

Project #3 - PAD

Connected Components in Hadoop

Federico Conte

fconte90@gmail.com

<https://github.com/Draxent>

Università degli studi di Pisa & Scuola Superiore Sant'Anna
Department of Computer Science and Information Engineering
Master in Computer Science and Networking

26th January 2016

Given a graph, the algorithm identifies the **connected components** (Clusters). We have tried to implement the “**The Alternating Algorithm**” proposed in the paper **Connected Components in MapReduce and Beyond**.

Algorithm 1 The Alternating Algorithm

INPUT: Edges (u, v) as a set of key-value pairs $\langle u; v \rangle$.

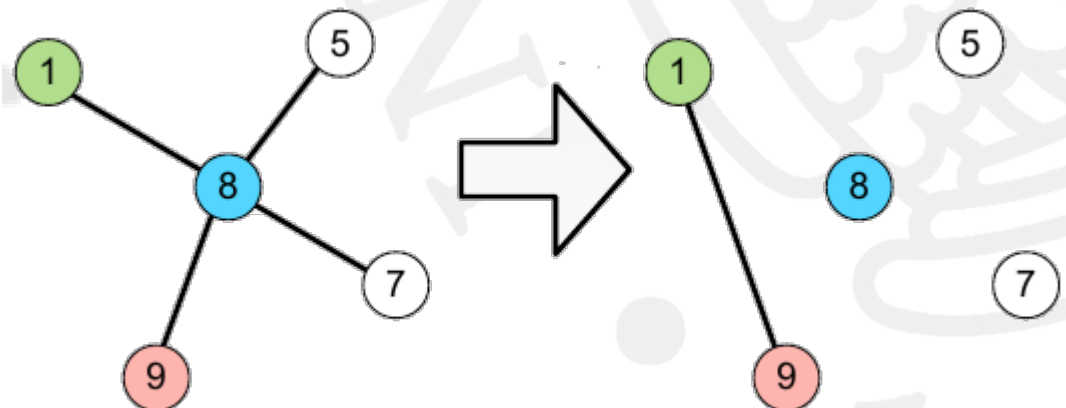
INPUT: A unique label ℓ_v for every node $v \in V$.

- 1: repeat
- 2: **Large-Star**
- 3: Small-Star
- 4: until Convergence

Algorithm 2 The Large-Star operation

- 1: procedure MAP($u; v$)
- 2: Emit $\langle u; v \rangle$
- 3: Emit $\langle v; u \rangle$
- 4: end procedure
- 5: procedure REDUCE($u; \Gamma(u)$)
- 6: $m \leftarrow \arg \min_{v \in \Gamma^+(u)} \ell_v$
- 7: Emit $\langle v; m \rangle \forall v \text{ where } \ell_v > \ell_u$
- 8: end procedure

Large-Star(v): Connect all strictly **larger** neighbours to the **min** neighbour including **self**.



Given a graph, the algorithm identifies the **connected components** (Clusters).
 We have tried to implement the “*The Alternating Algorithm*” proposed in the paper *Connected Components in MapReduce and Beyond*.

Algorithm 1 The Alternating Algorithm

INPUT: Edges (u, v) as a set of key-value pairs $\langle u; v \rangle$.

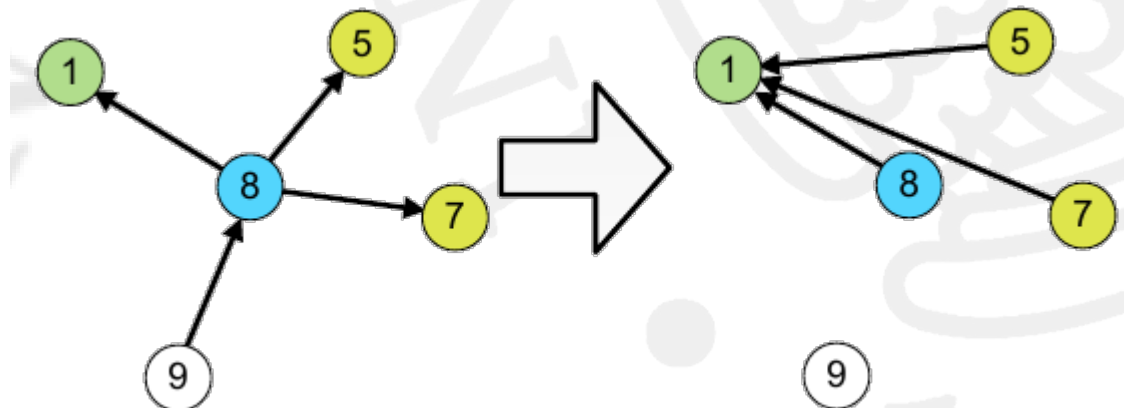
INPUT: A unique label ℓ_v for every node $v \in V$.

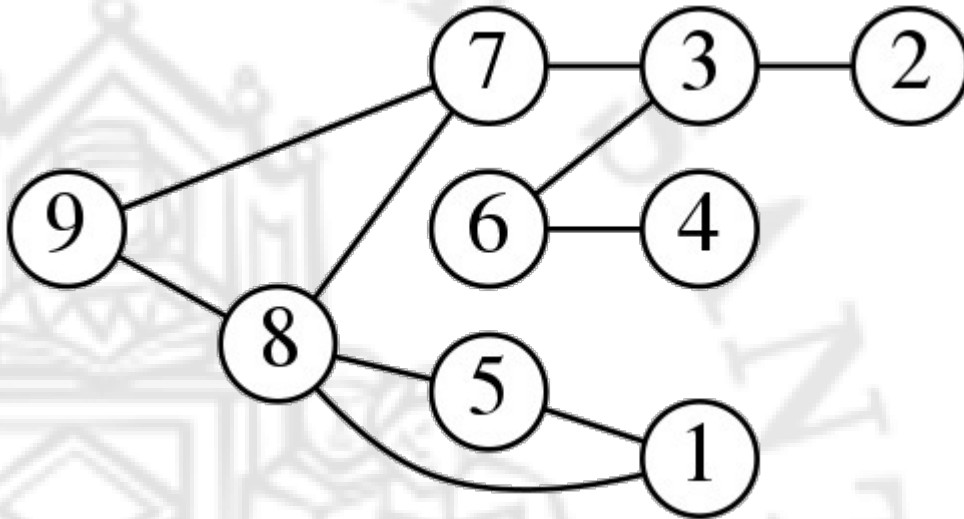
- 1: repeat
- 2: Large-Star
- 3: Small-Star

Algorithm 3 The Small-Star operation

- 1: procedure MAP($u; v$)
 - 2: if $\ell_v \leq \ell_u$ then
 - 3: Emit $\langle u; v \rangle$
 - 4: else
 - 5: Emit $\langle v; u \rangle$
 - 6: end if
 - 7: end procedure
 - 8: procedure REDUCE($u; N \subseteq \Gamma(u)$)
 - 9: $m \leftarrow \arg \min_{v \in N \cup \{u\}} \ell_v$
 - 10: Emit $\langle v; m \rangle \forall v \in N$
 - 11: end procedure
-

Small-Star(v): Connect all **smaller** neighbours and **self** to the **min** neighbour.





Clique List:

1 5 8
8 9 7
7 3
3 2
3 6
6 4

Adjacency List:

1	5,8
2	3
3	2,6,7
4	6
5	1,8
6	3,4
7	3,8,9
8	1,5,7,9
9	7,8

Algorithm 4 Our implementation of the Alternating Algorithm

INPUT: $G = (V, E)$ represented as an adjacency list

INPUT: A unique and proper coloring of G

1: Initialization_Phase

2: repeat

3: Large-Star

4: Small-Star

5: until Convergence

6: Termination_Phase

7: Check_Phase

Clique List:

1	5	8
8	9	7
7	3	
3	2	
3	6	
6	4	

Adjacency List:

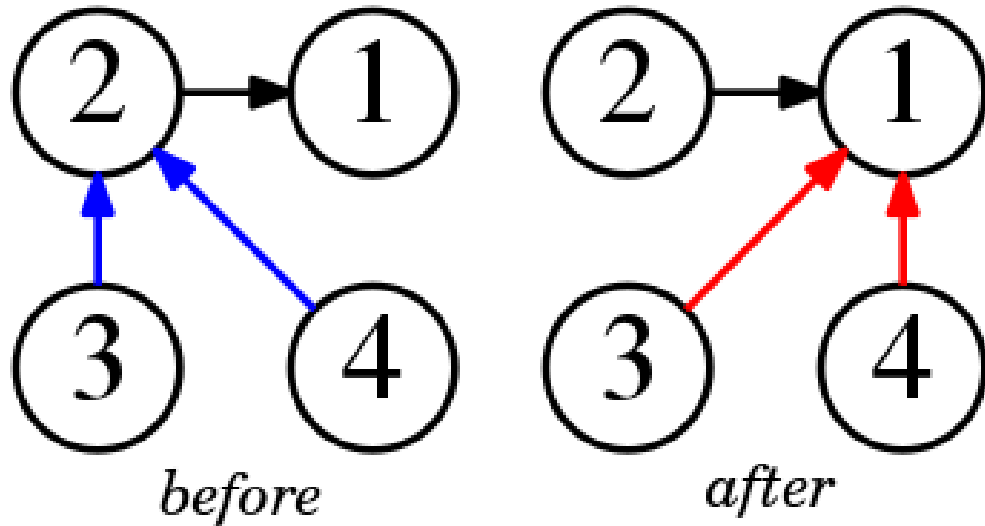
1	5,8
2	3
3	2,6,7
4	6
5	1,8
6	3,4
7	3,8,9
8	1,5,7,9
9	7,8

Edge List:

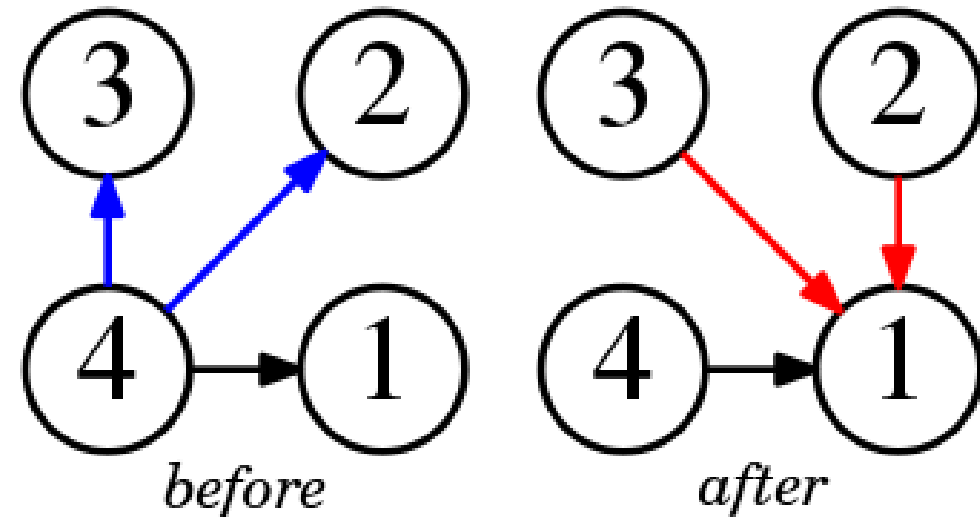
3	2
5	1
6	3
6	4
7	3
8	1
8	5
8	7
9	7
9	8

Convergence

LARGE-STAR



SMALL-STAR



5: until Convergence

6: Termination_Phase

7: Check_Phase

If $min \neq currentNode$ **then**
 $\#ModifiedEdges \leftarrow \#EmittedEdges$

LargeStar.#Changes + SmallStar.#Changes = 0

Algorithm 4 Our implementation of the Alternating Algorithm

INPUT: $G = (V, E)$ represented with an adjacency/cluster list.

INPUT: A unique and positive label ℓ_v for every node $v \in V$.

1: Initialization_Phase

2: **repeat**

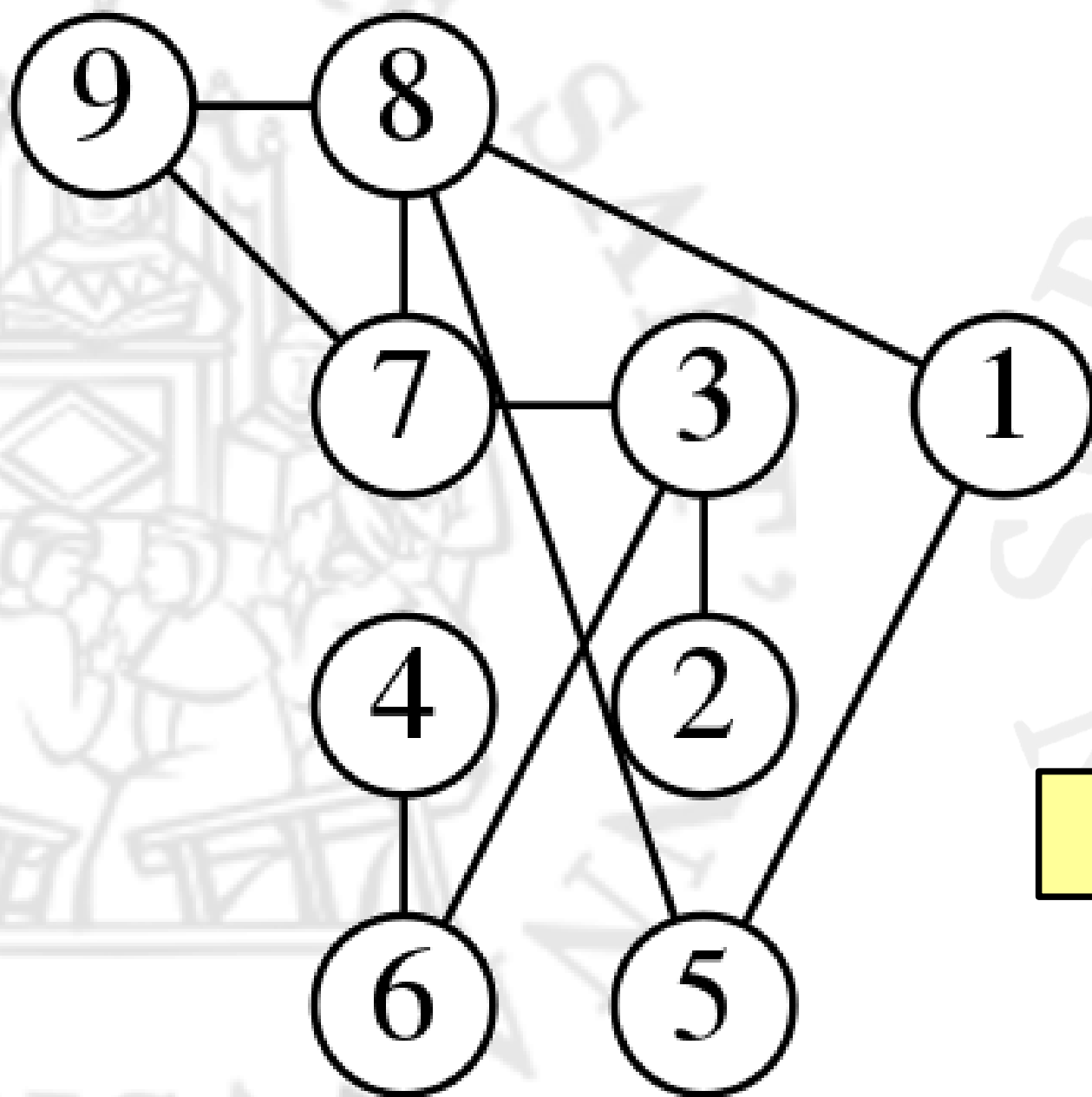
3: Large-Star

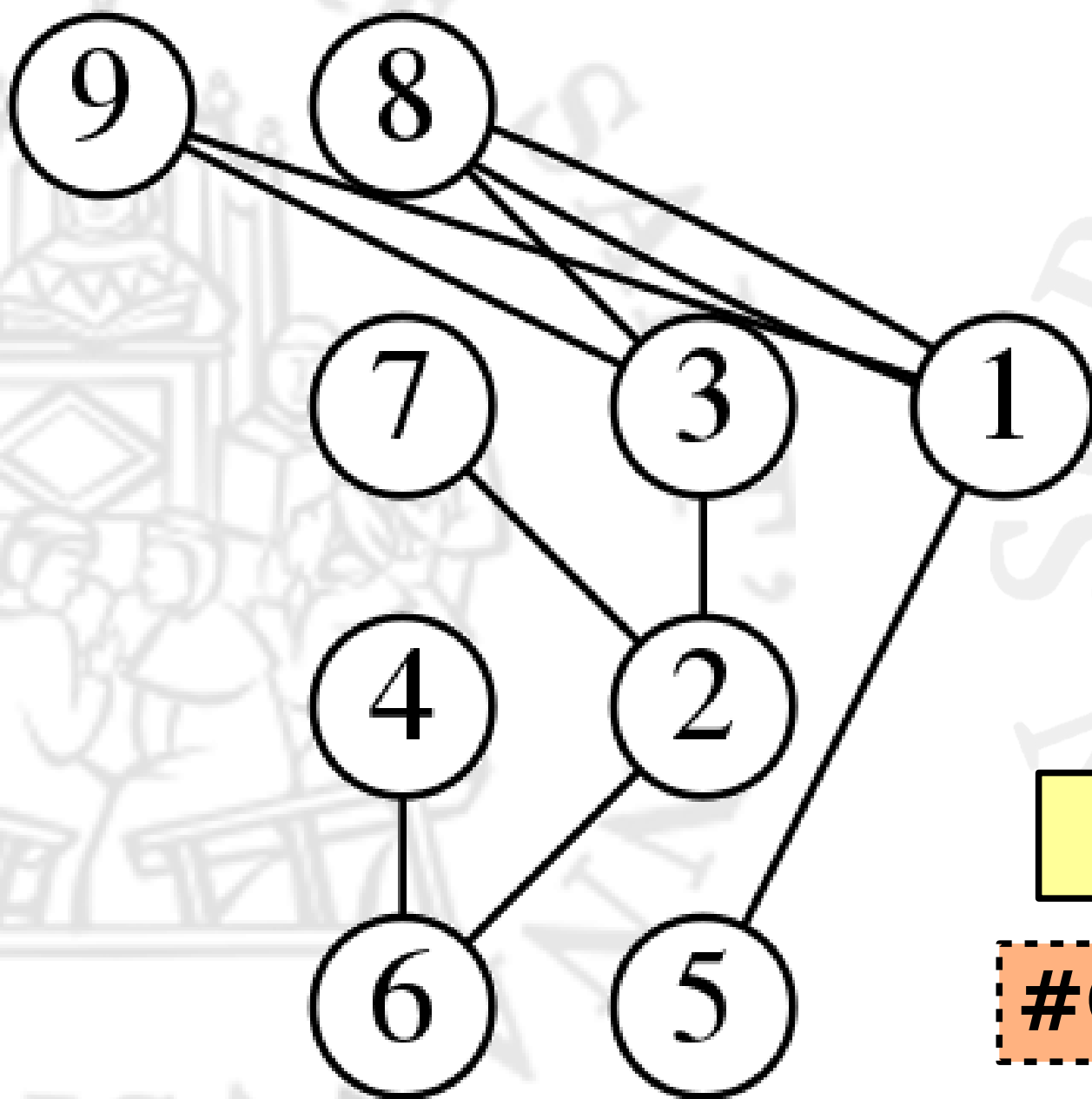
4: Small-Star

5: **until** Convergence

6: Termination_Phase

7: Check_Phase

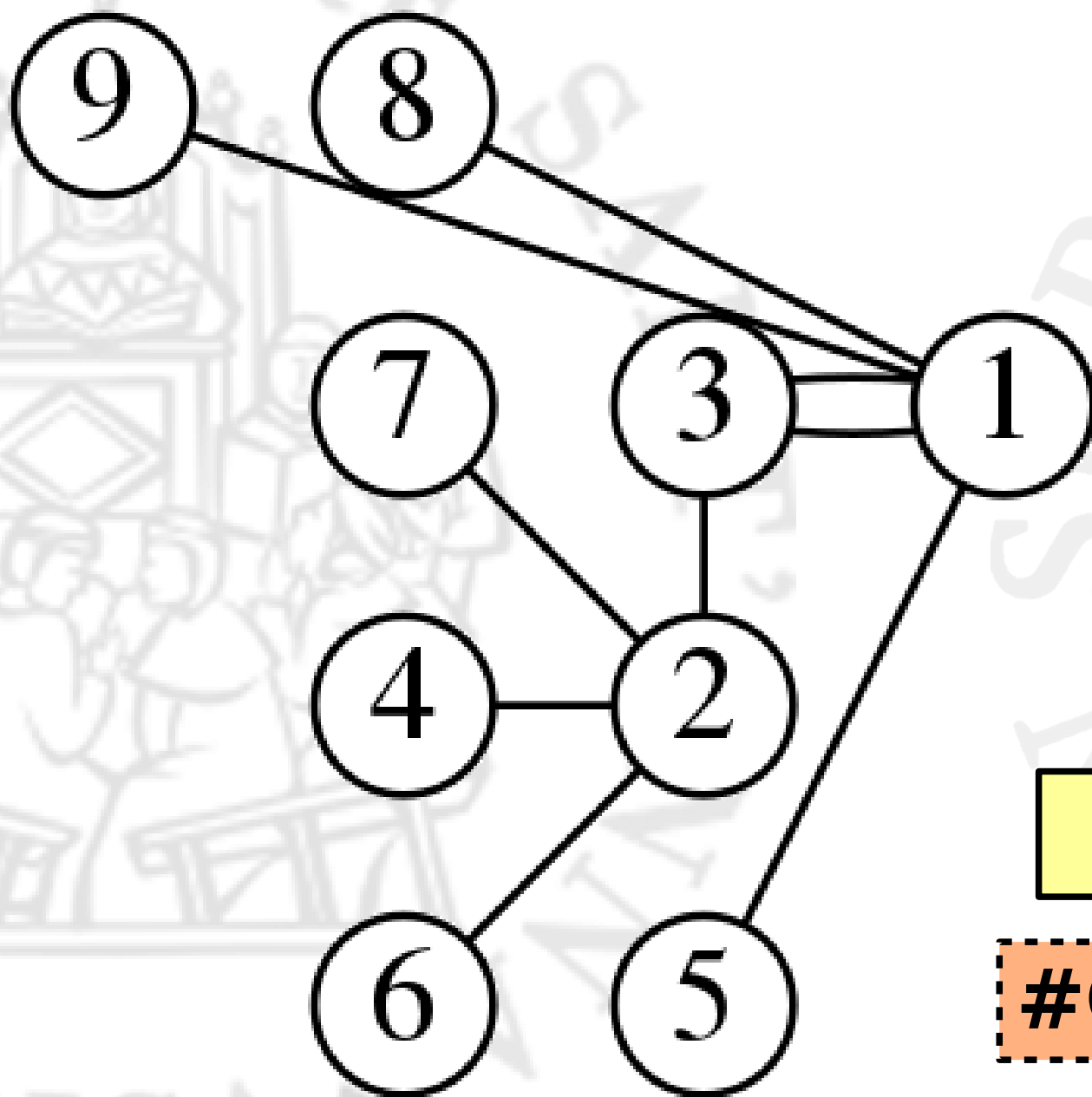
**STEP 0**



Large-Star

STEP 1

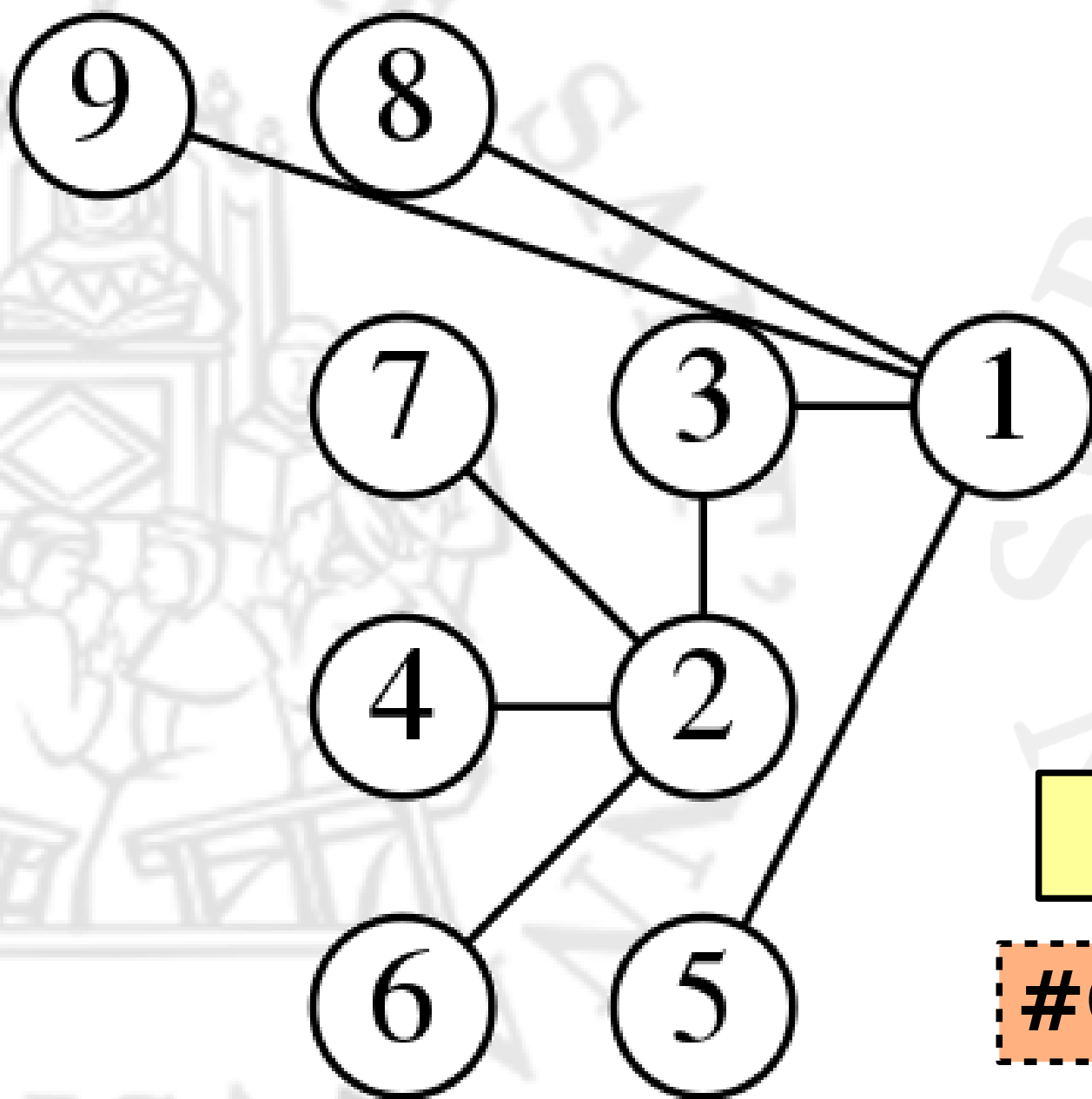
#Changes = 6



Small-Star

STEP 2

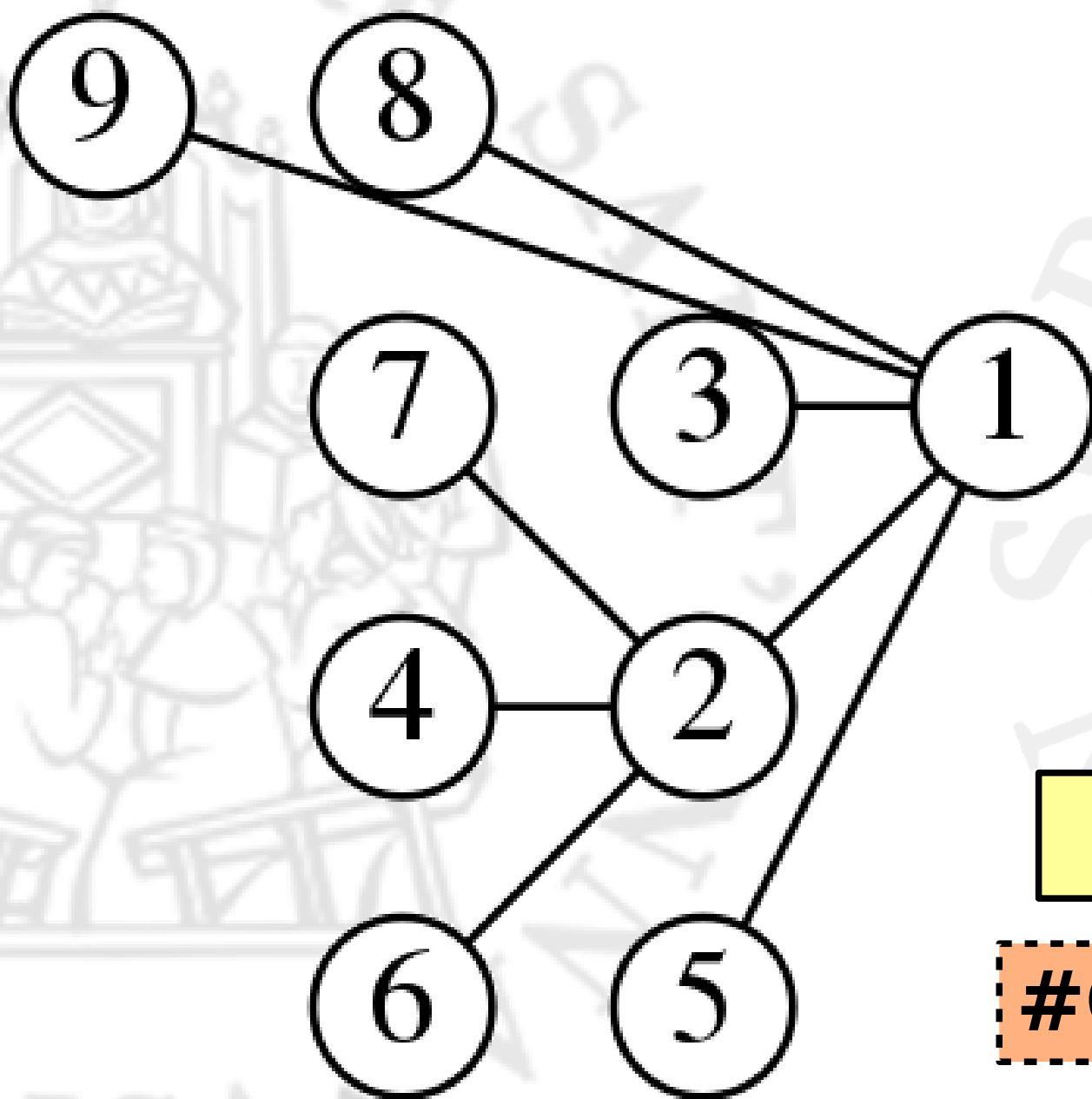
#Changes = 3



Large-Star

STEP 3

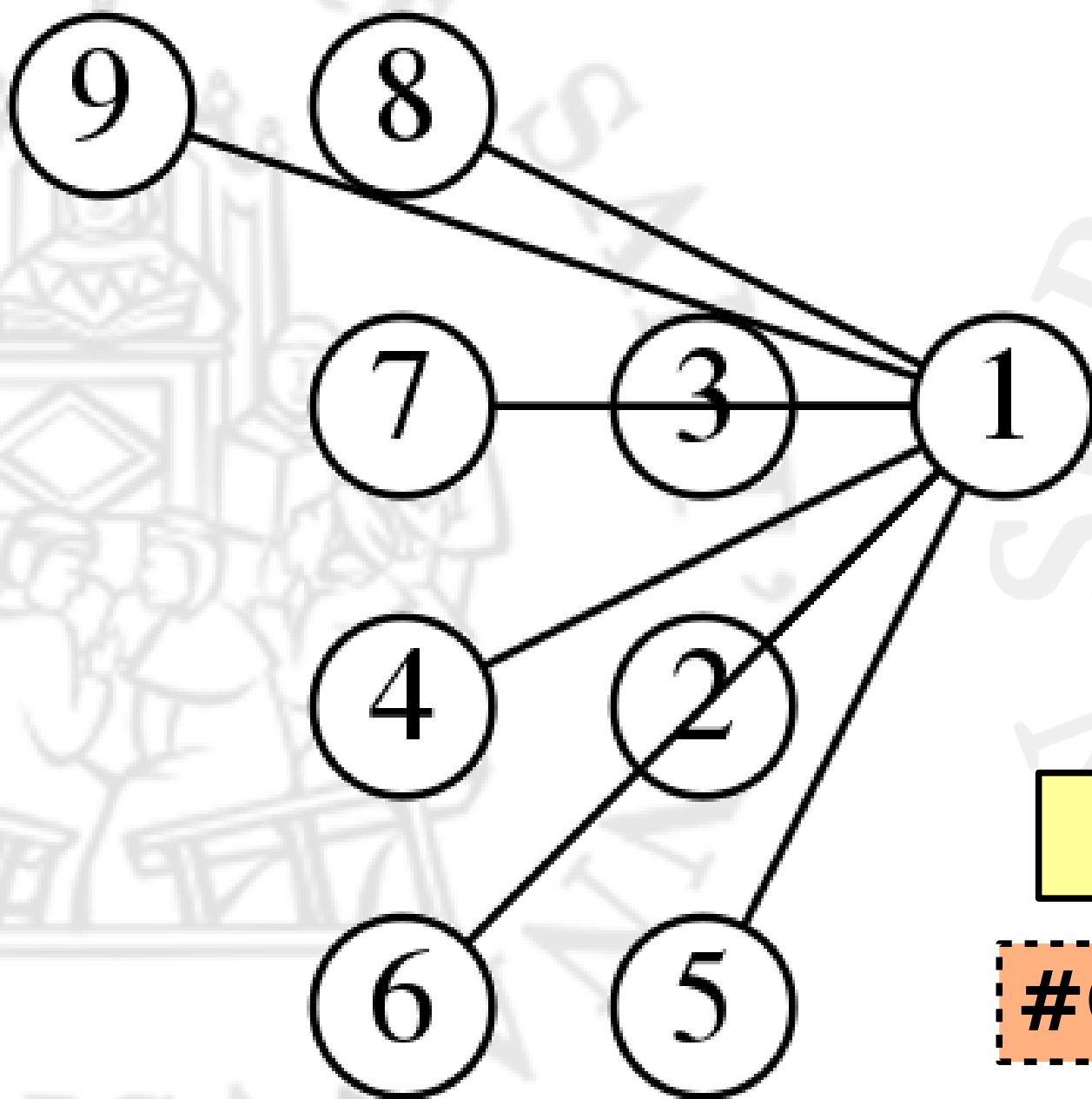
#Changes = 0



Small-Star

STEP 4

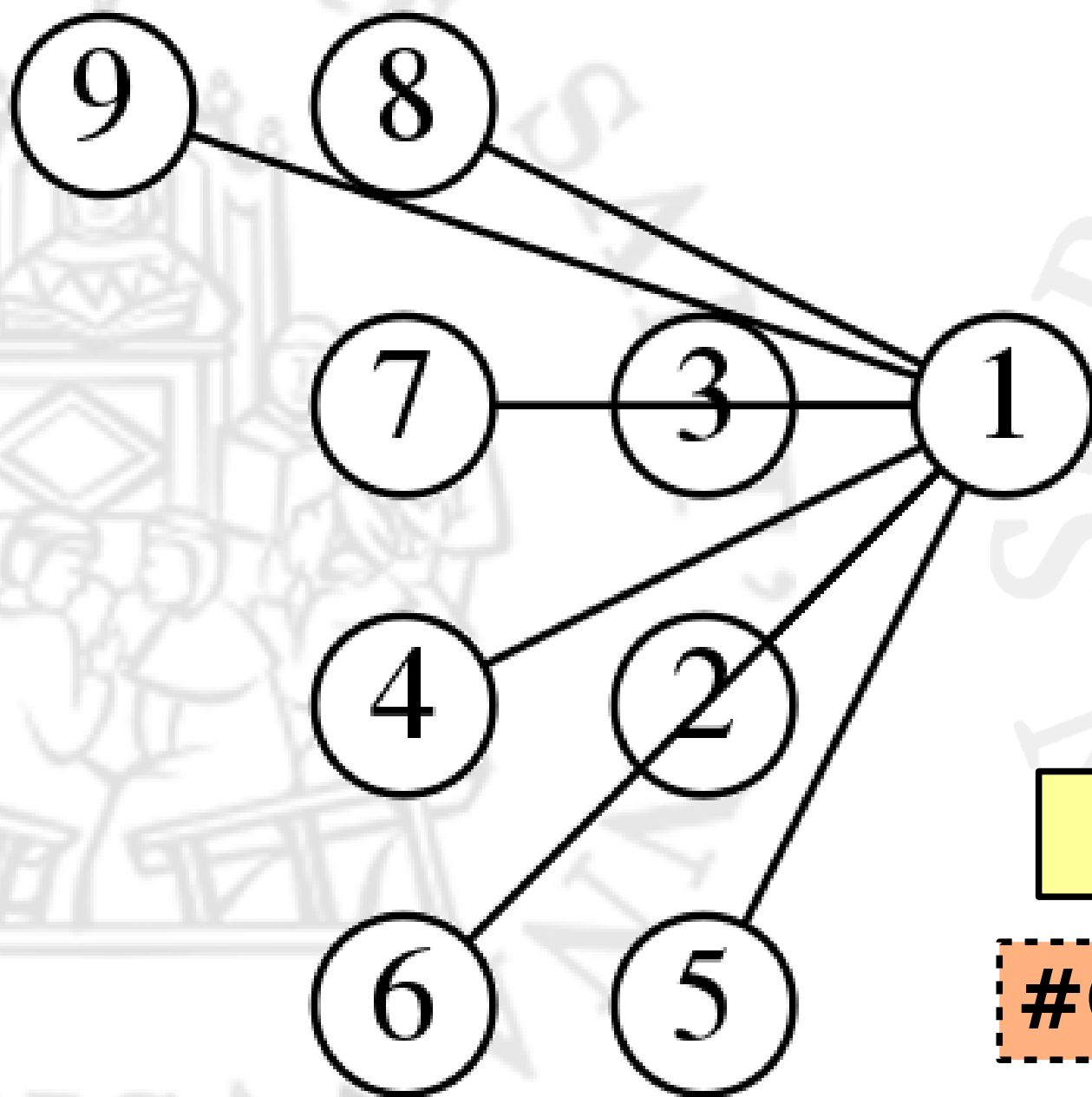
#Changes = 1



Large-Star

STEP 5

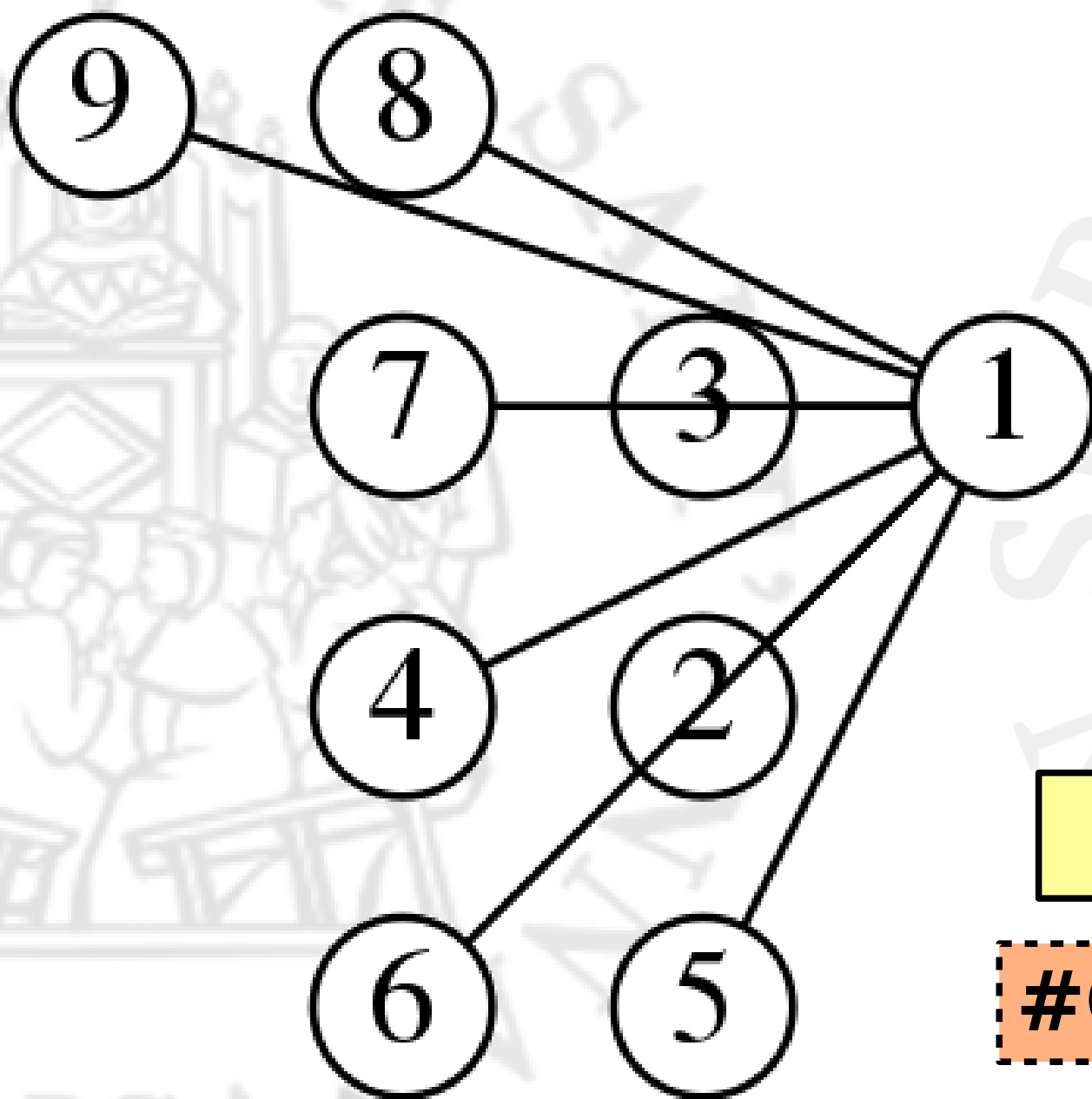
#Changes = 3



Small-Star

STEP 6

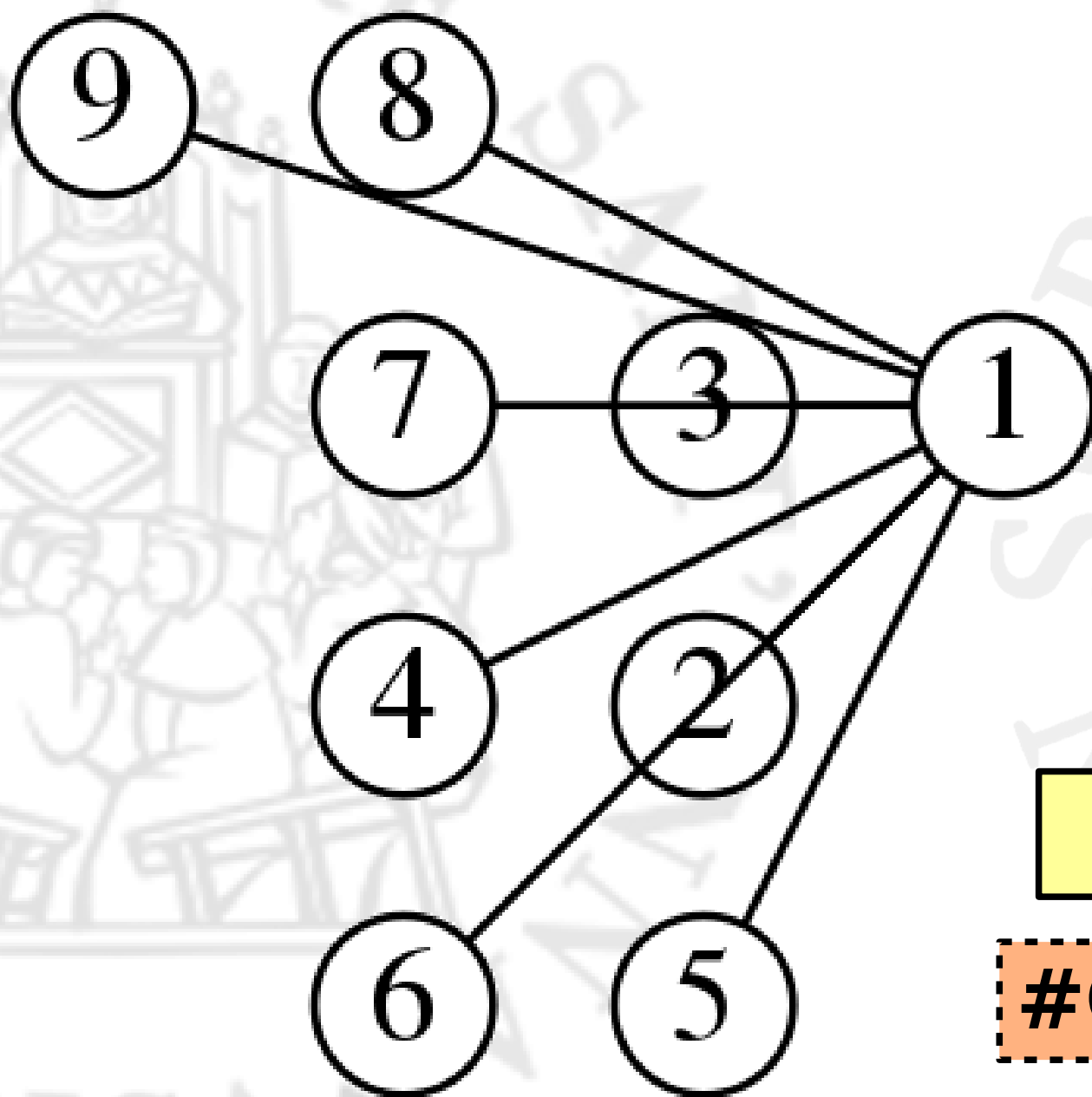
#Changes = 0



Large-Star

STEP 7

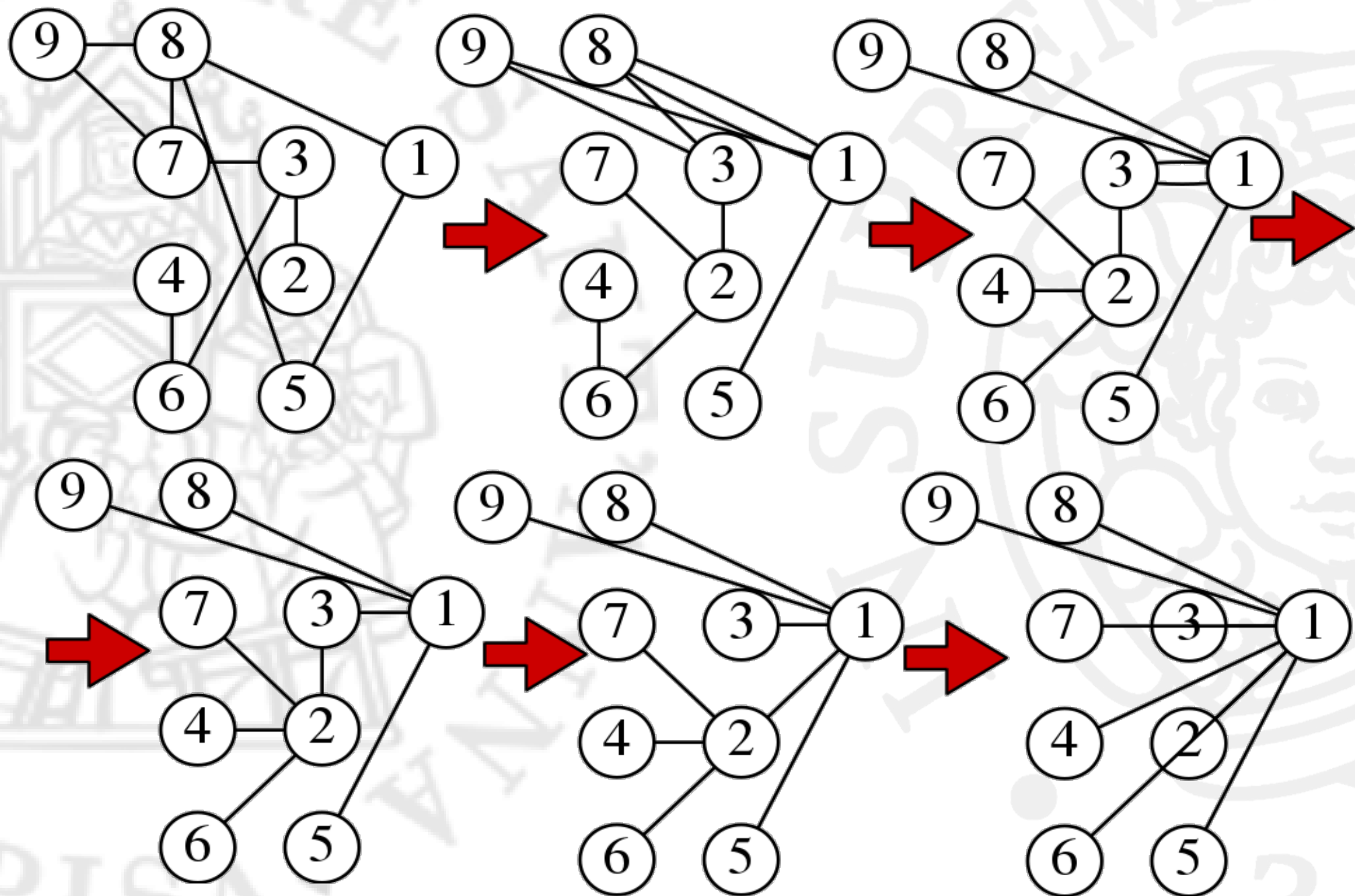
#Changes = 0



Small-Star

STEP 8

#Changes = 0



Algorithm 4 Our implementation

INPUT: $G = (V, E)$ represented

INPUT: A unique and positive

1: Initialization_Phase

2: **repeat**

3: Large-Star

4: Small-Star

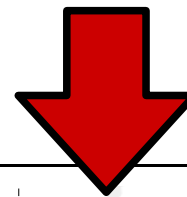
5: **until** Convergence

6: Termination_Phase

7: Check_Phase

Edge List:

2	1
3	1
4	1
5	1
6	1
7	1
8	1
9	1



Star List:

1 2 3 4 5 6 7 8 9

Algorithm 4 Our implementation of the Alternating Algorithm

INPUT: $G = (V, E)$ represented with an adjacency/cluster list.

INPUT: A unique and positive label ℓ_v for every node $v \in V$.

1: Initialization_Phase

2: repeat

3: Large-Star

4: Small-Star

5: until Convergence

6: Termination_Phase

7: **Check_Phase**

In Termination Phase, we ensure to **not store duplicate nodes within a cluster**.

A cluster is **malformed** if one of its node is present in another cluster.

Check that **all nodes are unique**.
If a node is found twice, it is surely present in two different clusters.

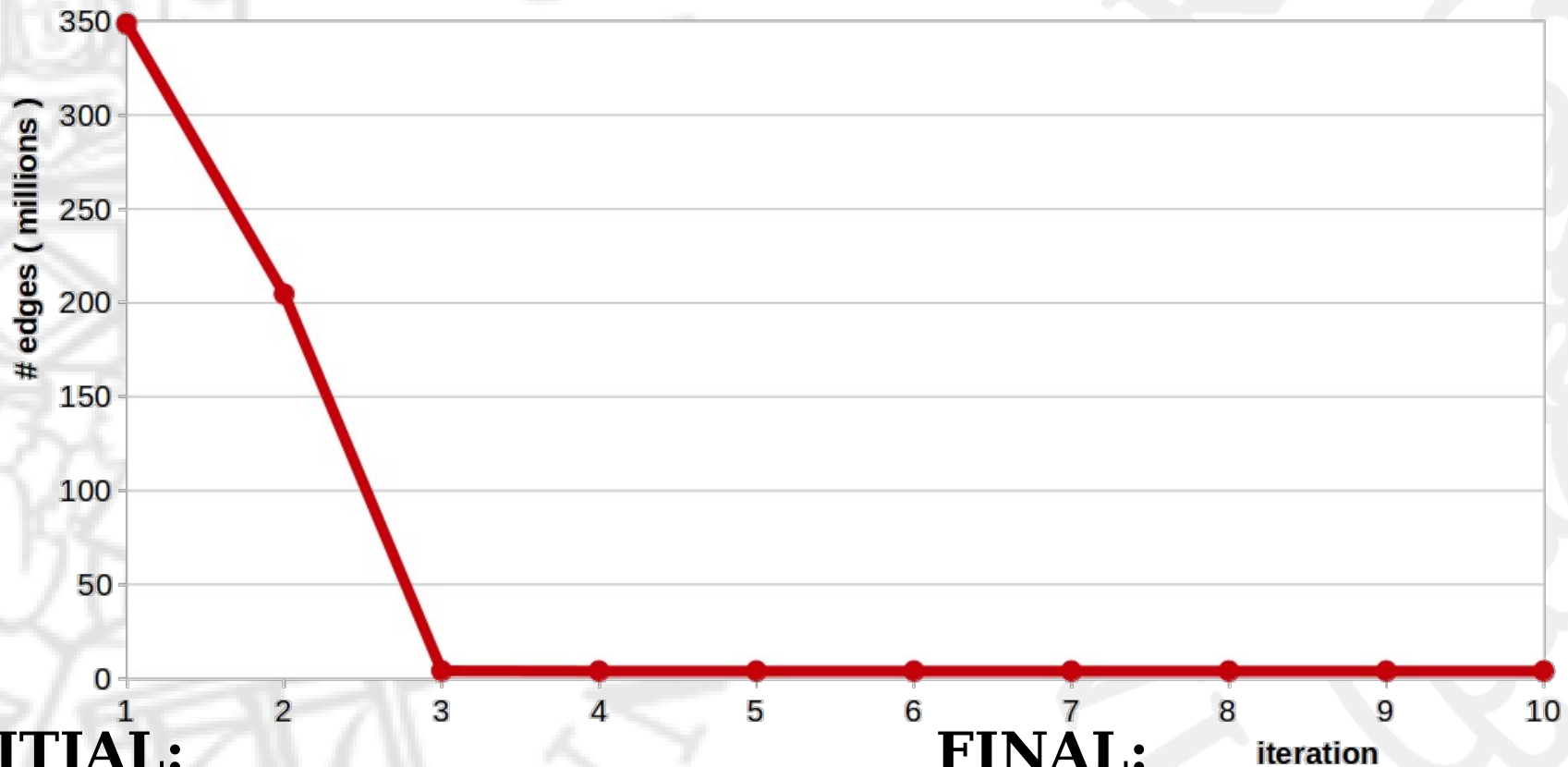
```
Processing input_0.txt.
Added hdfs://localhost:9000/user/draxent/input_0
ConnectedComponents Job started !
ConnectedComponents Job completed !
Input file format: CLUSTER_LIST.
Number of initial nodes: 9.
Number of initial Clusters: 6.
Number of final nodes: 9.
Number of final Clusters: 1.
TestOK: true.
TranslatorDriver Cluster2Text Job started !
TranslatorDriver Cluster2Text Job completed correctly !
16/01/26 11:31:35 INFO util.NativeCodeLoader: Loaded the n
Deleted hdfs://localhost:9000/user/draxent/input_0
Deleted hdfs://localhost:9000/user/draxent/out0
Deleted hdfs://localhost:9000/user/draxent/out0T
No differences between cluster_0.txt and cluster_out_0.txt
Deleted /home/draxent/Github/ConnectedComponents/data/clus
Test on input_0.txt compleated correctly !
Processing input_1.txt.
Added hdfs://localhost:9000/user/draxent/input_1
ConnectedComponents Job started !
ConnectedComponents Job completed !
Input file format: ADJACENCY_LIST.
Number of initial nodes: 20.
Number of initial Clusters: 0.
Number of final nodes: 20.
Number of final Clusters: 5.
TestOK: true.
TranslatorDriver Cluster2Text Job started !
TranslatorDriver Cluster2Text Job completed correctly !
16/01/26 11:34:36 INFO util.NativeCodeLoader: Loaded the n
Deleted hdfs://localhost:9000/user/draxent/input_1
Deleted hdfs://localhost:9000/user/draxent/out1
Deleted hdfs://localhost:9000/user/draxent/out1T
No differences between cluster_1.txt and cluster_out_1.txt
Deleted /home/draxent/Github/ConnectedComponents/data/clus
Test on input_1.txt compleated correctly !
```

```
Processing input_2.txt.
Added hdfs://localhost:9000/user/draxent/input_2
ConnectedComponents Job started !
ConnectedComponents Job completed !
Input file format: ADJACENCY_LIST.
Number of initial nodes: 25.
Number of initial Clusters: 0.
Number of final nodes: 25.
Number of final Clusters: 1.
TestOK: true.
TranslatorDriver Cluster2Text Job started !
TranslatorDriver Cluster2Text Job completed correctly !
16/01/26 11:38:43 INFO util.NativeCodeLoader: Loaded the n
Deleted hdfs://localhost:9000/user/draxent/input_2
Deleted hdfs://localhost:9000/user/draxent/out2
Deleted hdfs://localhost:9000/user/draxent/out2T
No differences between cluster_2.txt and cluster_out_2.txt
Deleted /home/draxent/Github/ConnectedComponents/data/clus
Test on input_2.txt compleated correctly !
Processing input_3.txt.
Added hdfs://localhost:9000/user/draxent/input_3
ConnectedComponents Job started !
ConnectedComponents Job completed !
Input file format: CLUSTER_LIST.
Number of initial nodes: 98.
Number of initial Clusters: 67.
Number of final nodes: 98.
Number of final Clusters: 6.
TestOK: true.
TranslatorDriver Cluster2Text Job started !
TranslatorDriver Cluster2Text Job completed correctly !
16/01/26 11:42:56 INFO util.NativeCodeLoader: Loaded the n
Deleted hdfs://localhost:9000/user/draxent/input_3
Deleted hdfs://localhost:9000/user/draxent/out3
Deleted hdfs://localhost:9000/user/draxent/out3T
No differences between cluster_3.txt and cluster_out_3.txt
Deleted /home/draxent/Github/ConnectedComponents/data/clus
Test on input_3.txt compleated correctly !
```

Clique List file:

- Size of 73.7MB

- #Iterations = 10



INITIAL:

- #Nodes = 5,869,938
- #Clique = three millions
- #Edges = 348,528,515

FINAL:

- #Nodes = 5,869,938
- #Star = 2,039,304
- #Edges = 3,830,634