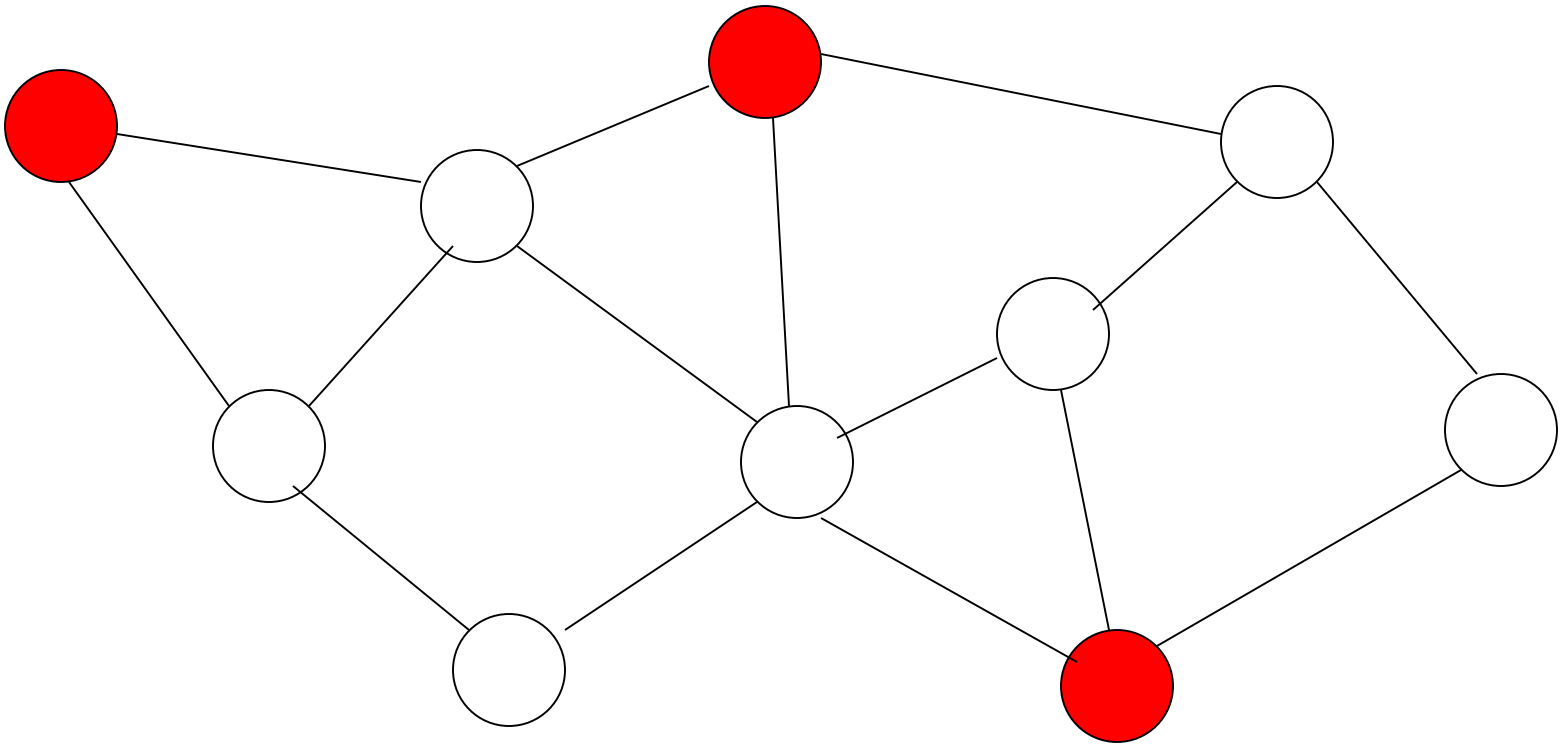
Maximal Independent Set

Credits: ***Guido Proietti***
Rezaul A. Chowdhury

Independent (or stable) Set (IS):



Independent Sets

Let $G = (V, E)$ be an undirected graph.

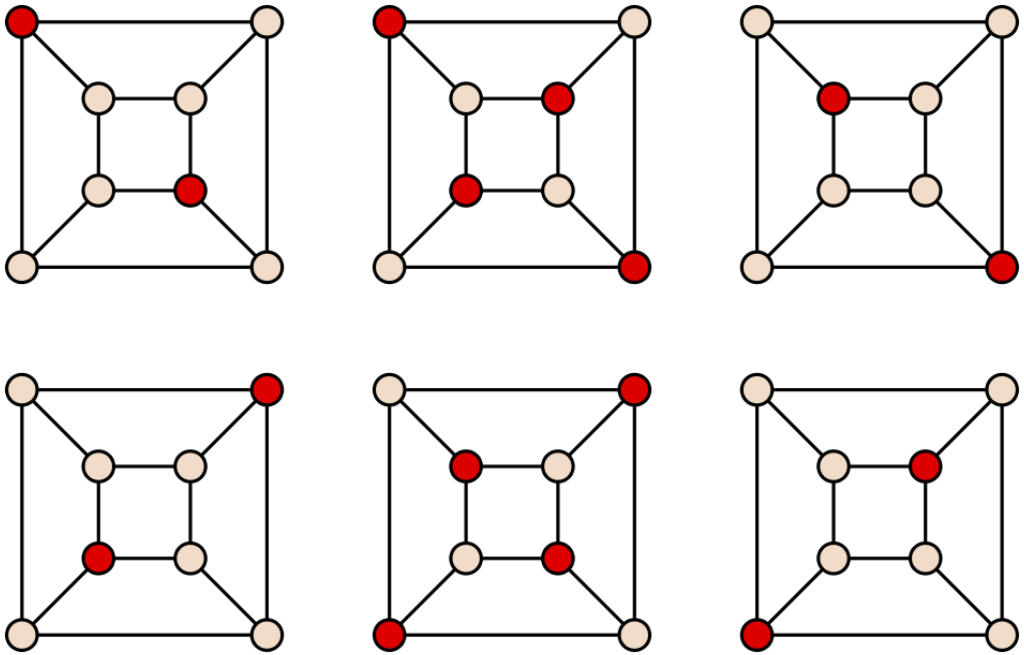
Independent Set: A subset $I \subseteq V$ is said to be *independent* provided for each $v \in I$ none of its neighbors in G belongs to I.

Maximal Independent Set: An independent set of G is *maximal* if it is not properly contained in any other independent set in G.

Maximum Independent Set: A maximal independent set of the largest size.

Finding a maximum independent set is NP-hard.

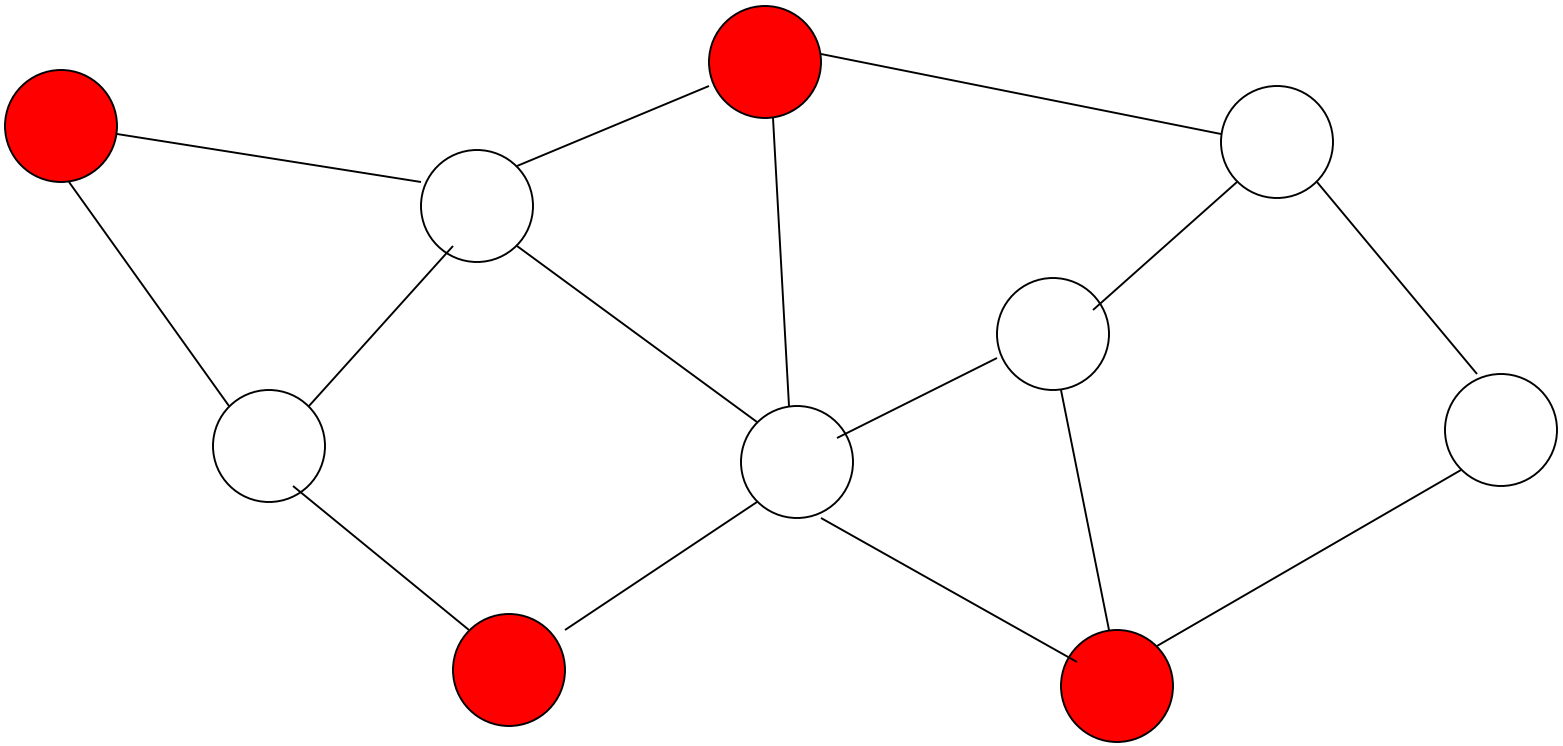
But finding a maximal independent set is trivial in the sequential setting.



Maximal Independent Sets (red vertices) of the Cube Graph
Source: Wikipedia

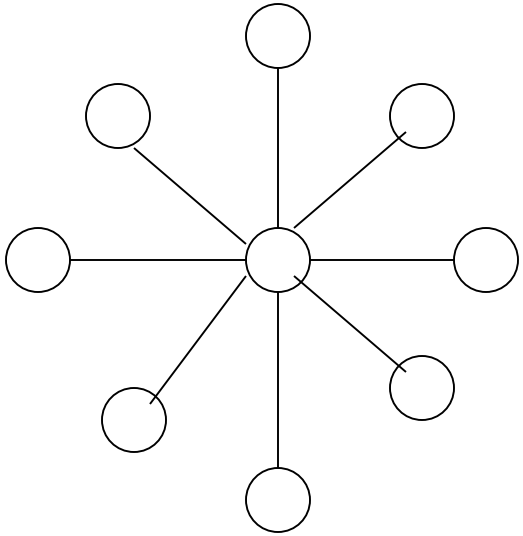
Maximal Independent Set (MIS):

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

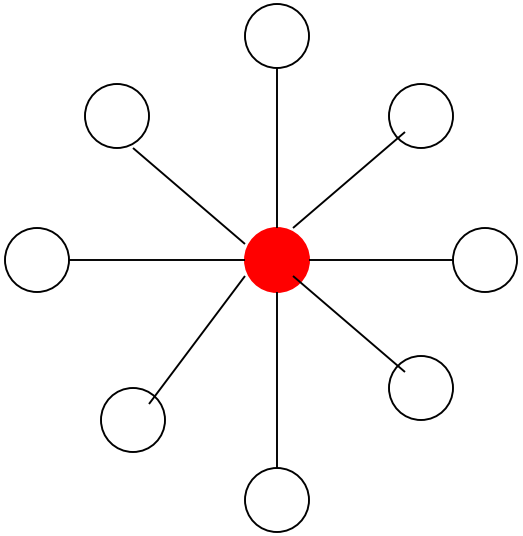


Size of Maximal Independent Sets

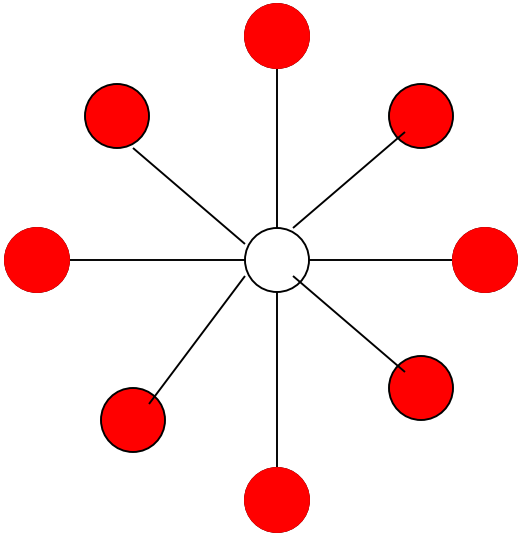
A graph G…



…a MIS of minimum size…



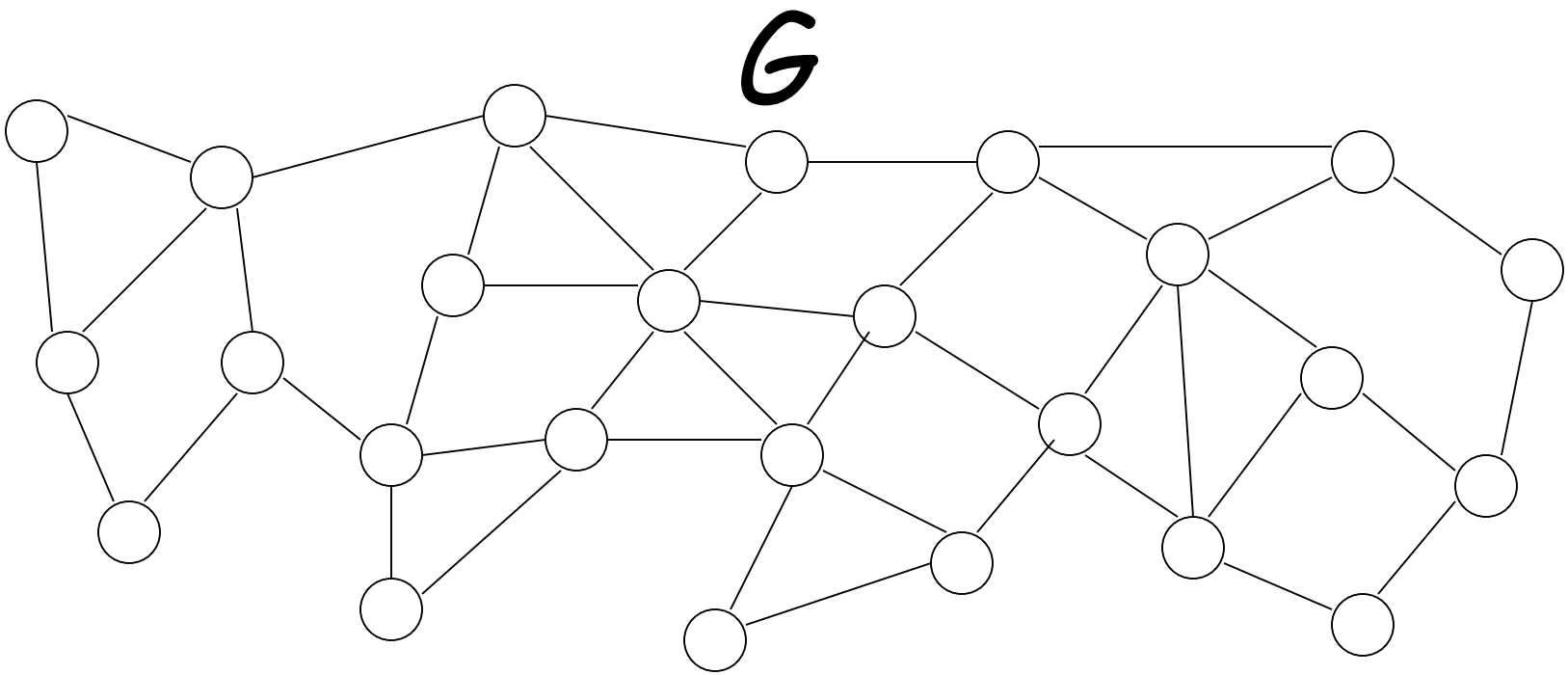
…a MIS of maximum size (a.k.a. maximum independent set)



A sequential algorithm to find a MIS

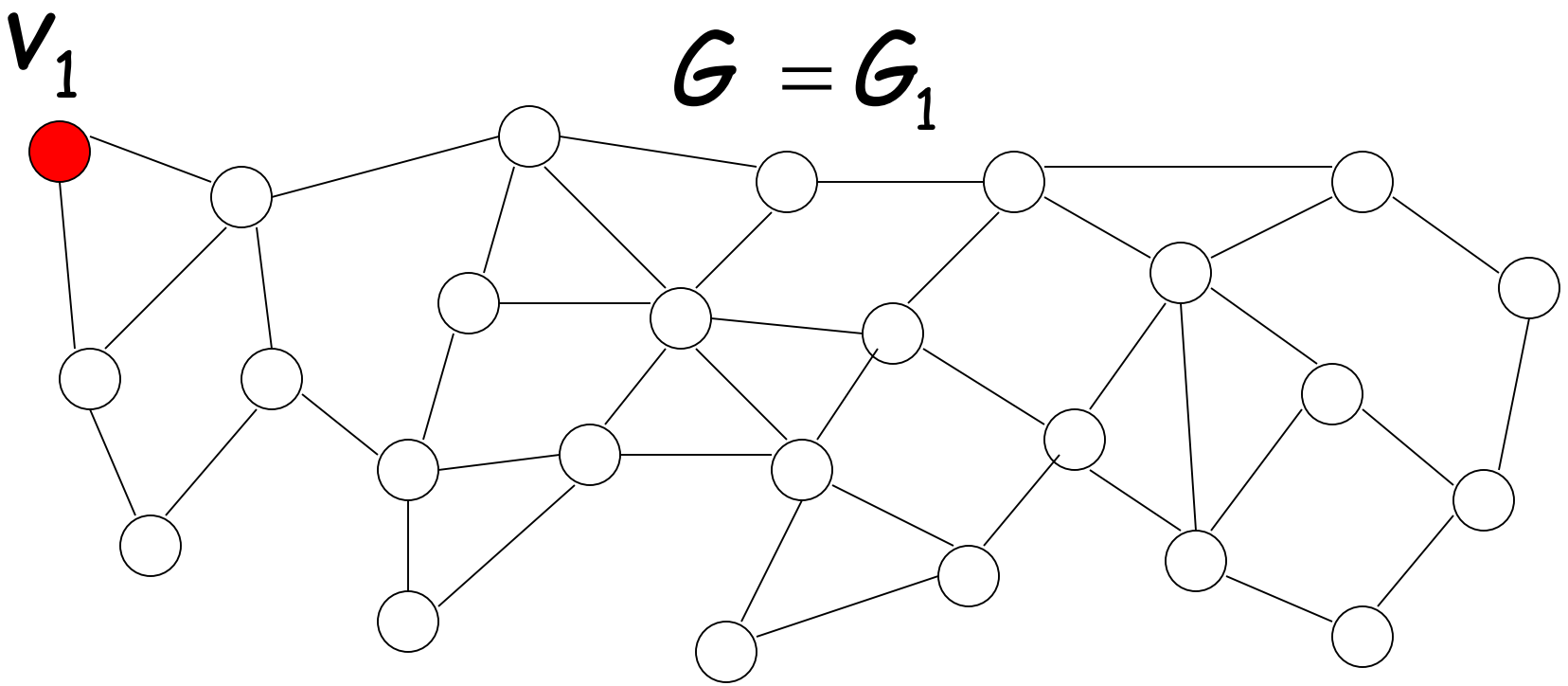
Suppose that I will hold the final MIS, and assume that we are not interested in any respect to the final size of the MIS

Initially $I = \emptyset$

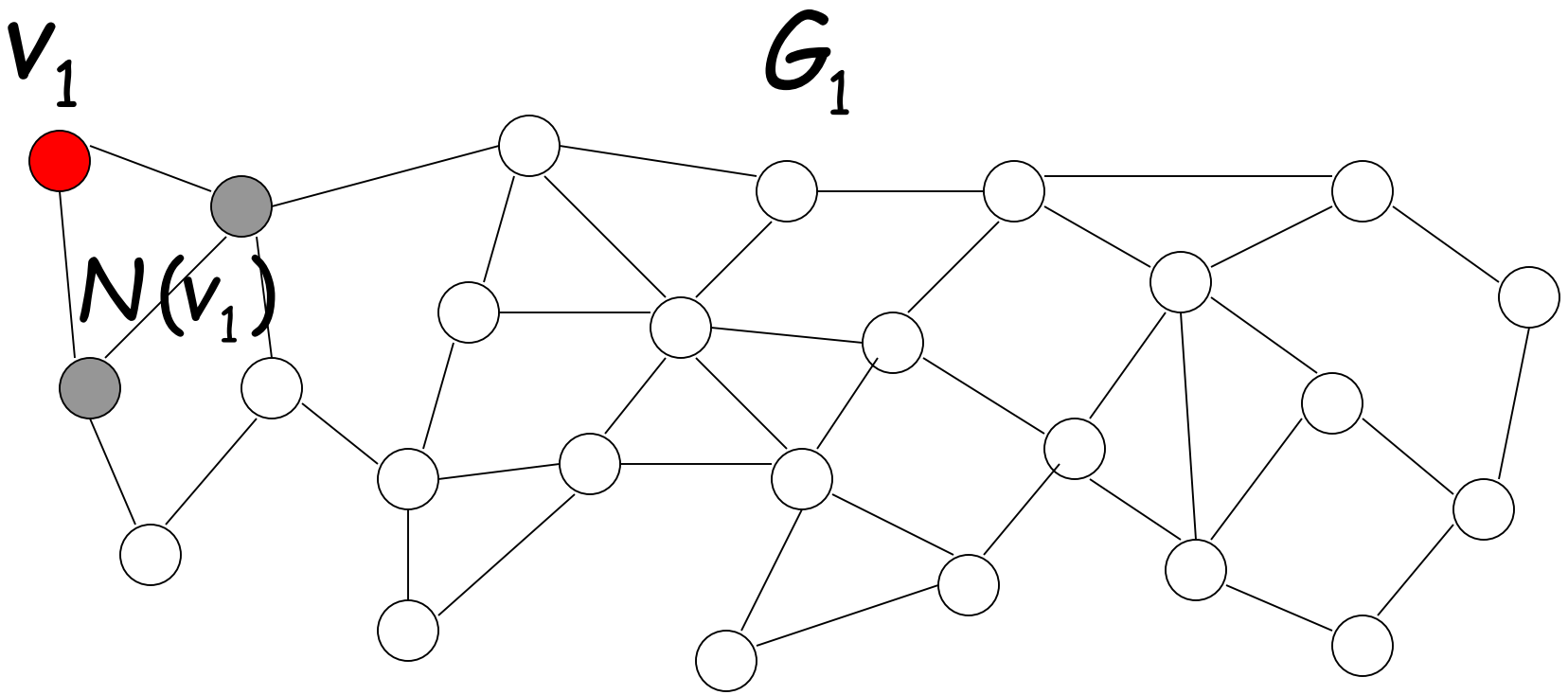


Phase 1:

Pick an arbitrary node v_1 and add it to I

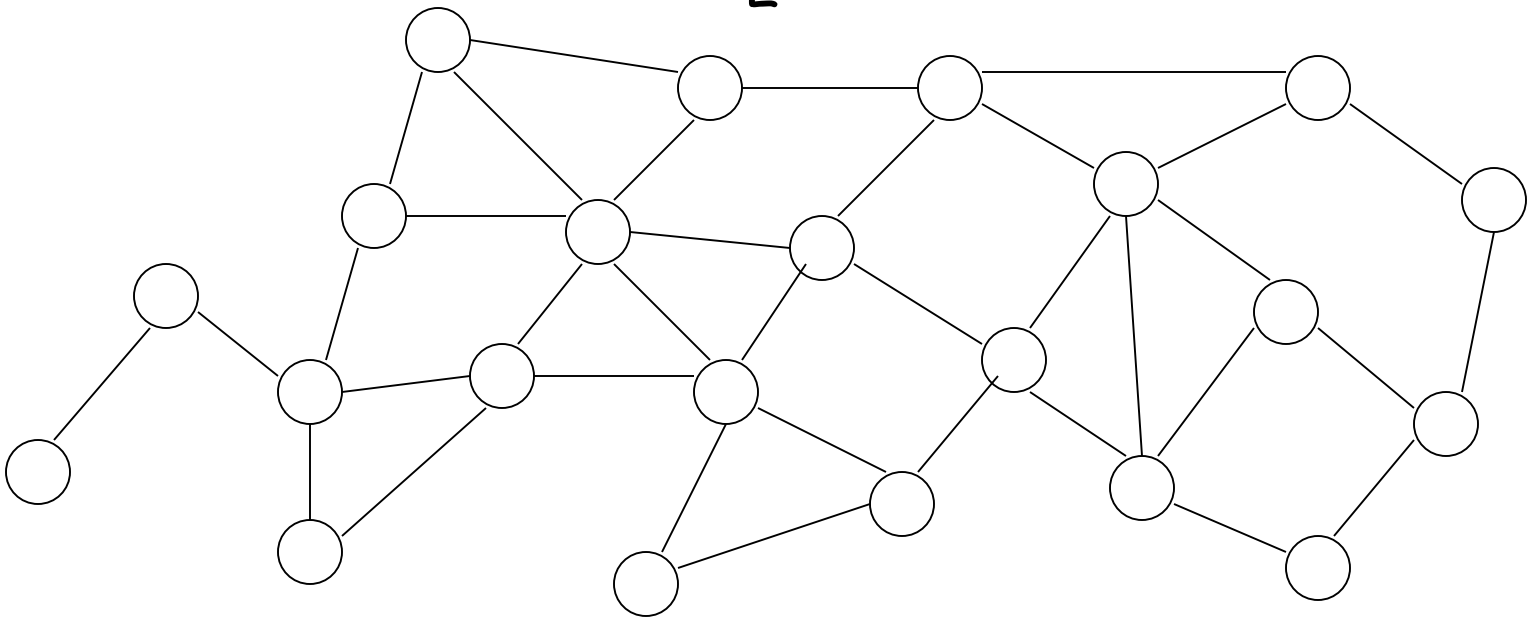


Identify the neighborhood of v_1, say $N(v_1)$



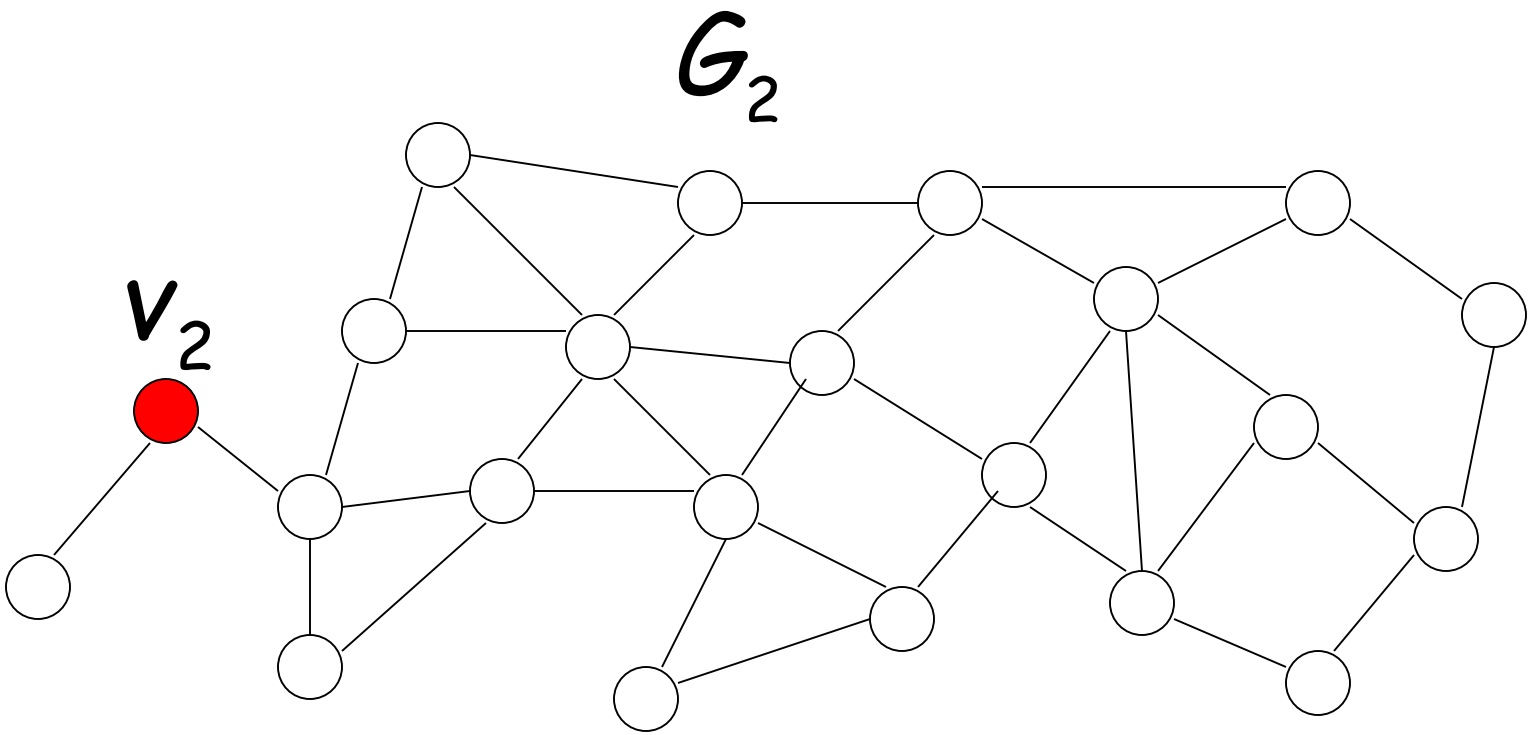
Remove v_1 and $N(v_1)$, and let G_2 be the resulting graph

G_2

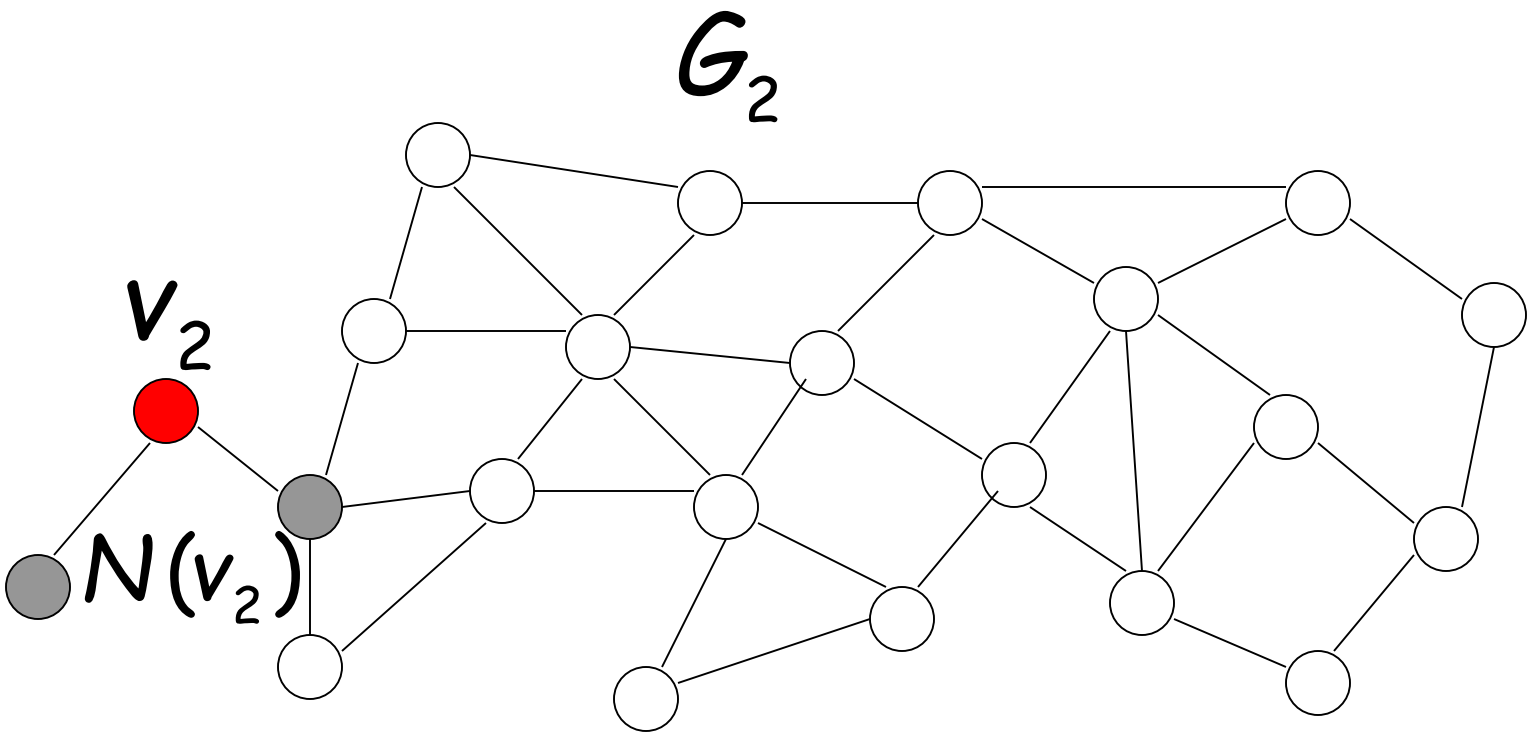


Phase 2:

Pick a node v_2 and add it to I

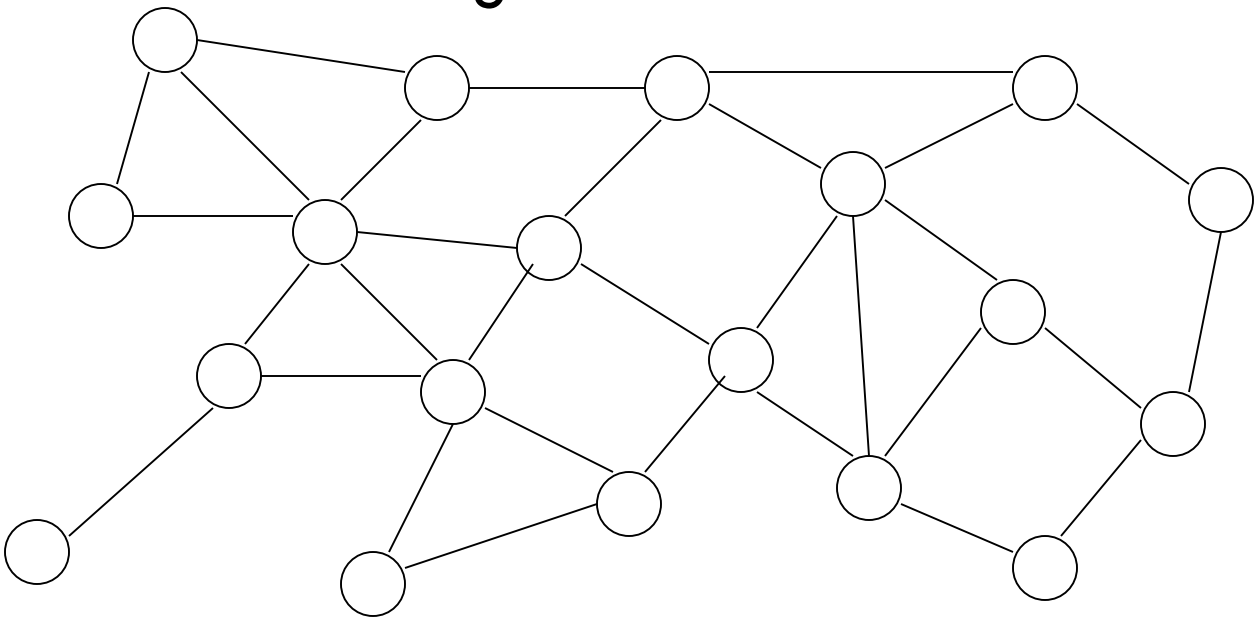


Identify v_2 and neighbors $N(v_2)$



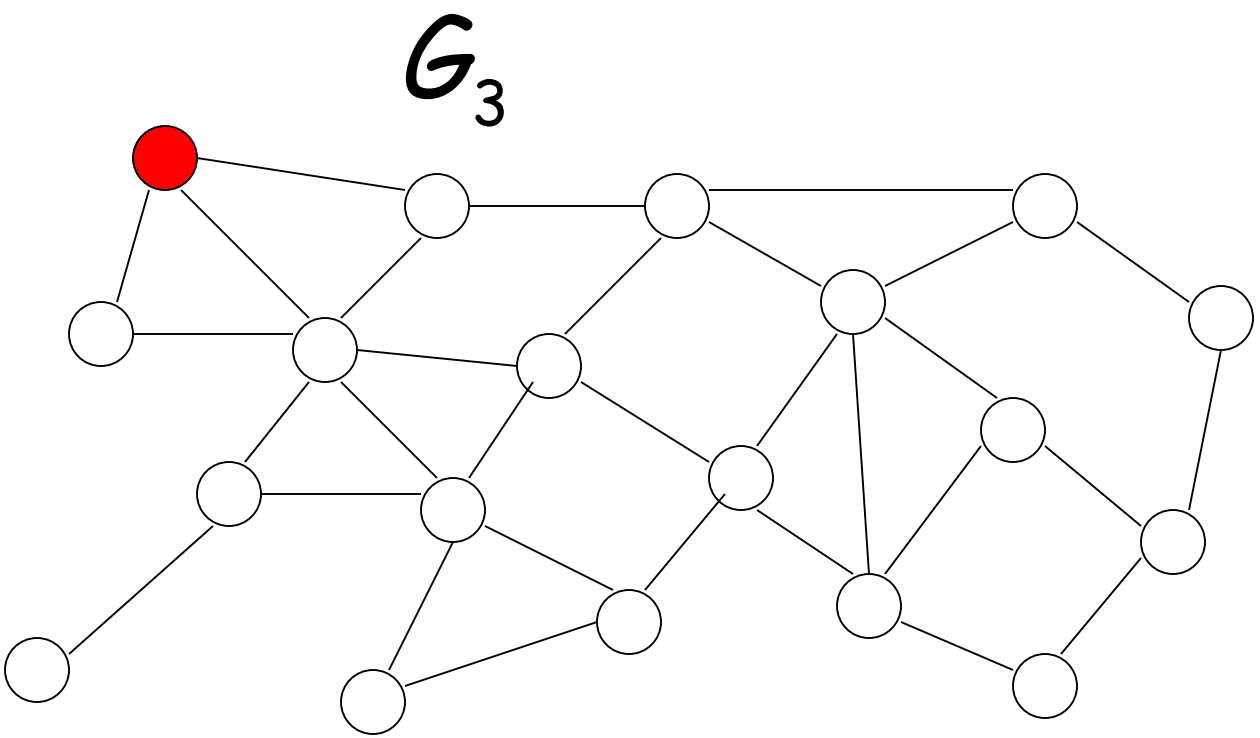
Remove v_2 and $N(v_2)$, and let G_3 be the resulting graph

G_3



Phases 3,4,5,…:

Repeat until all nodes are removed



Phases 3,4,5,…,x:

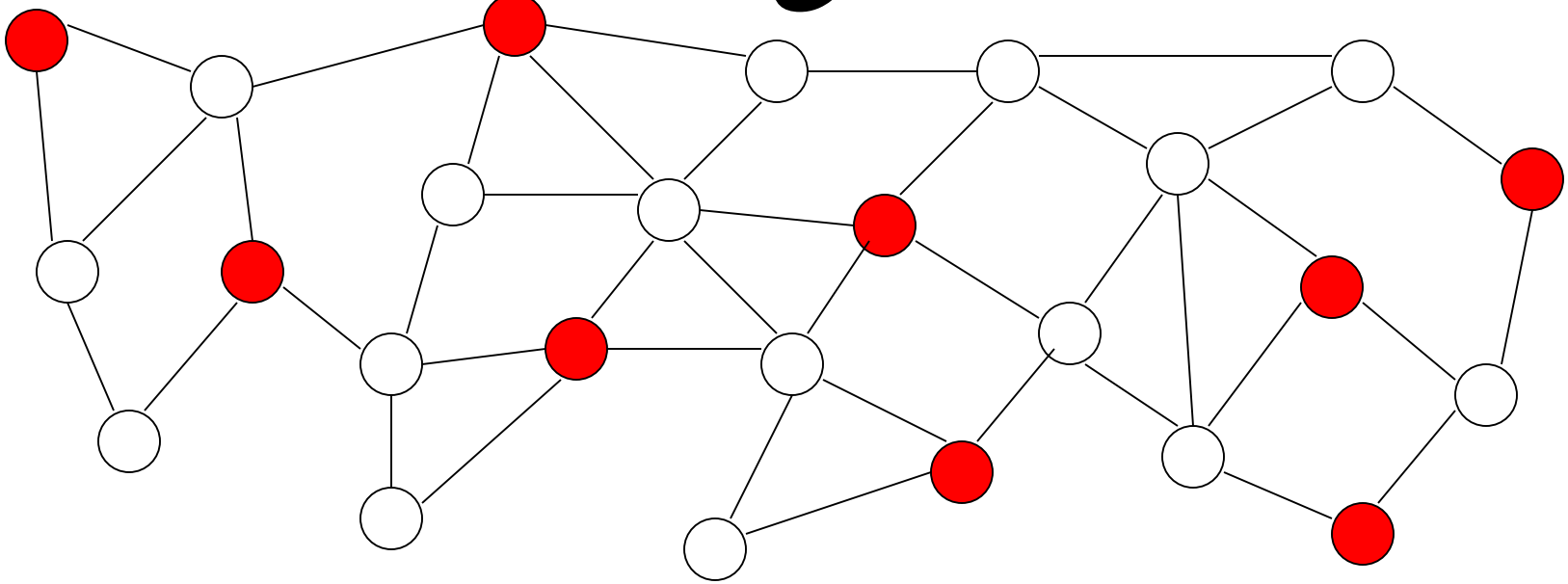
Repeat until all nodes are removed

G_{x+1}

No remaining nodes

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

G



Finding a Maximal Independent Set Sequentially

Input: V is the set of vertices, and E is the set of edges. For each $v \in V$, we denote by $\Gamma(v)$ the set of neighboring vertices of v.

Output: A maximal independent set *MIS* of the input graph.

```
Serial-Greedy-MIS ( V, E )
1. MIS ← φ
2. for v ← 1 to |V| do
3.     if MIS ∩ Γ( v ) = φ then MIS ← MIS ∪ { v }
4. return MIS
```

This algorithm can be easily implemented to run in $\Theta(n + m)$ time, where n is the number of vertices and m is the number of edges in the input graph.

The output of this algorithm is called the *Lexicographically First MIS* (LFMIS).

Finding a Maximal Independent Set Sequentially

Input: V is the set of vertices, and E is the set of edges. For each $v \in V$, we denote by $\Gamma(v)$ the set of neighboring vertices of v.

Output: A maximal independent set *MIS* of the input graph.

```
Serial-Greedy-MIS-2 ( V, E )
1. MIS ← φ
2. while |V| > 0 do
3.     pick an arbitrary vertex v ∈ V
4.     MIS ← MIS ∪ { v }
5.     R ← { v } ∪ Γ( v )
6.     V ← V \ R
7.     E ← E \ { ( v1, v2 ) | v1 ∈ R or v2 ∈ R }
8. return MIS
```

Always choosing the vertex with the smallest id in the current graph will produce exactly the same MIS as in *Serial-Greedy-MIS*.

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! But actually, we can avoid these LE steps, as explained in the following

A Generalized Algorithm For Computing a MIS

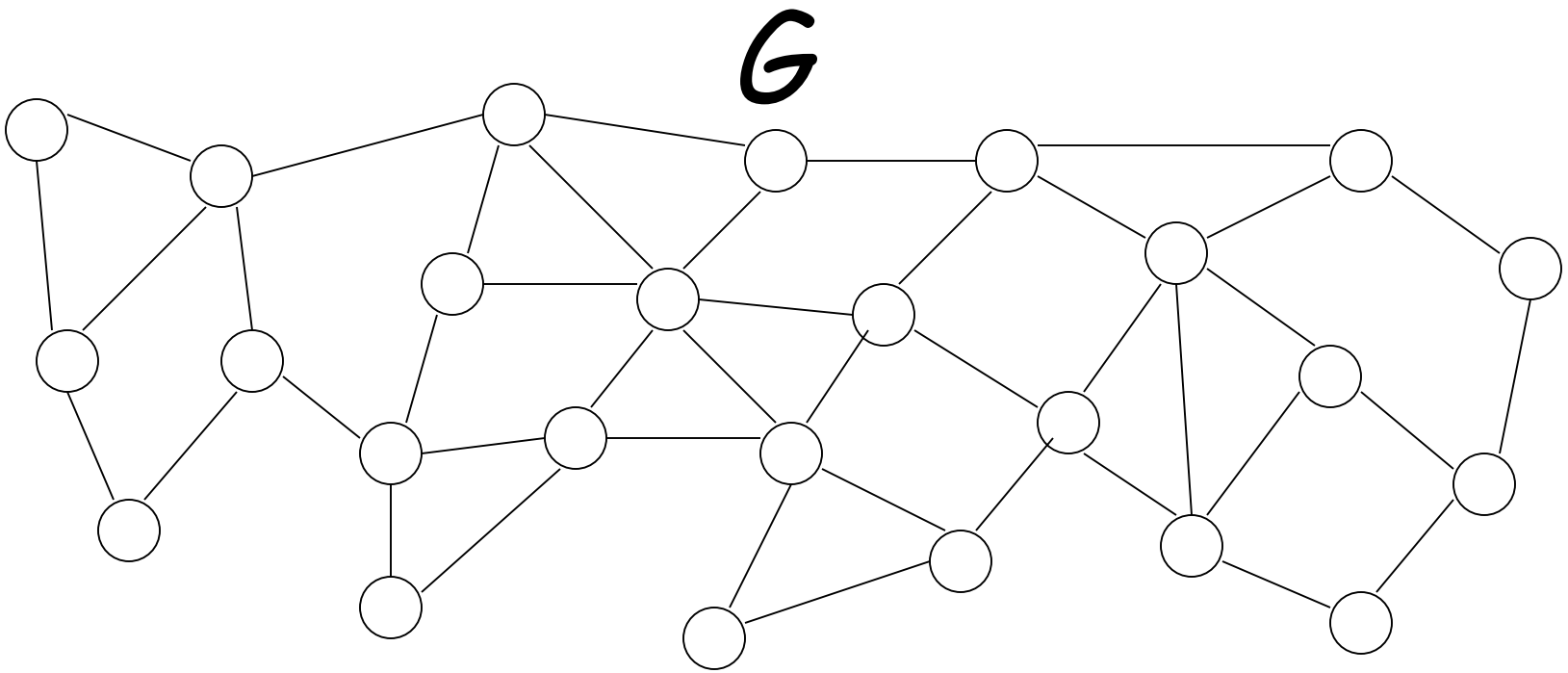
Same as the previous algorithm, but at each phase, instead of a single node, we now select any independent set (this selection should be seen as a black box at this stage, i.e., we do not know/specify how such independent set is selected)

$\Rightarrow$ 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 $I = \emptyset$

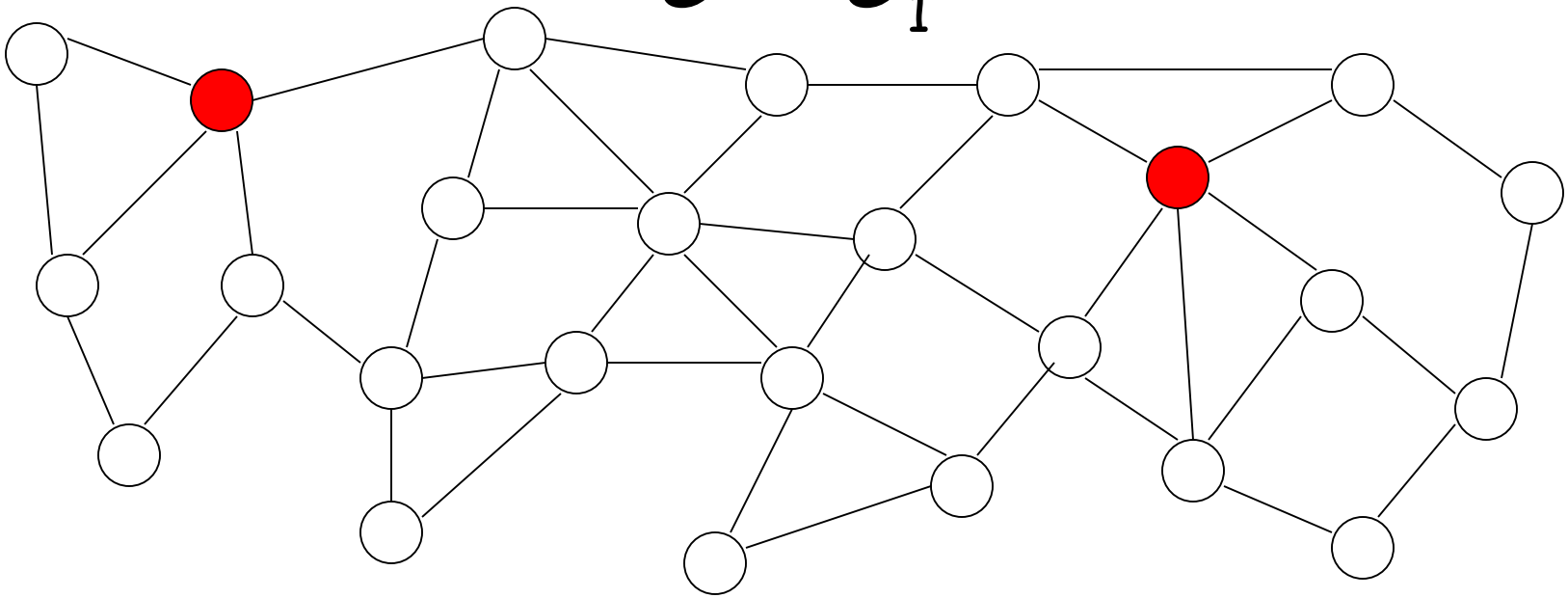


Phase 1:

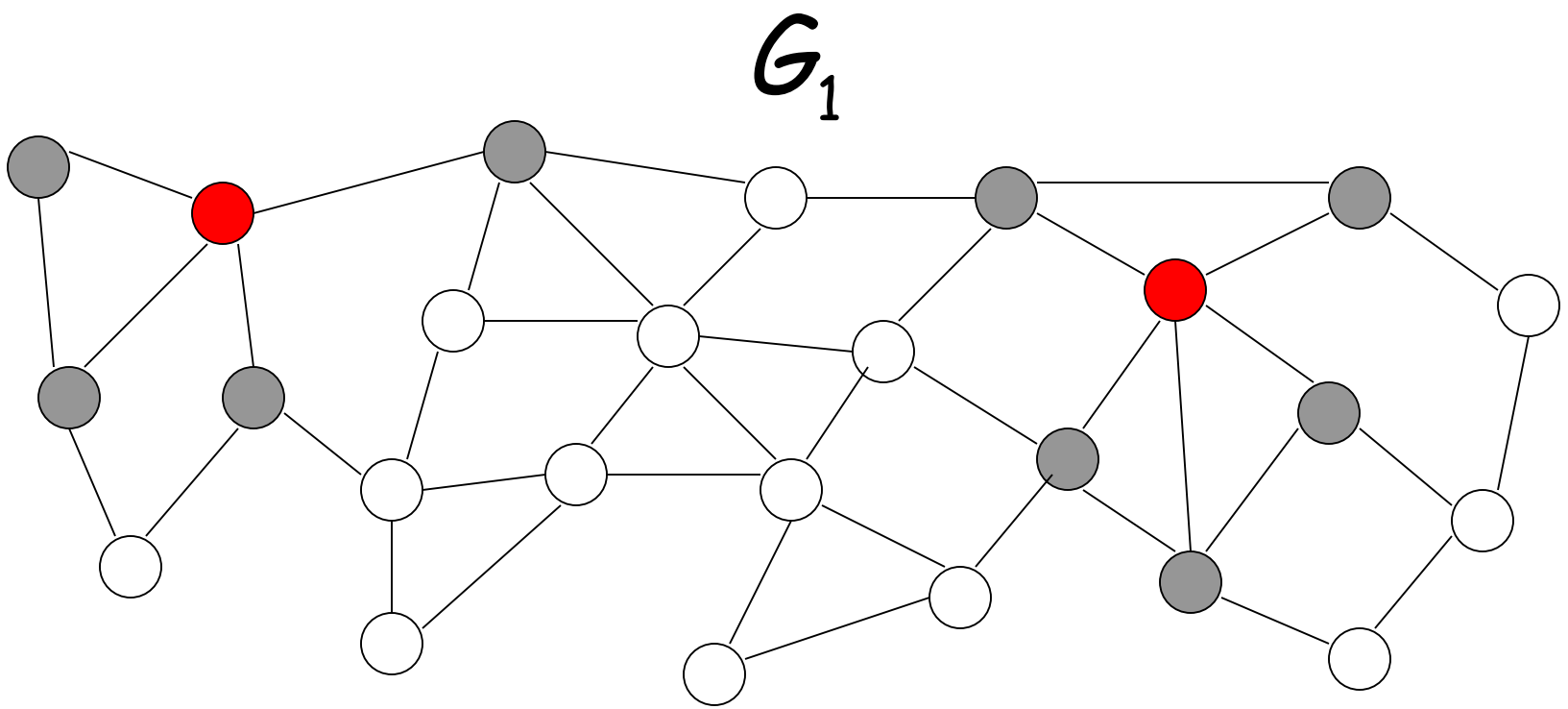
Find any independent set I_1

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

$G = G_1$

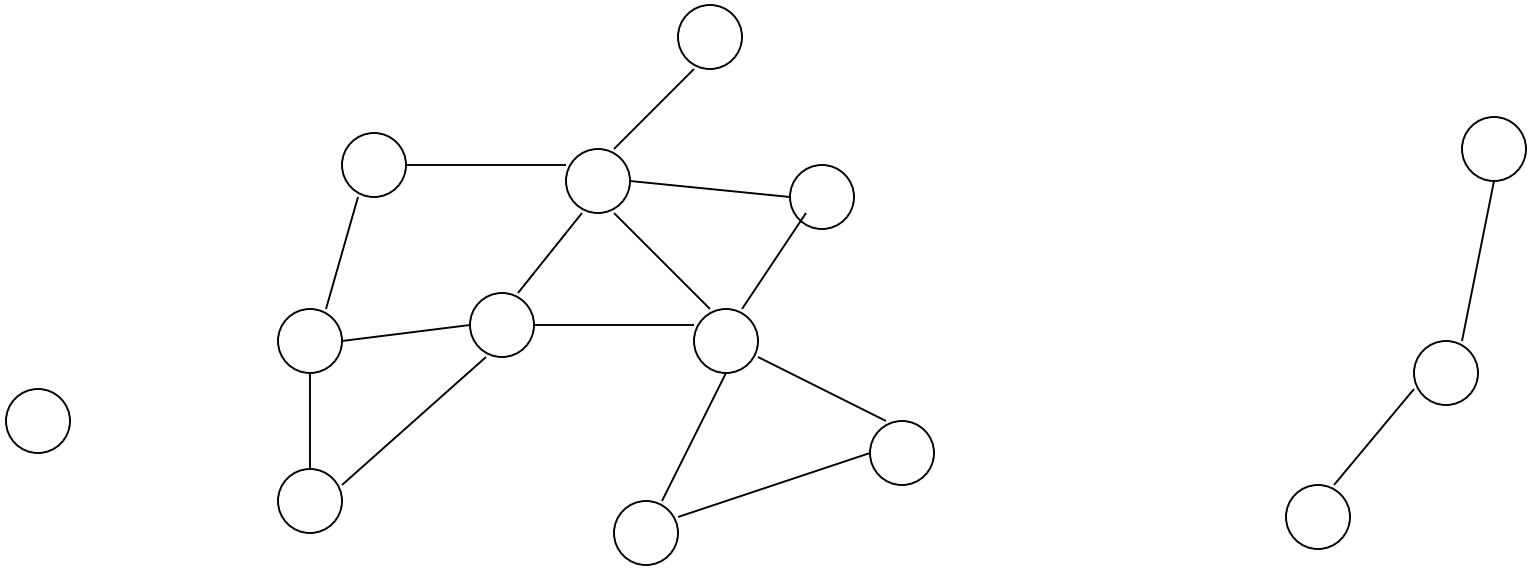


Identify the neighborhood of I_1, say $N(I_1)$



Remove I_1 and $N(I_1)$, and let G_2 be the resulting graph

G_2



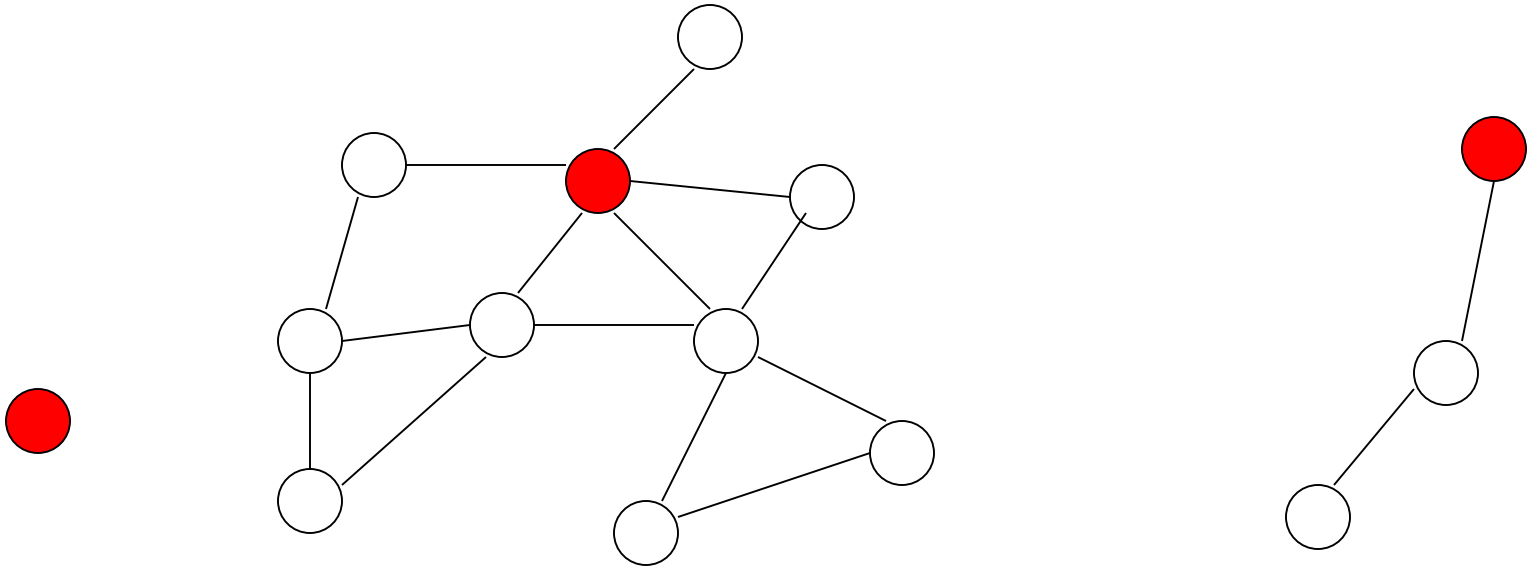
Phase 2:

On new graph

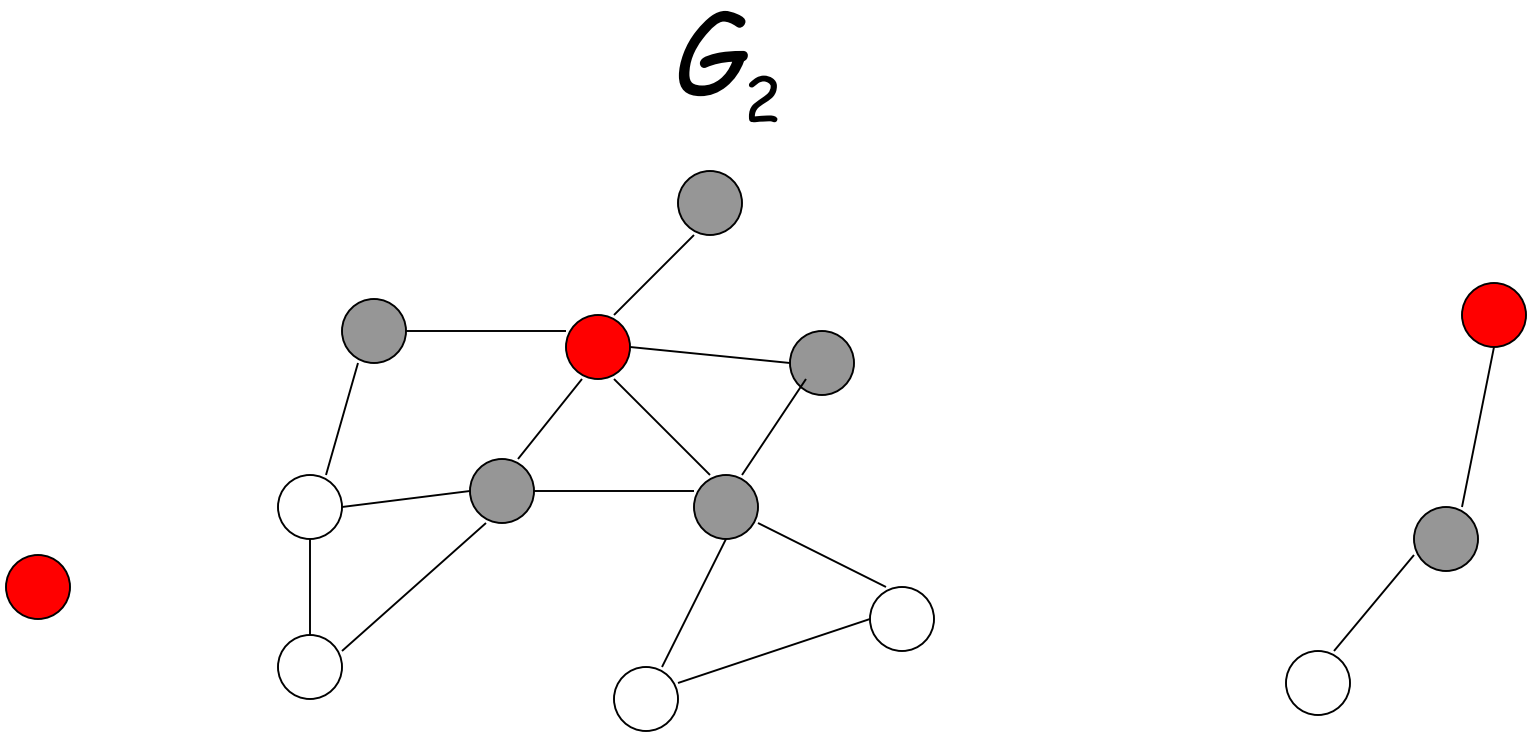
Find any independent set I_2

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

G_2

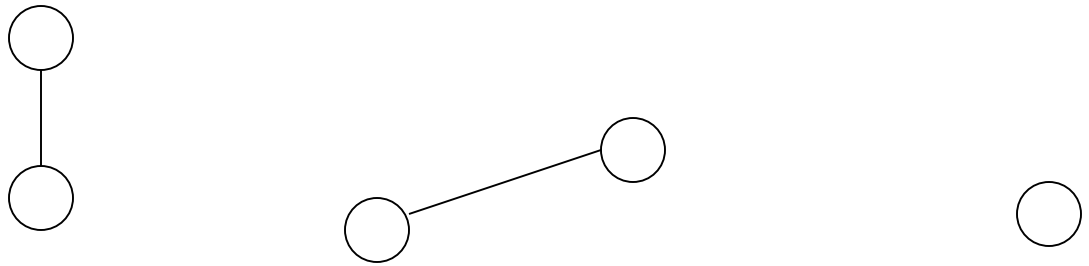


Identify the neighborhood of I_2, say $N(I_2)$



Remove I_2 and $N(I_2)$, and let G_3 be the resulting graph

G_3



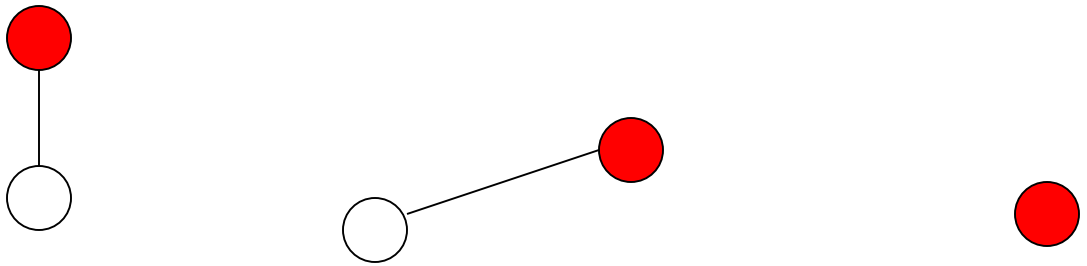
Phase 3:

On new graph

Find any independent set I_3

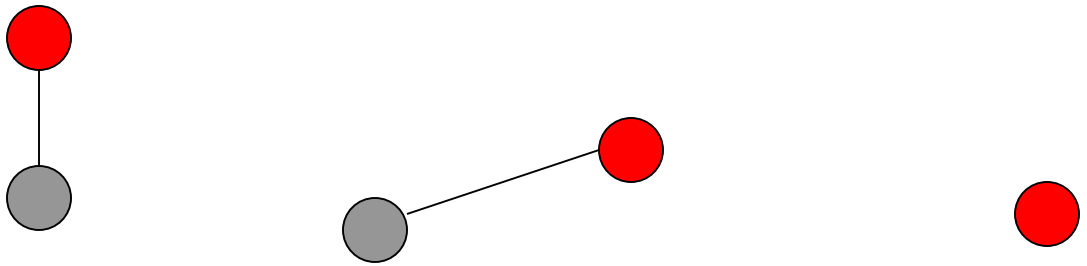
and add I_3 to I: $I \leftarrow I \cup I_3$

G_3



Identify the neighborhood of I_3, say $N(I_3)$

G_3



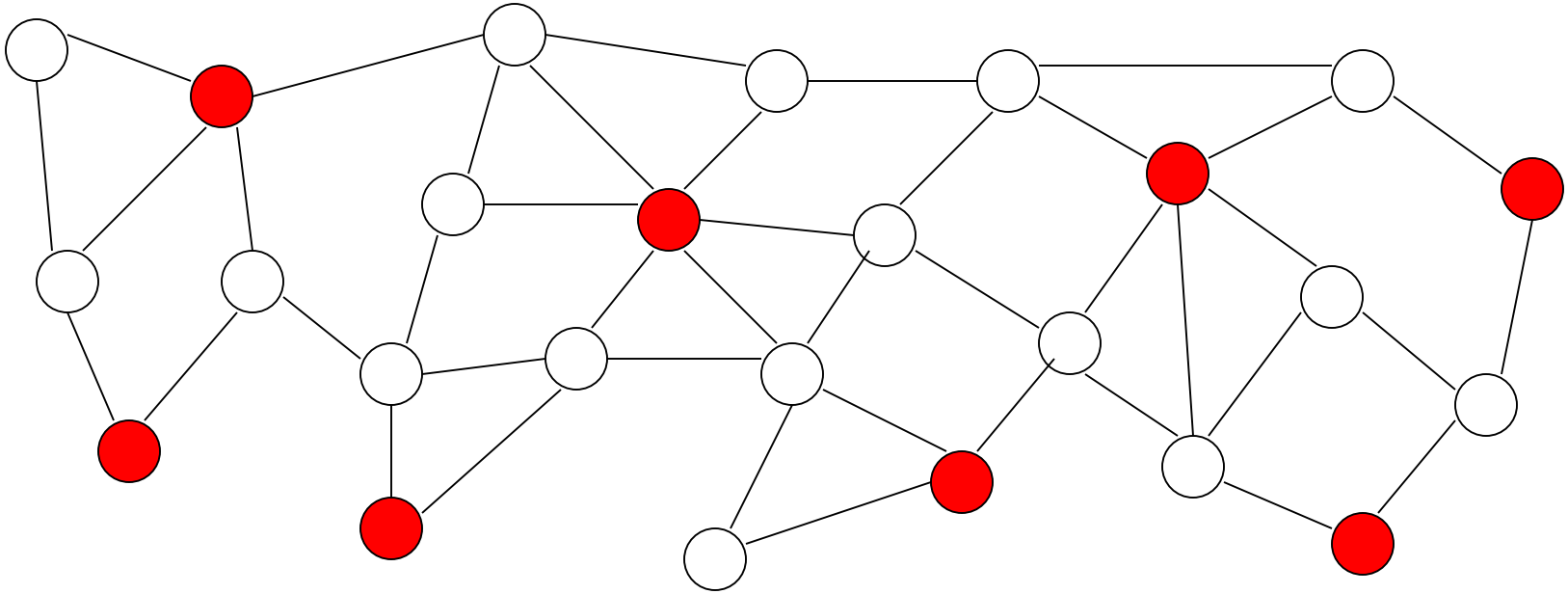
Remove I_3 and $N(I_3)$, and let G_4 be the resulting graph

G_4

No nodes are left, termination

Final MIS I

G



A Randomized Synchronous 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

Finding a Maximal Independent Set Sequentially

Input: V is the set of vertices, and E is the set of edges. For each $S \subseteq V$, we denote by $\Gamma(S)$ the set of neighboring vertices of S.

Output: A maximal independent set *MIS* of the input graph.

Serial-Greedy-MIS-3 (V, E)

1. $MIS \leftarrow \phi$
2. *while* $|V| > 0$ *do*
3. find an independent set $S \subseteq V$
4. $MIS \leftarrow MIS \cup S$
5. $R \leftarrow S \cup \Gamma(S)$
6. $V \leftarrow V \setminus R$
7. $E \leftarrow E \setminus \{ (v_1, v_2) \mid v_1 \in R \text{ or } v_2 \in R \}$
8. *return* *MIS*

Parallelizing Serial-Greedy-MIS-3

- Number of iterations can be kept small by finding in each iteration an S with large $S \cup \Gamma(S)$. But this is difficult to do.
- Instead in each iteration we choose an S such that a large fraction of current edges are incident on $S \cup \Gamma(S)$.
- To select S we start with a random $S' \subseteq V$.
  - By choosing lower degree vertices with higher probability we are likely to have very few edges with both end-points in S'.
  - We check each edge with both end-points in S', and drop the end-point with lower degree from S'. Our intention is to keep $\Gamma(S')$ as large as we can.
  - After removing all edges as above we are left with an independent set. This is our S.

Serial-Greedy-MIS-3 (V, E)

1. $MIS \leftarrow \phi$
2. *while* $|V| > 0$ *do*
3. find an independent set $S \subseteq V$
4. $MIS \leftarrow MIS \cup S$
5. $R \leftarrow S \cup \Gamma(S)$
6. $V \leftarrow V \setminus R$
7. $E \leftarrow E \setminus \{ (v_1, v_2) \mid v_1 \in R \text{ or } v_2 \in R \}$
8. *return* MIS

Randomized Maximal Independent Set (MIS)

Input: n is the number of vertices, and for each vertex $u \in [\,1, n\,]$, $V[\,u\,]$ is set to u. E is the set of edges sorted in non-decreasing order of the first vertex. For every edge $(\,u, v\,)$ both $(\,u, v\,)$ and $(\,v, u\,)$ are included in E.

Output: For all $u \in [\,1, n\,]$, $MIS[\,u\,]$ is set to 1 if vertex u is in the MIS.

$d[\,u\,]$ (i.e., degree of vertex u) can now be computed easily by subtracting $c[\,u-1\,]$ from $c[\,u\,]$

if both end-points of an edge is marked, unmark the one with the lower degree

remove marked vertices along with their neighbors as well as the corresponding edges

```
Par-Randomized-MIS ( n, V, E, MIS )
1.  while |V| > 0 do
2.      array d[ 1 : |V| ], c[ 1 : |V| ] = { 0 }, M[ 1 : |V| ] = { 0 }
3.      parallel for i ← 1 to |E| do
4.          if i = |E| then k ← n else k ← E[ i + 1 ].u − 1
5.          parallel for j ← E[ i ].u to k do c[ j ] ← i
6.      parallel for u ← 1 to |V| do
7.          if u = 1 then d[ u ] ← c[ u ] else d[ u ] ← c[ u ] − c[ u − 1 ]
8.          if d[ u ] = 0 then M[ u ] ← 1
9.          else M[ u ] ← 1 ( with probability 1 / ( 2d[ u ] ) )
10.     parallel for each ( u, v ) ∈ E do
11.         if M[ u ] = 1 and M[ v ] = 1 then
12.             if d[ u ] ≤ d[ v ] then M[ u ] ← 0 else M[ v ] ← 0
13.     parallel for u ← 1 to |V| do
14.         if M[ u ] = 1 then MIS[ V[ u ] ] ← 1
15.     ( V, E ) ← Par-Compress ( V, E, M )
```

for each u find the edge with the largest index i such that $E[\,i\,].u \le u$, and store that i in $c[\,u\,]$

mark lower-degree vertices with higher probability

add all marked vertices to MIS

Removing Marked Vertices and Their Neighbors

Input: Arrays V and E, and bit array $M[\,1: |V|\,]$. Each entry of E is of the form $(\,u, v\,)$, where $1 \le u, v \le |V|$. If for some u, $M[\,u\,] = 1$, then u and all v such that $(\,u, v\,) \in E$ must be removed from V along with all edges $(\,u, v\,)$ from E.

Output: Updated V and E.

marked vertices will be removed

find new indices for surviving vertices & edges

move surviving edges to the smaller array F

```
Par-Compress ( V, E, M )
1.  array S_V[ 1 : |V| ] = { 1 }, S'_V[ 1 : |V| ], S_E[ 1 : |E| ] = { 1 }, S'_E[ 1 : |E| ]
2.  parallel for u ← 1 to |V| do
3.      if M[ u ] = 1 then S_V[ u ] ← 0
4.  parallel for i ← 1 to |E| do
5.      u ← E[ i ].u, v ← E[ i ].v
6.      if M[ u ] = 1 or M[ v ] = 1 then S_V[ u ] ← 0, S_V[ v ] ← 0, S_E[ i ] ← 0
7.  S'_V ← Par-Prefix-Sum ( S_V, + ), S'_E ← Par-Prefix-Sum ( S_E, + )
8.  array U[ 1 : S'_V[ |V| ] ], F[ 1 : S'_E[ |E| ] ]
9.  parallel for u ← 1 to |V| do
10.     if S_V[ u ] = 1 then U[ S'_V[ u ] ] ← V[ u ]
11. parallel for i ← 1 to |E| do
12.     if S_E[ i ] = 1 then F[ S'_E[ i ] ] ← E[ i ]
13. parallel for i ← 1 to |F| do
14.     u ← F[ i ].u, v ← F[ i ].v
15.     F[ i ].u ← S'_V[ u ], F[ i ].v ← S'_V[ v ]
16. return ( U, F )
```

initialize

neighbors of marked vertices & corresponding edges must go

move surviving vertices to the smaller array U

update the end-points of the surviving edges to new vertex indices