Algorithms in Nature

Maximal Independent Set (MIS)

Overview

A minimum connected dominating set (MCDS) is a useful substructure of a graph representing a network:

- routing
- access control
- coverage

Computing a MCDS in a general graph is NP-complete …
… But an approximation may be much easier to compute.

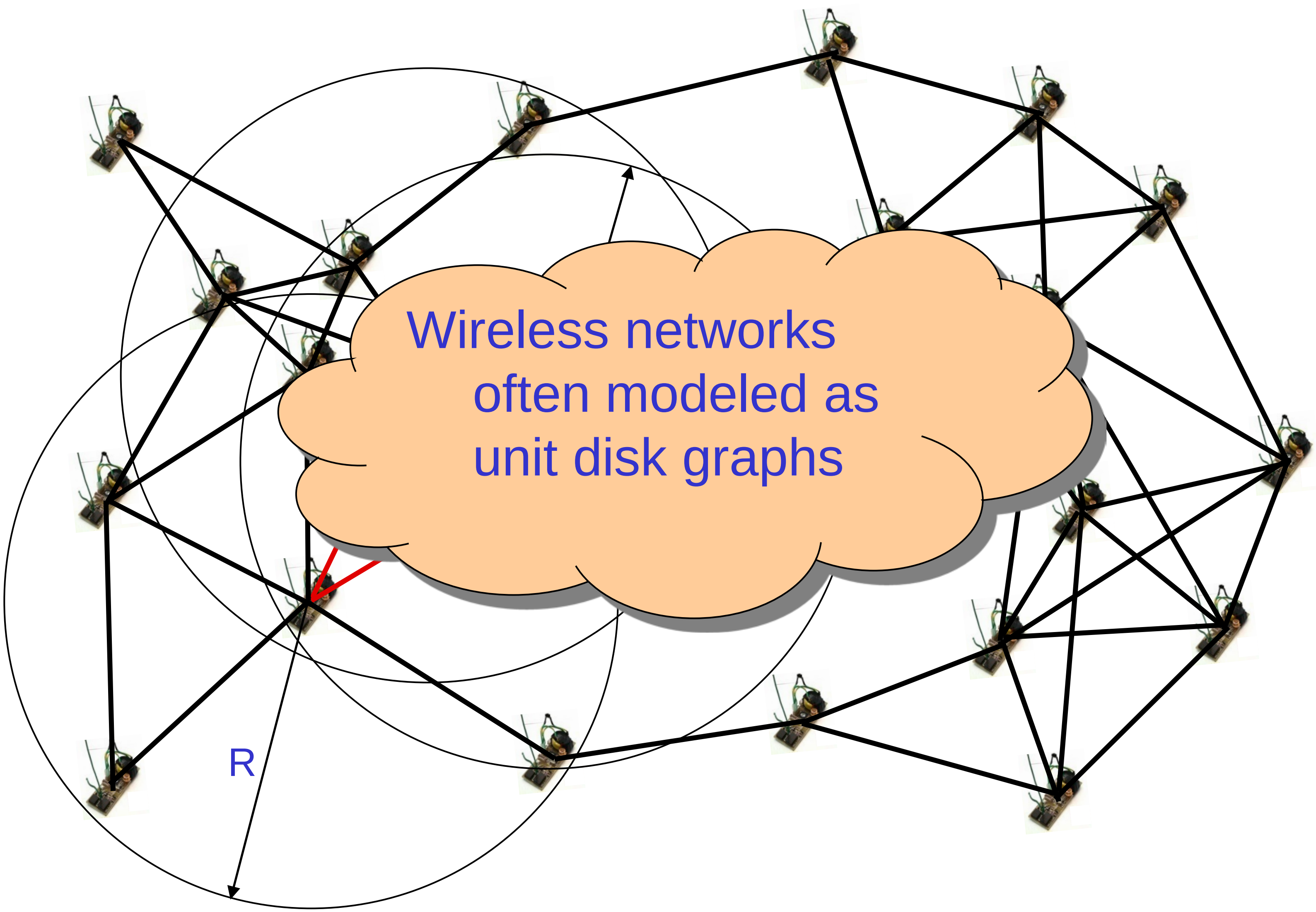
Overview

A

independent set is a constant approximation of a MCDS

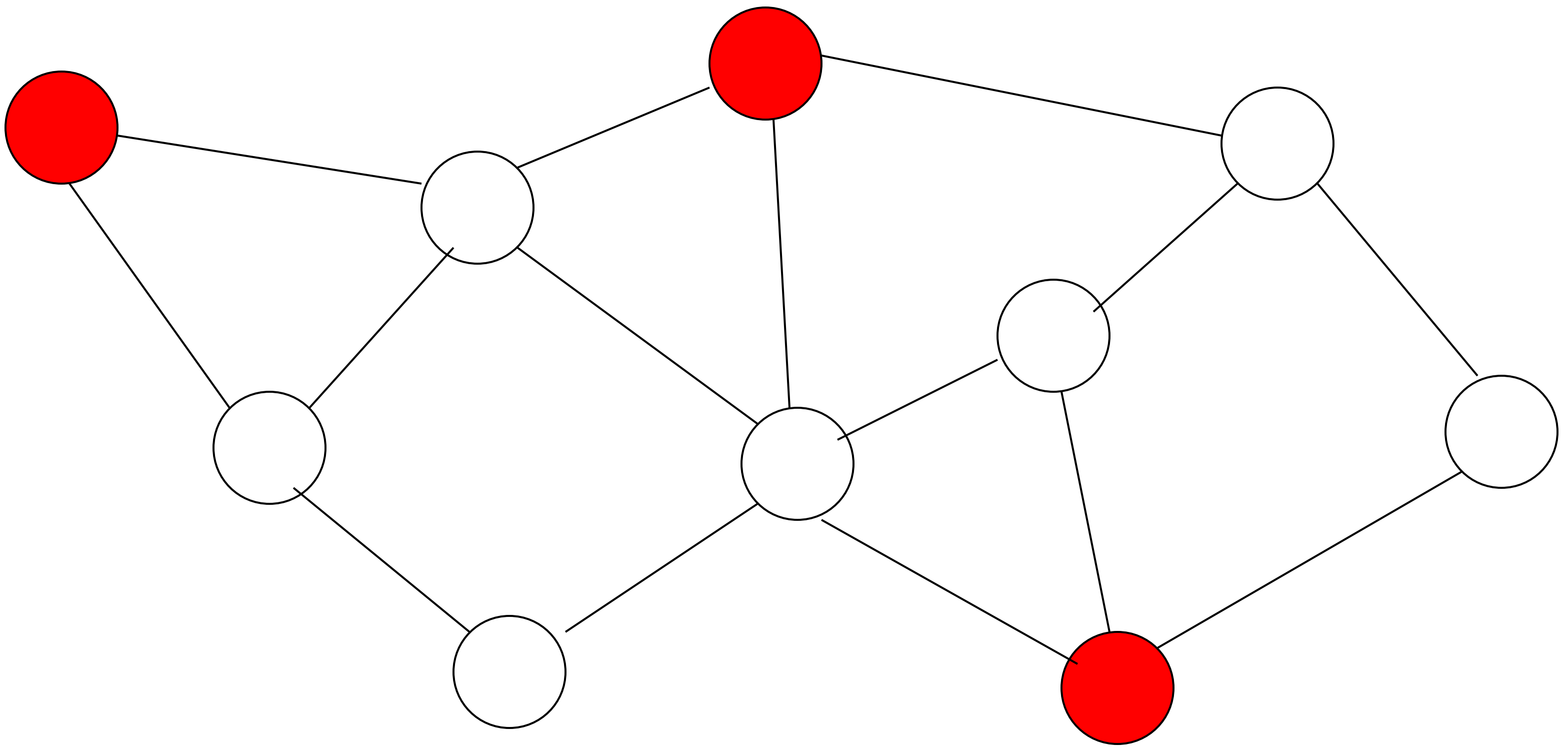
- A MIS is an independent subset S of the nodes of a graph (none of the nodes in S are neighbors), and no superset of S is independent

Unit Disk Graphs



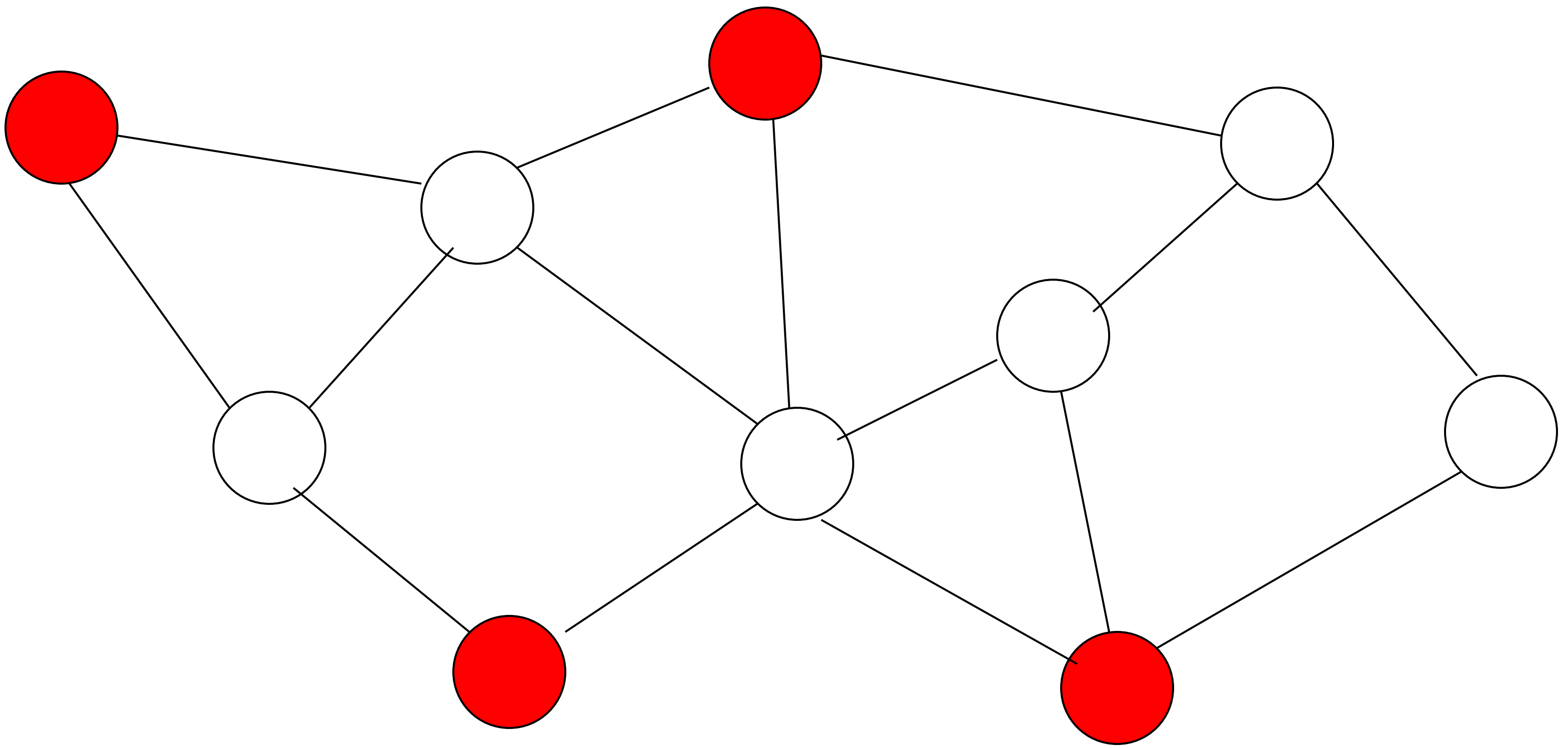
Independent Set (IS):

Any set of nodes that are not adjacent



Maximal Independent Set (MIS):

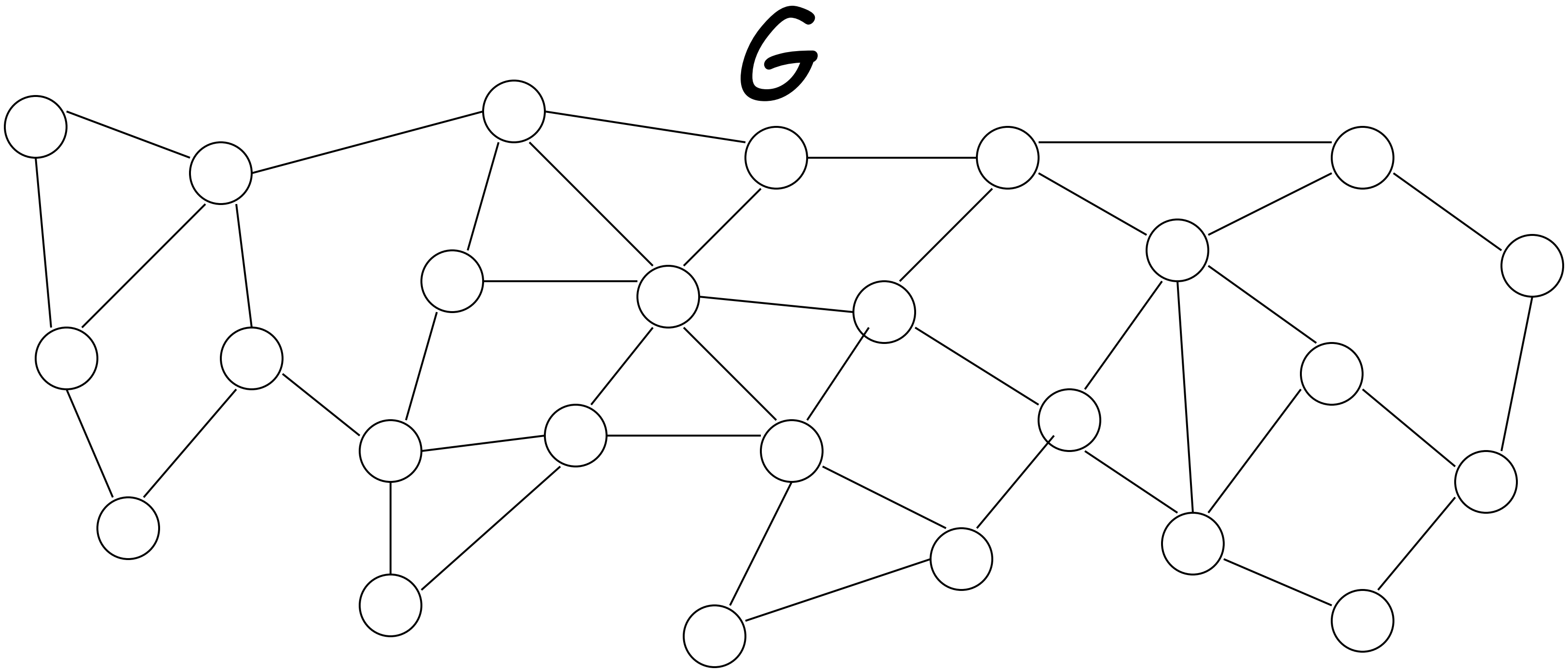
An independent set that is no subset of any other independent set



A Sequential Greedy algorithm

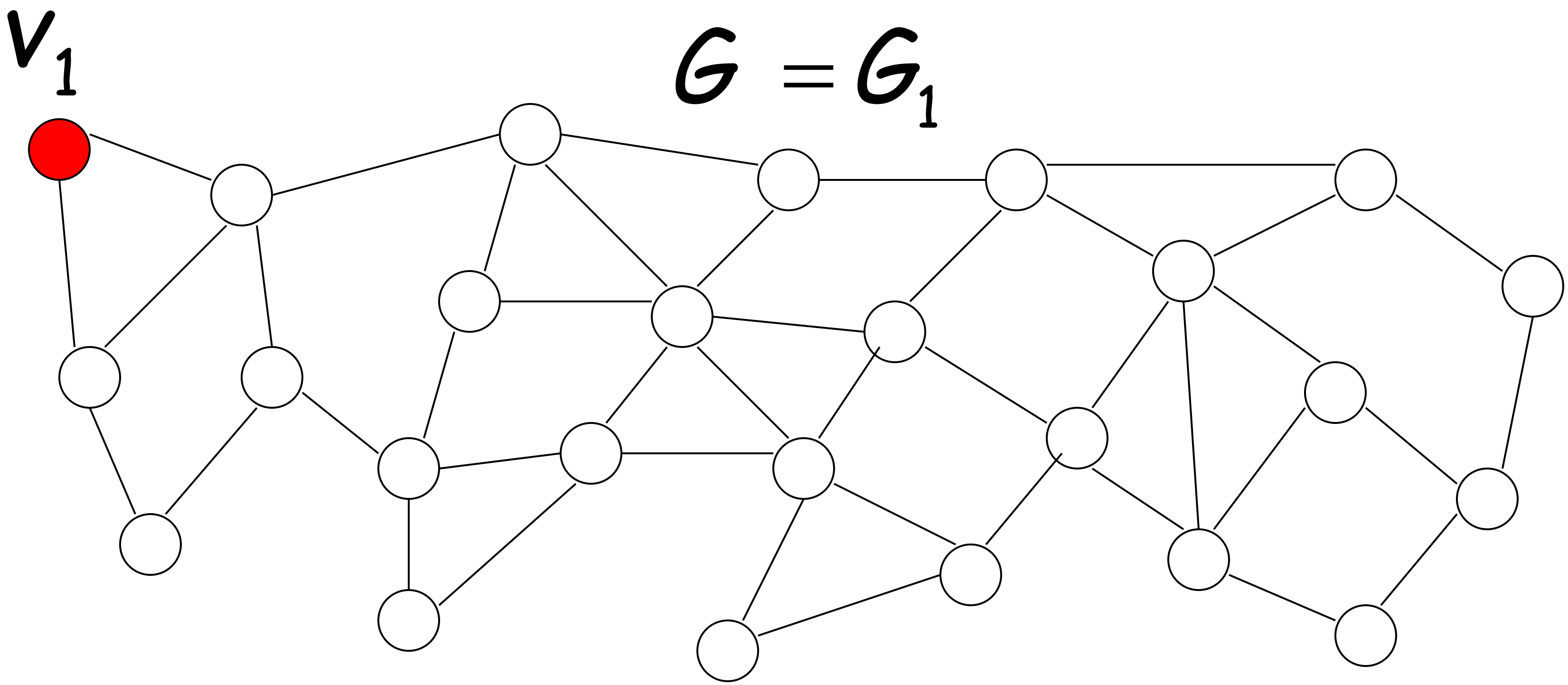
Suppose that I will hold the final MIS

Initially $I = \emptyset$



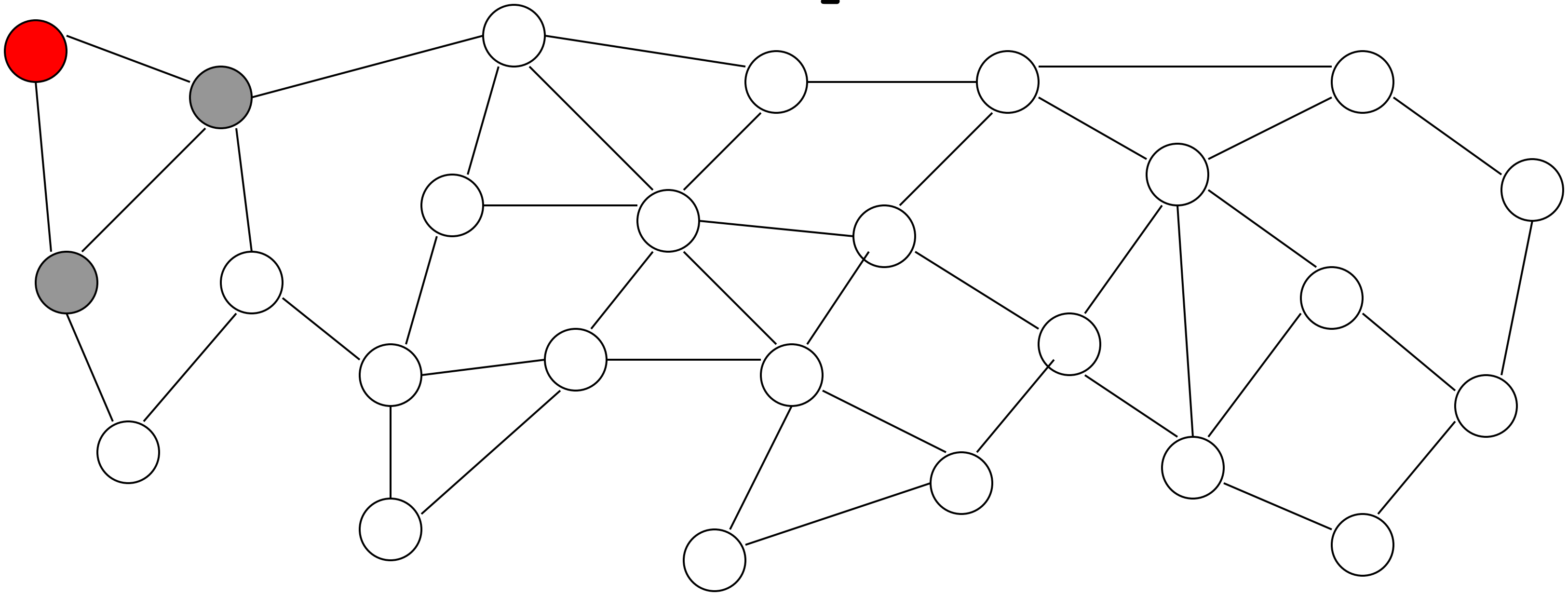
Phase 1:

Pick a node v_1 and add it to I



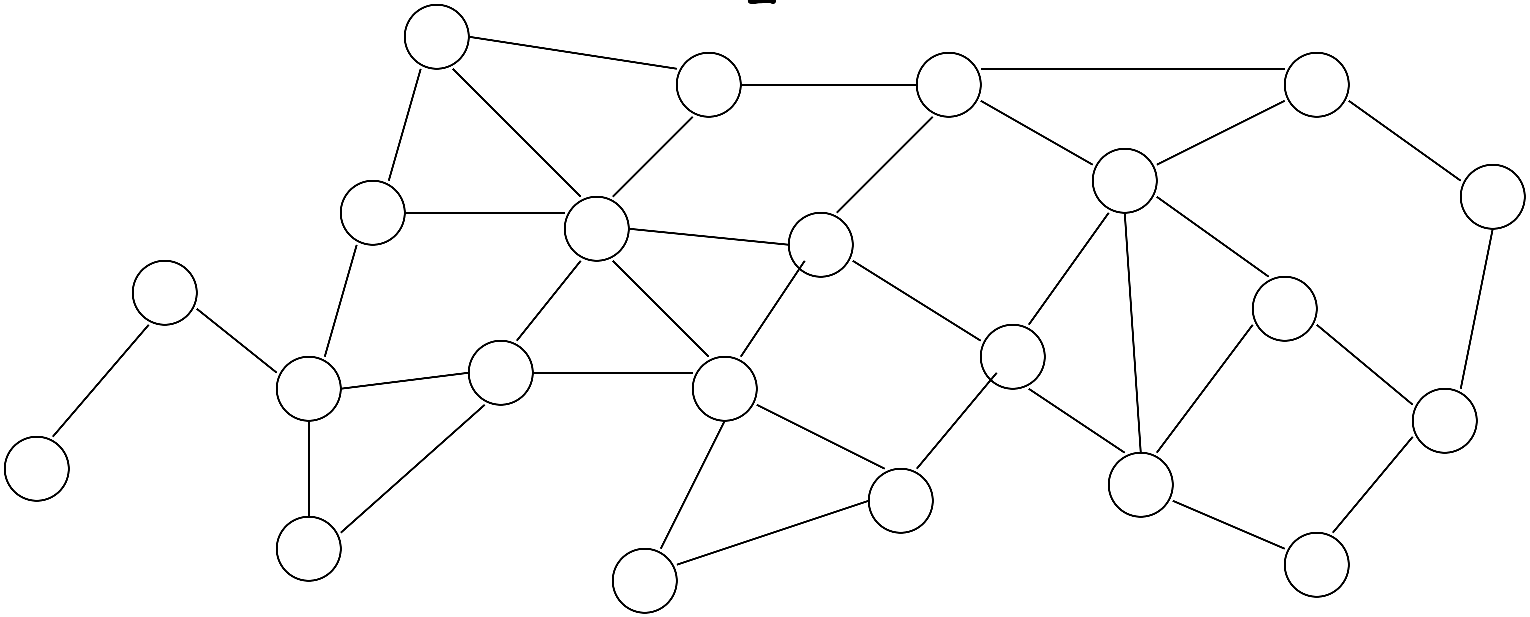
Remove v_1 and neighbors $N(v_1)$

G_1



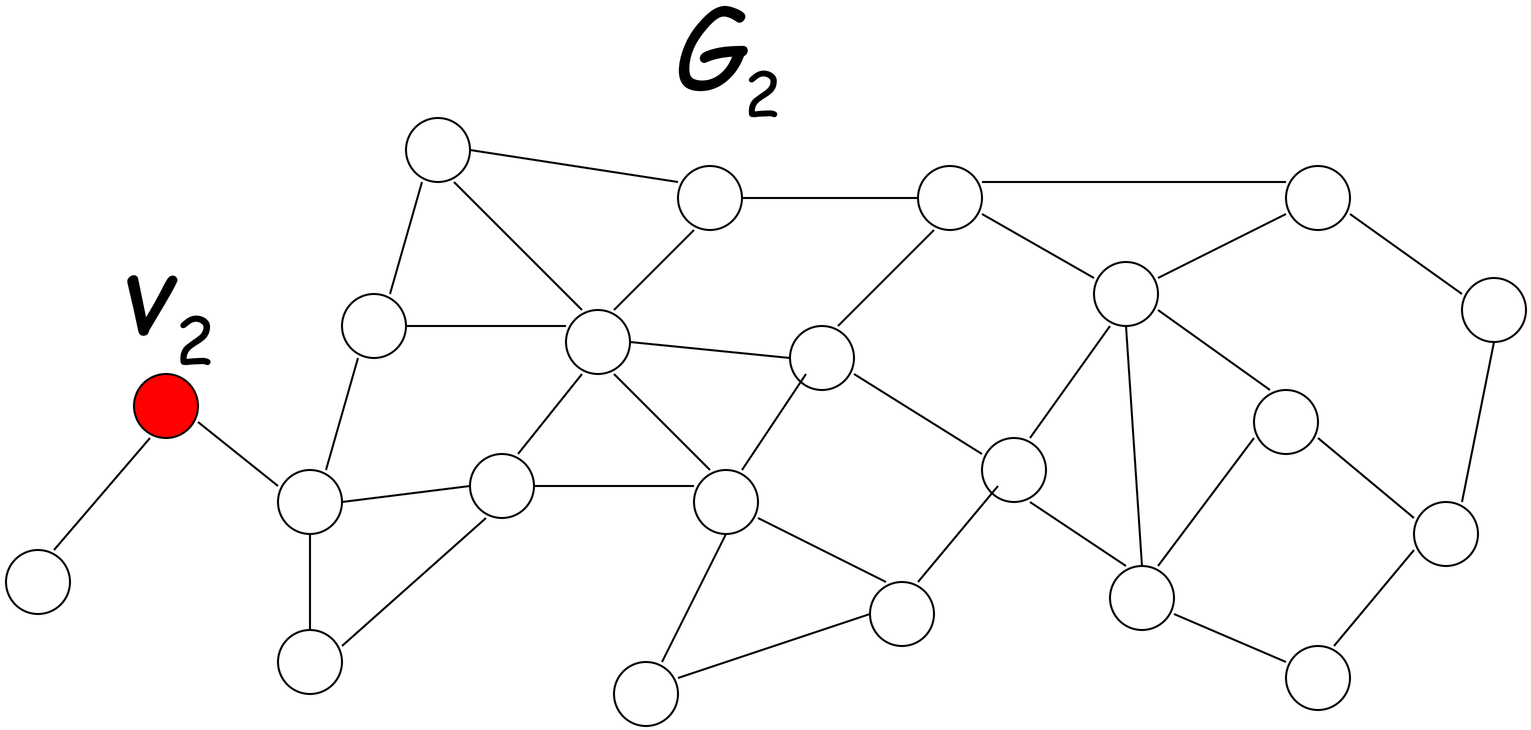
Remove v_1 and neighbors $N(v_1)$

G_2

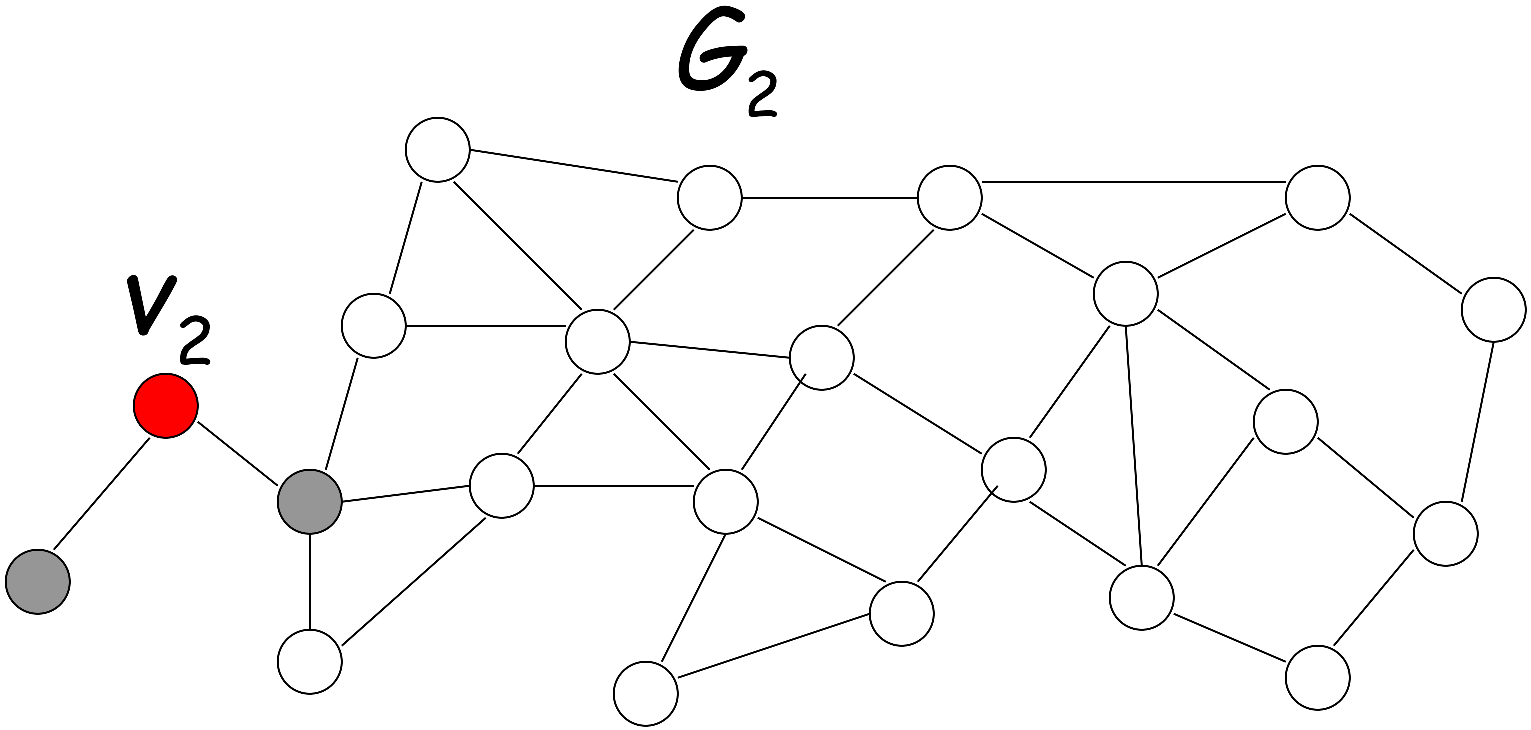


Phase 2:

Pick a node v_2 and add it to I

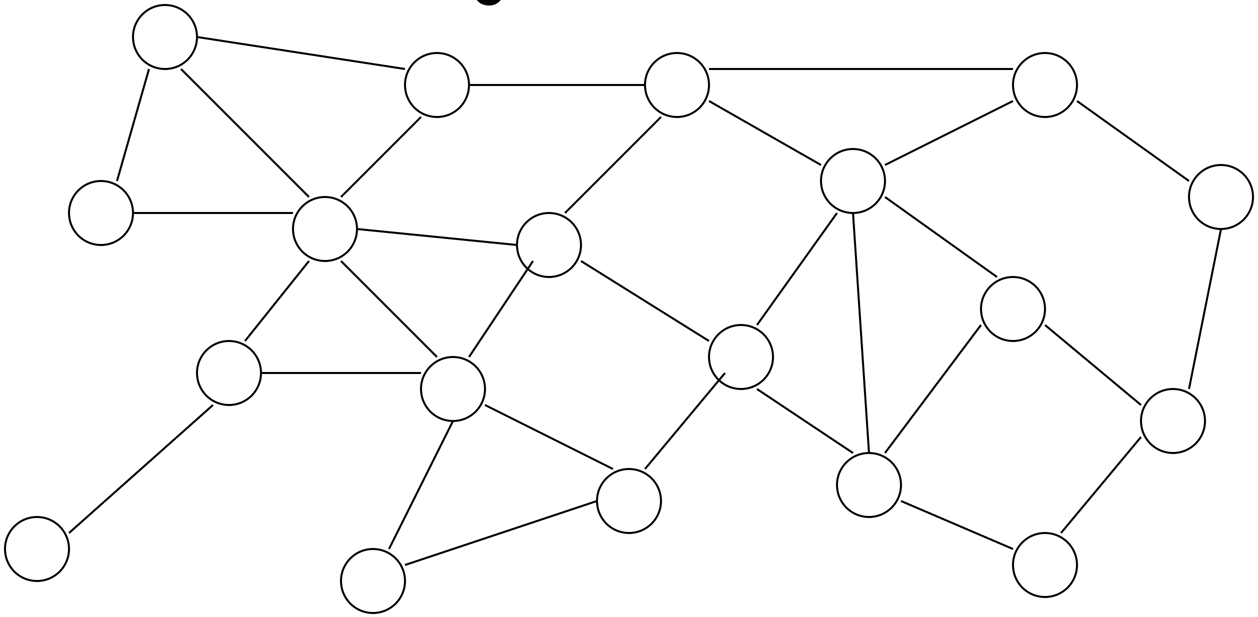


Remove v_2 and neighbors $N(v_2)$



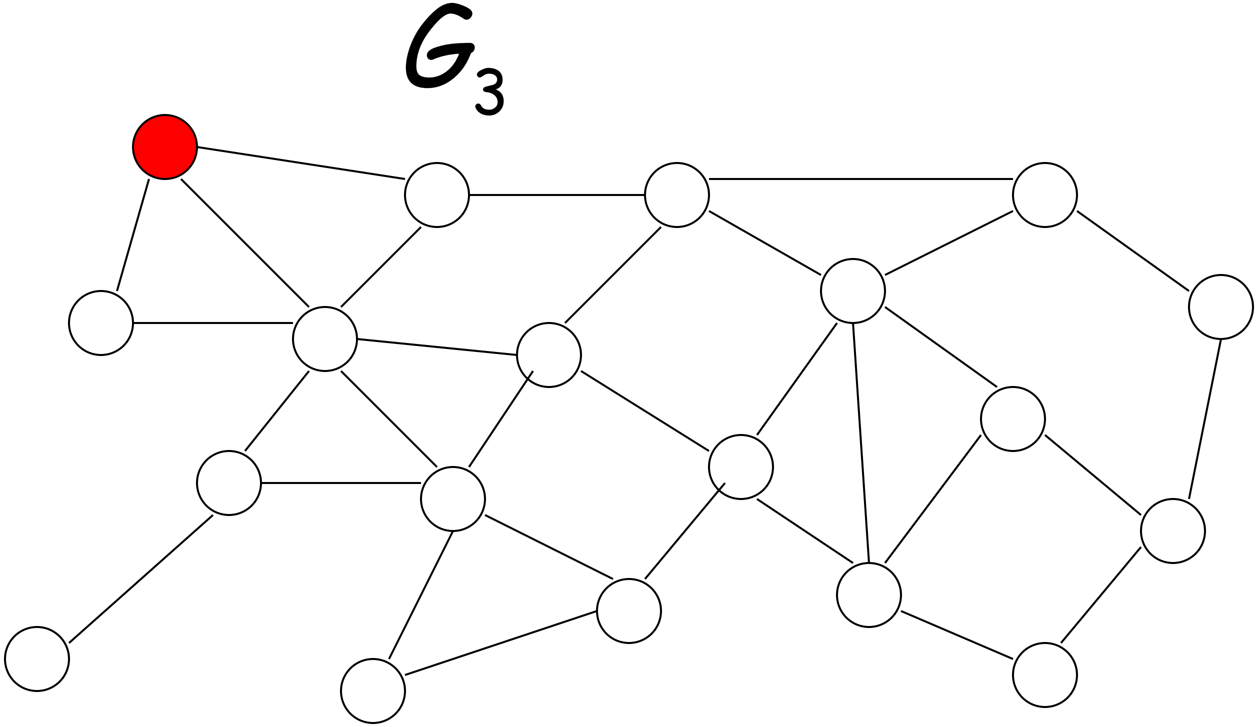
Remove v_2 and neighbors $N(v_2)$

G_3



Phases 3,4,5,...:

Repeat until all nodes are removed



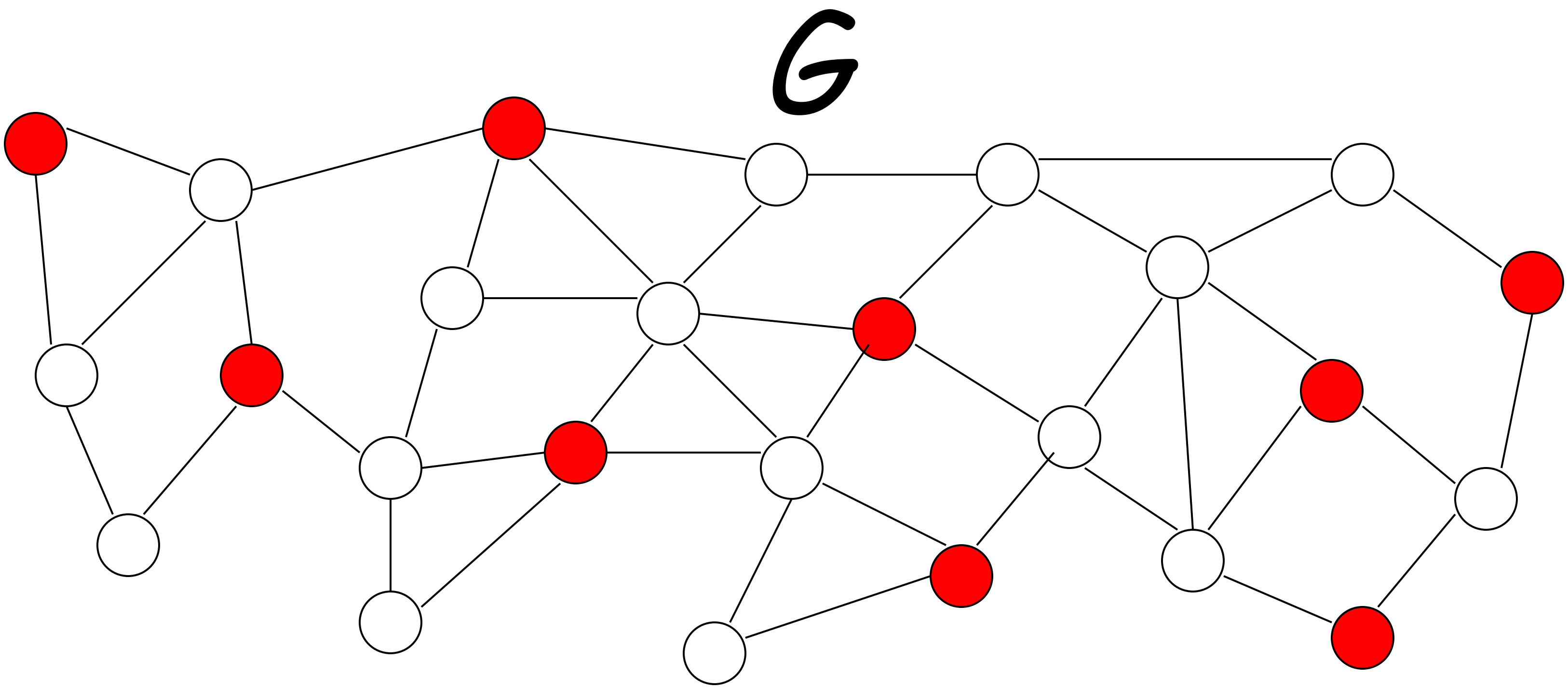
Phases 3,4,5,…,x:

Repeat until all nodes are removed

$$G_{x+1}$$

No remaining nodes

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



Running time of algorithm: $O(n)$

Worst case graph:

n nodes

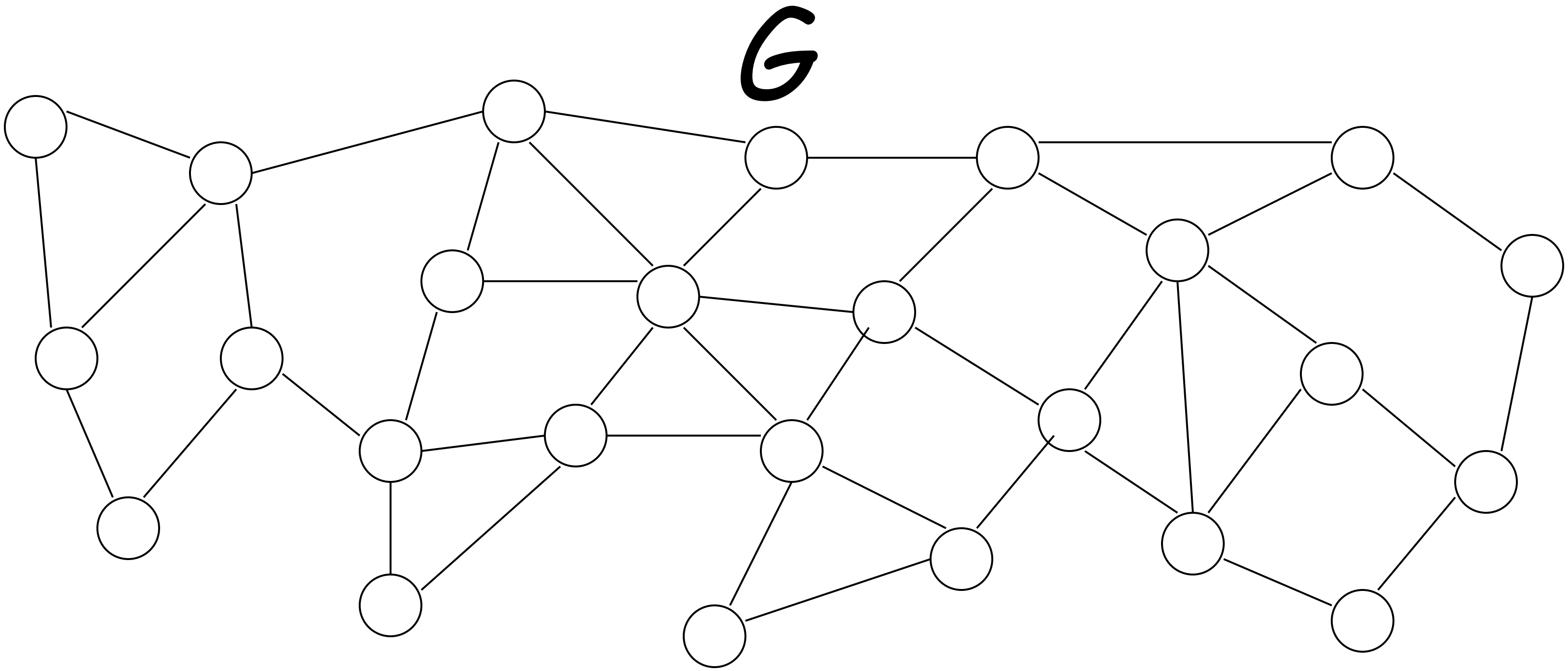
A General Algorithm For Computing MIS

Same as the sequential greedy algorithm,
but
at each phase we may select
any independent set (instead of a single node)

Example:

Suppose that I will hold the final MIS

Initially $I = \emptyset$

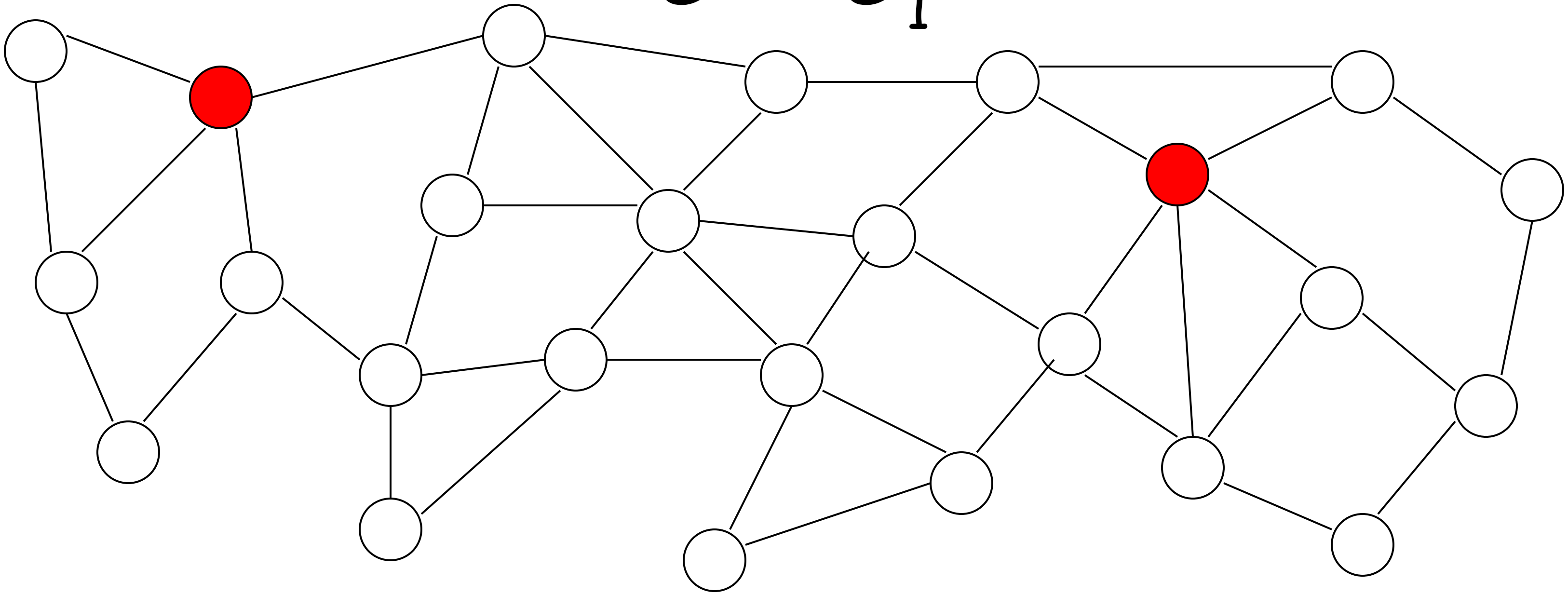


Phase 1:

Find any independent set I_1

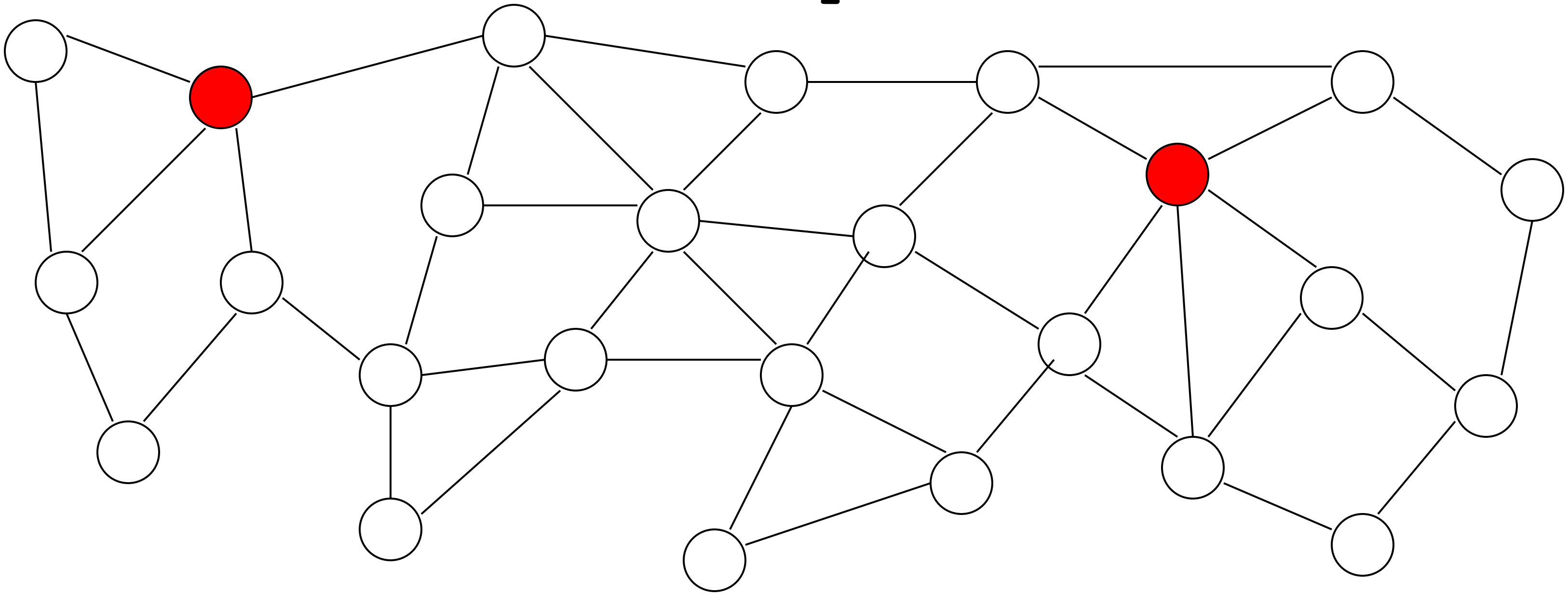
And insert I_1 to I : $I \leftarrow I \cup I_1$

$G = G_1$



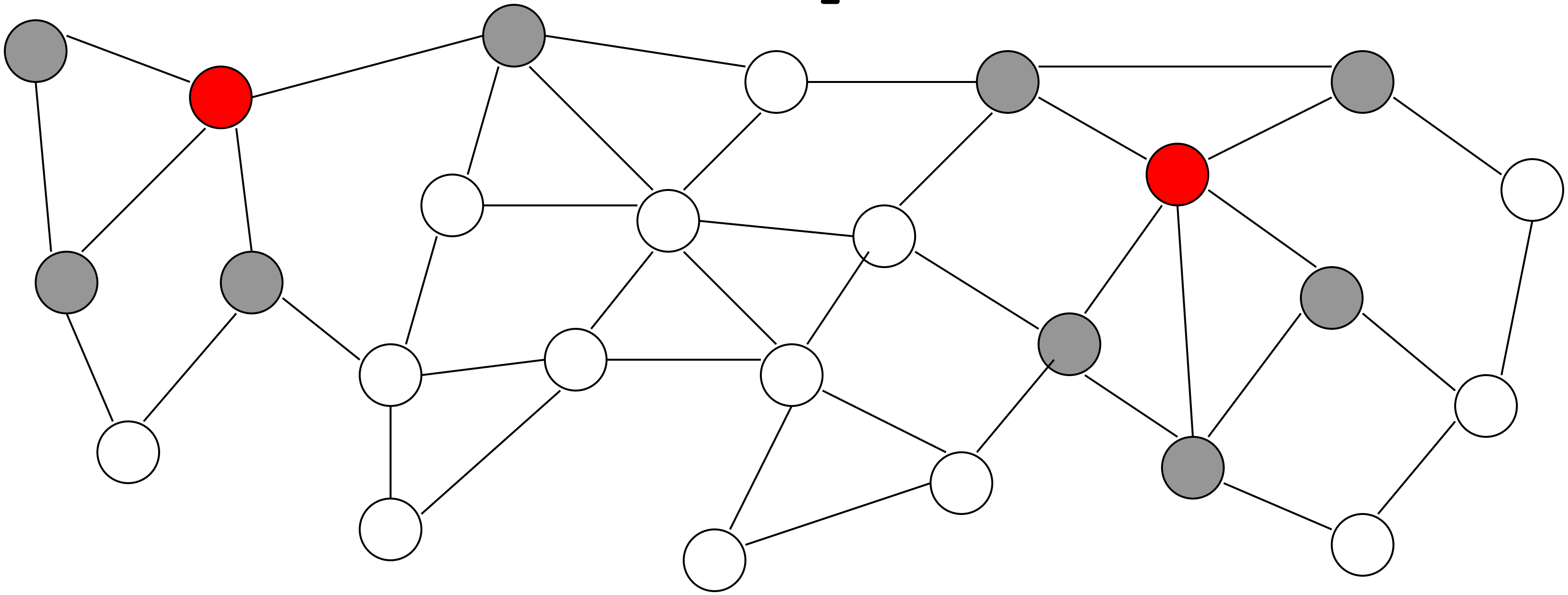
remove I_1 and neighbors $N(I_1)$

G_1



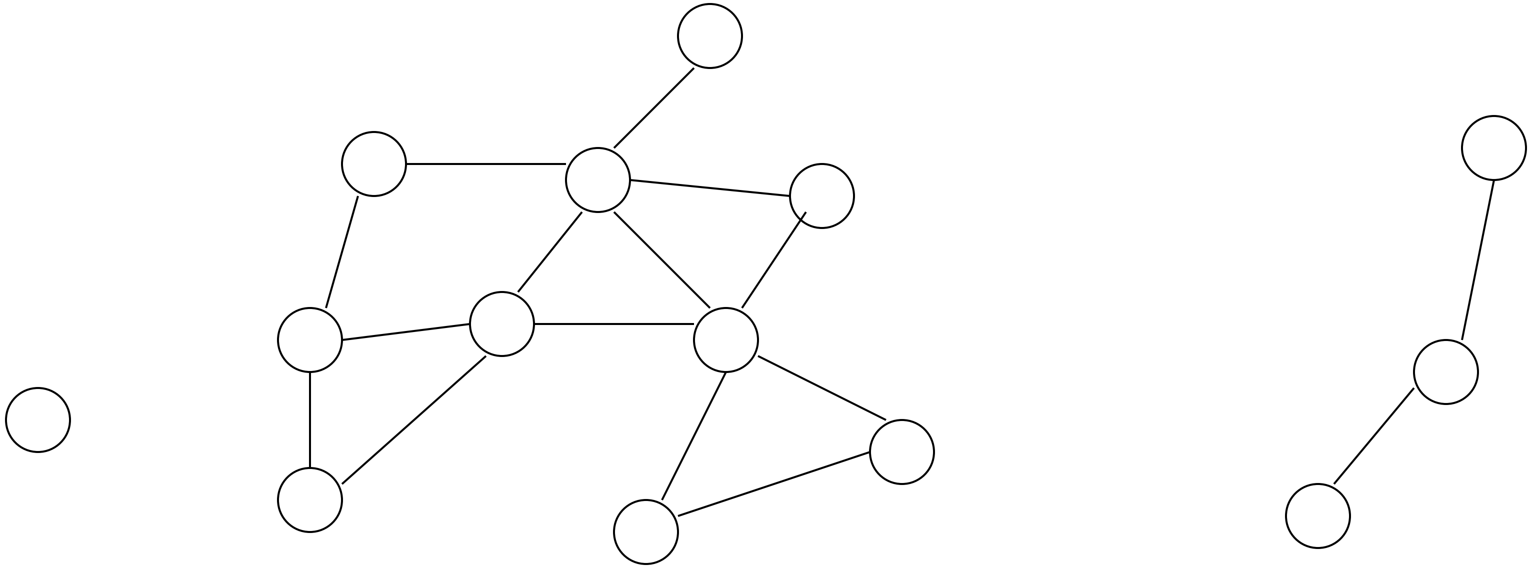
remove I_1 and neighbors $N(I_1)$

G_1



remove I_1 and neighbors $N(I_1)$

G_2



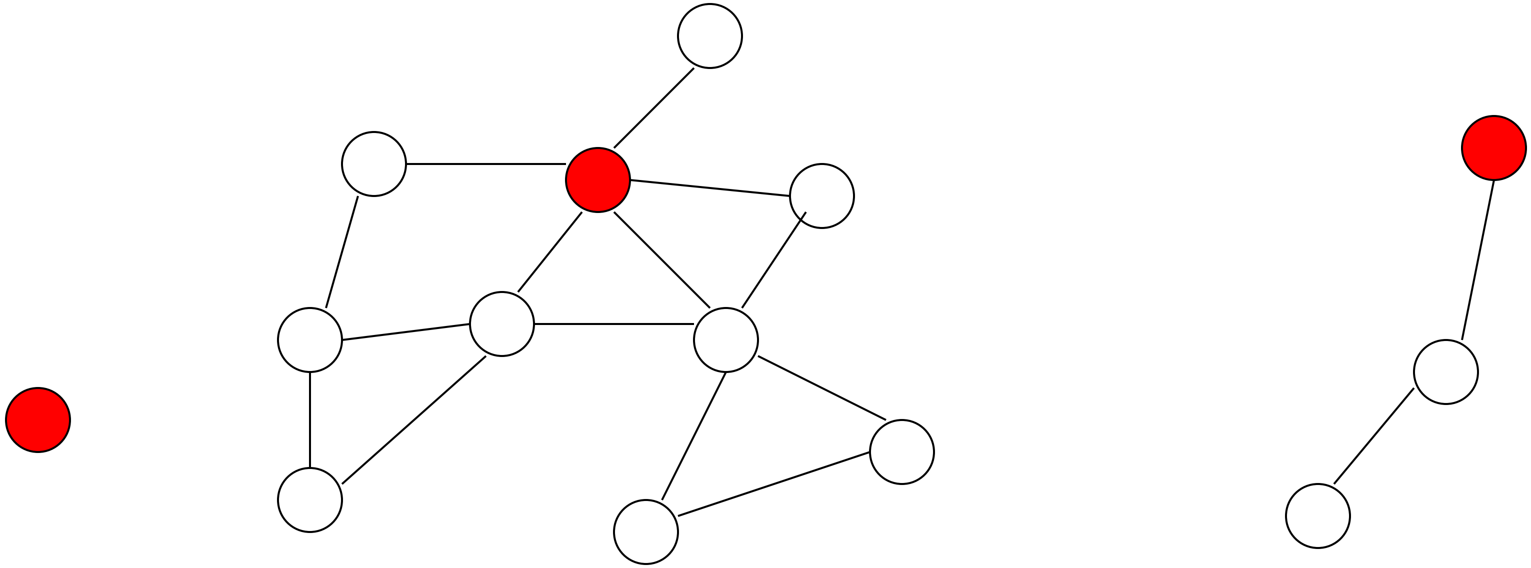
Phase 2:

On new graph

Find any independent set I_2

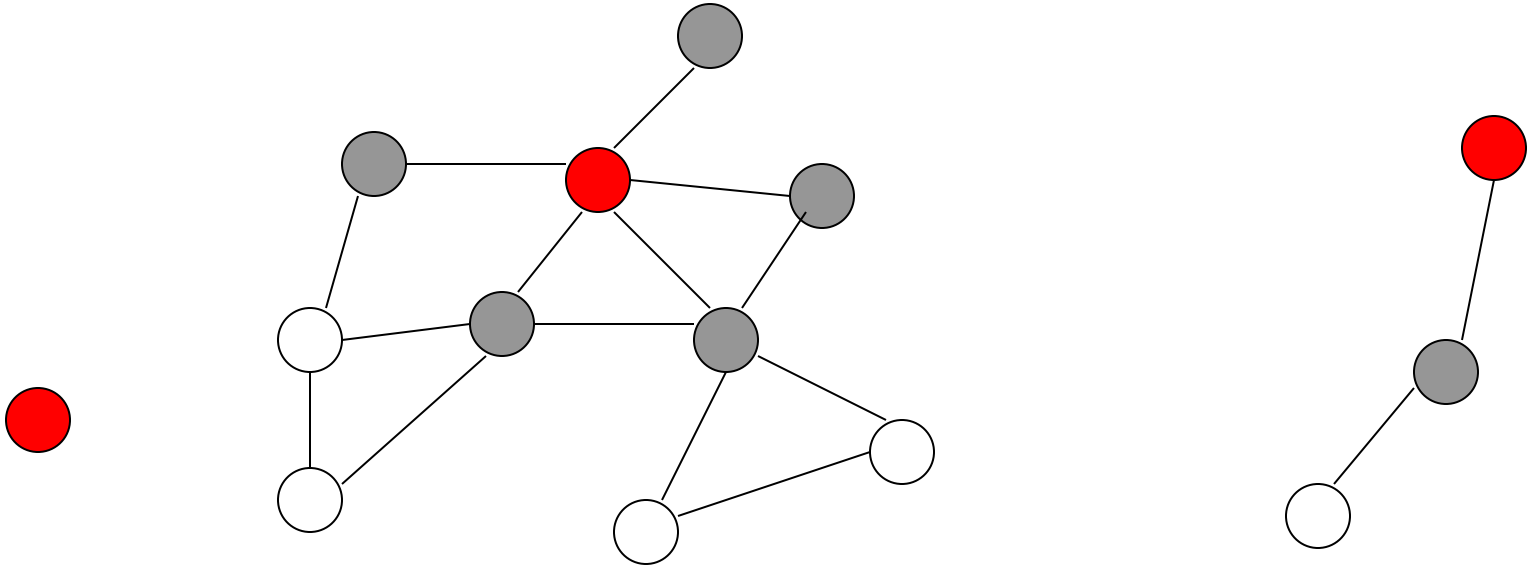
And insert I_2 to I : $I \leftarrow I \cup I_2$

G_2



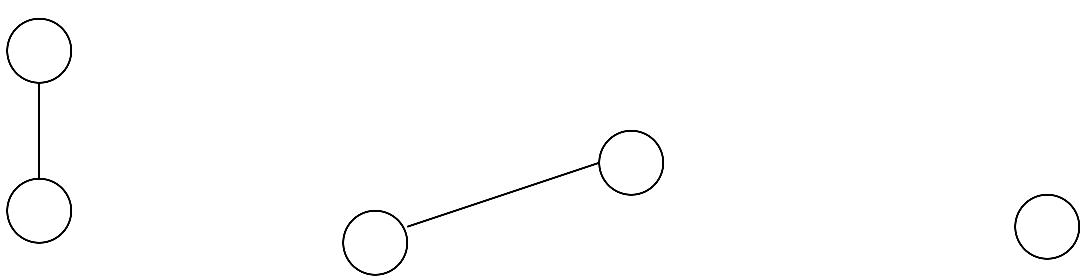
remove I_2 and neighbors $N(I_2)$

G_2



remove I_2 and neighbors $N(I_2)$

G_3



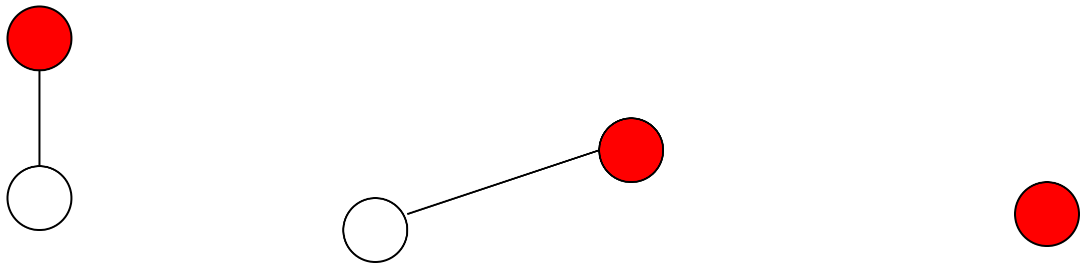
Phase 3:

On new graph

Find any independent set I_3

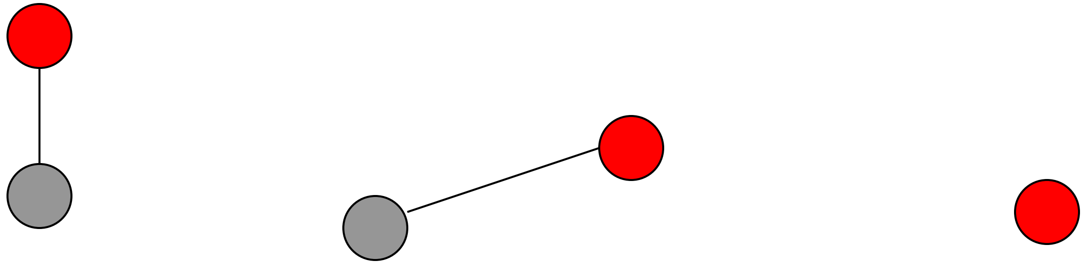
And insert I_3 to I : $I \leftarrow I \cup I_3$

G_3



remove I_3 and neighbors $N(I_3)$

G_3

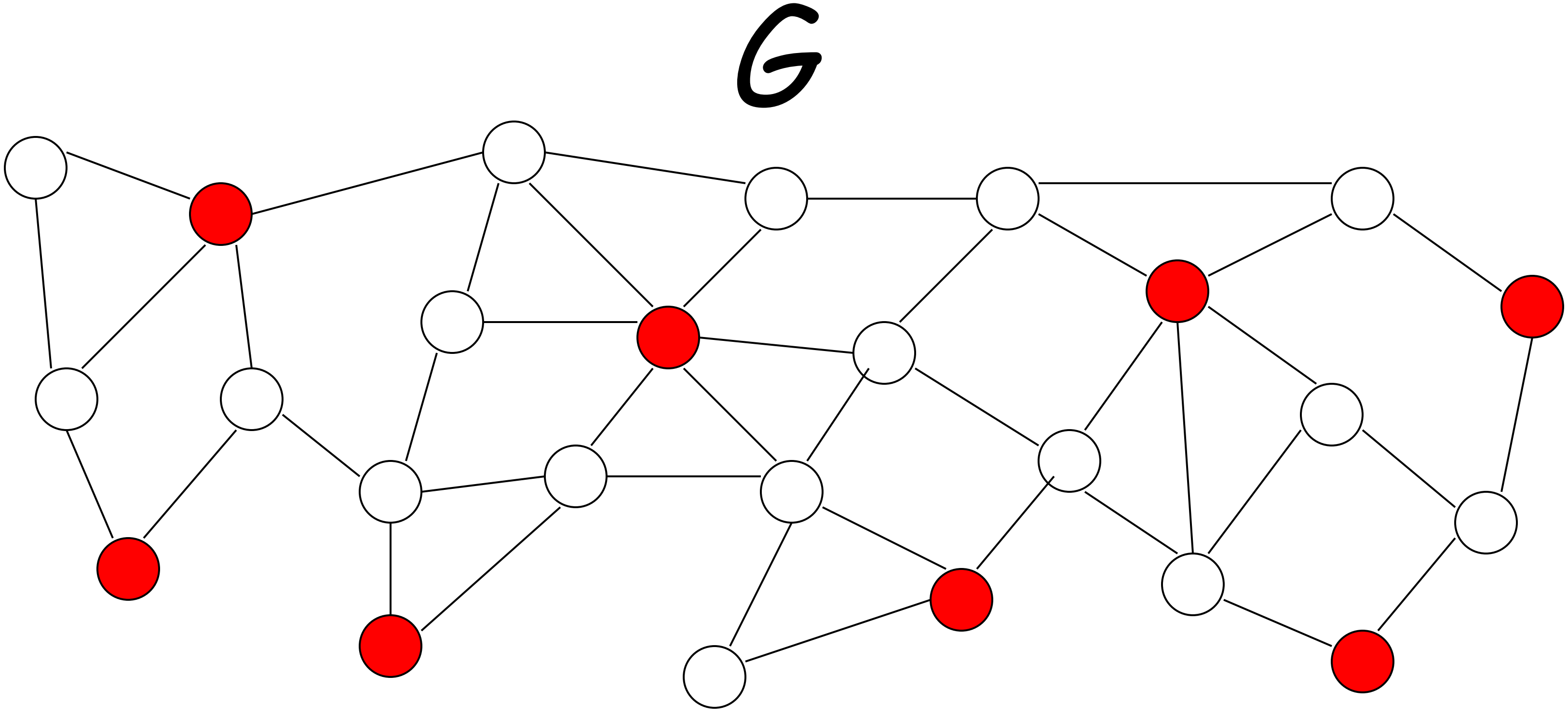


remove I_3 and neighbors $N(I_3)$

G_4

No nodes are left

Final MIS I



Observation:

The number of phases depends on the choice of independent set in each phase:

The larger the independent set at each phase the faster the algorithm

Example: If I_1 is MIS,
1 phase is needed

Example: If each I_k contains one node,
$O(n)$ phases are needed

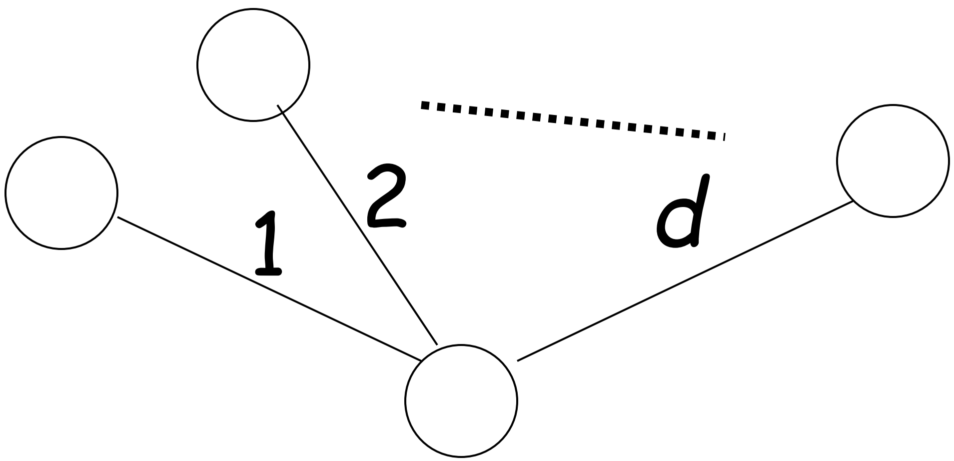
(sequential greedy algorithm)

A Simple Distributed Algorithm

Same as the general MIS algorithm

At each phase the independent set
is chosen randomly so that it includes
many nodes of the remaining graph

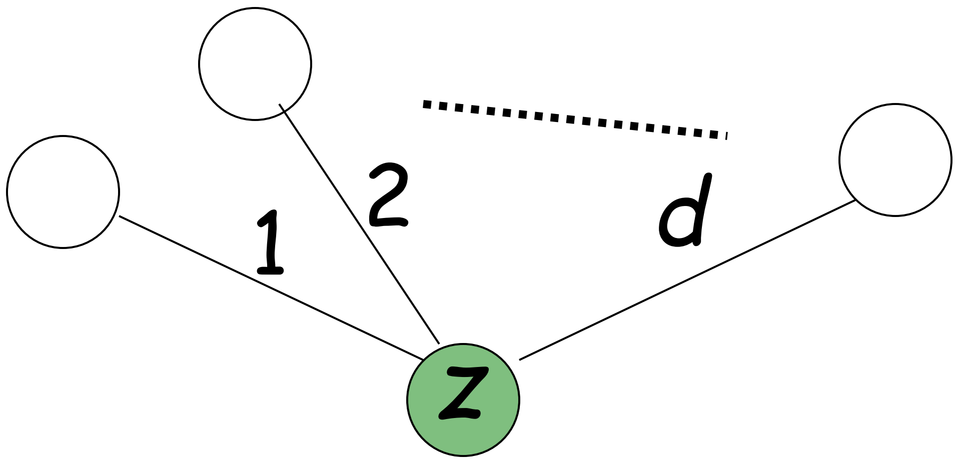
Let d be the maximum node degree in the whole graph



Suppose that d is known to all the nodes

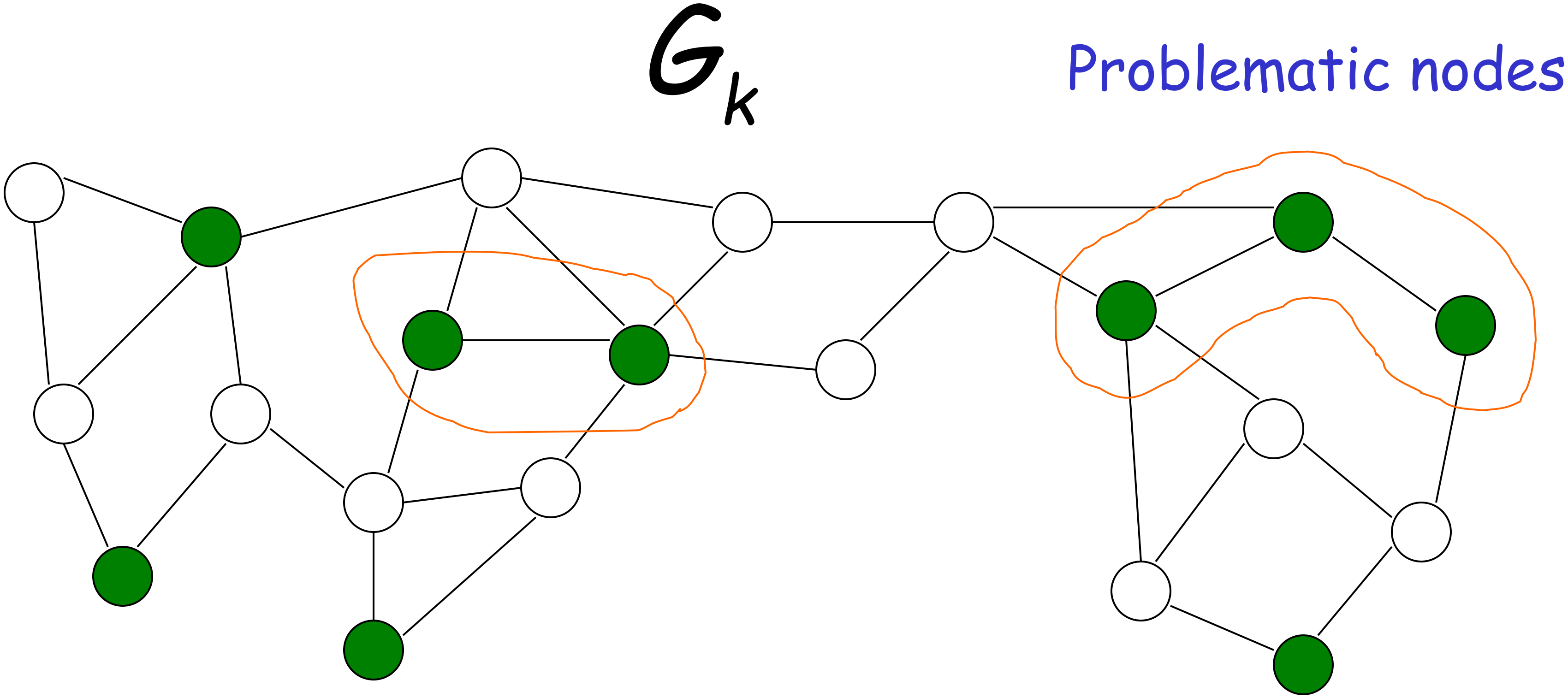
At each phase k:

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

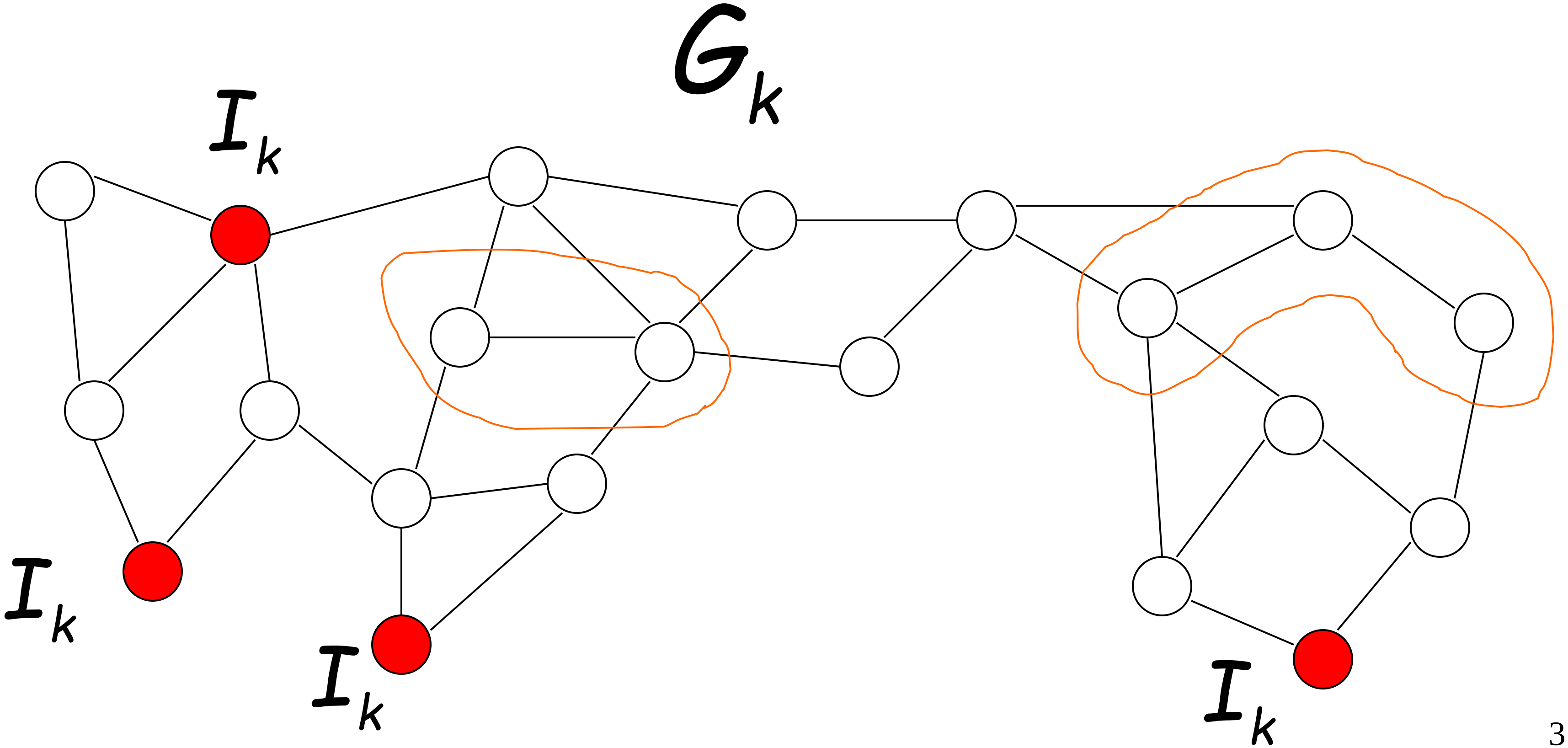


Elected nodes are candidates for independent set I_k

However, it is possible that neighbor nodes may be elected simultaneously



All the problematic nodes must be un-elected. The remaining elected nodes form independent set I_k



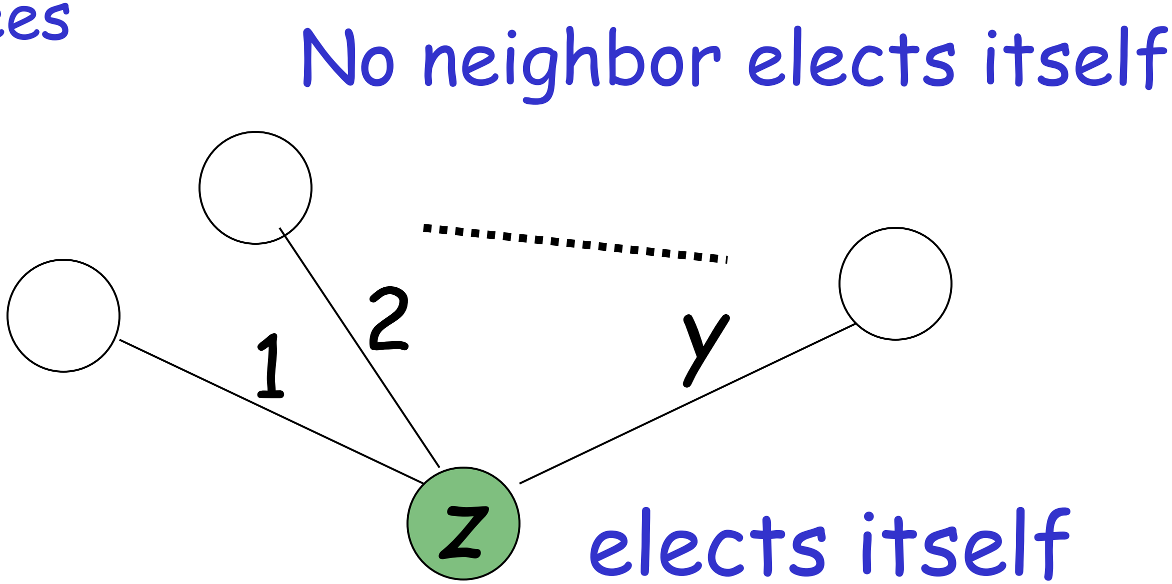
Analysis:

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

z disappears at end of phase k

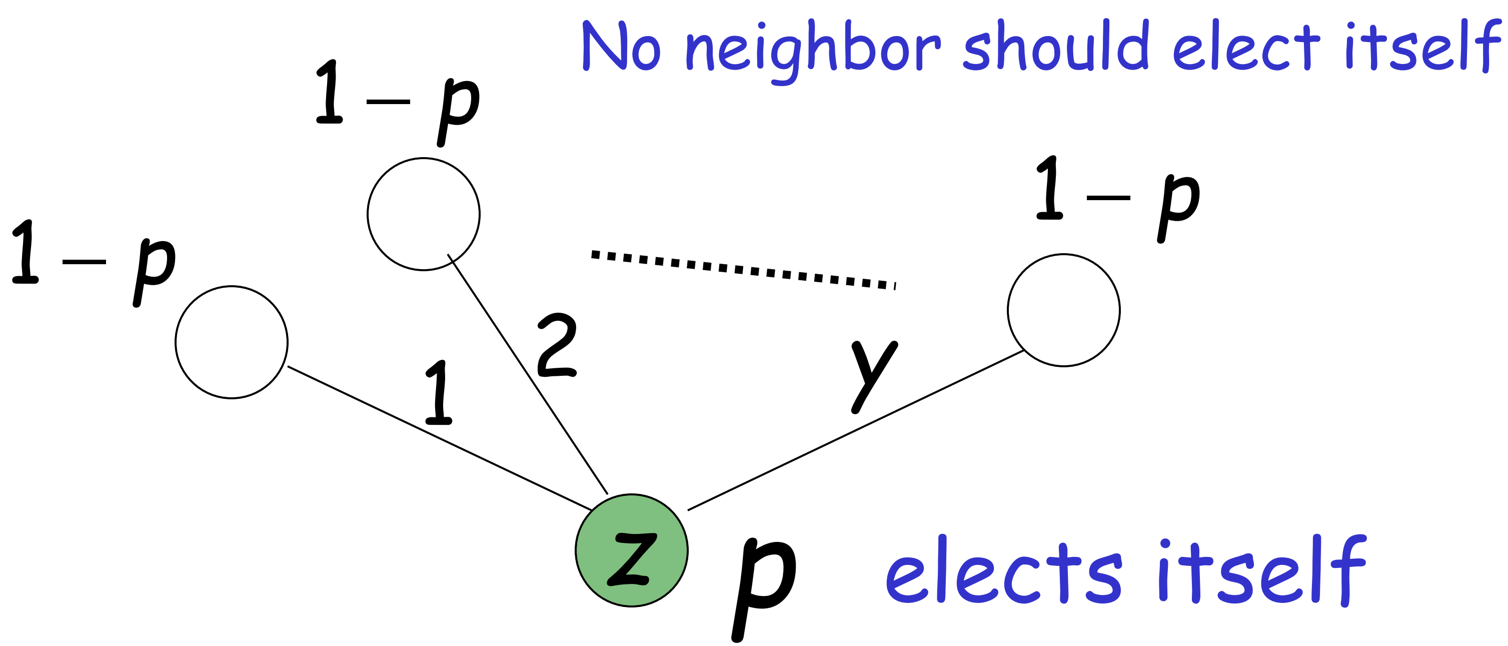
(enters I_k or $N(I_k)$)

A good scenario that guarantees success



Probability of success in phase:

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



Fundamental inequalities

$d > 0$

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

Probability of success in phase:

At least

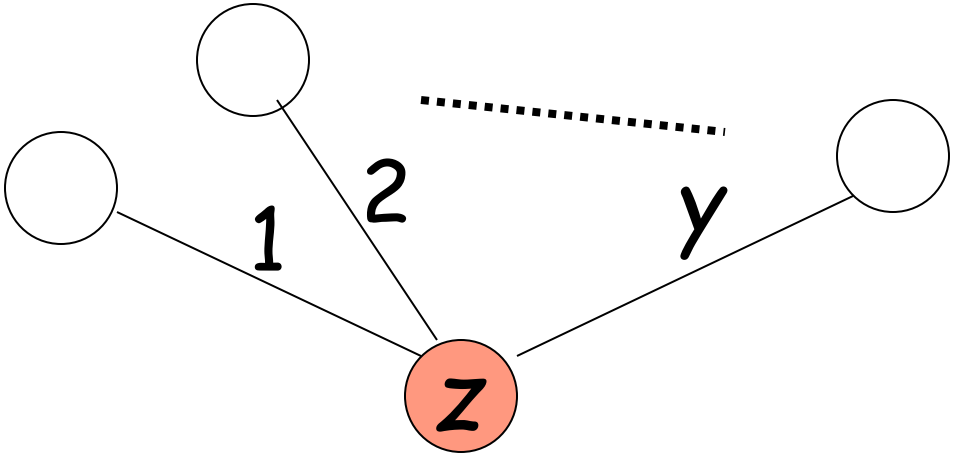
$$p(1-p)^y \geq p(1-p)^d$$

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

$$\cong \frac{1}{ed}$$

For $d \geq 2$

Therefore, node z will enter I_k and disappear in phase k with probability at least $\frac{1}{ed}$



Expected number of phases until node z disappears:

at most $\dfrac{1}{\text{probability of sucess in phase}} = ed$ phases

Bad event for node z:

after $2ed \ln n$ phases

node z did not disappear

Probability:

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

Bad event for any node in G :

after $2ed \ln n$ phases

at least one node did not disappear

Probability:

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

Good event for all nodes in G :

within $2ed \ln n$ phases

all nodes disappear

Probability:

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

(high probability)

Total number of phases:

$$2ed \ln n = O(d \log n)$$

with high probability

Time duration of each phase: $O(1)$

Total time: $O(d \log n)$

Still could be very large if the max degree is O(n)

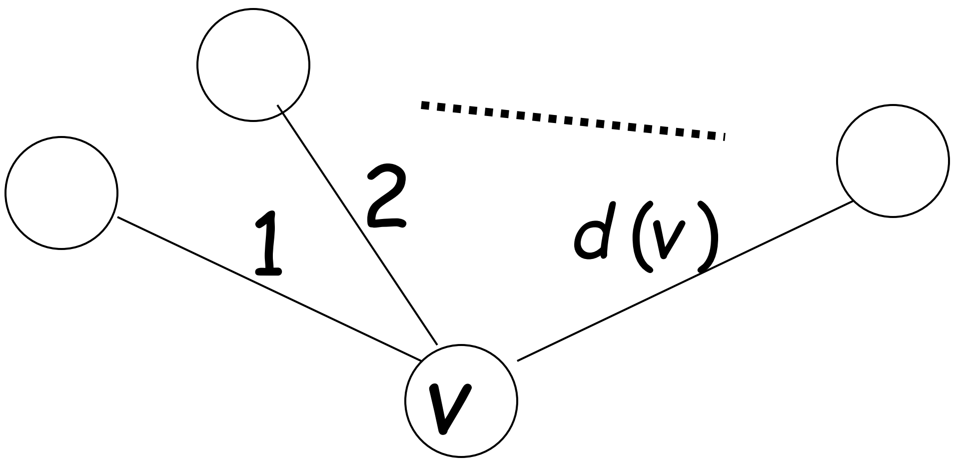
Luby's MIS Distributed Algorithm

Runs in time $O(\log n)$ in expected case

$\boxed{O(\log d \cdot \log n)}$ with high probability

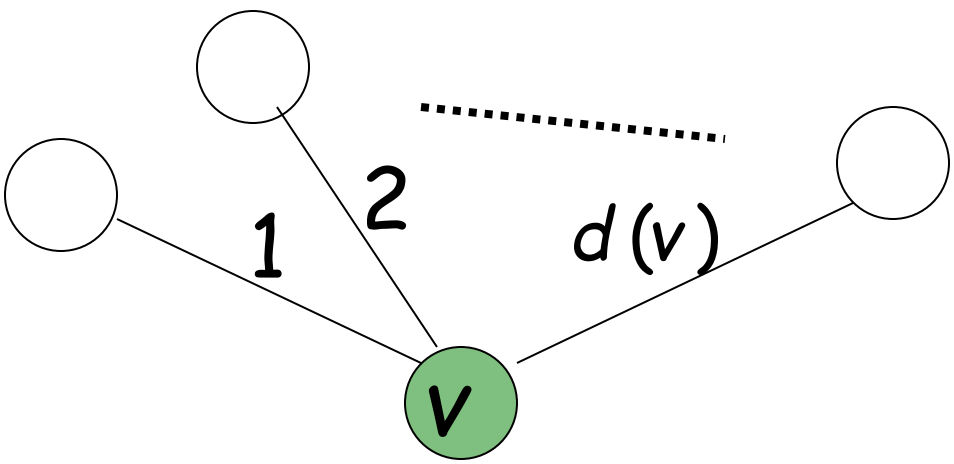
this algorithm is asymptotically better than the previous

Let $d(v)$ be the degree of node v



At each phase k:

Each node $v \in G_k$ elects itself with probability $p(v) = \frac{1}{2d(v)}$

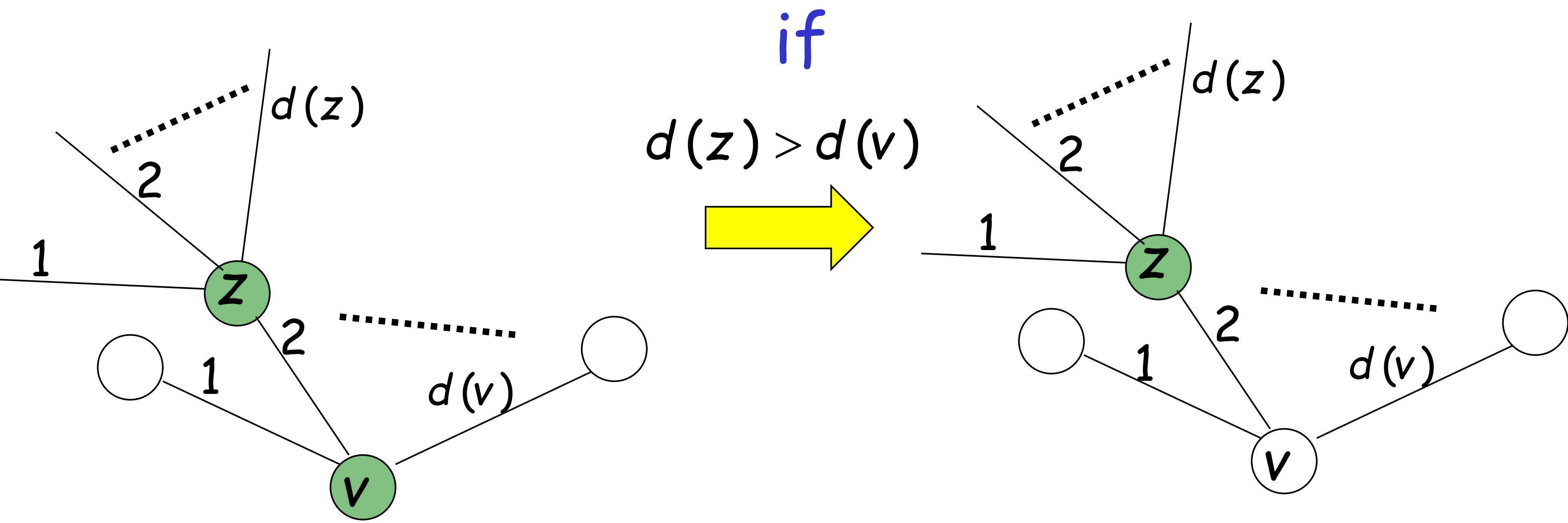


degree of v in G_k

Elected nodes are candidates for the independent set I_k

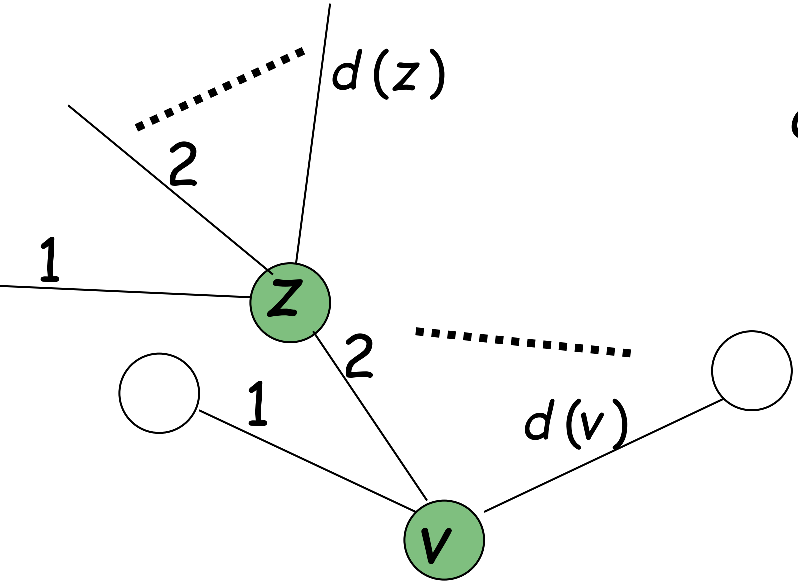
If two neighbors are elected simultaneously, then the higher degree node wins

Example:



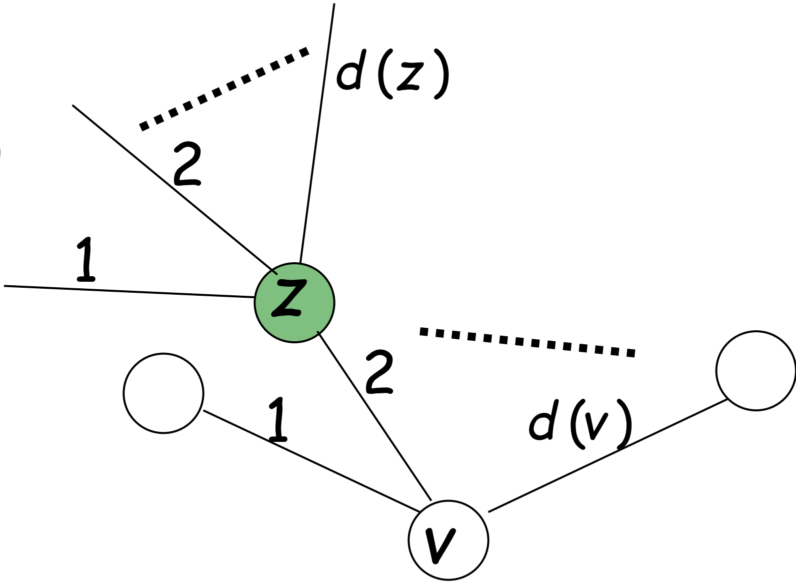
If both have the same degree,
ties are broken arbitrarily

Example:

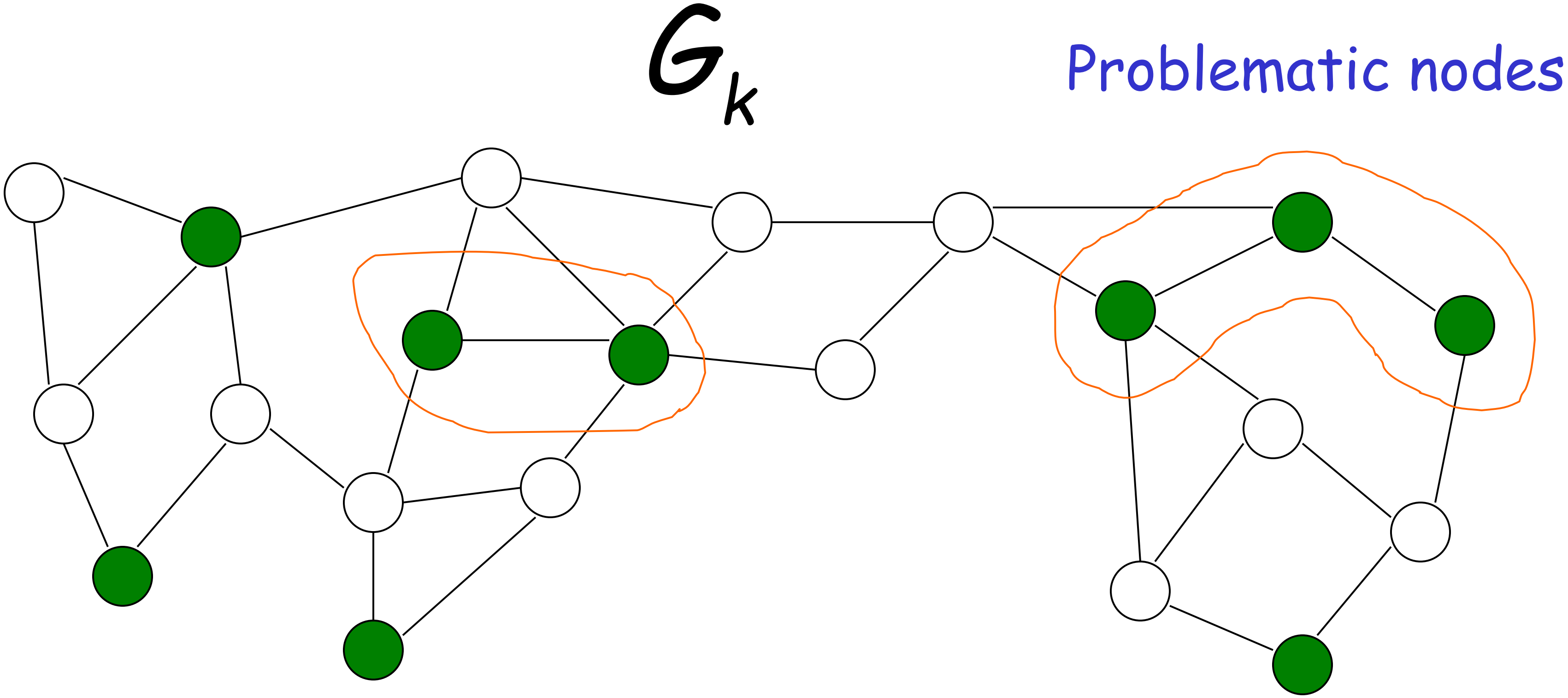


if

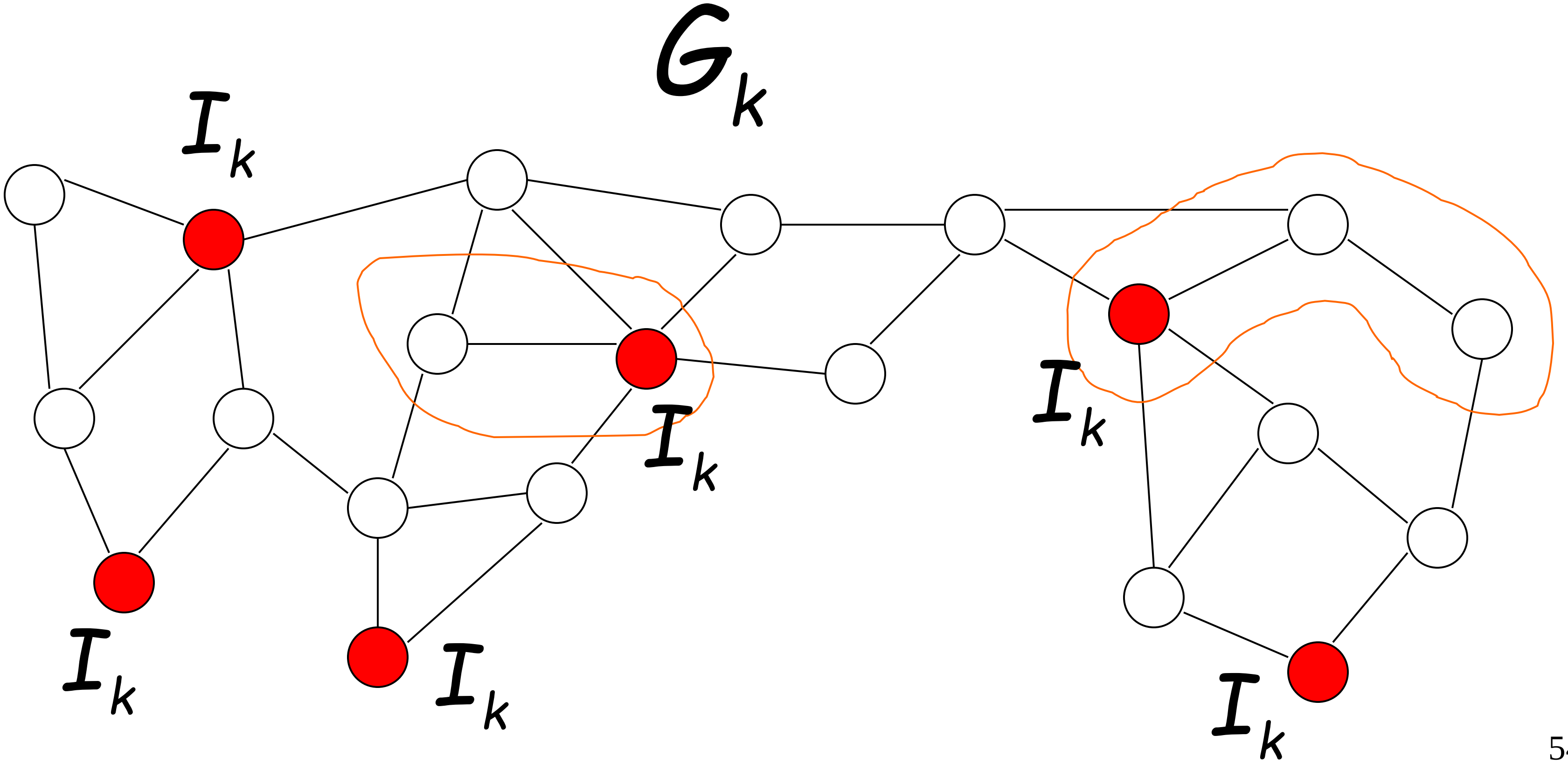
$$d(z) = d(v)$$



Using previous rules, problematic nodes are removed



The remaining elected nodes form independent set I_k



Luby's MIS Distributed Algorithm (1982)

this algorithm is asymptotically better than the previous

Runs in time $O(\log n)$ in expected case

$\boxed{O(\log d \cdot \log n)}$ with high probability