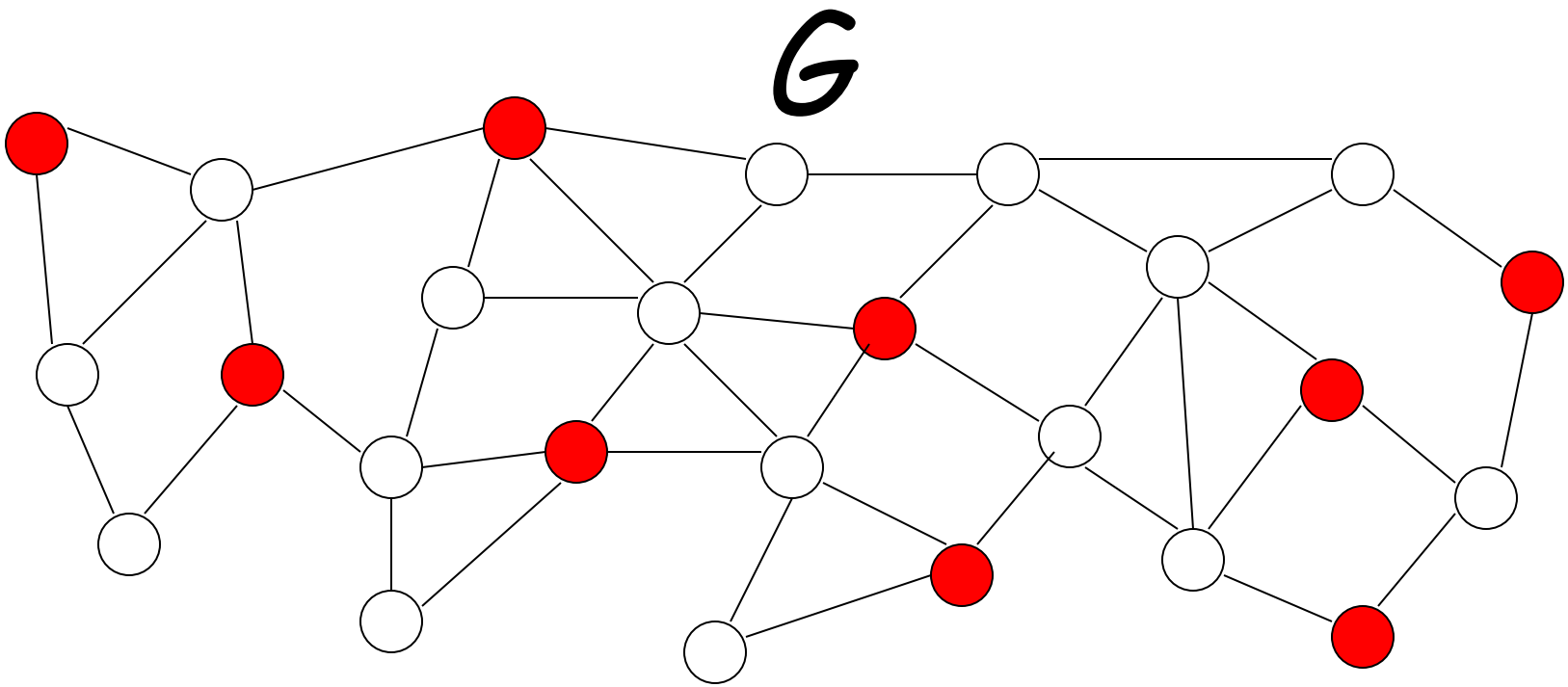
Phases 3,4,5,…,x:

Repeat until all nodes are removed

$$G_{x+1}$$

No remaining nodes

At the end, set I will be a MIS of G

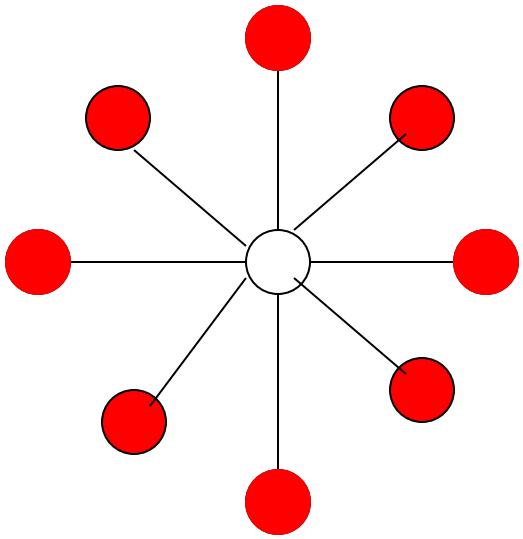


Analysis

Let $G=(V,E)$ and $n=|V|$ and $m=|E|$. If the algorithm is implemented in a centralized setting, then its running time is $\Theta(m)$.

On the other hand, the number of phases of the algorithm is equal to the size of the found MIS, and clearly it is $O(n)$.

Worst case graph (for number of phases):



n nodes, n-1 phases

Question

Can you see a distributed version of the just given algorithm?

Yes, we can elect a leader at each phase, and then add it to the MIS. In this distributed version, we would like to minimize the number of phases, since in this way the number of leader elections will be minimized!

A Generalized Algorithm For Computing a MIS

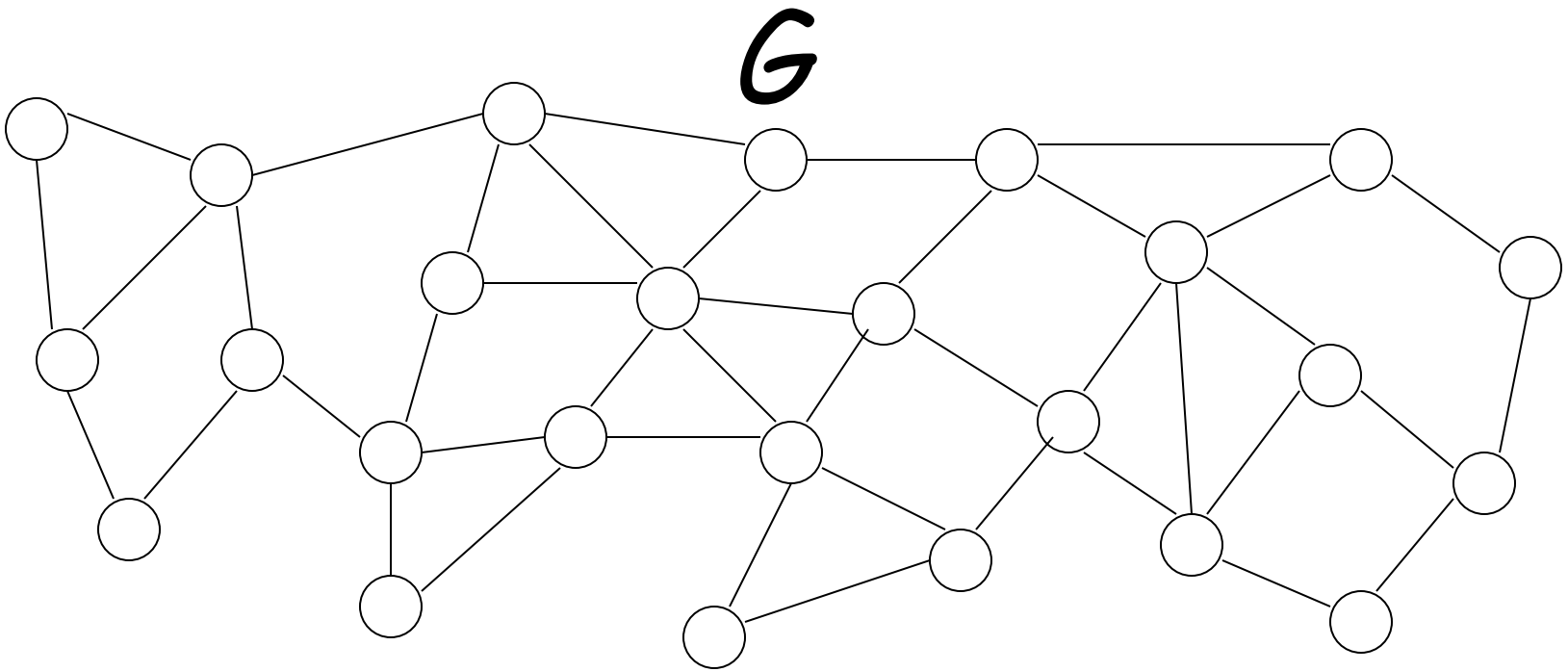
Same as the previous algorithm, but at each phase, instead of a single node, we now select any independent set (this selection should be seen as a black box at this stage, i.e., we do not know/specify how such independent set is selected)

$\Rightarrow$ The underlying idea is that this approach will be useful for a distributed algorithm, since it will reduce the number of phases

Example:

Suppose that I will hold the final MIS

Initially $I = \emptyset$

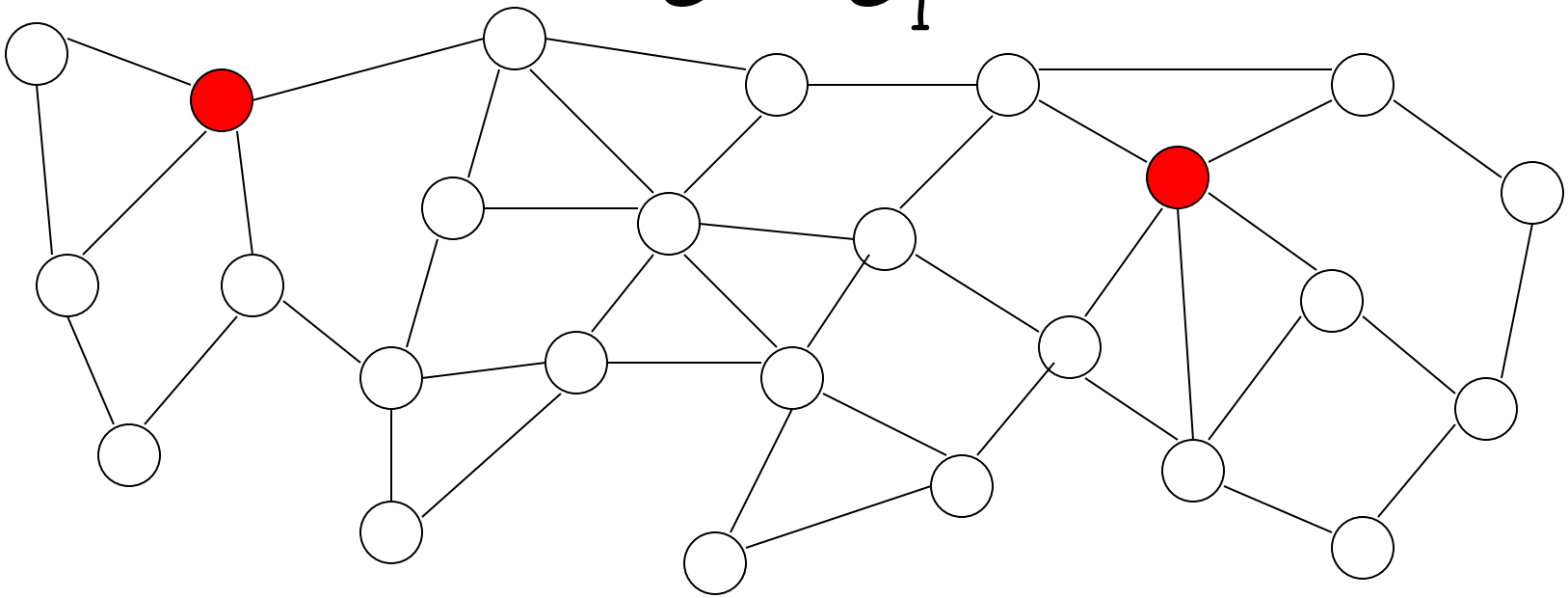


Phase 1:

Find any independent set I_1

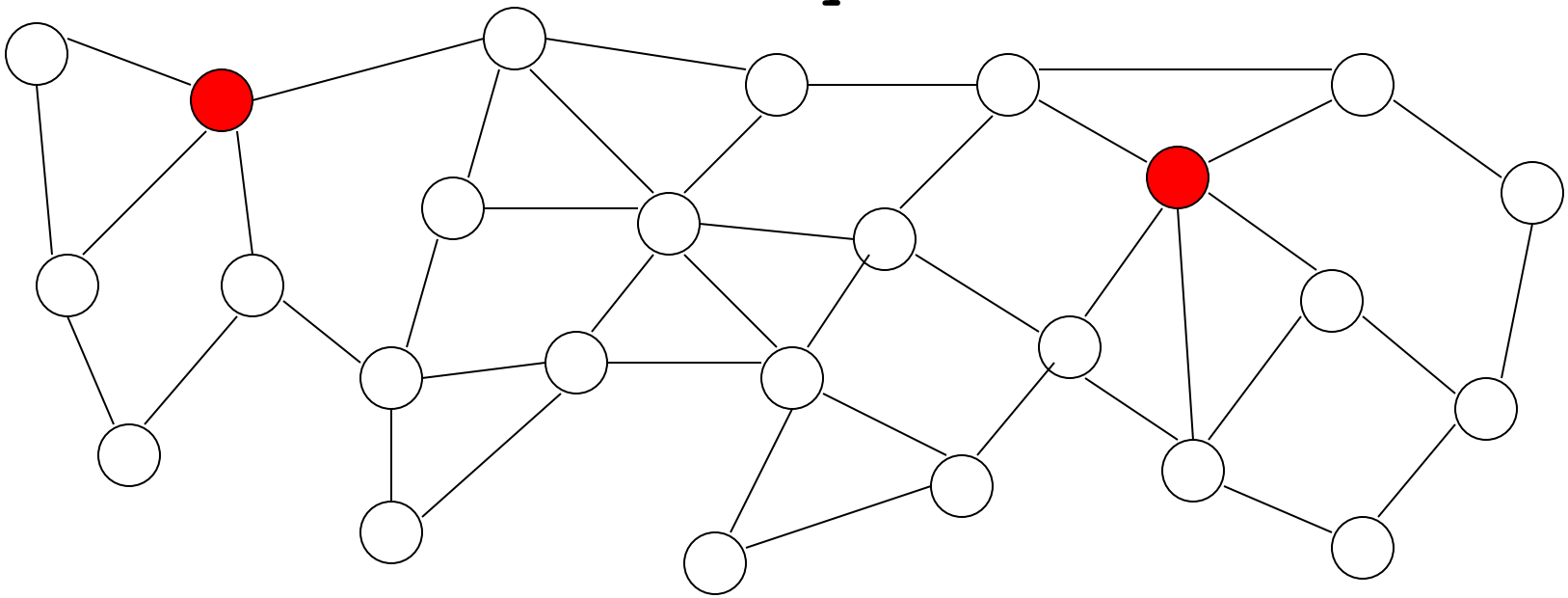
and add I_1 to I: $I \leftarrow I \cup I_1$

$G = G_1$



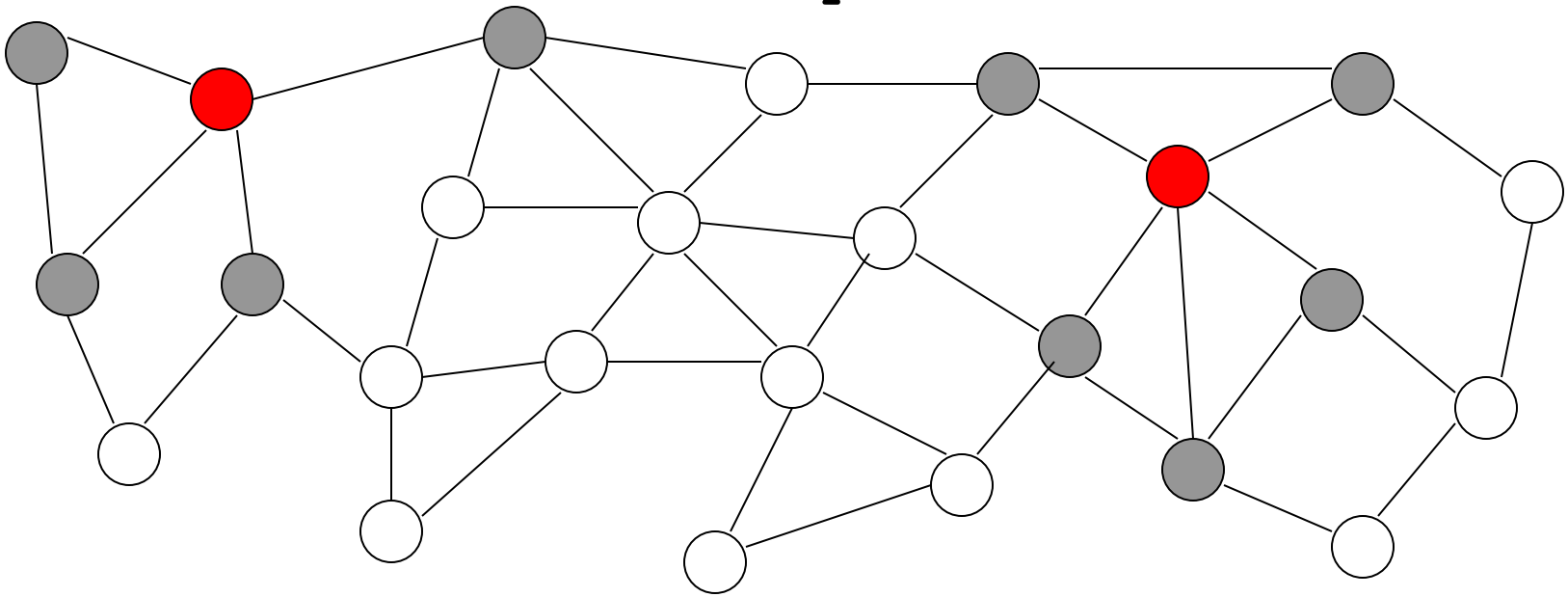
remove I_1 and neighbors $N(I_1)$

G_1



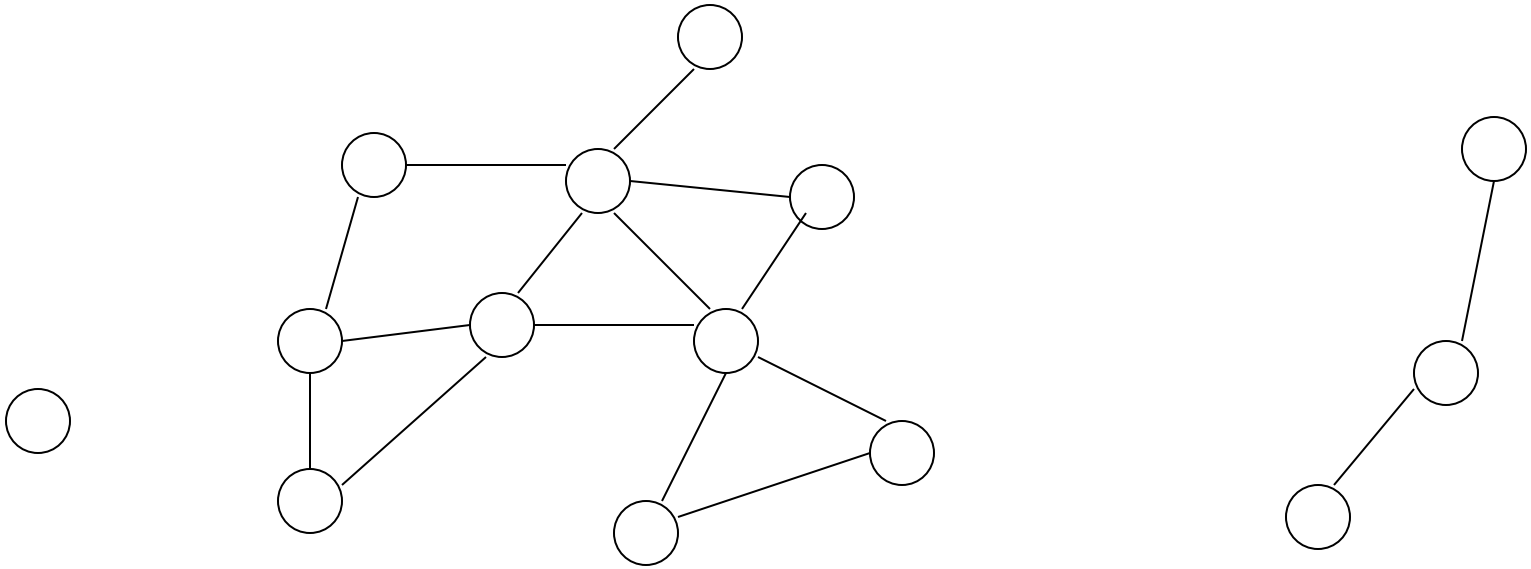
remove I_1 and neighbors $N(I_1)$

G_1



remove I_1 and neighbors $N(I_1)$

G_2



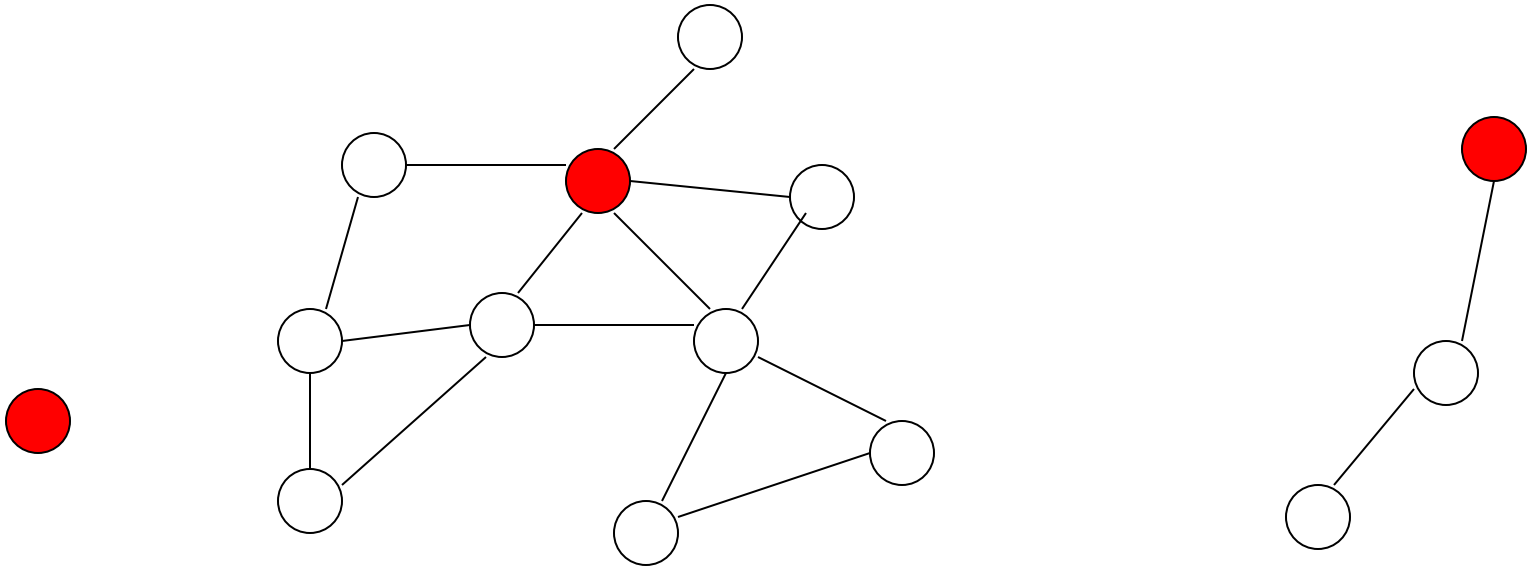
Phase 2:

On new graph

Find any independent set I_2

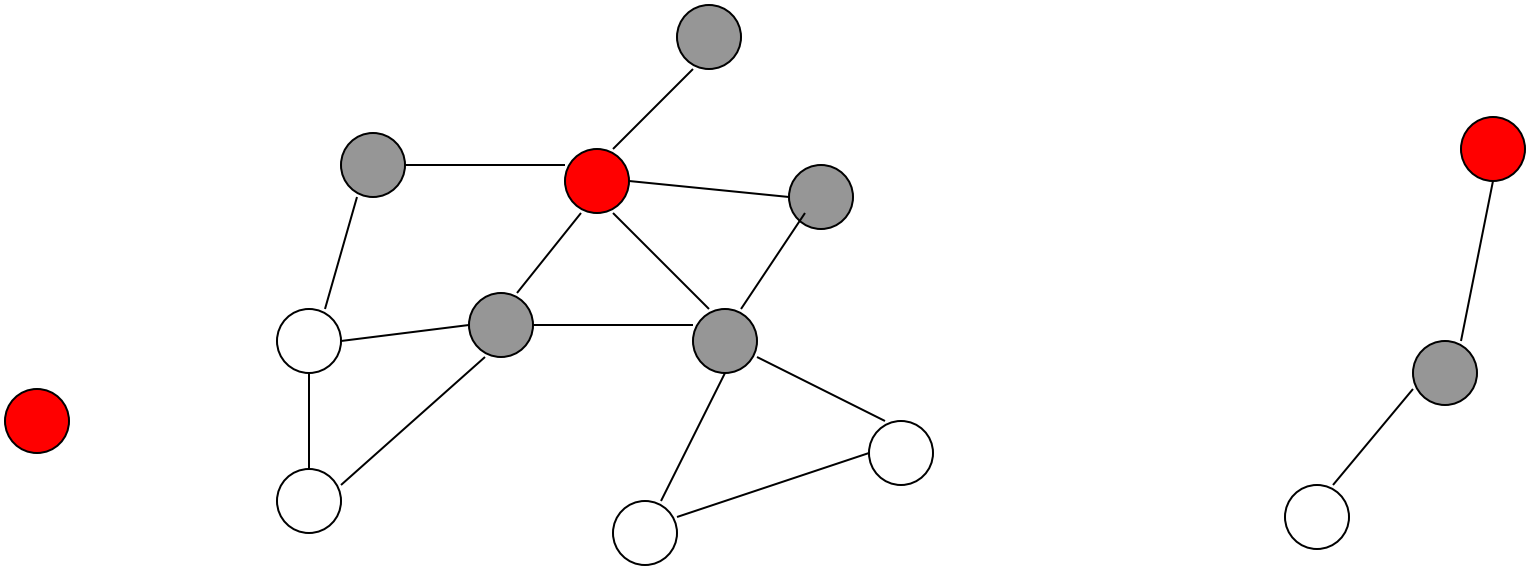
and add I_2 to I: $I \leftarrow I \cup I_2$

G_2



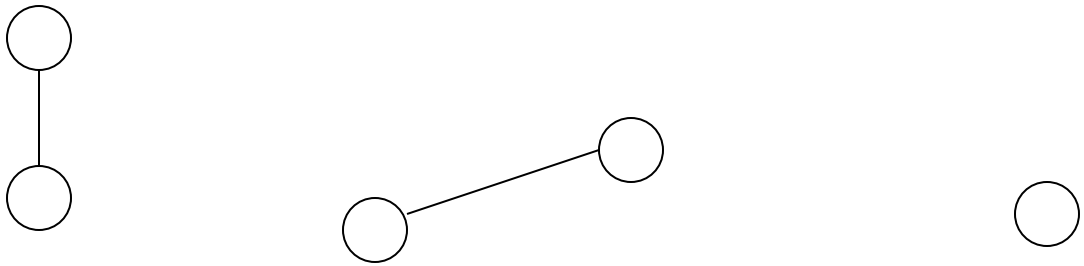
remove I_2 and neighbors $N(I_2)$

G_2



remove I_2 and neighbors $N(I_2)$

G_3



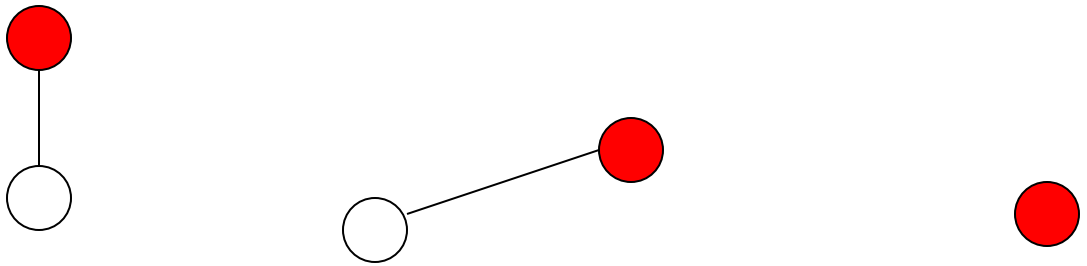
Phase 3:

On new graph

Find any independent set I_3

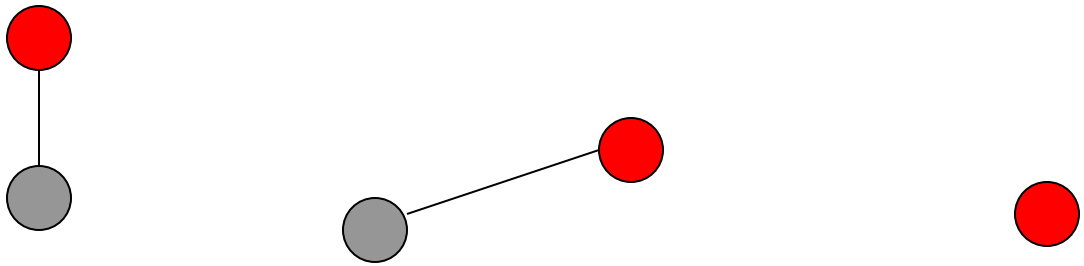
and add I_3 to I: $I \leftarrow I \cup I_3$

G_3



remove I_3 and neighbors $N(I_3)$

G_3

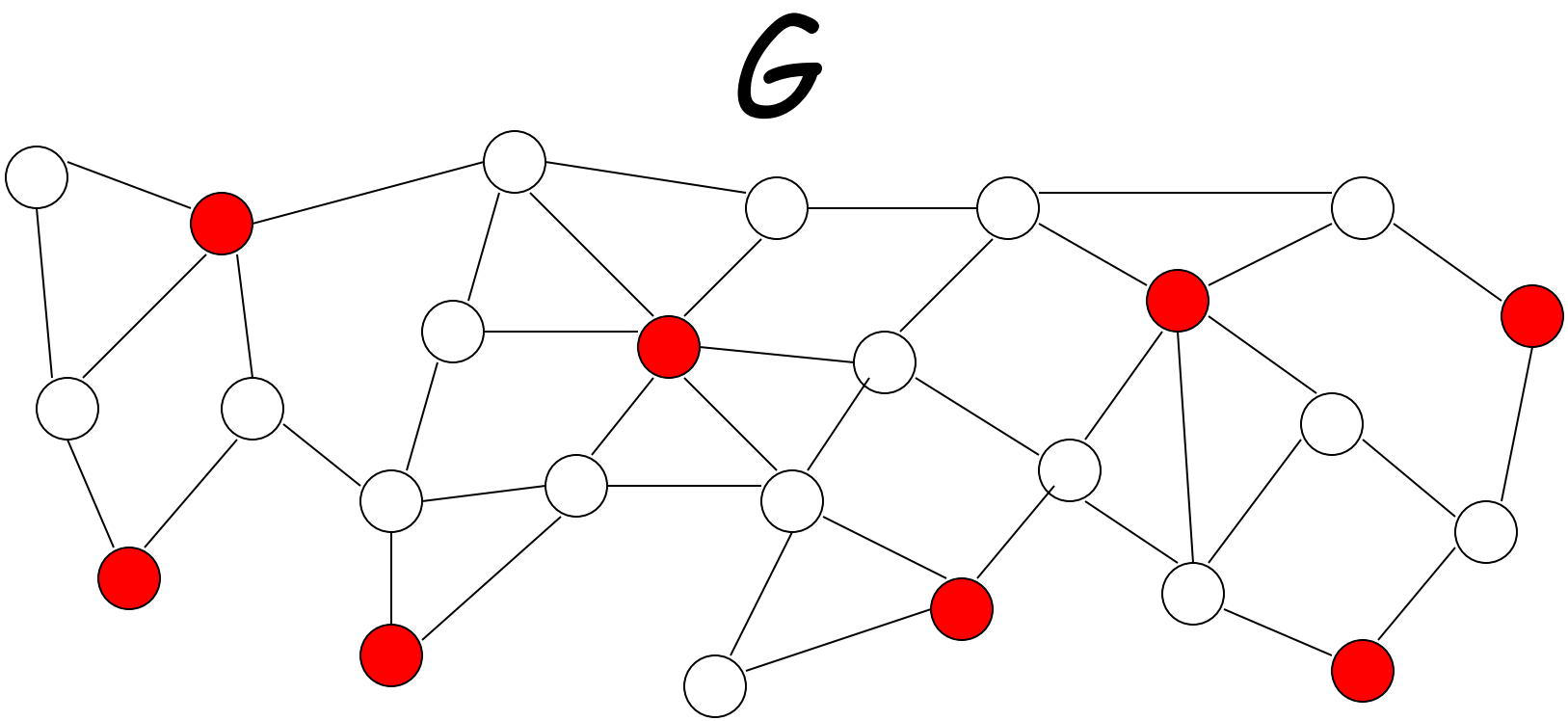


remove I_3 and neighbors $N(I_3)$

G_4

No nodes are left

Final MIS I



Analysis

1. The algorithm is correct, since independence and maximality follow by construction
2. Running time is now $\Theta(m)$ (the time needed to remove the edges), plus the time needed at each phase to find an independent set (this is really the crucial step!)
3. The number of phases now depends on the choice of the independent set in each phase: The larger the subgraph removed at the end of a phase, the smaller the residual graph, and then the faster the algorithm. Then, how do we choose such a set, so that independence is guaranteed and the convergence is fast?

Example: If I_1 is a MIS,
one phase is enough!

Example: If each I_k contains one node,
$\Theta(n)$ phases may be needed

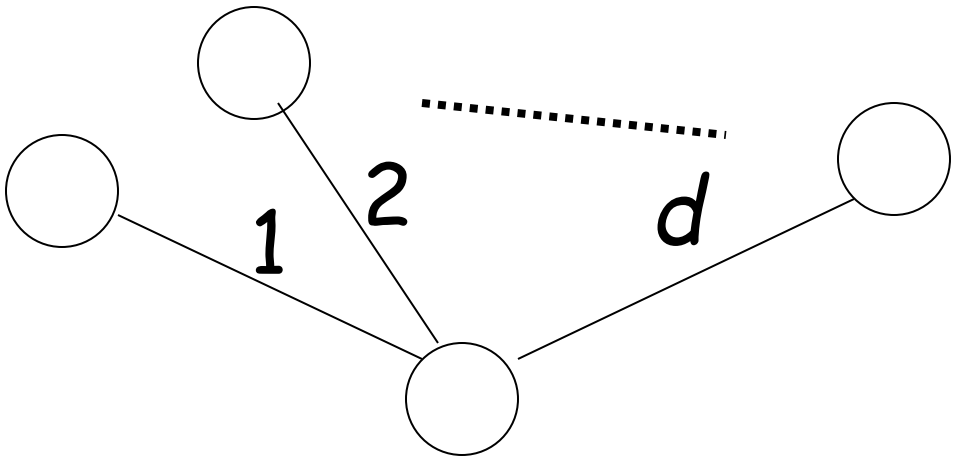
(sequential greedy algorithm)

A Randomized Sync. Distributed Algorithm

- Implements in a distributed setting the latter MIS algorithm, by choosing randomly at each phase the independent set, in such a way that it is expected to remove many nodes from the current residual graph
- Works with synchronous, uniform models, and does not make use of the processor IDs

Remark: It is randomized in a Las Vegas sense, i.e., it uses randomization only to reduce the **expected** running time, but always terminates with a correct result (against a Monte Carlo sense, in which the running time is **fixed**, while the result is correct with a certain probability)

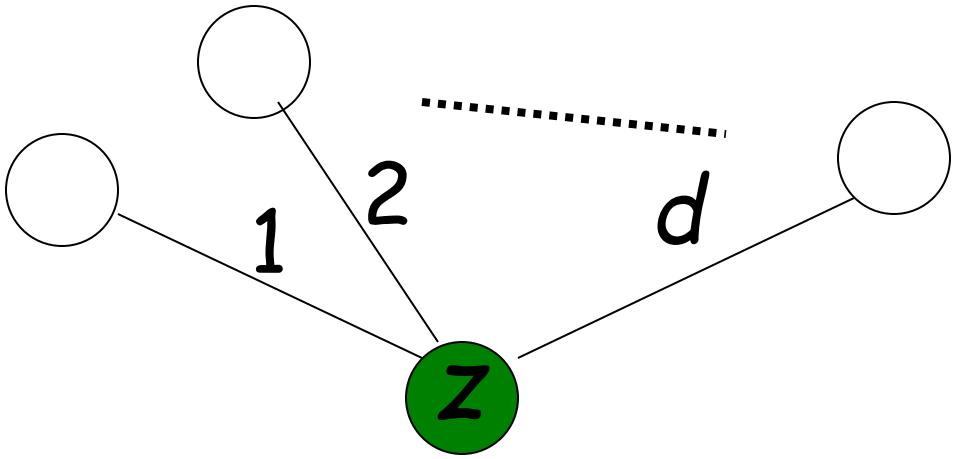
Let d be the maximum node degree in the whole graph G



Suppose that d is known to all the nodes (this may require a pre-processing)

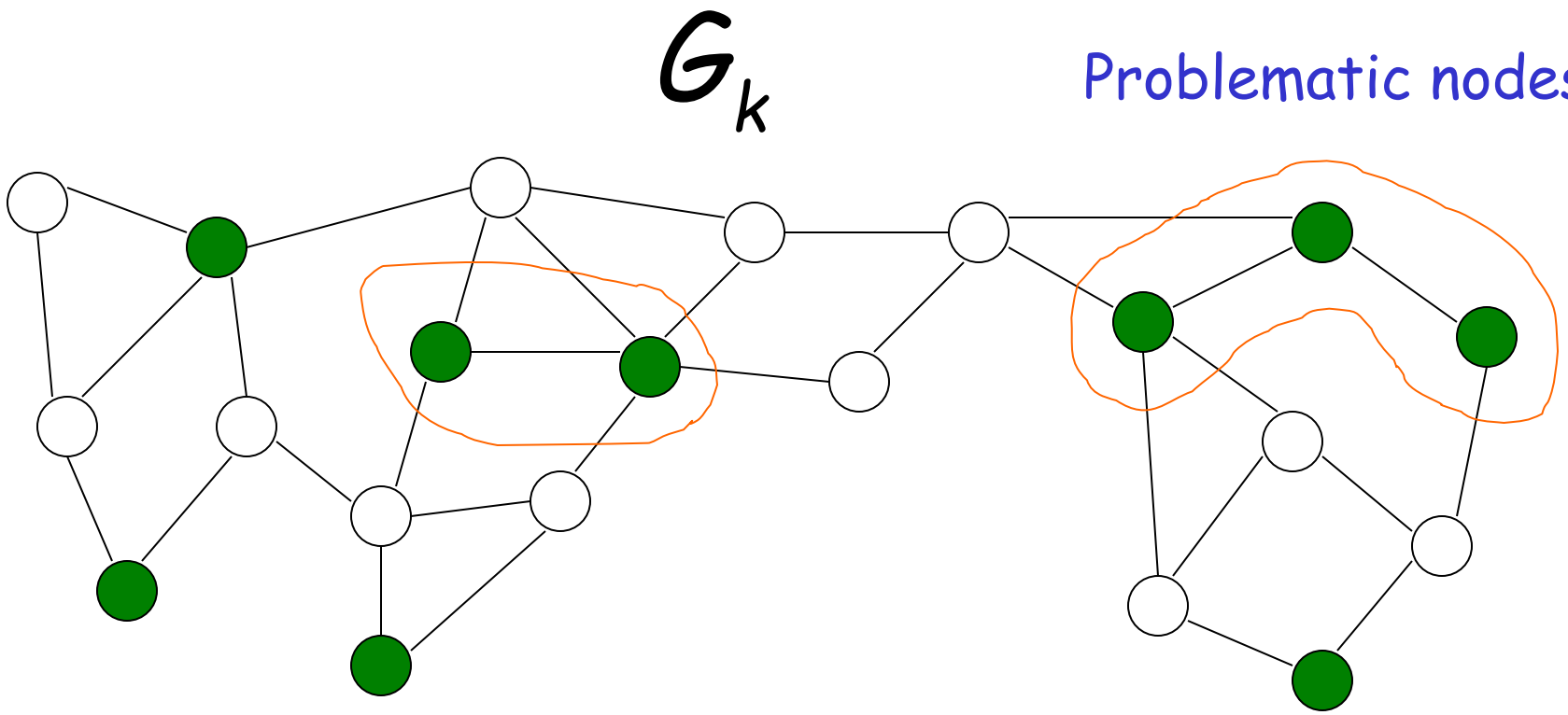
At each phase k:

Each node $z \in G_k$ elects itself with probability $p = \frac{1}{d}$

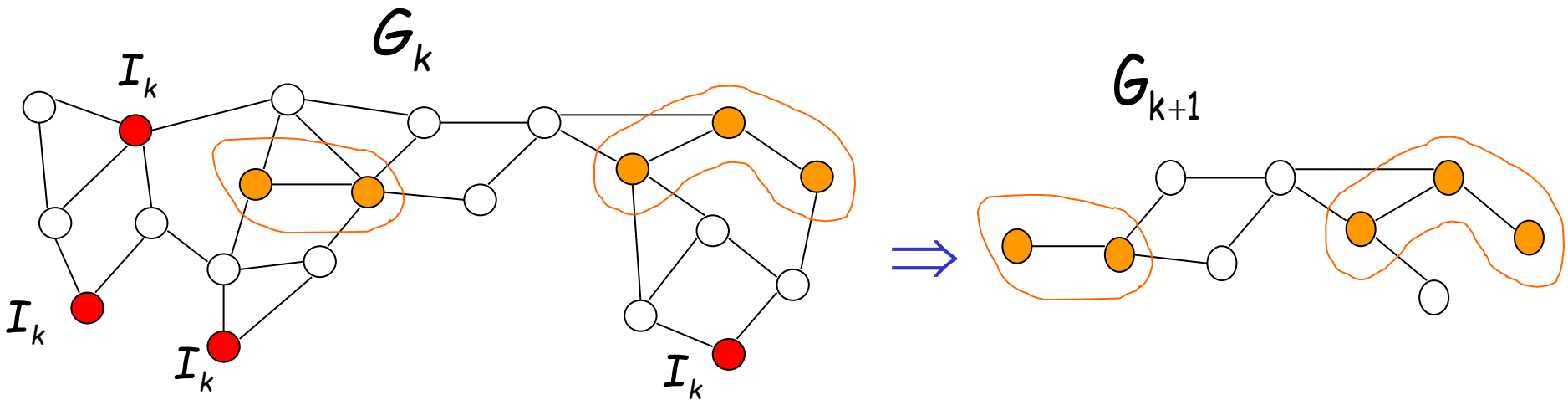


Elected nodes are candidates for independent set I_k

However, it is possible that neighbor nodes are elected simultaneously (nodes can check it out by testing their neighborhood)



All the problematic nodes step back to the unelected status, and proceed to the next phase. The remaining elected nodes form independent set I_k, and $G_{k+1} = G_k \setminus (I_k \cup N(I_k))$



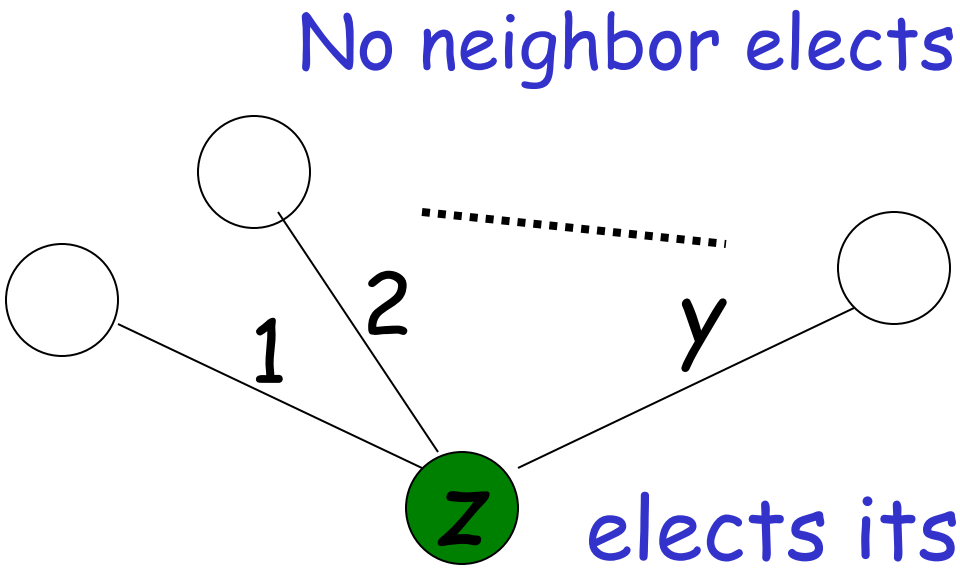
Analysis:

Success for a node $z \in G_k$ in phase k :

z disappears at the end of phase k

(enters I_k or $N(I_k)$)

A good scenario that guarantees success for z and all of its neighbors



Basics of Probability

Let A and B denote two events in a probability space; let

1. $\neg A$ (i.e., not A) be the event that A does not occur;
2. $A \cap B$ be the event that both A and B occur;
3. $A \cup B$ be the event that A or (non-exclusive) B occurs.

Then, we have that:

1. $P(\neg A)=1-P(A)$;
2. if A and B are **independent**, then $P(A \cap B)=P(A) \cdot P(B)$

   Example: if two coins are flipped the chance of both being heads is $\frac{1}{2} \cdot \frac{1}{2}=\frac{1}{4}$
3. if A and B are **mutually exclusive**, then $P(A \cup B)=P(A)+P(B)$, otherwise $P(A \cup B)=P(A)+P(B)-P(A \cap B)$

   Example: the chance of rolling a 1 **or** 2 on a six-sided dice is 1/6+1/6=1/3 (mutual exclusion), while the chance of rolling a value $\leq$ 3 **or** an even value is 1/2 +1/2 - 1/6 = 5/6 (the two events are not mutually exclusive, since if 2 is rolled, then both events are verified, so we have to subtract the probabilty 1/6 that 2 is rolled)

Fundamental inequality

$x \geq 1, x \in \Re$

$|t| \leq x, t \in \Re$

$e = 2,7182\ldots$

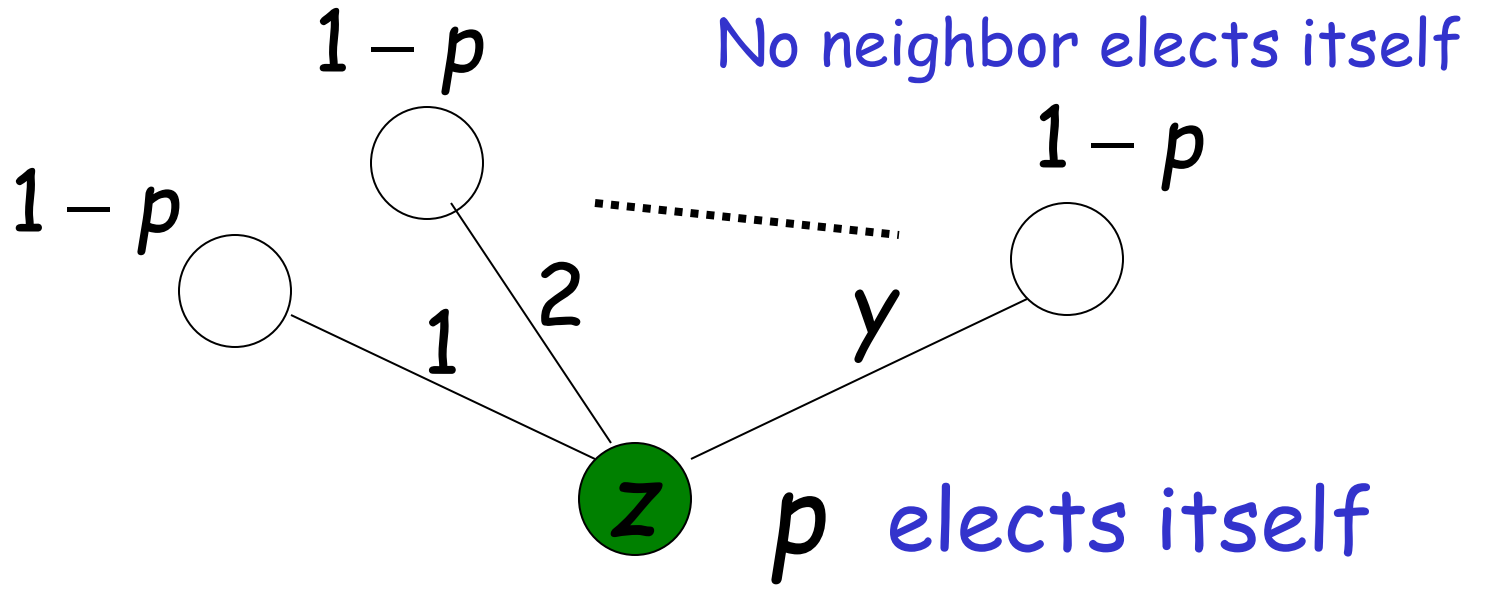
$$e^t\left(1-\frac{t^2}{x}\right) \leq \left(1+\frac{t}{x}\right)^x \leq e^t$$

Probability for a node z of success in a phase:

$$P(\text{success } z) = P((z \text{ enters } I_k) \text{ OR } (z \text{ enters } N(I_k)))$$
$$\geq P(z \text{ enters } I_k)$$

i.e., it is at least the probability that it elects itself **AND** no neighbor elects itself, and since these events are **independent**, if $y=|N(z)|$, then

$$P(z \text{ enters } I_k) = p\cdot(1-p)^y \quad \text{(recall that } p=1/d\text{)}$$



Probability of success for a node in a phase:

At least $p(1-p)^y \geq p(1-p)^d$

$$= \frac{1}{d}\left(1-\frac{1}{d}\right)^d$$

Fundamental inequality

$$e^t\left(1-\frac{t^2}{x}\right) \leq \left(1+\frac{t}{x}\right)^x$$

$$\geq \frac{1}{ed}\left(1-\frac{1}{d}\right)$$

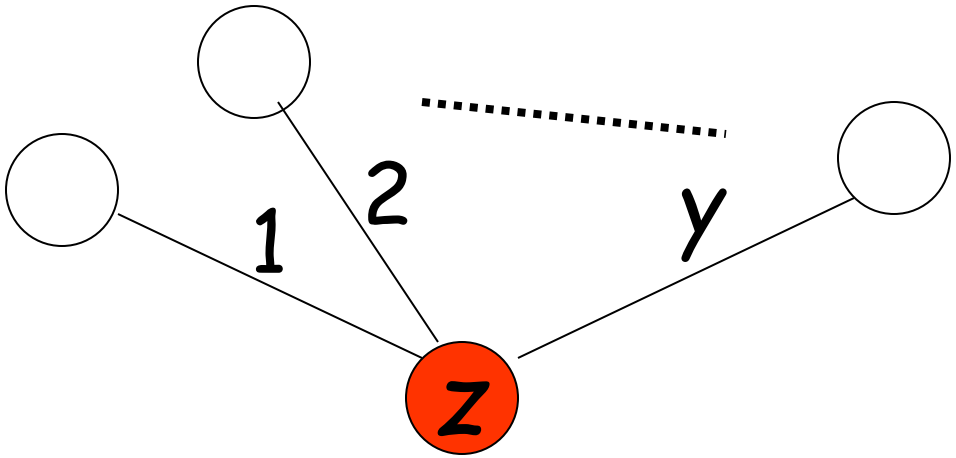
with t=-1 and x=d≥1:

$(1-1/d)^d \geq (1-(-1)^2/d)e^{(-1)}$

i.e., $(1-1/d)^d \geq 1/e\cdot(1-1/d)$

$$\geq \frac{1}{2ed} \quad \text{for } d \geq 2$$

Therefore, node z disappears at the end of a phase with probability at least $\frac{1}{2ed}$



$\Rightarrow$ Node z does not disappear at the end of a phase with probability at most $1-\frac{1}{2ed}$

Definition: Bad event for node z:

after $4ed \ln n$ phases

node z did not disappear

Independent events

This happens with probability

$P(AND_{k=1,\ldots,4ed\ln n}$ (z does not disappear at the end of phase k))

i.e., at most:

$$\left(1-\frac{1}{2ed}\right)^{4ed\cdot\ln n} = \left(\left(1-\frac{1}{2ed}\right)^{2ed}\right)^{2\ln n} \leq \frac{1}{e^{2\ln n}} = \frac{1}{n^2}$$

(fund. ineq. $(1+t/x)^x \leq e^t$ with $t=-1$ and $x=2ed$)

Definition: Bad event for graph G:

after $4ed \ln n$ phases

at least one node did not disappear
(i.e., computation has not yet finished)

This happens with probability (notice that events are **not mutually exclusive**):

$$P(OR_{z \in G}(\text{bad event for } z)) \leq$$

$$\sum_{z \in G} P(\text{bad event for } z) \leq n \frac{1}{n^2} = \frac{1}{n}$$

Definition: Good event for graph G:

within $4ed \ln n$ phases

all nodes disappear (i.e., computation has finished)

This happens with probability:

$$1 - [\text{probability of bad event for } G] \geq 1 - \frac{1}{n}$$

(i.e., with high probability (w.h.p.), since it goes to 1 as n goes to infinity)

Time complexity

Total number of phases:

$$4ed \ln n = O(d \log n) \quad \text{(w.h.p.)}$$

rounds for each phase: 3

1. In round 1, each node adjusts its neighborhood (according to round 3 of the previous phase), and then elects itself with probability 1/d; then, it notifies its neighbors on whether it succeeded or not;
2. In round 2, each node receives notifications of election from its neighbors (if any), decide whether it is in I_k, and if this is the case, it notifies its neighbors, and stops;
3. In round 3, each node receiving notifications from elected neighbors, realizes to be in $N(I_k)$, notifies its neighbors about that, and stops.

$\Rightarrow$ total # of rounds: $O(d \log n)$ (w.h.p.)

Homework

Can you provide a good bound on the total number of messages?

Can you see an asynchronous version of the algorithm?