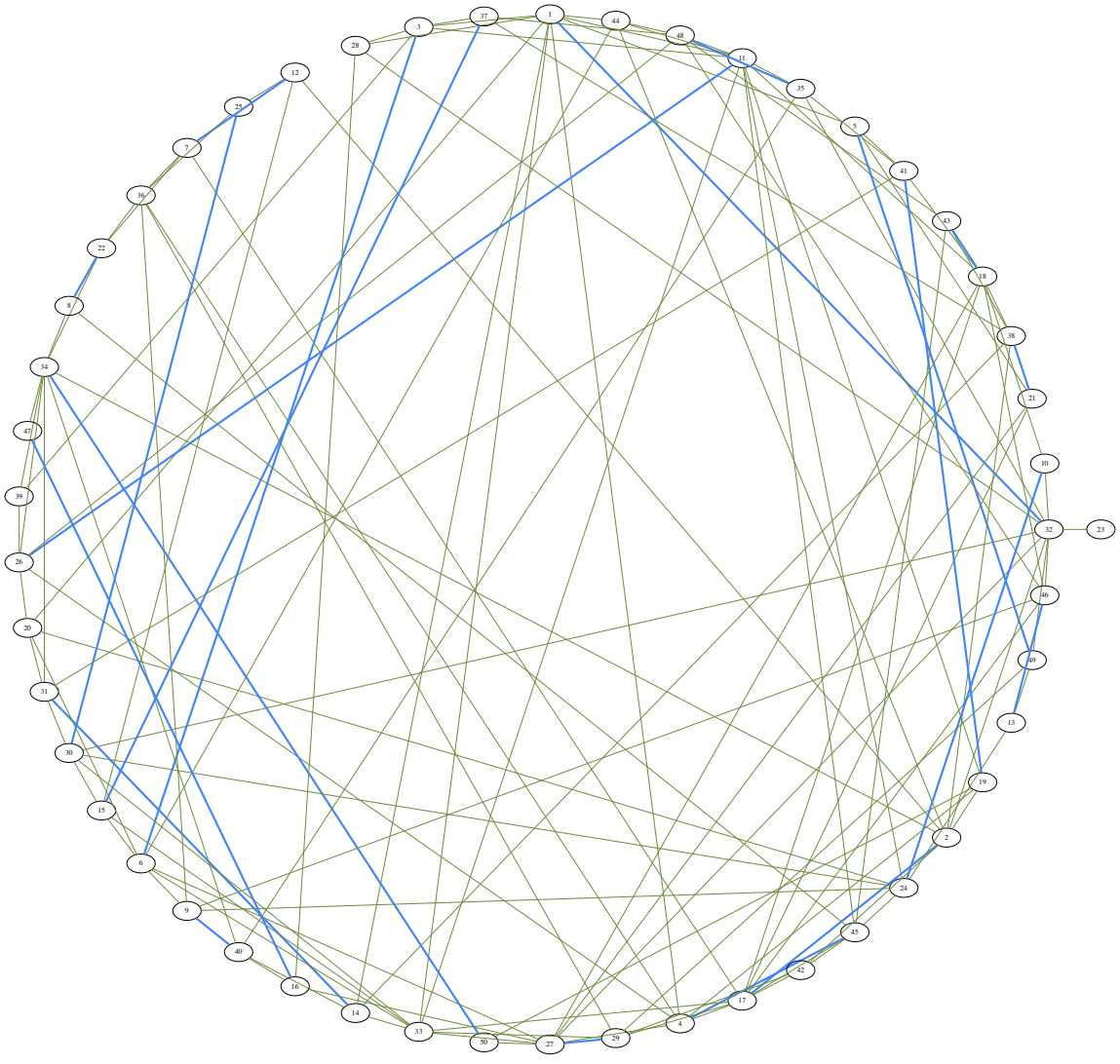
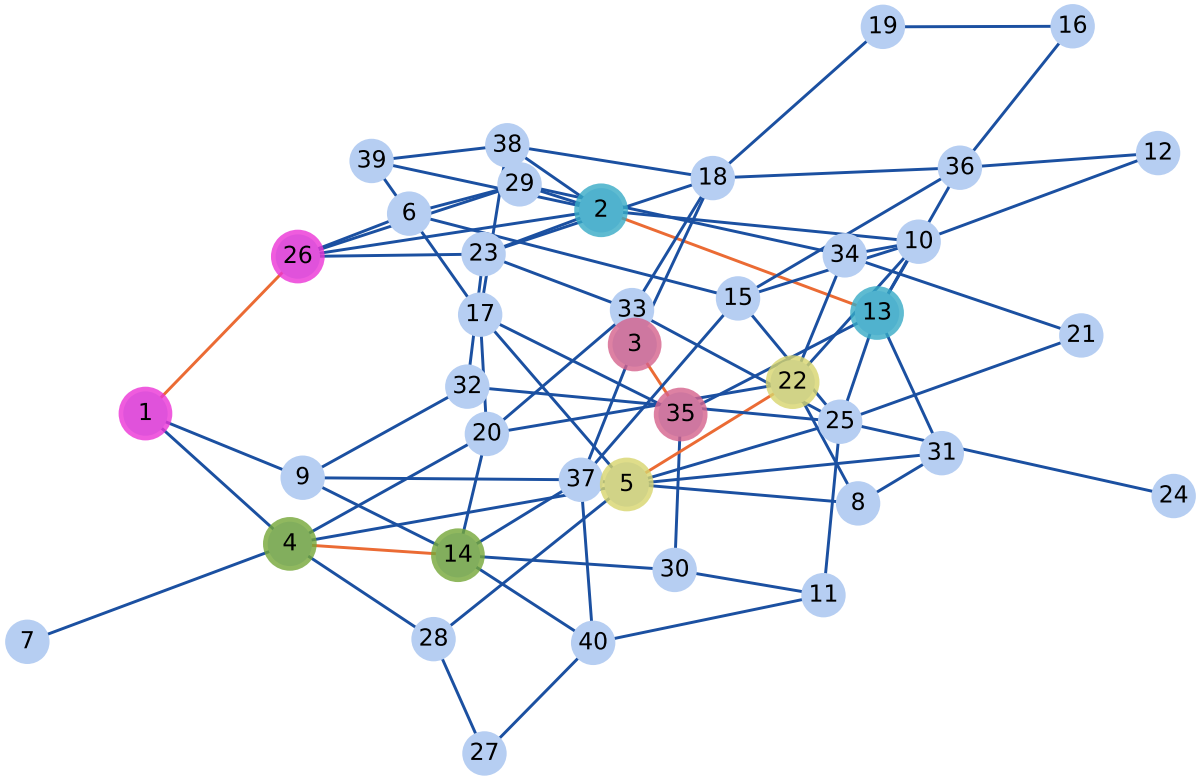
# Algorithmic complexity and graphs: the matching problem

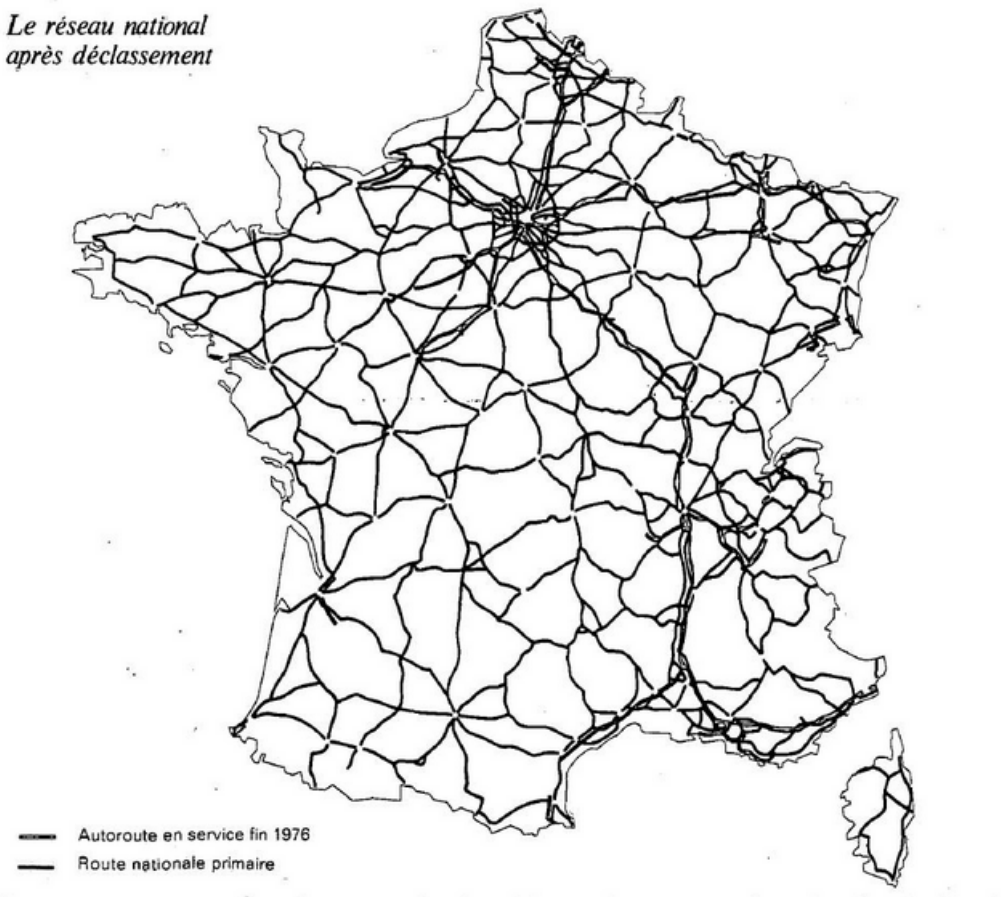
October 1, 2022



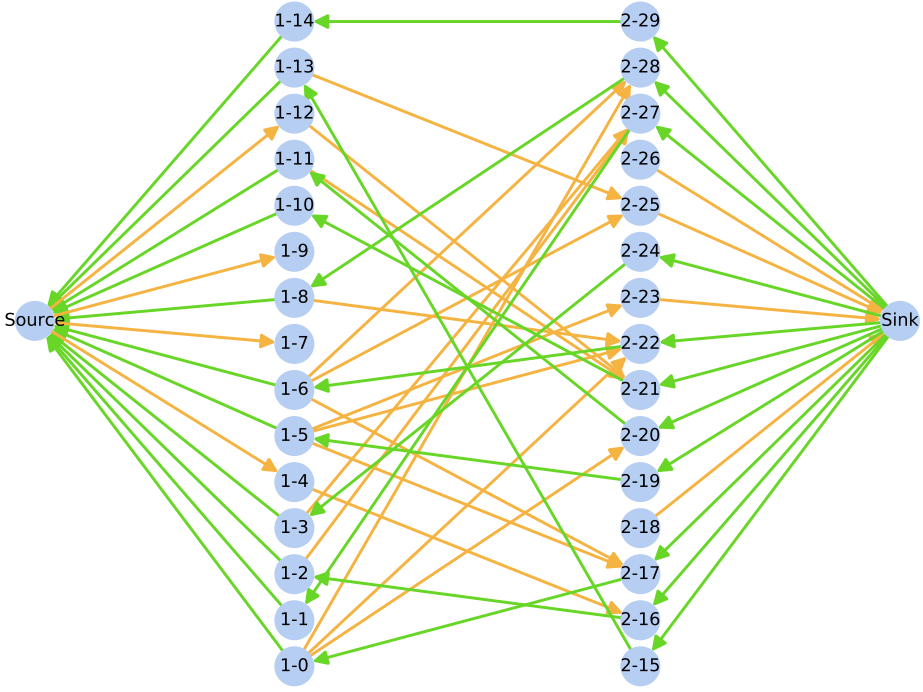

Matching size: 21
Algo step: 128
Nb nodes: 50

Matching size: 5
Algo step: 19
Nb nodes: 40

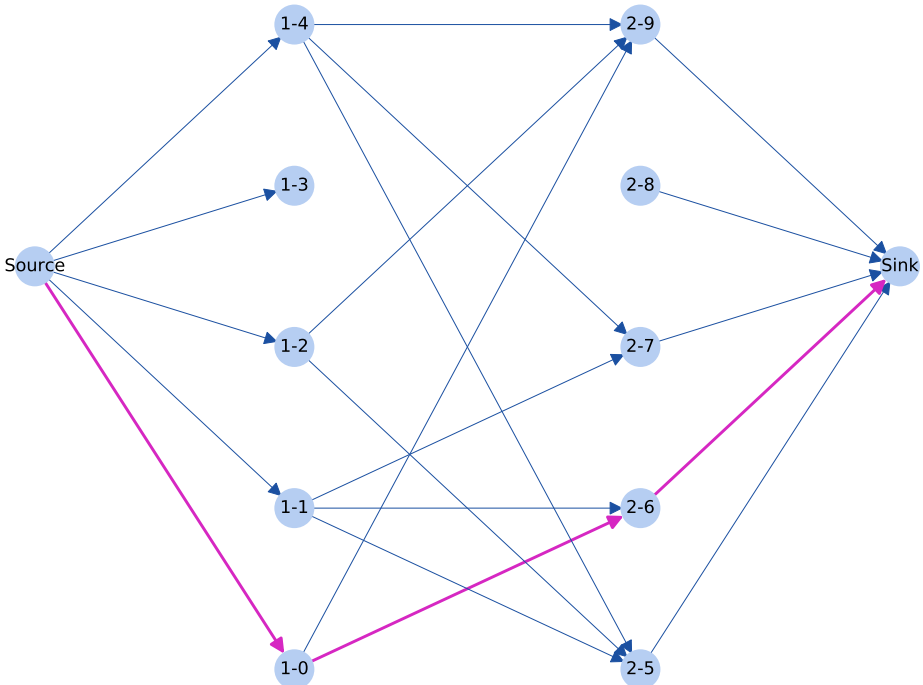



Le réseau national
après déclassement
Autoroute en service fin 1976
Route nationale primaire

residual graph step 12



augmenting path step 1



# The mathing problem

The matching problem
- Definition of the problem
- Experimental solutions
- Brute force algorithm
- Greedy algorithm

# Introductory example 1 : Max Flow

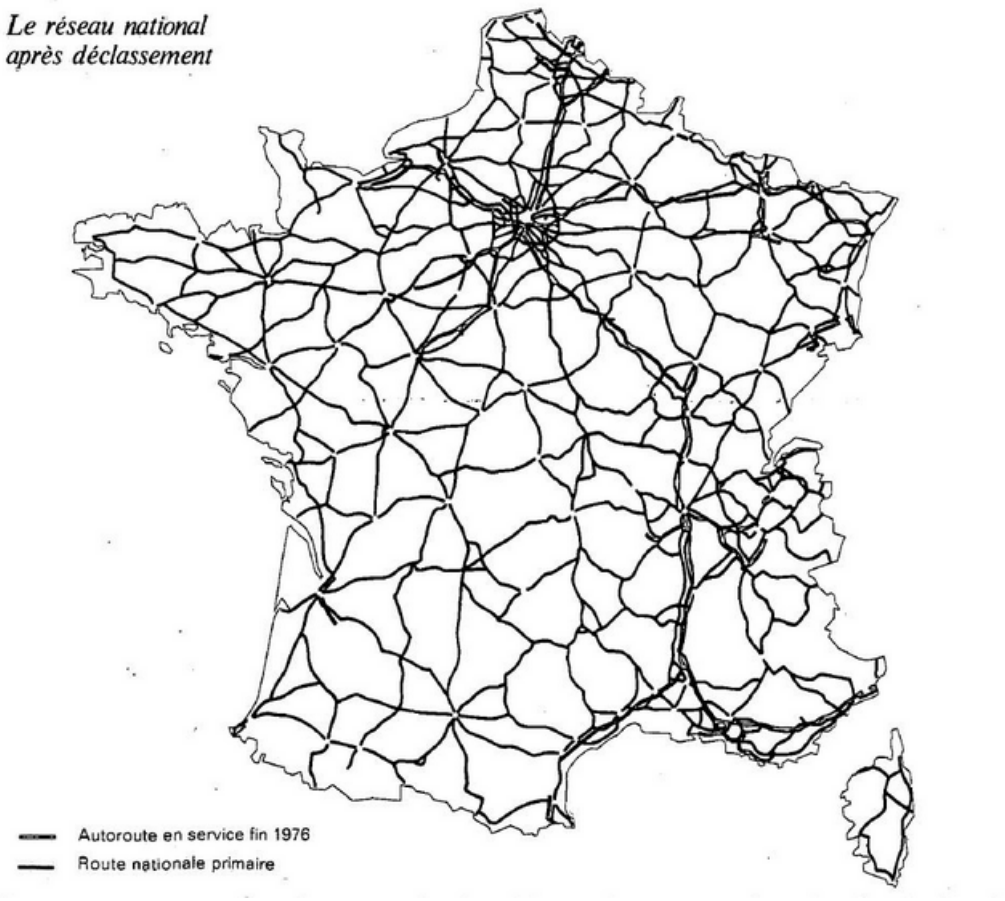



Figure: Problem 1 : transporting merchandise through a network

# Introductory example 2 : Maximum matching (Optimal assignment, problème d'affectation)

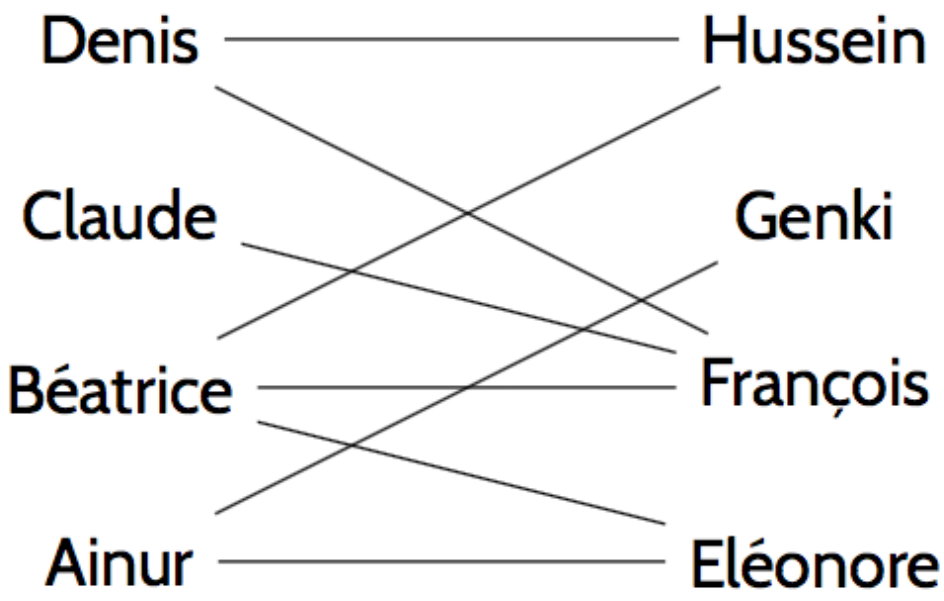



Figure: Problem 2 : Building the largest possible number of teams of 2 persons.

# Introductory example 2

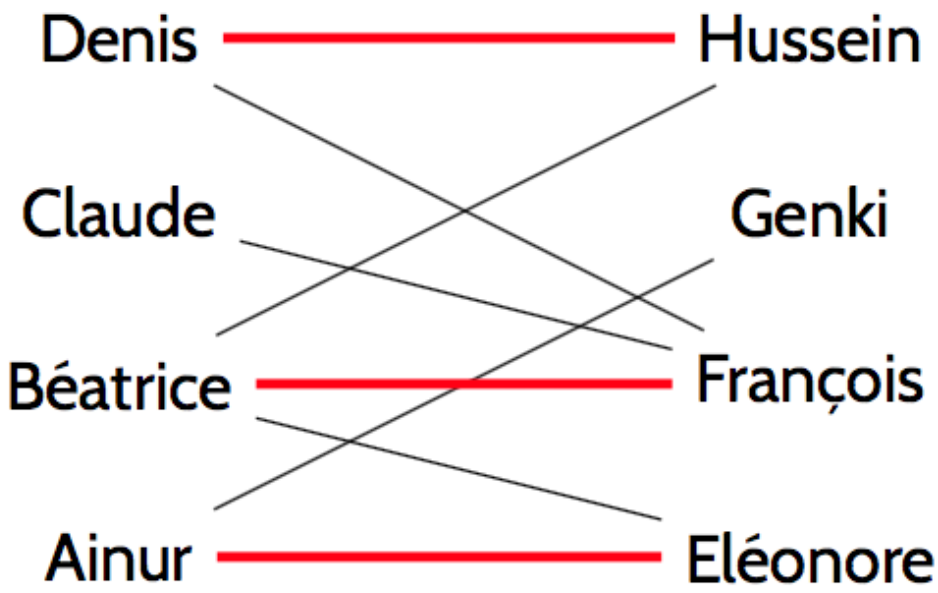



Figure: Problem 2 : not optimal assignment

# Introductory example 2

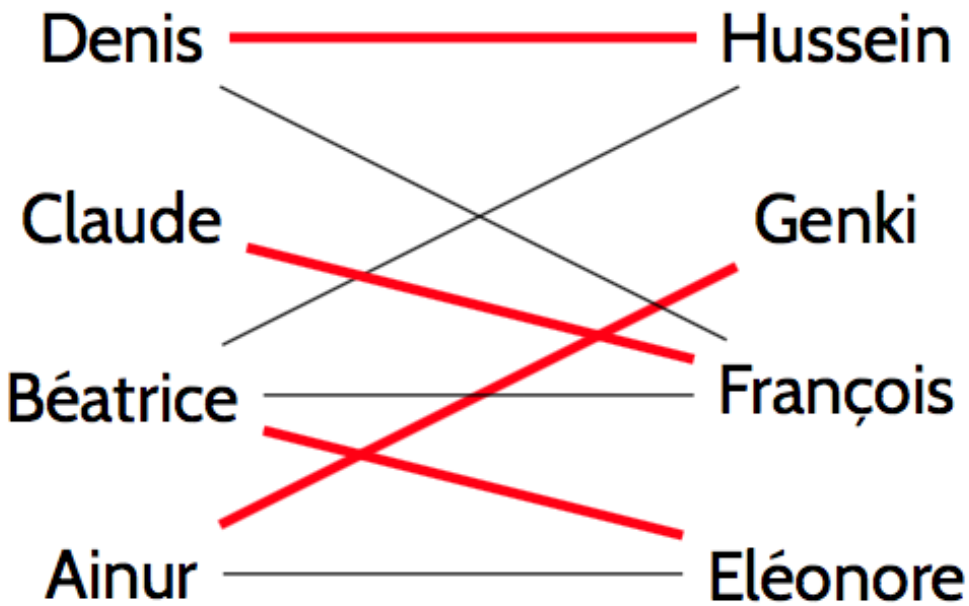



Figure: Problem 2 : optimal assignment

# Other examples

- Assigning students to internships

# Other examples

- Assigning students to internships
- Assigning machines to a task

# Summary

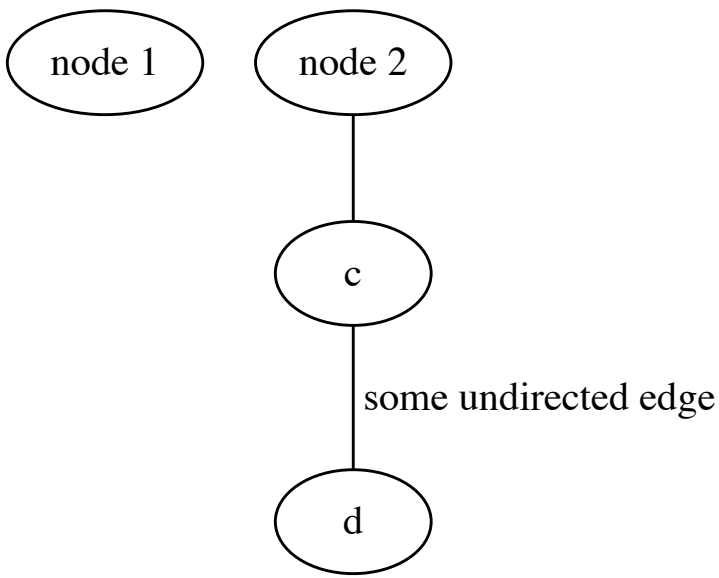
- Today we will work on **connnecting the two problems**.

# Summary

- Today we will work on **connnecting the two problems**.
- In some specific cases, the two problems **equivalent**.

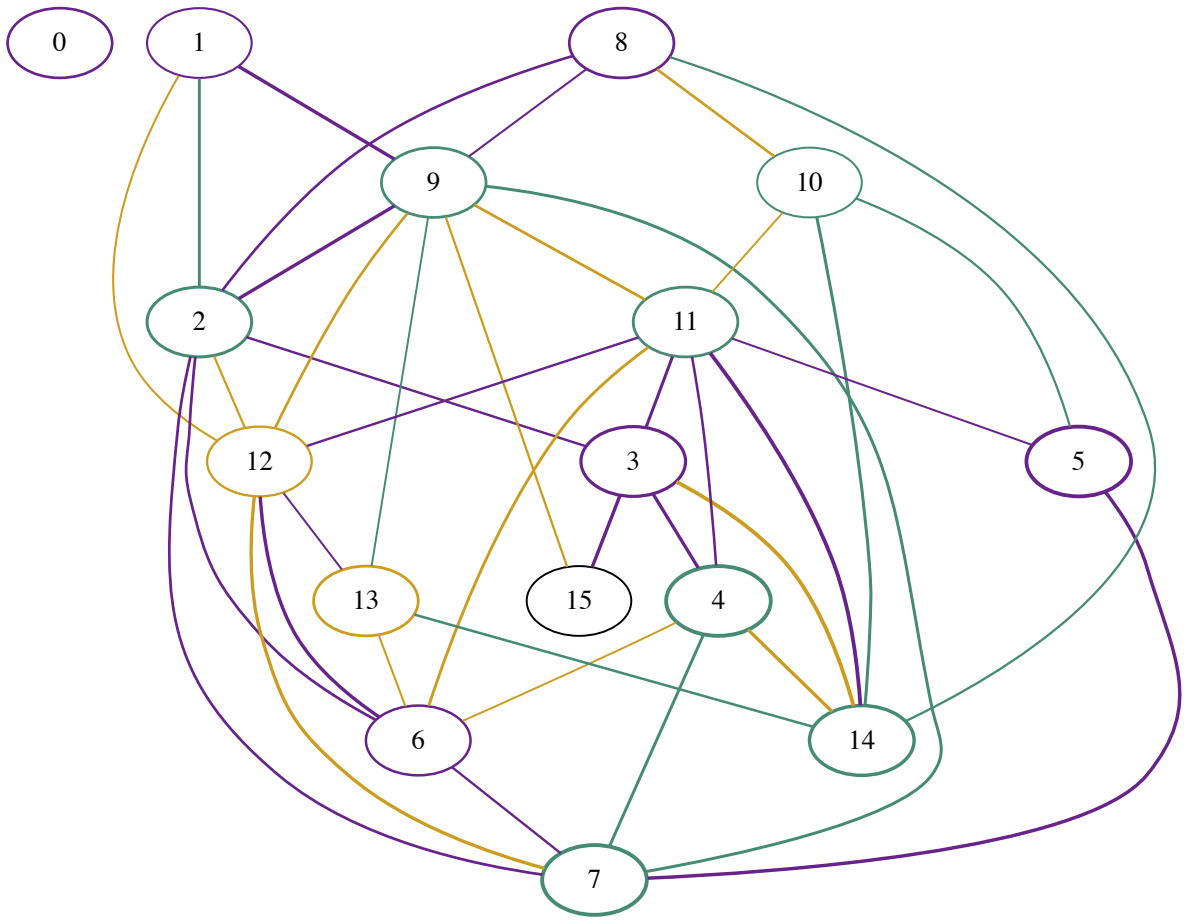
# Reminders on graphs

- A graph is defined by set of **vertices** (or **nodes** ) $V$ and a set of **edges** $E$.



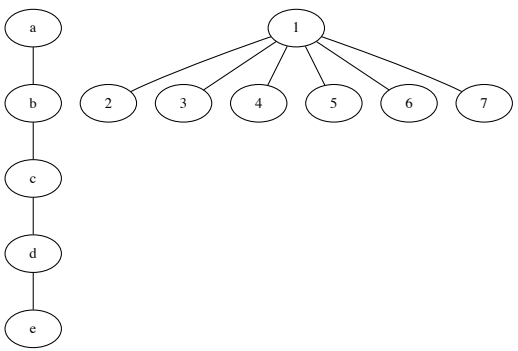
# Reminders on graphs

## Undirected graph

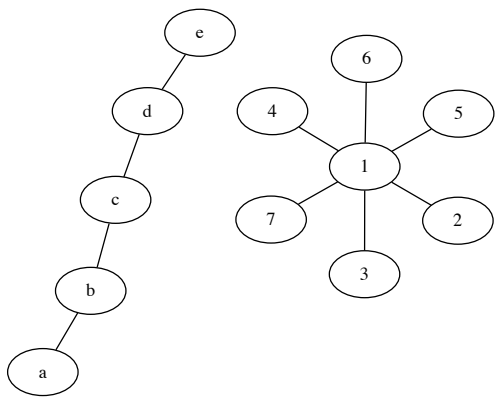


# Other available tool : graphviz

- A tool to visualize graphs
- Several **generator programs** : dot, neato



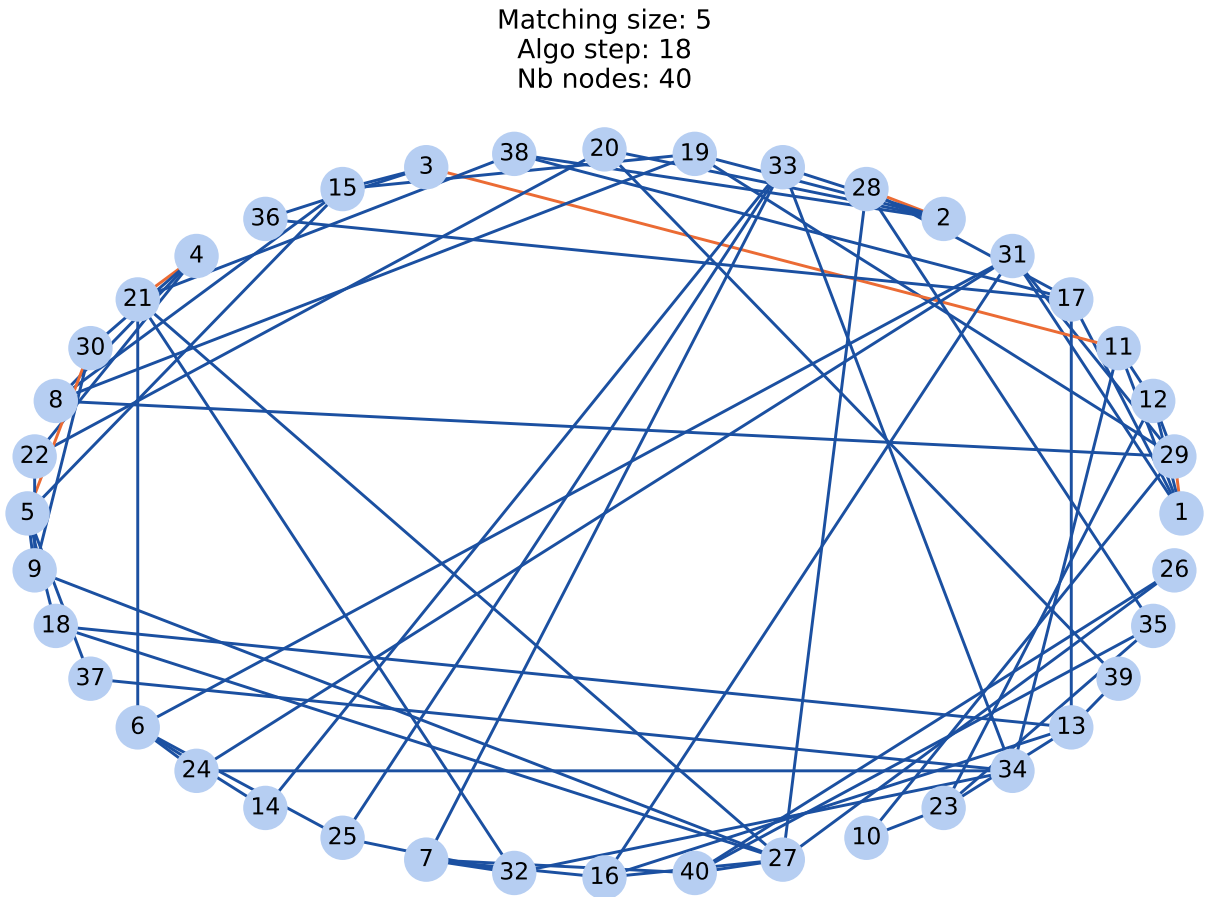

(a) Image generated with **dot**



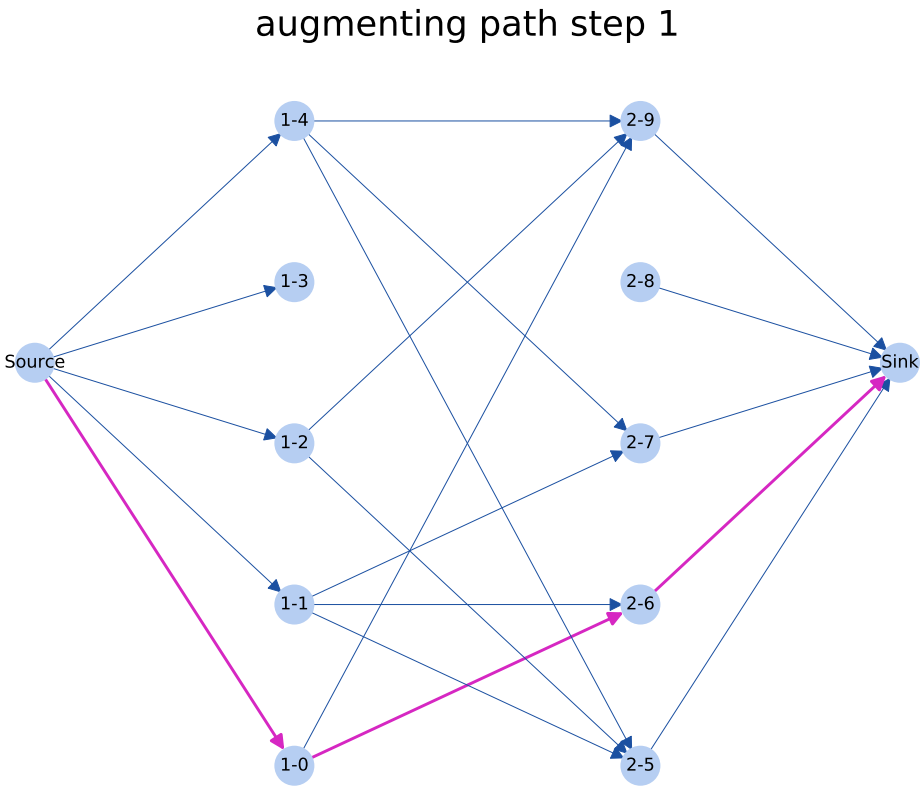

(b) Image generated with **neato**

# Networkx

We will use networkx.



# Networkx

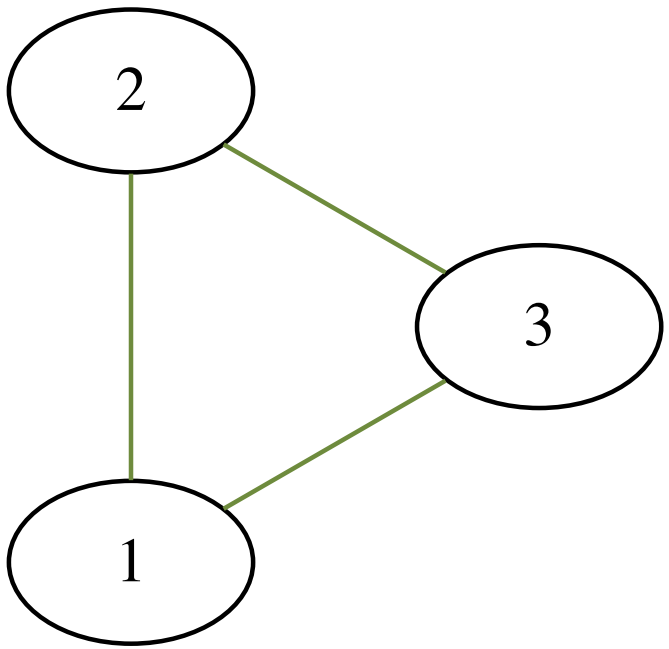


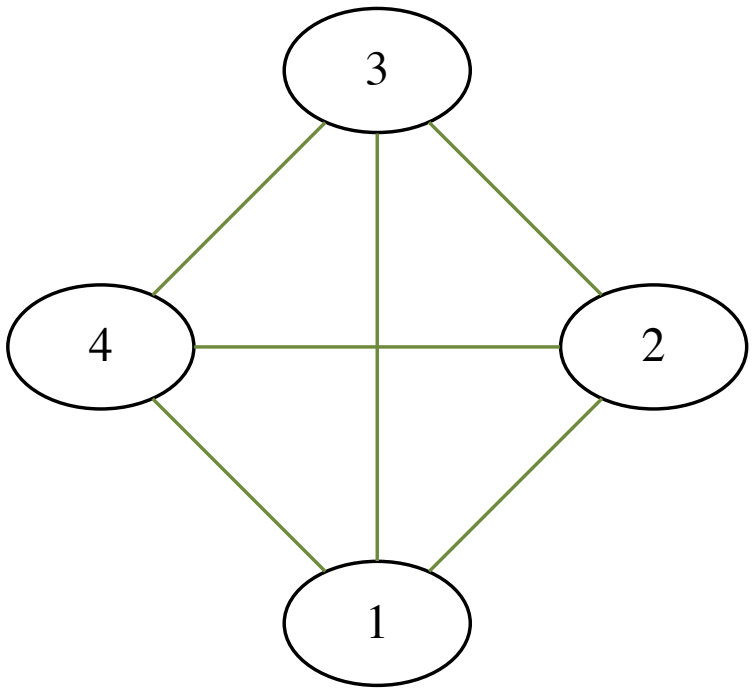
# Warm up question

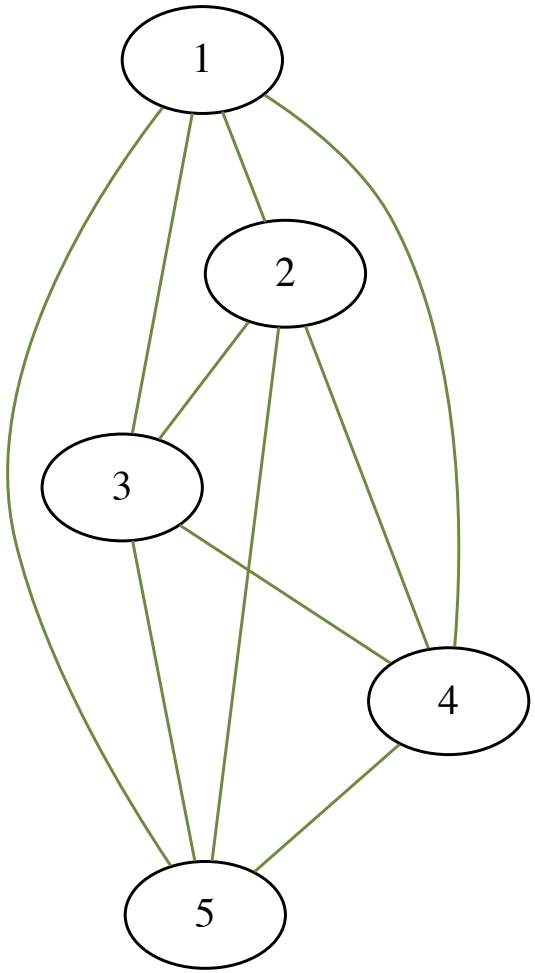
Given an **undirected** graph with $n$ nodes, how many edges can we build ?
Notation of a graph : $G(V, E)$

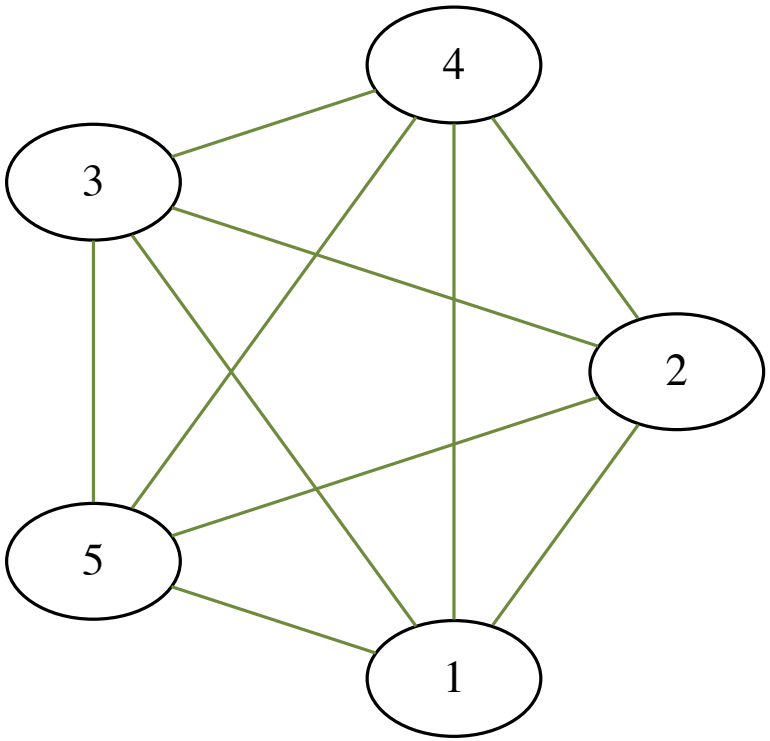
- $V$ : set of $n$ vertices
- $E$ : set of edges

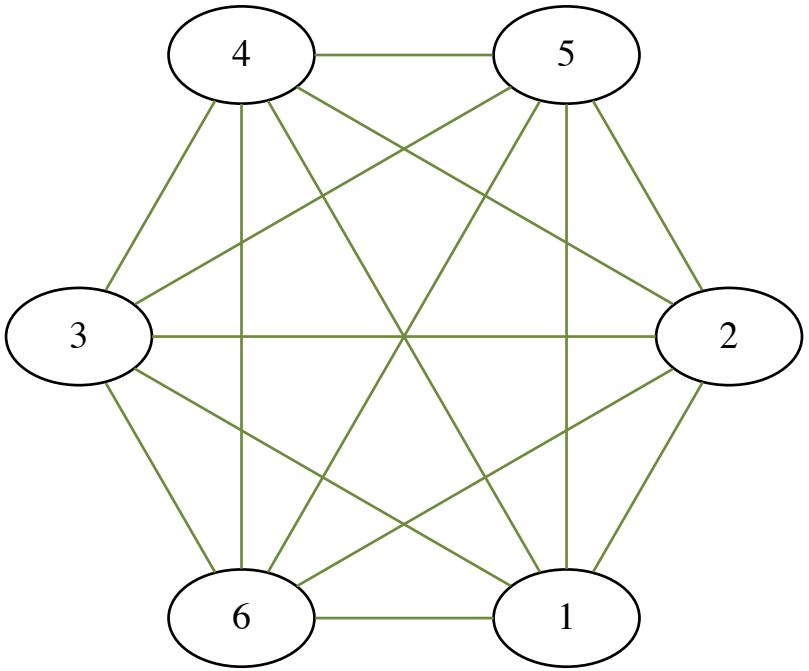
1
2


2
3
1


3
4
2
1


1
2
3
4
5


4
3
2
5
1


4
5
3
2
6
1

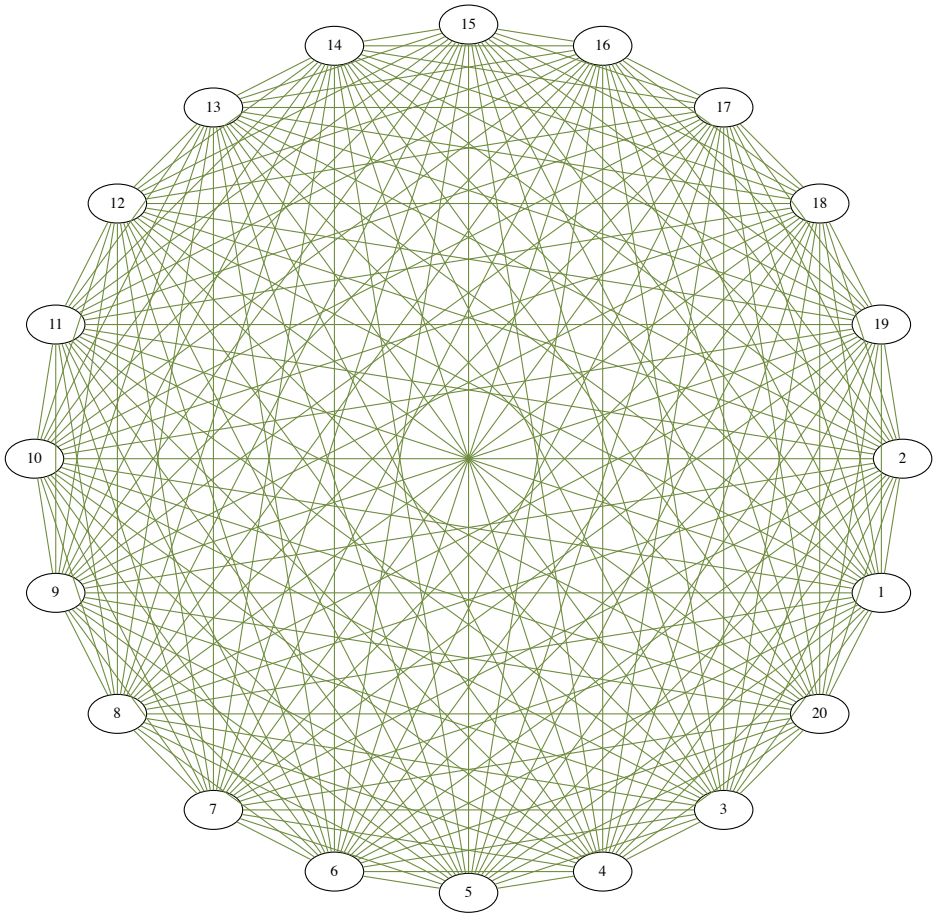
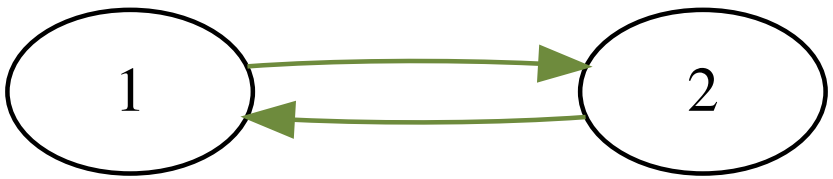
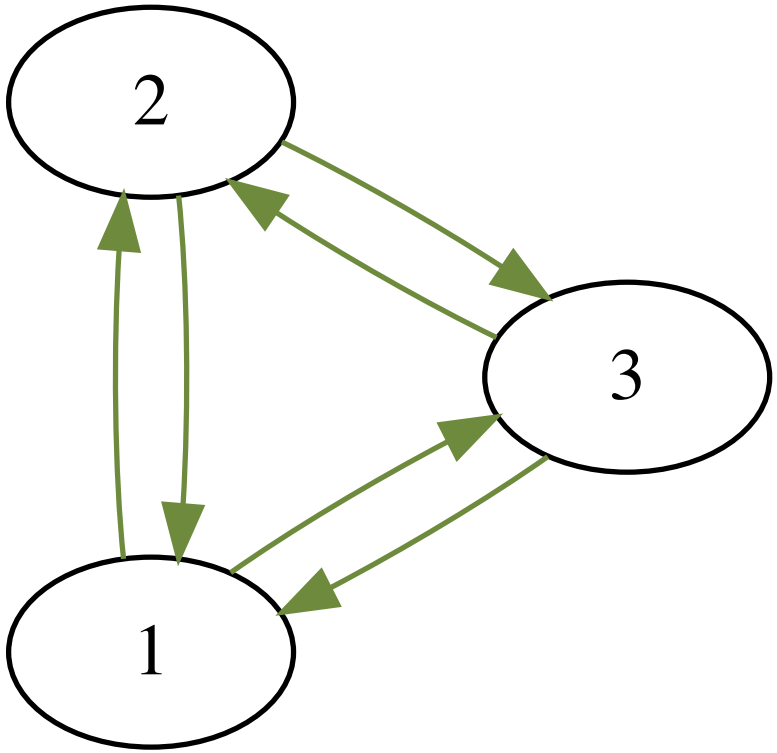



Figure: We cannot count anymore

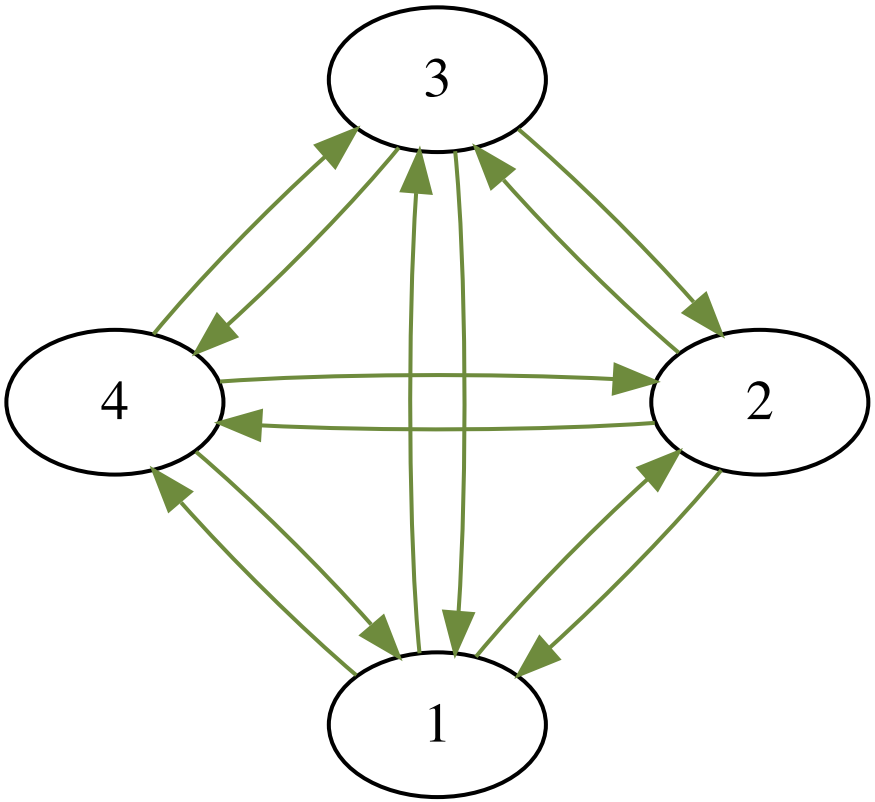
# What if the graph is directed ?



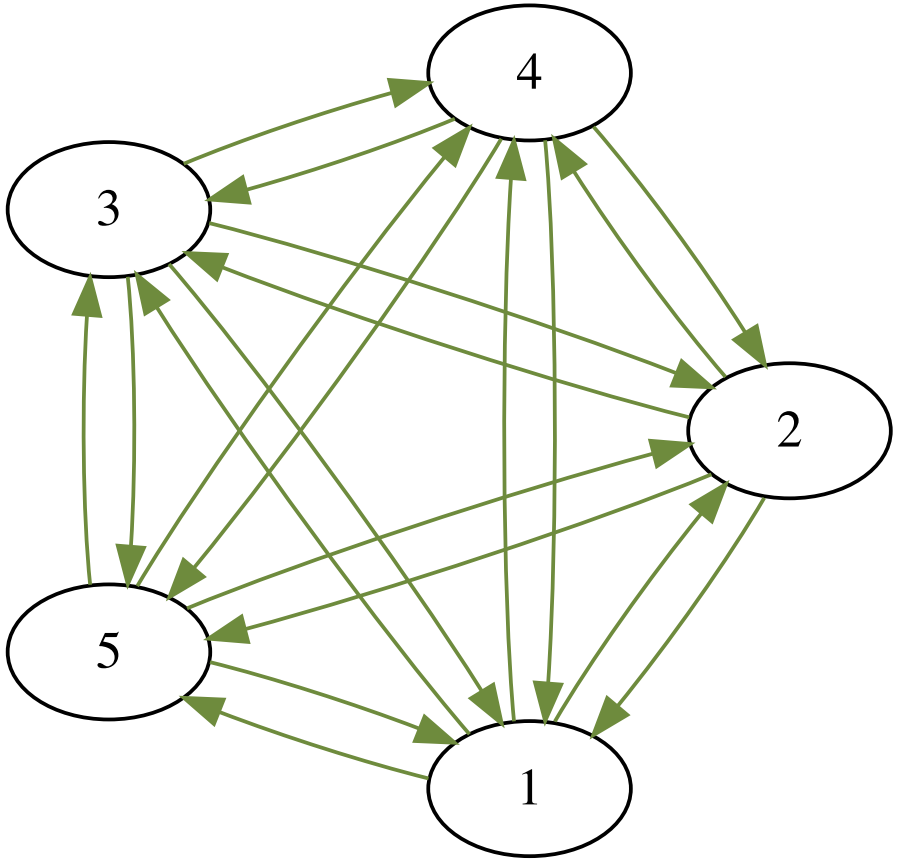
# What if the graph is directed ?



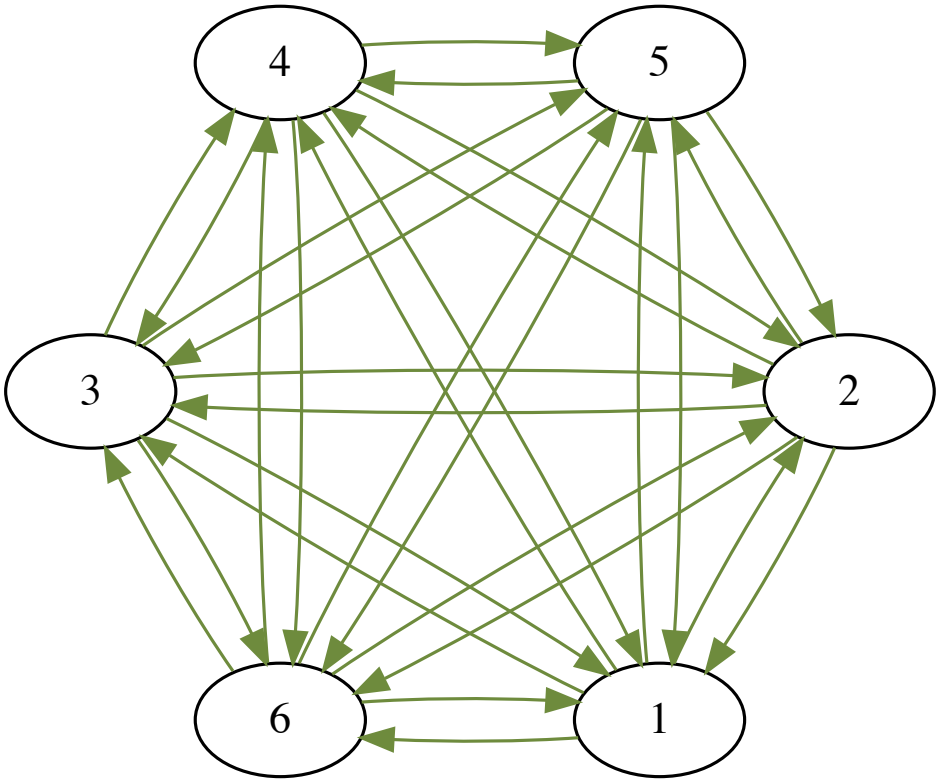
# What if the graph is directed ?



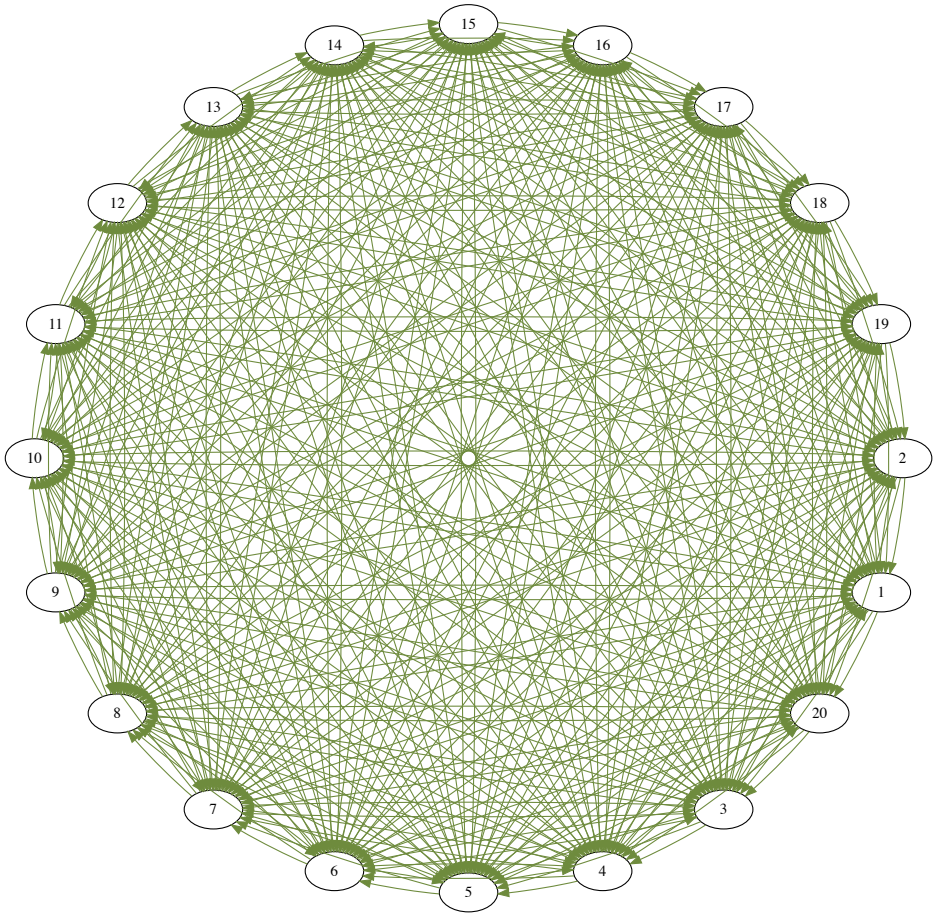
# What if the graph is directed ?



# What if the graph is directed ?



# What if the graph is directed ?



# Warm up question

Given an **directed** graph with $n$ nodes, how many edges can we build ?

# Warm up question

Given an **directed** graph with $n$ nodes, how many edges can we build ?

$$n(n-1) \tag{1}$$

# Warm up question

Given an **directed** graph with $n$ nodes, how many edges can we build ?

$$n(n-1) \tag{2}$$

So if the graph is **undirected**, we can build :

$$\frac{n(n-1)}{2} \tag{3}$$

edges.

# Remark

$\frac{n(n-1)}{2}$ is also the number of subsets of size 2 in a set of size $n$.

$$\binom{n}{2} = \frac{n!}{2!(n-2)!} \tag{4}$$

# Famous graph problem

- Dominating set

# Famous graph problem

- Dominating set
- Maximum clique

# Famous graph problem

- Dominating set
- Maximum clique
- Coloring

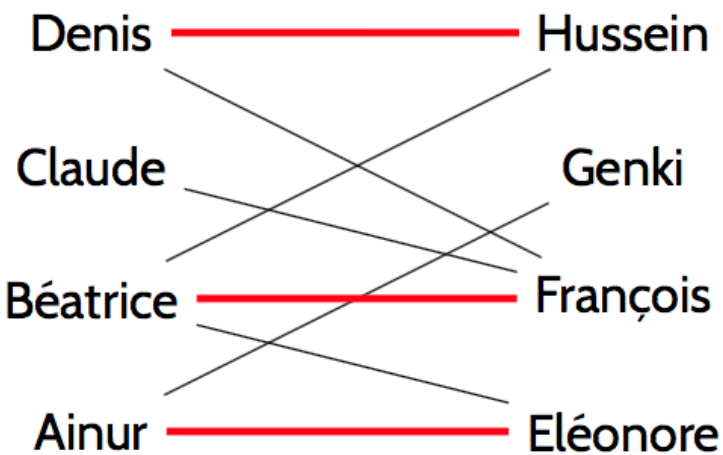
# Matching problem

Let us now focus on the **matching problem** (problème du **couplage** )

# Back to our problem

Given a **undirected** graph $G = (V, E)$, we want a **matching** $M$, which means:

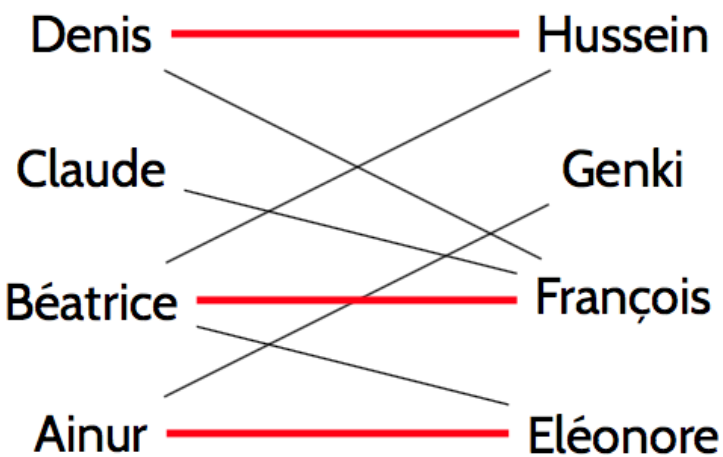
- A subset of edges $M \subset E$



# Back to our problem

Given a **undirected** graph $G = (V, E)$, we want a **matching**, which means:

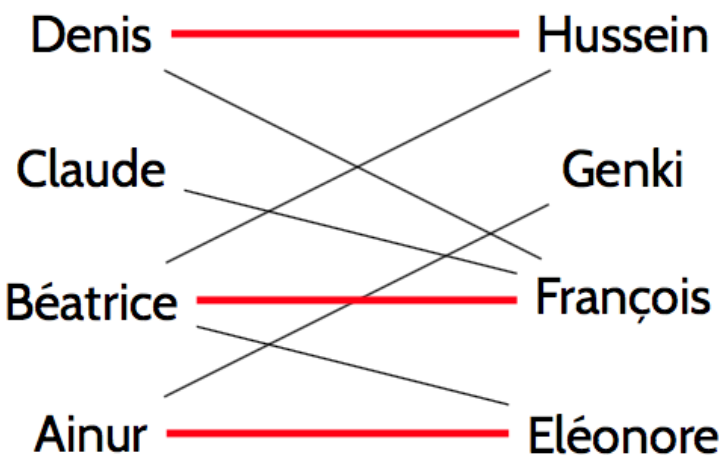
- A subset of edges $M \subset E$
- Such that no pairs of edges of $M$ are incident
- Equivalently, each node in the graph is **at most** in one edge of $M$.



# Back to our problem

Given **undirected** a graph $G = (V, E)$, we want a **matching**, which means:

- A subset of edges $M \subset E$
- Equivalently, each node in the graph is **at most** in one edge of $M$.
- No pairs of edges of $M$ are incident



# Maximum matching

- The **size** of a matching is the number of edges it contains.

# Maximum matching

- The **size** of a matching is the number of edges it contains.
- We want to find the matching of maximum size in a given graph.

# Example 1

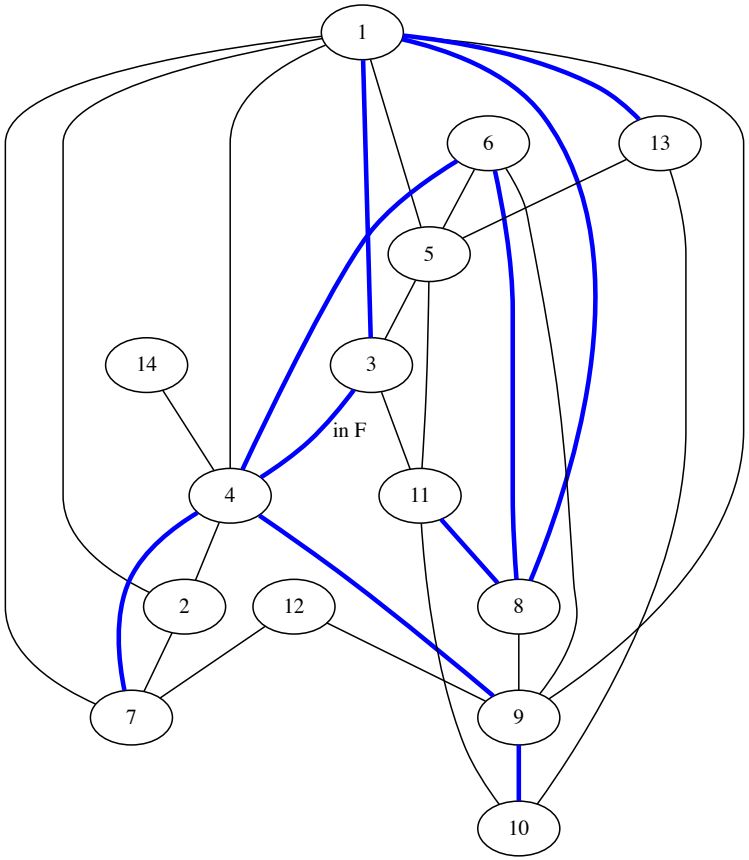



Figure: Is this a matching ?

# Example 2

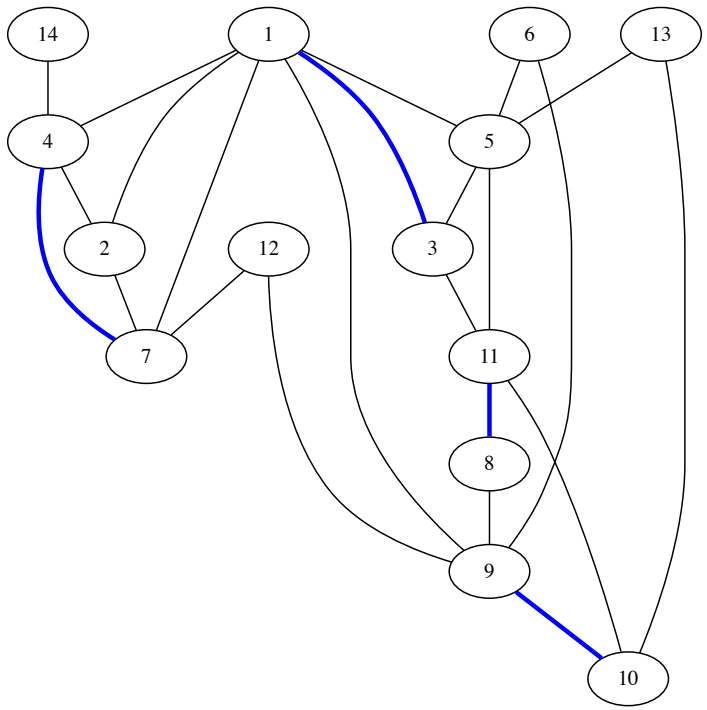



Figure: Is this a matching ?

# Example 3

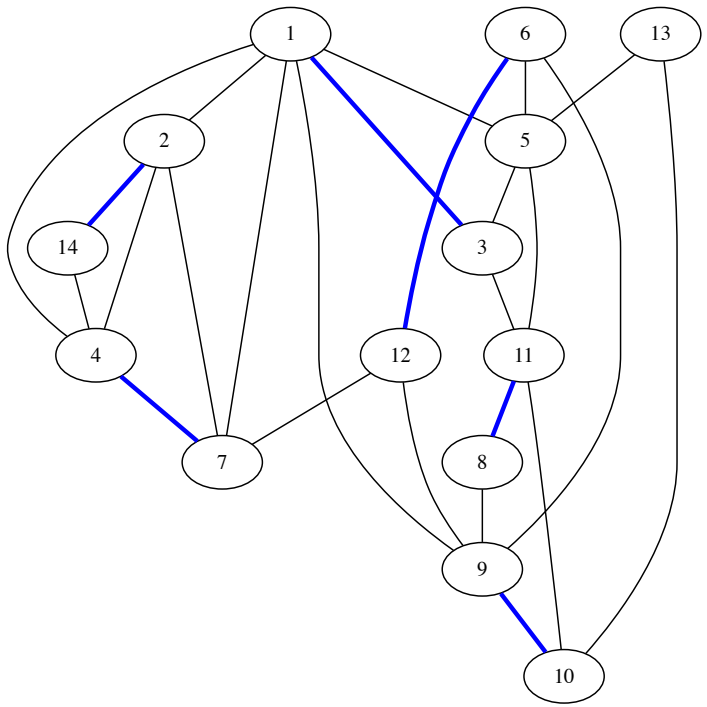



Figure: Is this an optimal matching ?

# Example 4

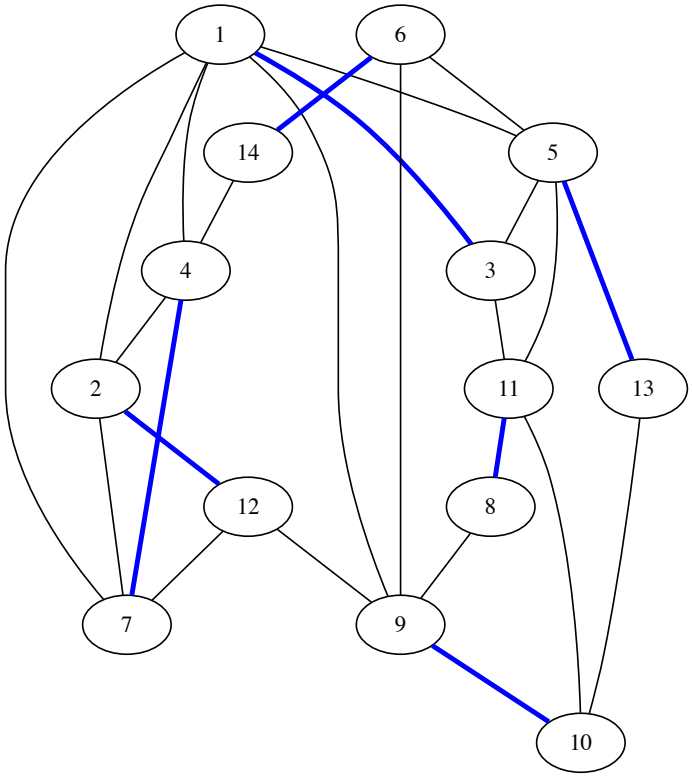



Figure: Is this an optimal matching ?

# Example 5

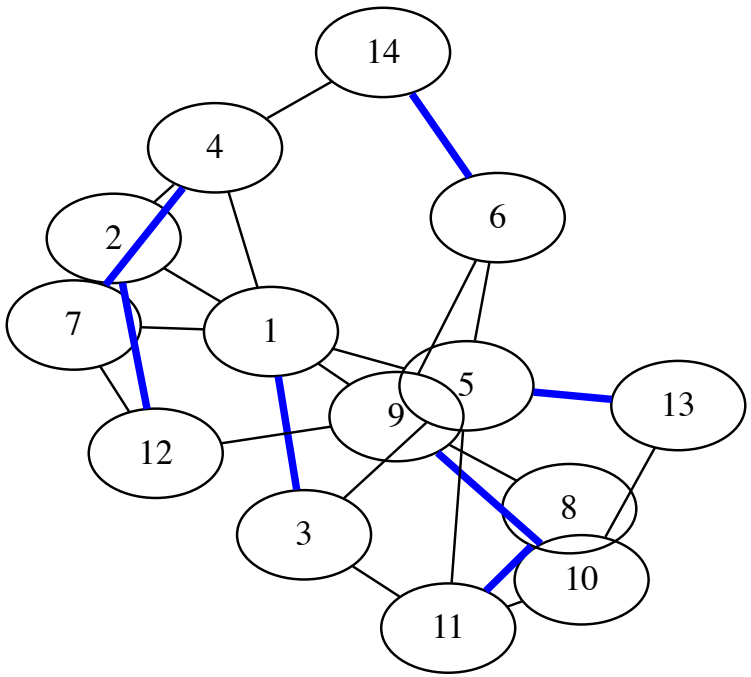



Figure: With neato

# Optimal matching

Exercice 1 : Given a graph of size $n$, what is maximum size possible for a **matching** ?

# Optimal matching

Exercice 1 : Given a graph of size $n$, what is maximum size possible for a **matching ?**

- If $n$ is even : $\frac{n}{2}$
- Else $n$ is odd : $\frac{n-1}{2}$

# Optimal matching

Exercice 1 : Given a graph of size $n$, what is maximum size possible for a **matching ?**

- If $n$ is even : $\frac{n}{2}$
- Else $n$ is odd : $\frac{n-1}{2}$

Hence,

$$\lfloor \frac{n}{2} \rfloor \tag{5}$$

# Optimal matching

Exercice 1 : Can you think of a graph with $n$ nodes that contains a matching of size $\frac{n}{2}$ ? (assuming $n$ is even)

# Optimal

Exercice 1 : Can you think of a graph with $n$ nodes that contains a matching of size $\frac{n}{2}$ ? (assuming $n$ is even)

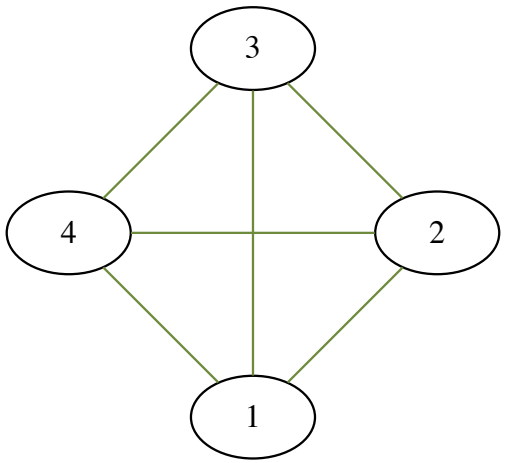


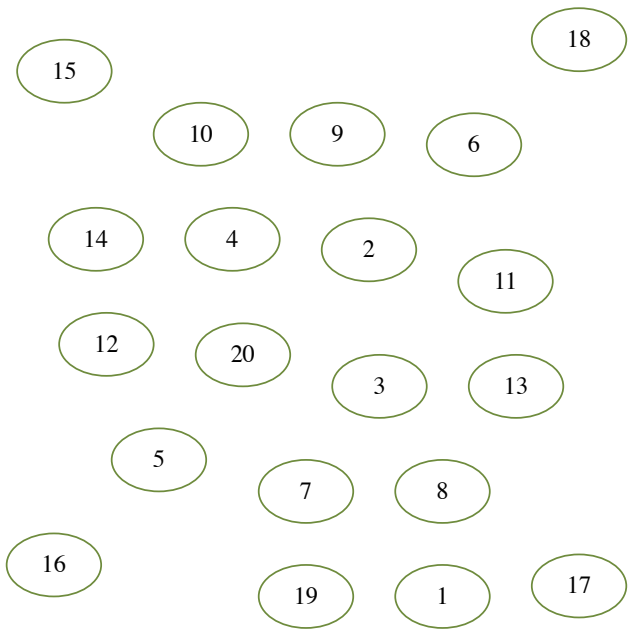

Figure: The complete graph works

# Optimal matching

Exercice 1 : Can you think of a graph with $n$ nodes that does **not** contains a matching of size $\frac{n}{2}$ ? (assuming $n$ is even)

# Optimal matching

Exercice 1 : Can you think of a graph with $n$ nodes that does **not** contains a matching of size $\frac{n}{2}$ ? (assuming $n$ is even)



# Optimal matching

Exercice 1 : Can you think of a **non trivial** graph that does **not** contains a matching of size $\frac{n}{2}$ ? (assuming $n$ is even)

# Optimal matching

Exercice 2 : Can you think of a **non trivial** graph that does **not** contains a matching of size $\frac{n}{2}$ ? (assuming $n$ is even)

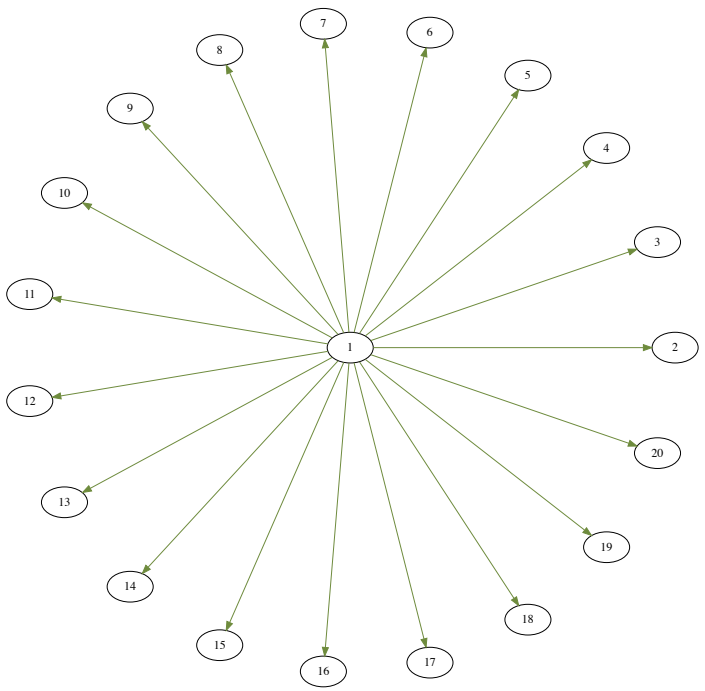


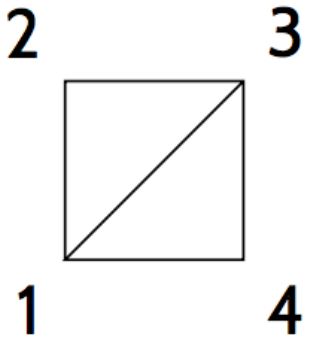

Figure: Star graph

# Experiments

Possibilities to code a graph:

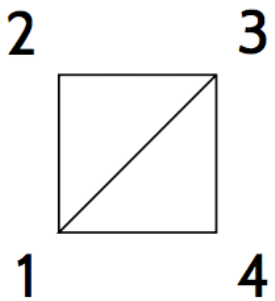
- list of sets of size 2 (for an undirected graph)
- a dictionary of successors (directed of undirected)

# Coding a graph : as a list



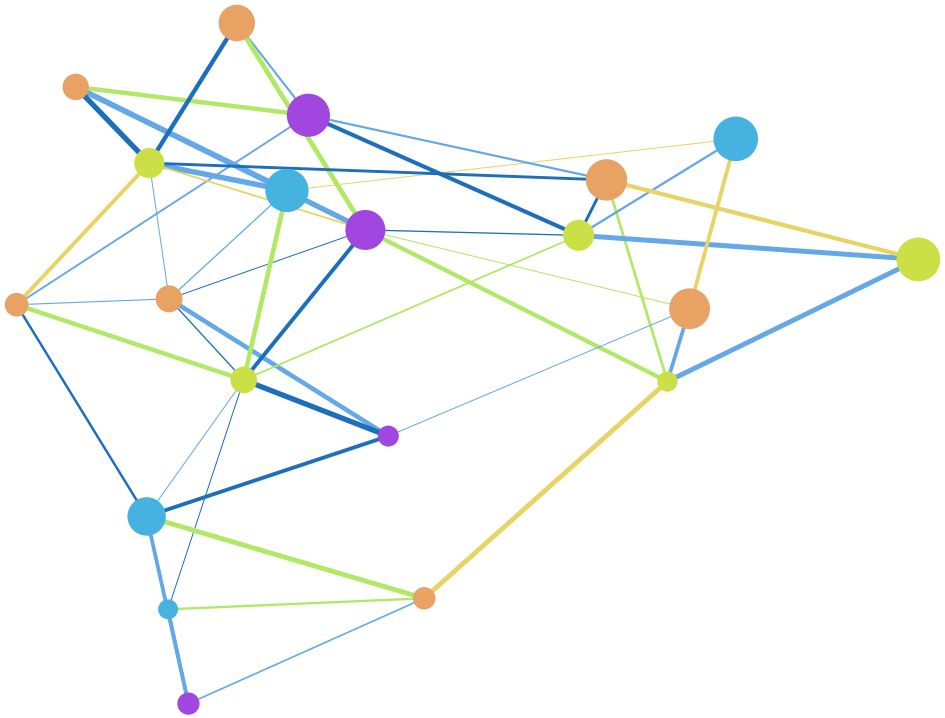

```
g1 = [{1,2},{1,3},{2,3},{3,4},{1,4}]
```

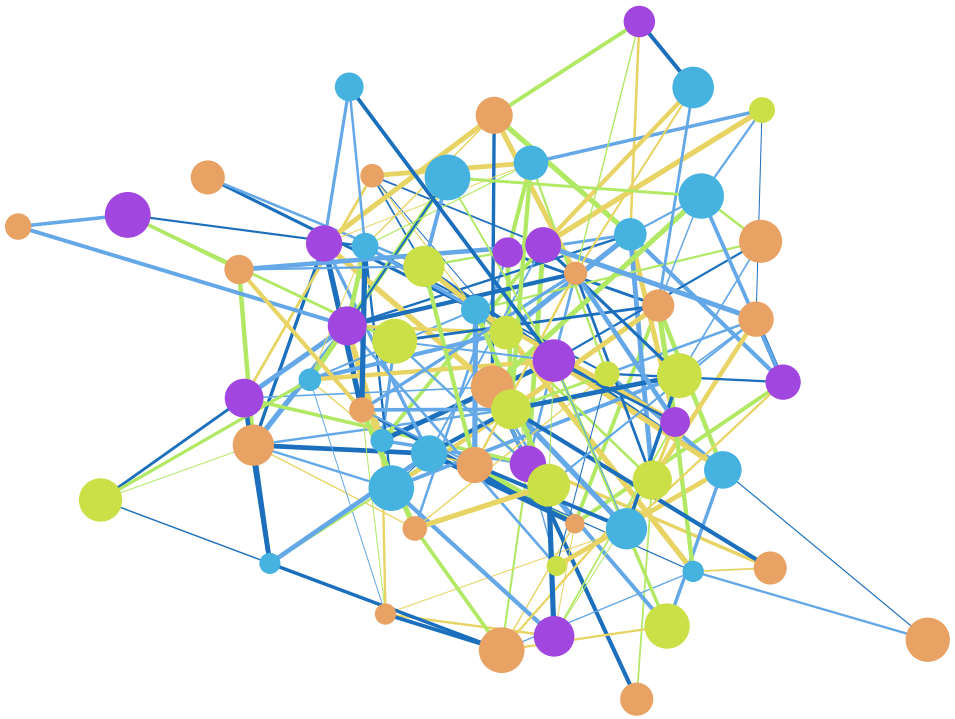
# Coding a graph : as a dictionary



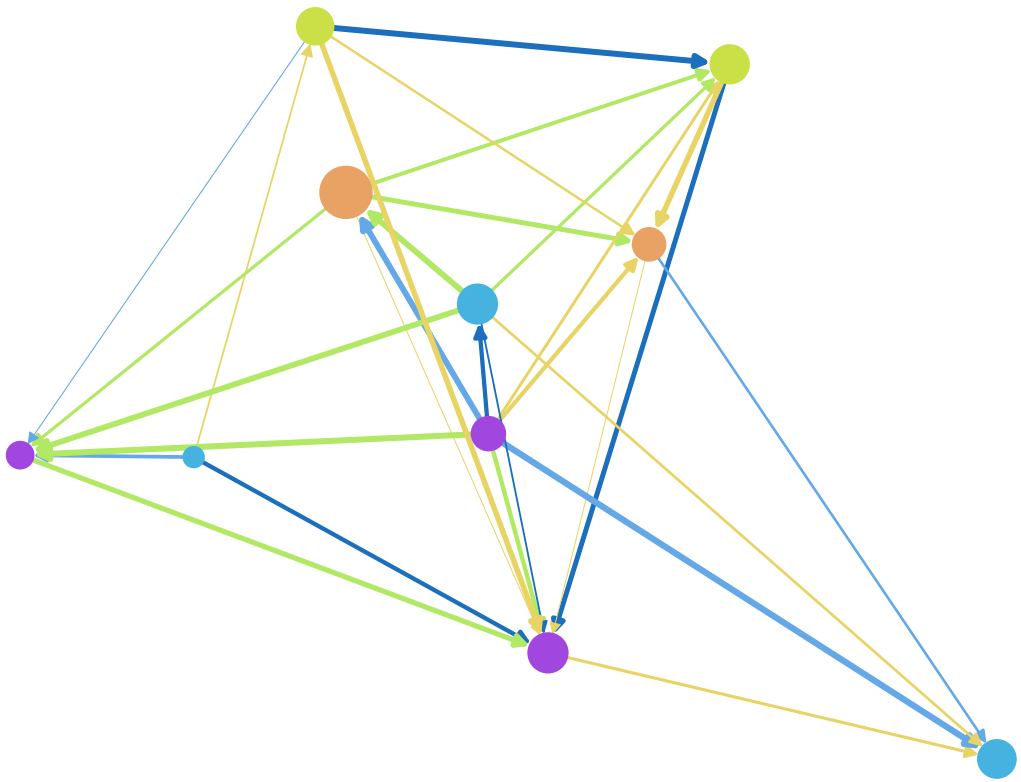

```
g1 = { 1:{2,3,4}, 2:{1,3}, 3:{1,2,4}, 4:{1,3} }
```

# Generating graphs with networks.

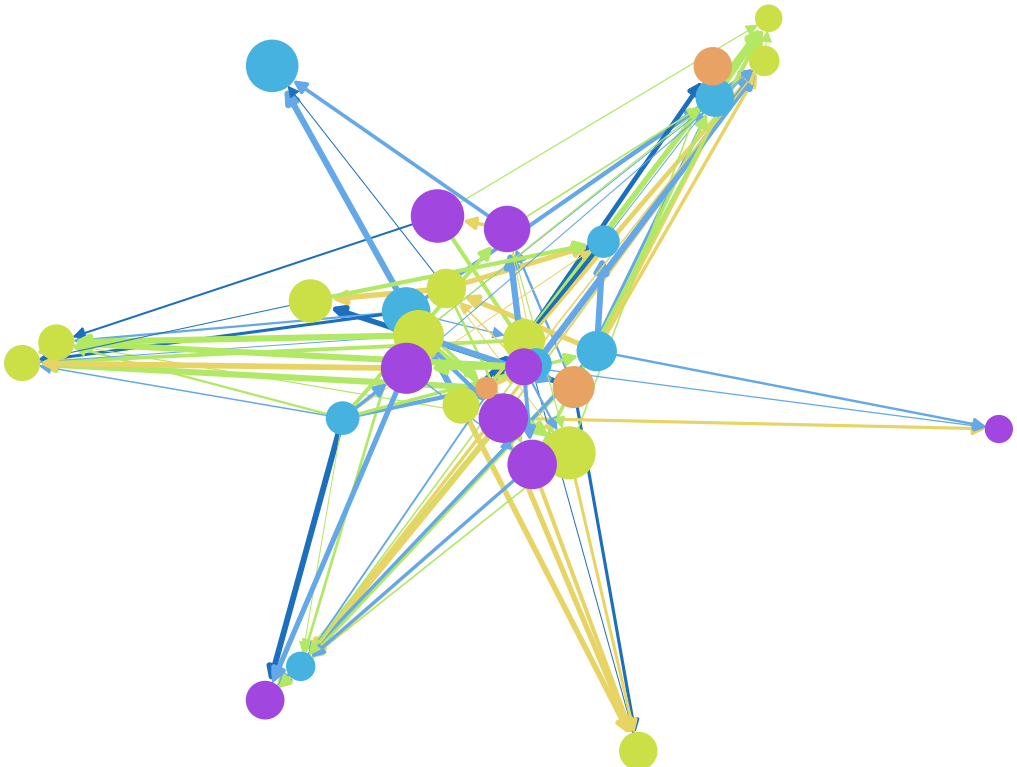




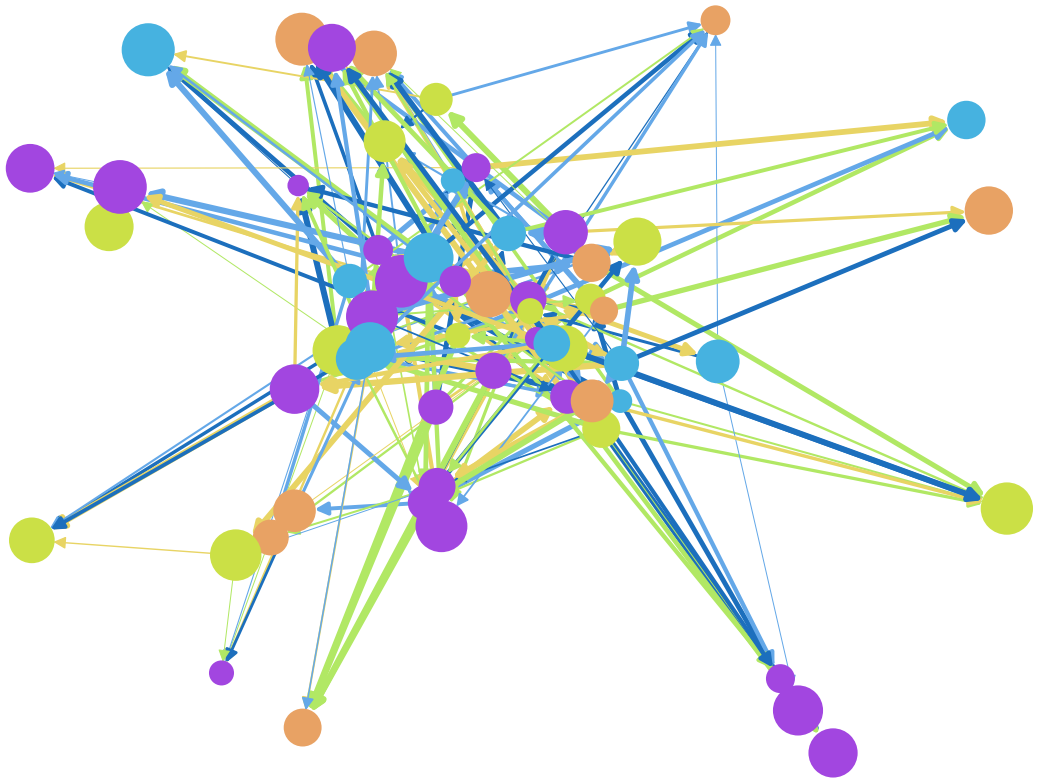
# Directed graph



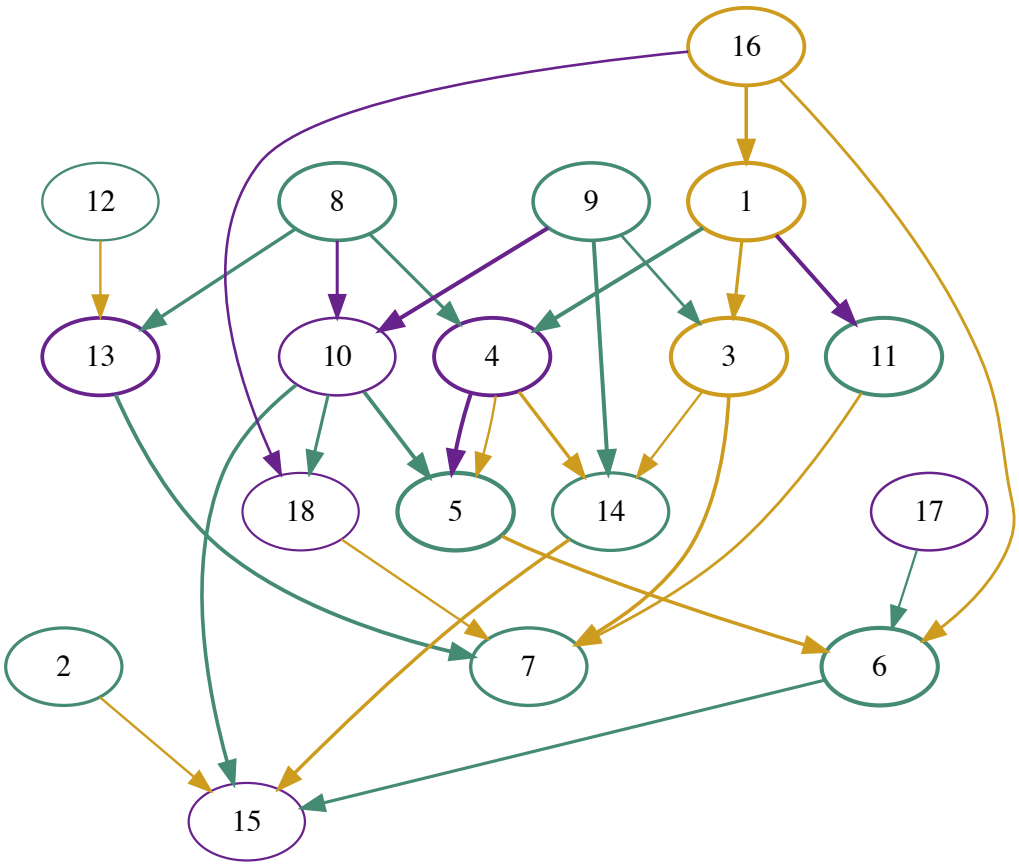
# Directed graph II



# Directed graph III



# Example directed graph



# Manual matching

Exercice 3 : Please manually find an **optimal matching** in your **undirected** graph.

# Big graph

We could not manually find an optimal matching in this graph :

# Summary

- We have defined the matching problem.
- When the size of the problem is large, we can not manually find an optimal matching.

# Brute force approach

Exercice 4 : Enumeration

- Given a graph, what would a brute force approach on the matching problem be ?

# Brute force approach

Exercice 4 : Exhaustive search

- Given a graph what would a brute force approach on the matching problem be ?
  - 1) Enumerate all possible subsets in the set of the edges.
  - 2) Check if each subset is a matching.
  - 3) Return the biggest one obtained.

# Brute force approach

Exercice 4 : Exhaustive search

- Given a graph what would a brute force approach on the matching problem be ?
  - 1) Enumerate all possible subsets in the set of the edges.
  - 2) Check if each subset is a matching.
  - 3) Return the biggest one obtained.

If the graph contains $n$ nodes, and given a subset of edges, what if the number of computations needed to perform step 2 ?

# Brute force approach

Exercice 4 : Exhaustive search

- Given a graph what would a brute force approach on the matching problem be ?
  - 1) Enumerate all possible subsets in the set of the edges.
  - 2) Check if each subset is a matching.
  - 3) Return the biggest one obtained.

If the graph contains $n$ nodes, and given a subset of edges, what if the number of computations needed to perform step 2 ?
You can give a rough approximation.

# Brute force approach

Exercice 4 : Exhaustive search

- Given a graph what would a brute force approach on the matching problem be ?
  - 1) Enumerate all possible subsets in the set of the edges.
  - 2) Check if each subset is a matching.
  - 3) Return the biggest one obtained.

If the graph contains $n$ nodes, and given a subset of edges, what if the number of computations needed to perform step 2 ?
It is a **polynomial** number of computations : so it is ok.

# Brute force search

Exercice 5 : Complexity of brute force

- 1) Enumerate all possible subsets in the set of the edges.
- 2) Check if each subset is a matching.
- 3) Return the biggest one obtained.

What is the complexity of step 1 ?

# Brute force search

Exercice 5 : Complexity of brute force

- 1) Enumerate all possible subsets in the set of the edges.
- 2) Check if each subset is a matching.
- 3) Return the biggest one obtained.

What is the complexity of step 1 ?

The number of subsets is $2^{\frac{n(n-1)}{2}}$ (in the worst case), which is exponential. If $p$ is the numberof edges, we can also write it as $2^p$.

# Brute force search

Exercice 5 : **Complexity of brute force**

Assume that checking a subset requires 1 microsecond. How long should we wait in order to check all possible matchings in a graph with 100 nodes ?

# Summary II

- For the matching problem on a large graph, we can neither
  - manually find an optimal matching
  - perform the exhaustive search (brute force algorithm)

# Algorithms

- Hence, we need different algorithms to solve the problem.
- Let us first introduce some theoretical notions.

# Notion of maximal and maximum matching

We will say that a matching $M$ of cardinality (number of elements) $|M|$ is:

- **Maximum** if is has the maximum possible number of edges (is is thus optimal)

# Notion of maximal and maximum matching

We will say that a matching $M$ of cardinality $|M|$ is:

- **Maximum** if it has the maximum possible number of edges (is is thus optimal)
- **Maximal** if the set of edges obtained by adding any edge to it is **not a matching**. This means that $M \cup \{e\}$ is not a matching for any $e$ not in M.
- $\cup$ means union of sets.

Is being a **maximal** matching the same thing as beeing a **maximum** matching ?

# Maximum implies maximal

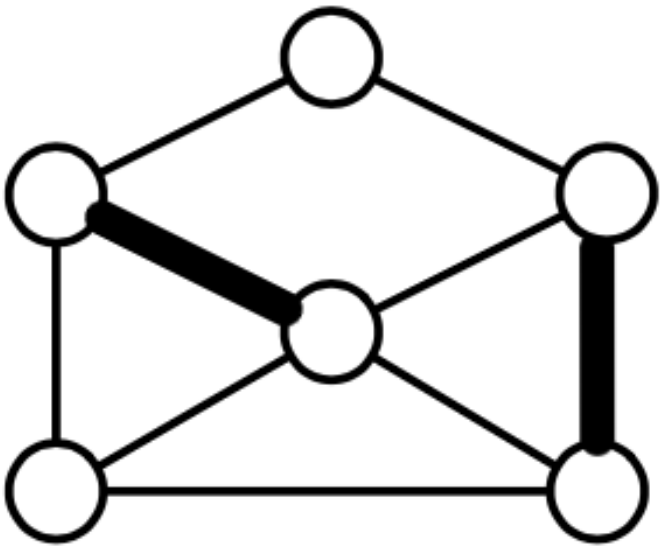
Let us show that a maximum matching is maximal.

# Counter Example

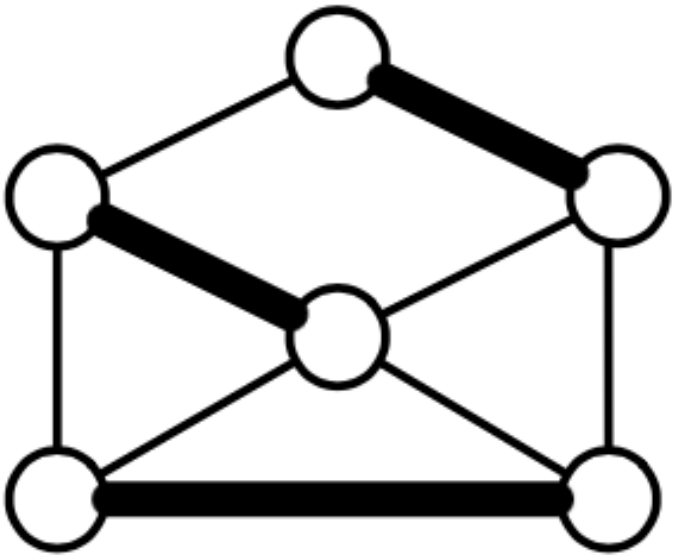
However, a matching that is maximal is **not necessary Maximum**.

# Counter Example

However, a matching that is maximal is **not necessary Maximum**.
Can you find an example ?



(a) A maximal matching not maximum



(b) A maximum matching

# Greedy algorithm

Can you propose a greedy algorithm to address the maximum matching problem ?

# Greedy algorithm

**Result :** Matching M
$M \leftarrow \emptyset$;
**for** $e \in E$ **do**
  **if** $M \cup \{e\}$ *is a matching* **then**
    $M \leftarrow M \cup \{e\}$
  **end**
**end**
return $M$

**Algorithme 0 :** Greedy algorithm to find a matching

# Greedy algorithm

- What is the type of matching algorithm returned by this algorithm ?
- What is the complexity of this algorithm ? (as a function of the number of nodes $n$ of the graph)

# Access times

`https://wiki.python.org/moin/TimeComplexity`

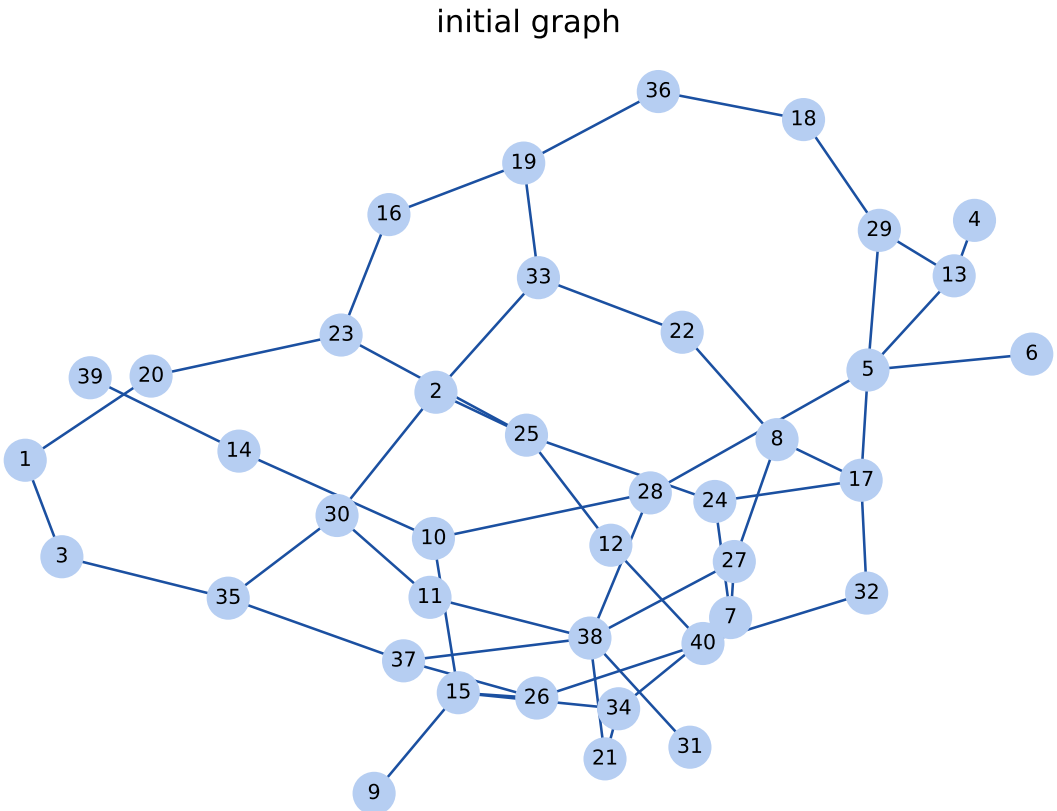
# Greedy algorithm

- The greedy algorithm returns a **maximal** matching (proof)
- Its complexity is smaller than $\mathcal{O}(np)$ ( $n$ nodes, $p$ edges) (proof)
- smaller than **cubic** in the number of nodes : $\mathcal{O}(n^3)$

# Greedy algorithm

- We will implement the greedy algorithm to find a maximal matching.

Exercice 6 : **cd matching_greedy/** and use **generate_graph.py** to build a graph with a least 30 nodes. The images are stored in **images/**, data stored in **data/**

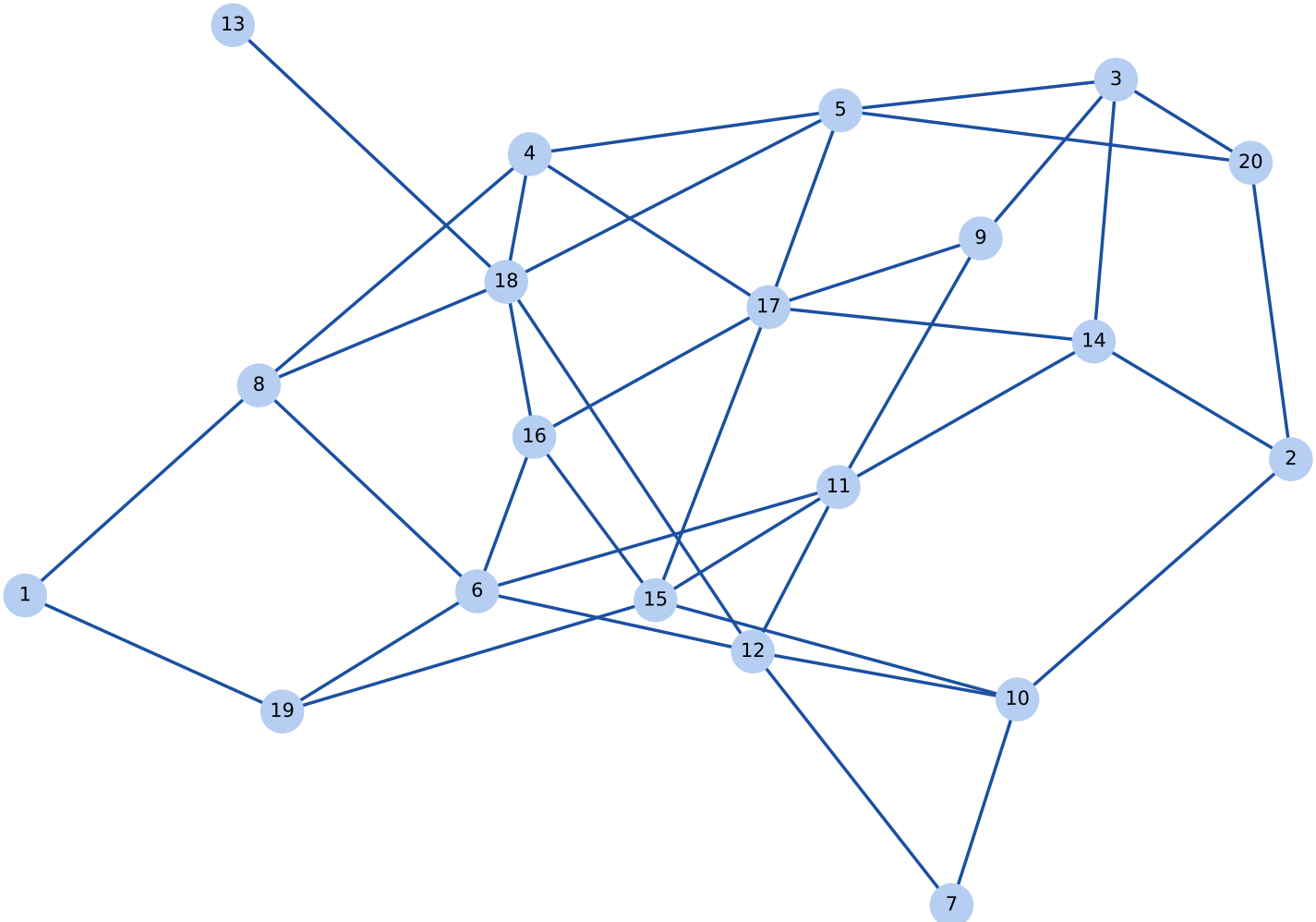


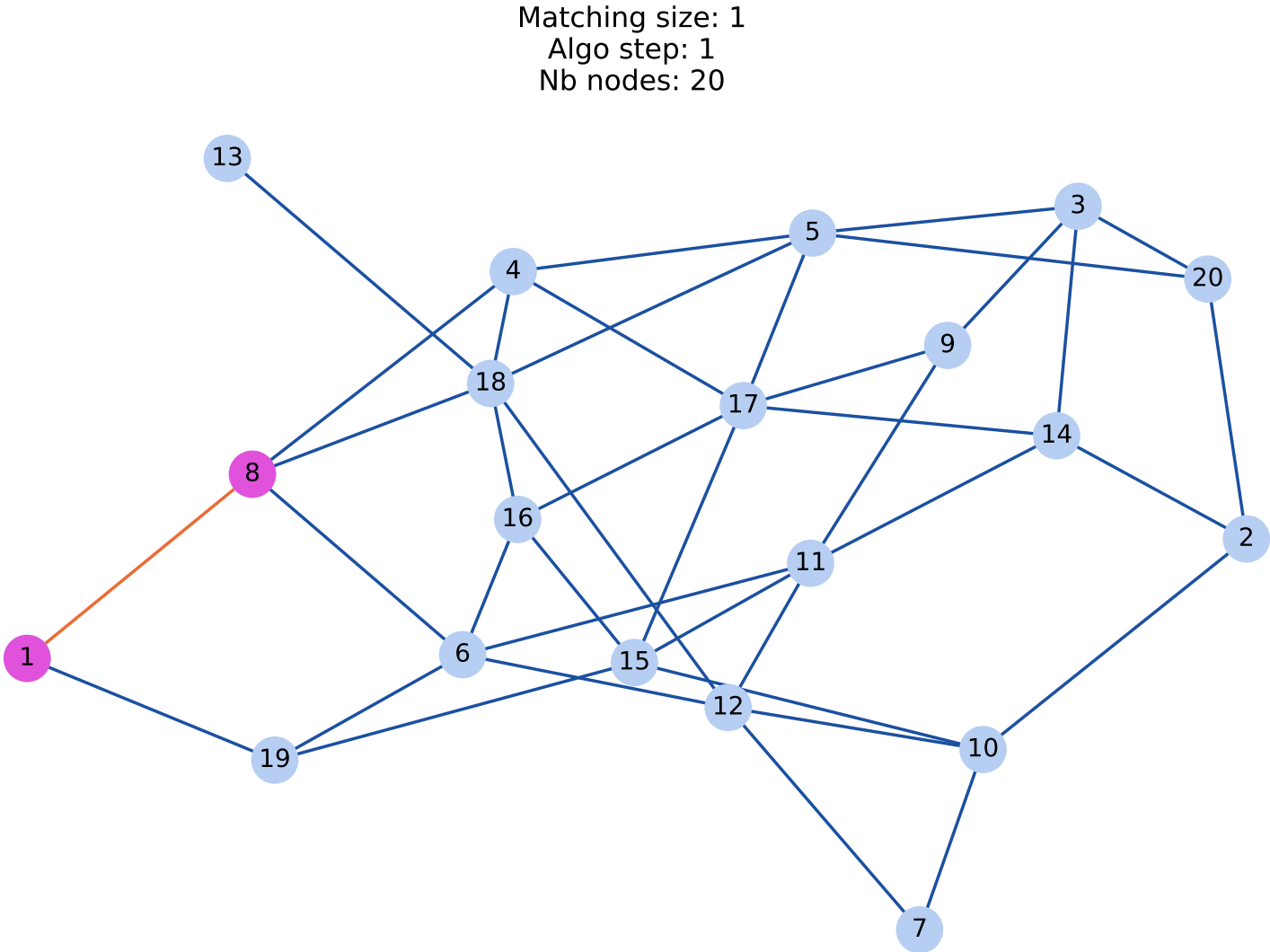
# Implementing the greedy algorithm

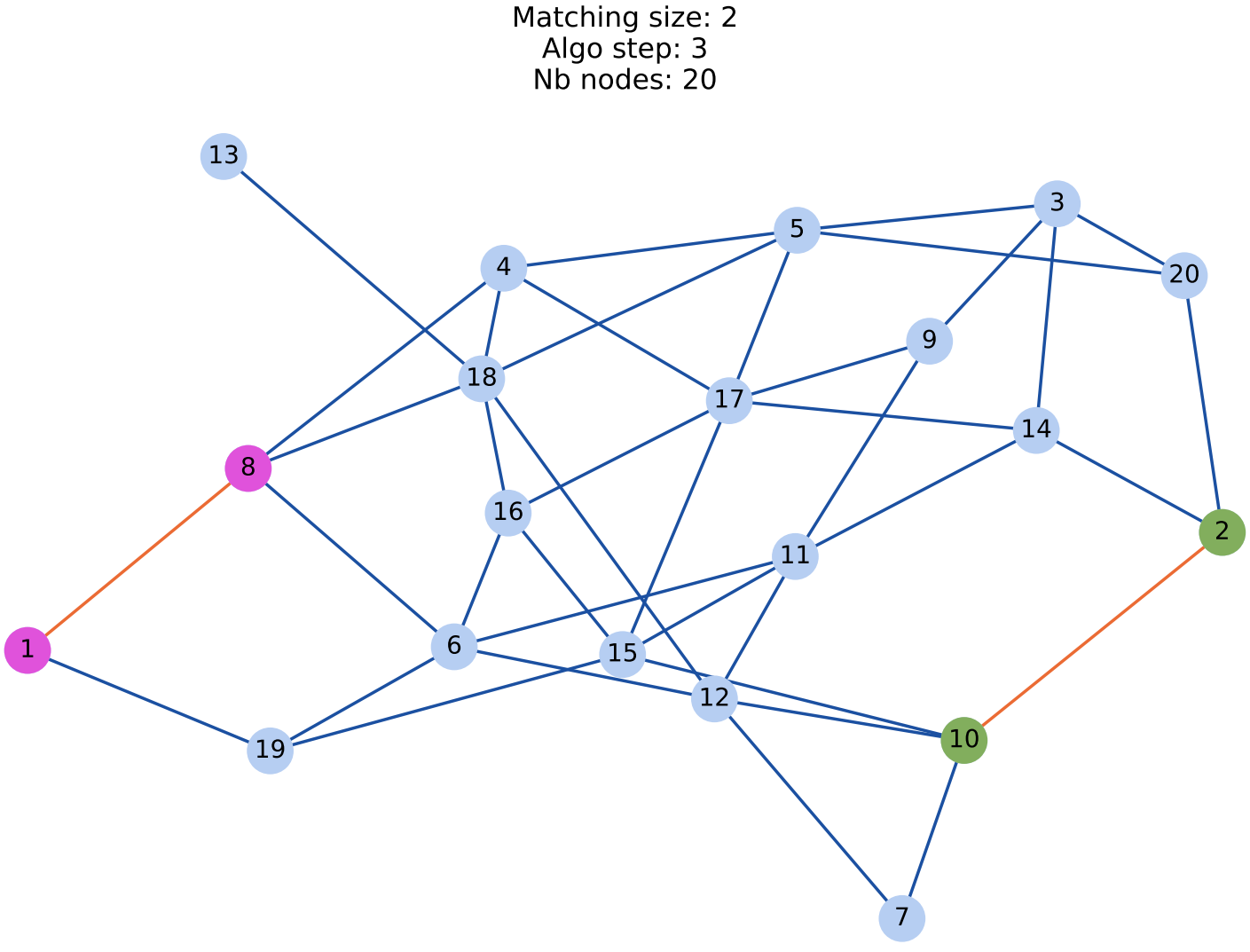
Exercice 6 : Implement the greedy algorithm on this graph.

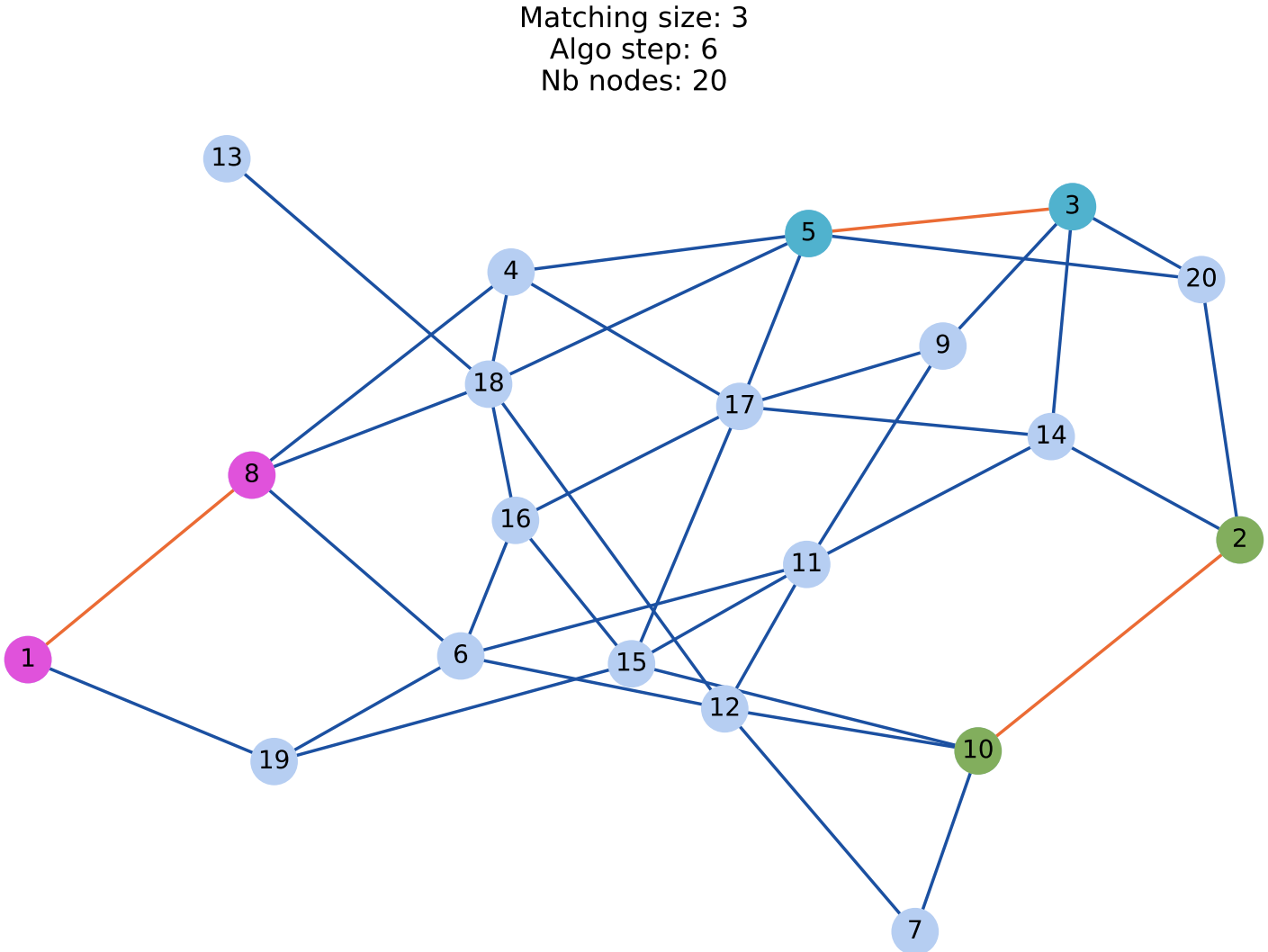
- Use the functions in **matching_functions.py** and call them from **apply_matching_algorithm.py**
- More details in the file.

initial graph

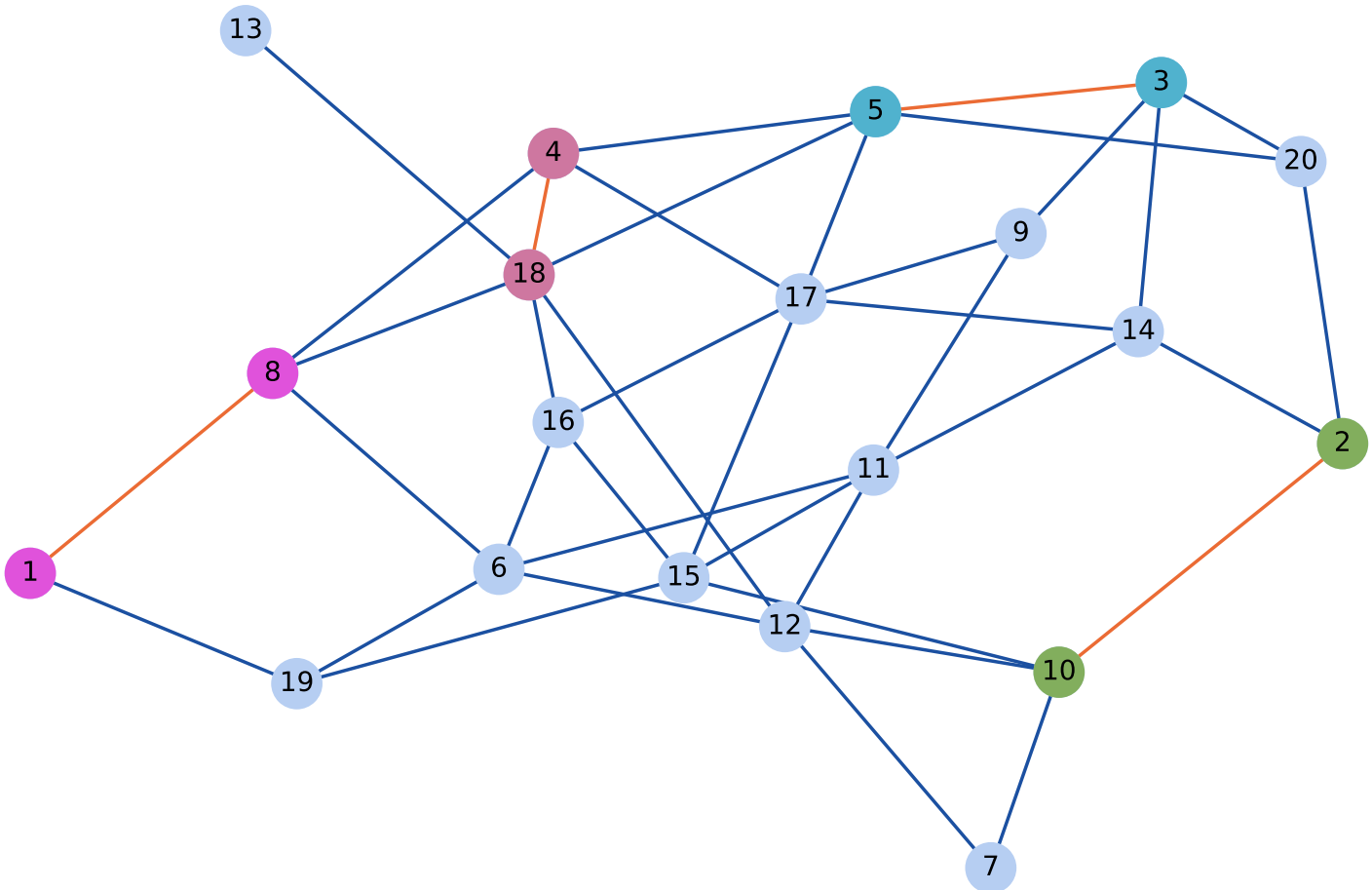


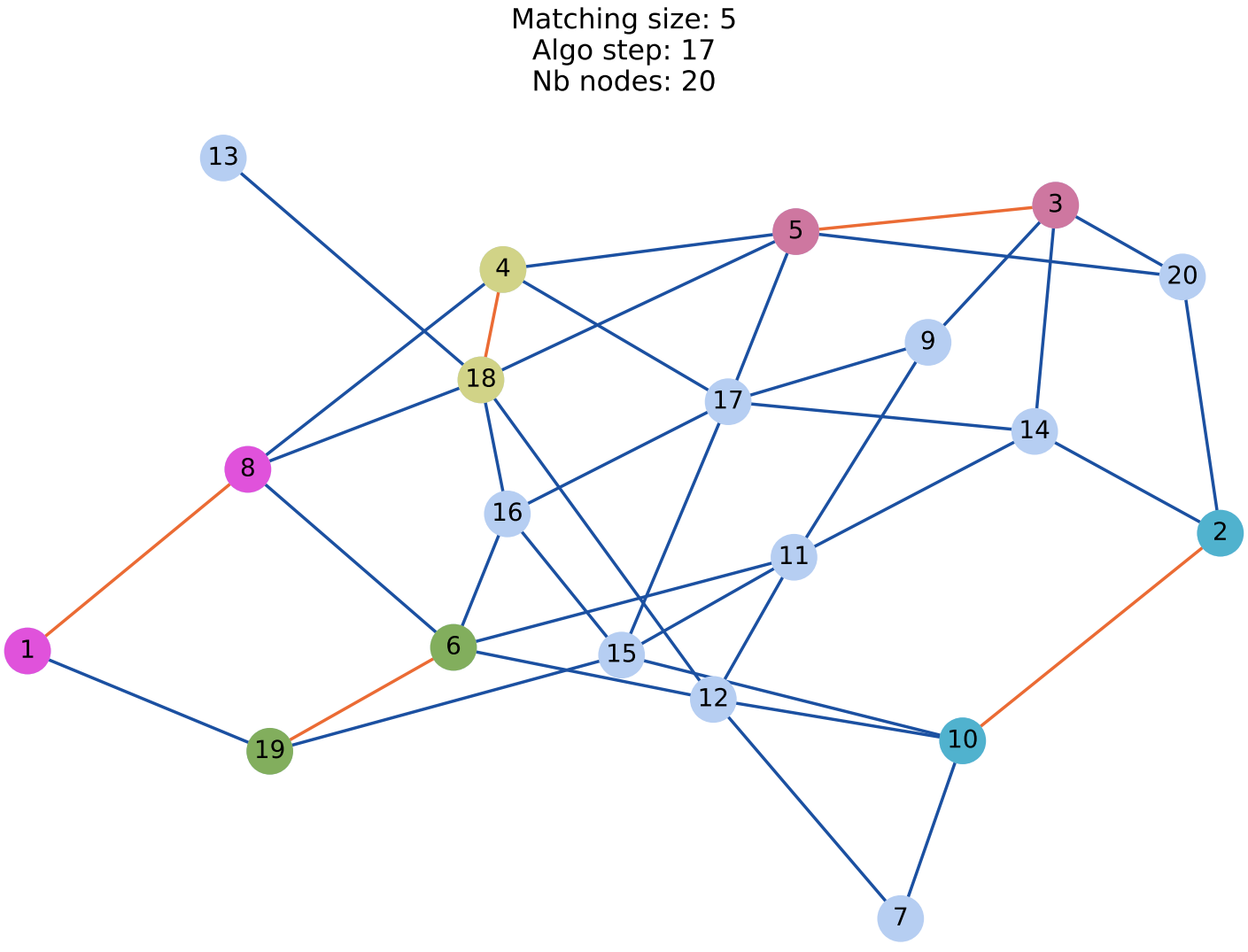

Matching size: 1
Algo step: 1
Nb nodes: 20
13
3
5
4
20
9
18
17
14
8
16
2
11
1
6
15
12
10
19
7


Matching size: 2
Algo step: 3
Nb nodes: 20
13
3
5
4
20
9
18
17
14
8
16
2
11
1
6
15
12
10
19
7

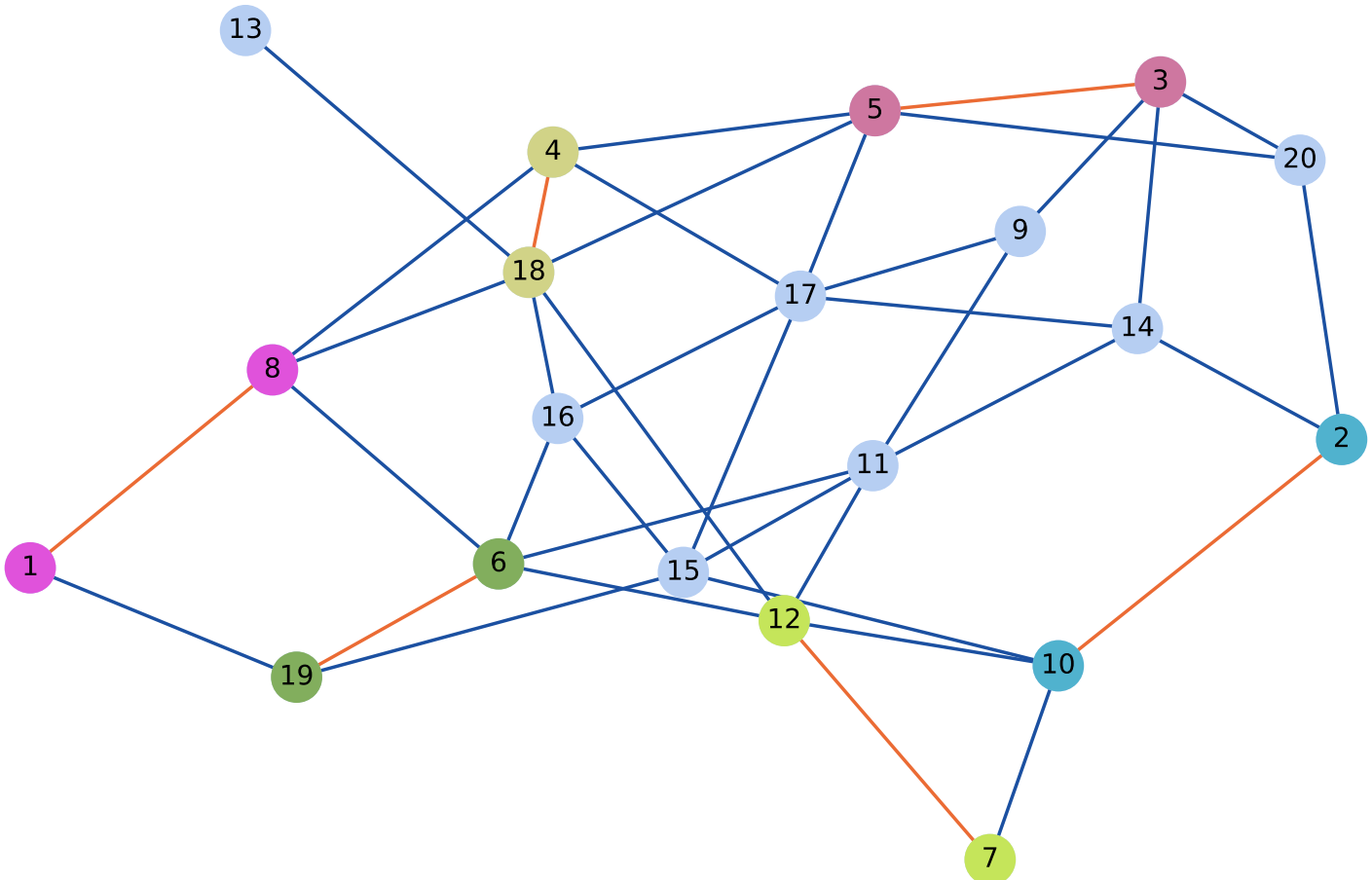

Matching size: 3
Algo step: 6
Nb nodes: 20
13
3
5
4
20
9
18
17
14
8
16
2
11
1
6
15
12
10
19
7

Matching size: 4
Algo step: 11
Nb nodes: 20

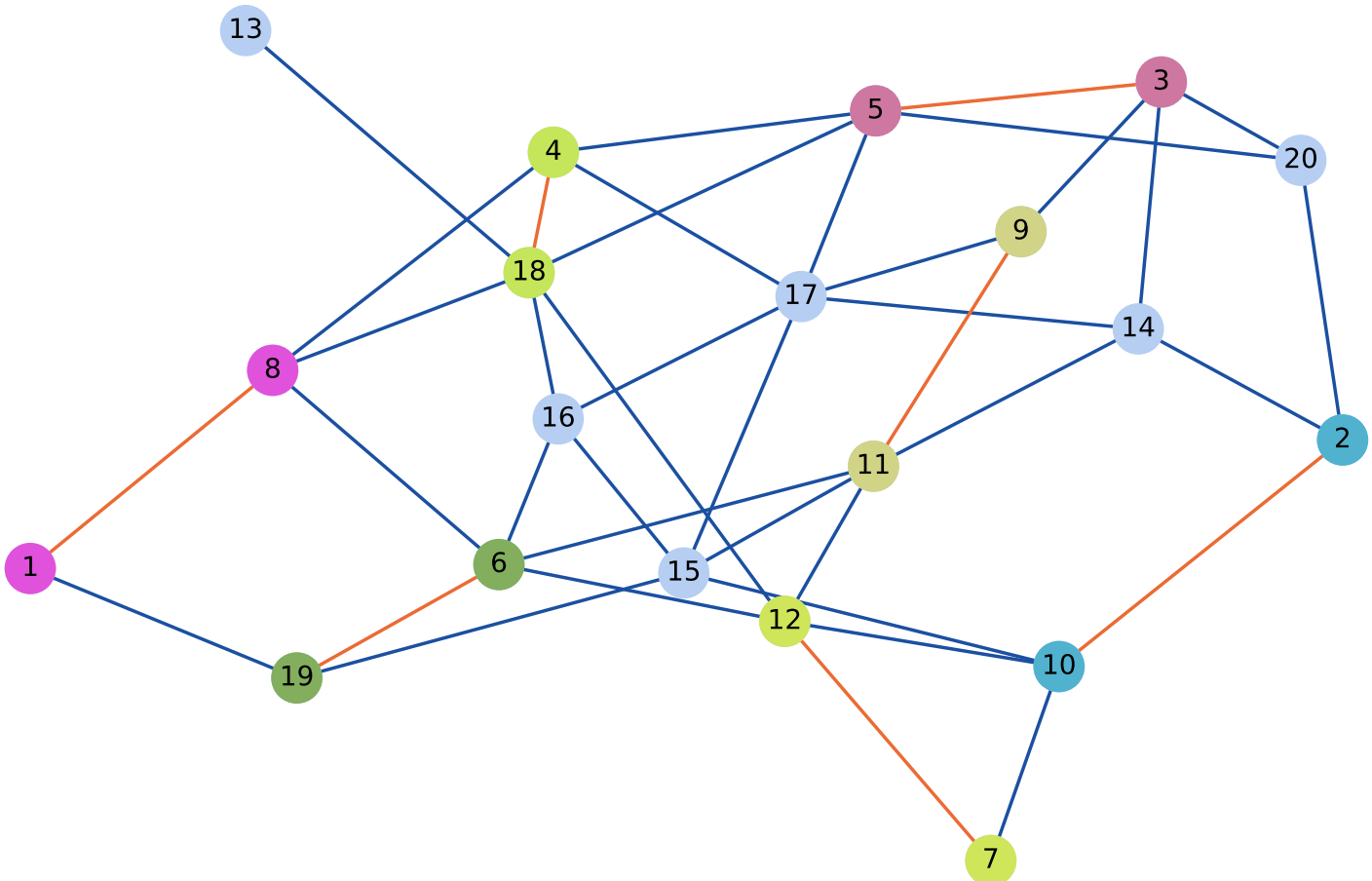



Matching size: 5
Algo step: 17
Nb nodes: 20
13
3
5
4
20
9
18
17
14
8
16
2
11
1
6
15
12
10
19
7

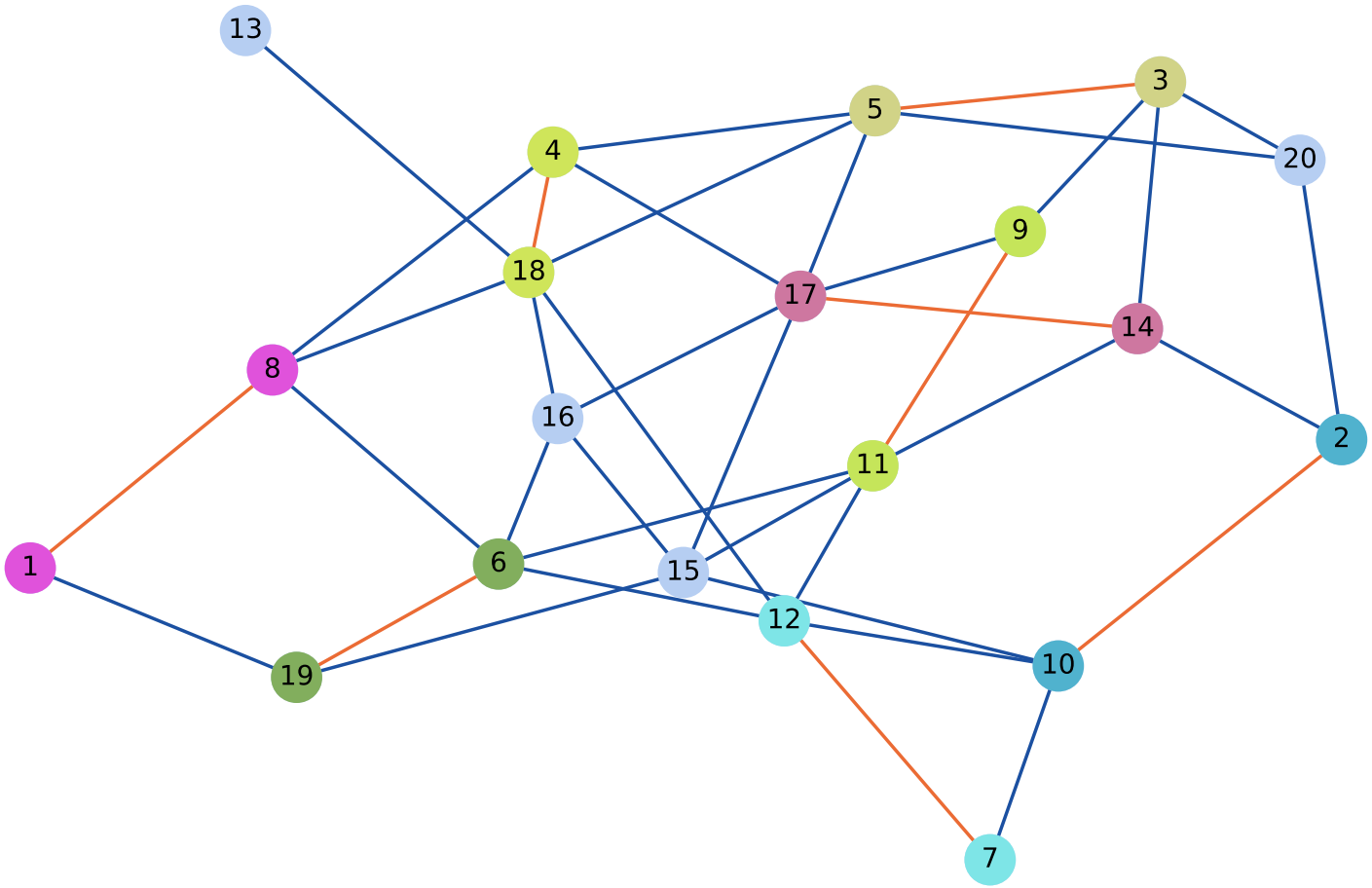
Matching size: 6
Algo step: 22
Nb nodes: 20



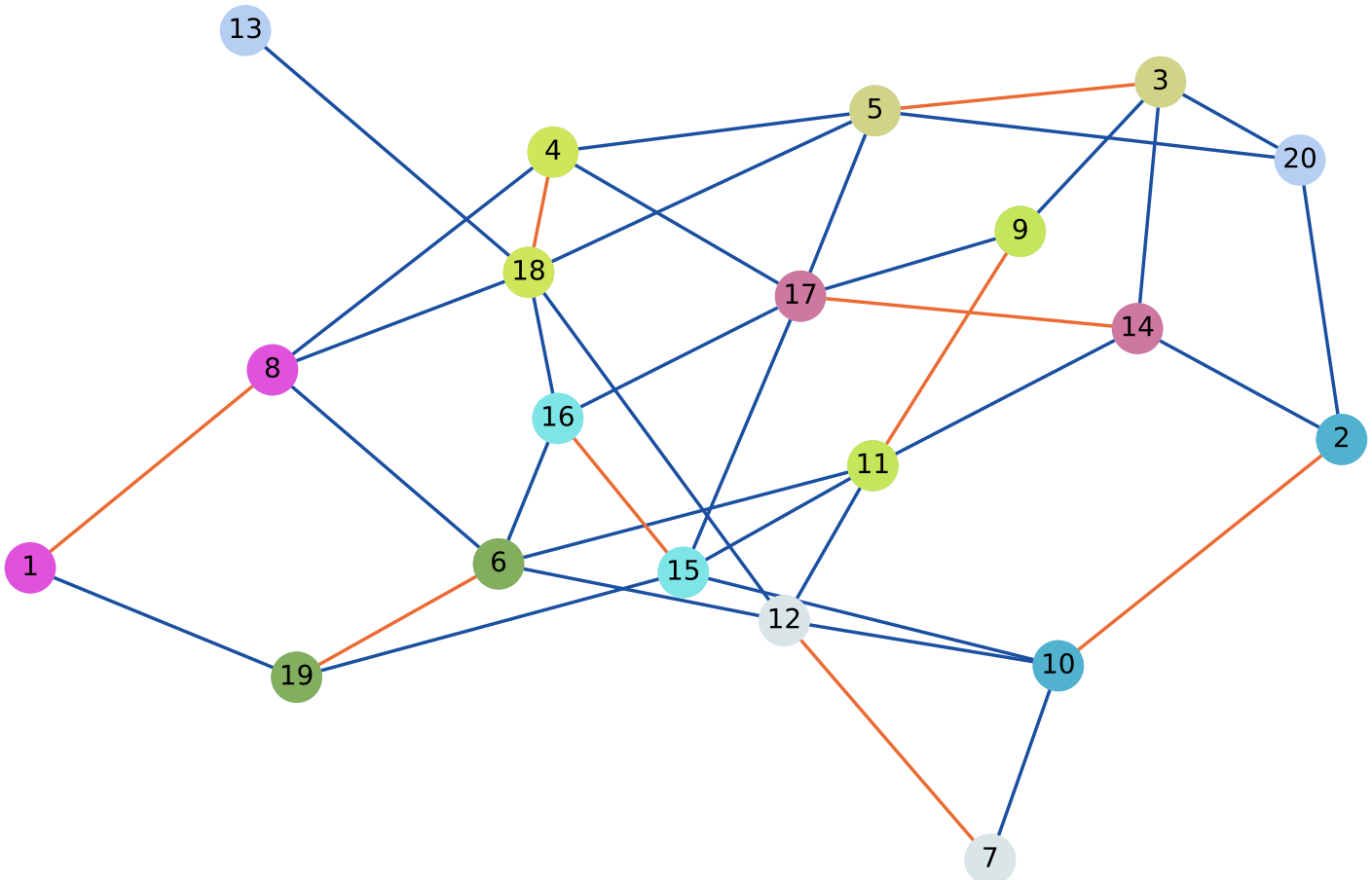
Matching size: 7
Algo step: 25
Nb nodes: 20



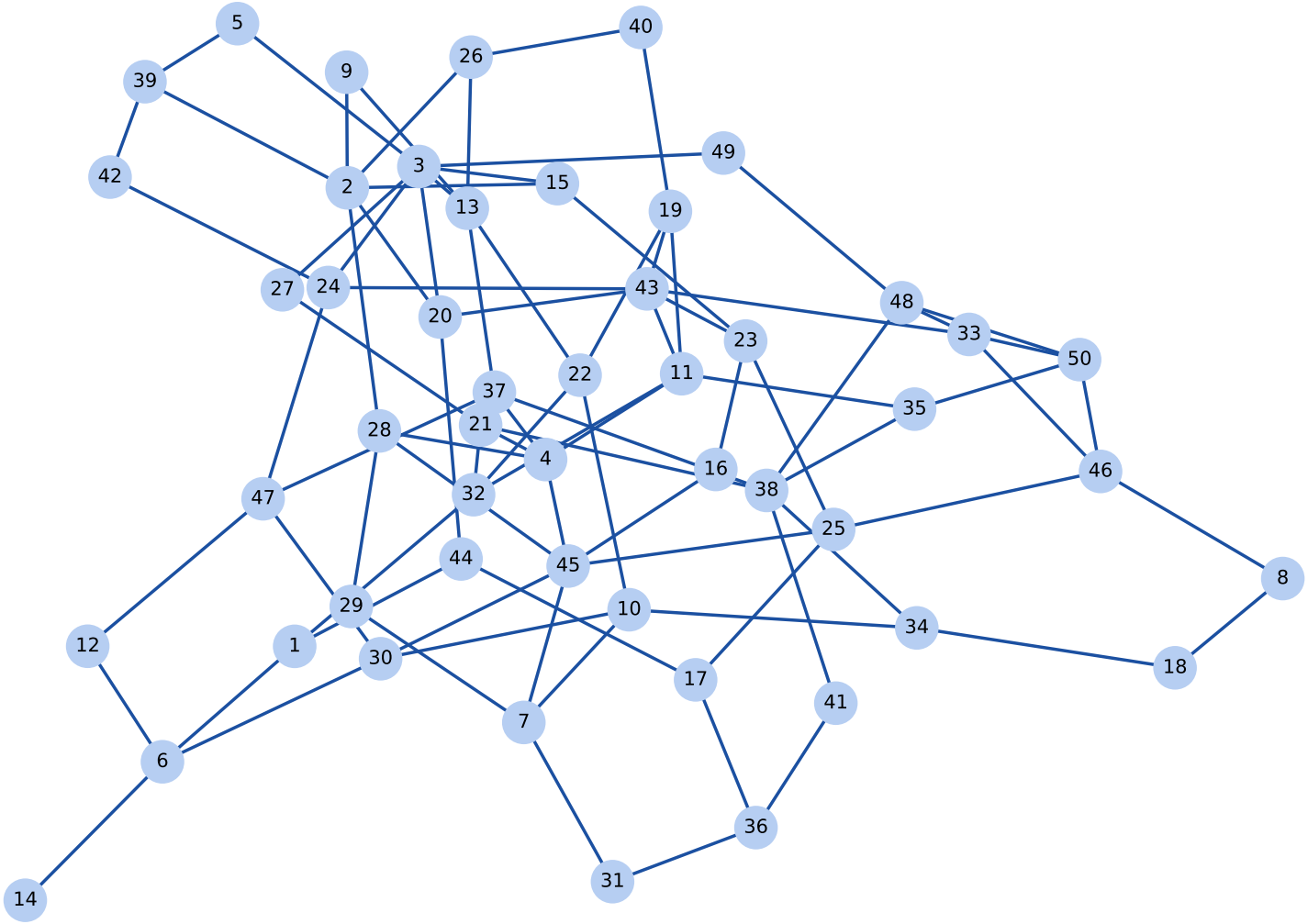
Matching size: 8
Algo step: 34
Nb nodes: 20

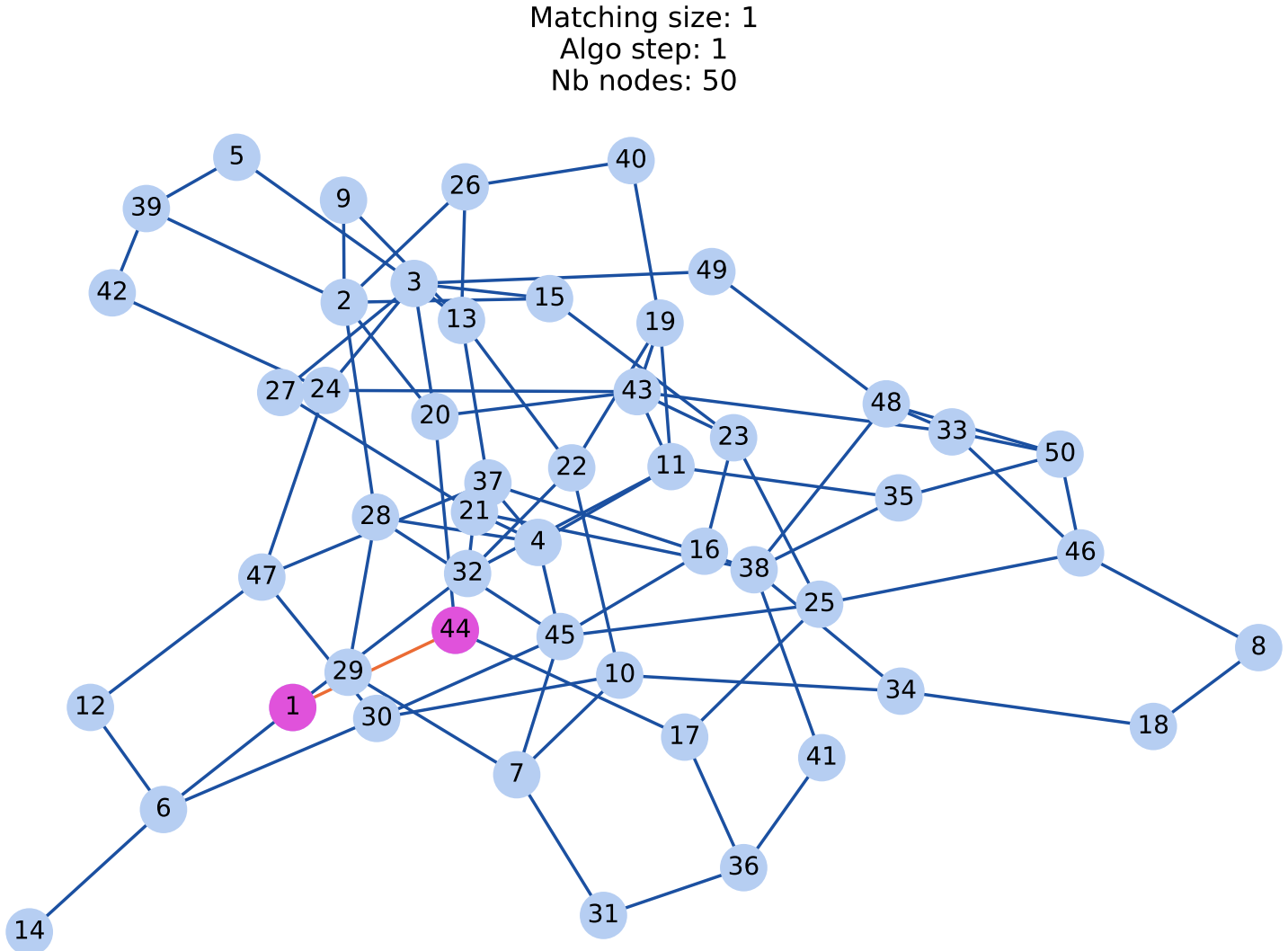


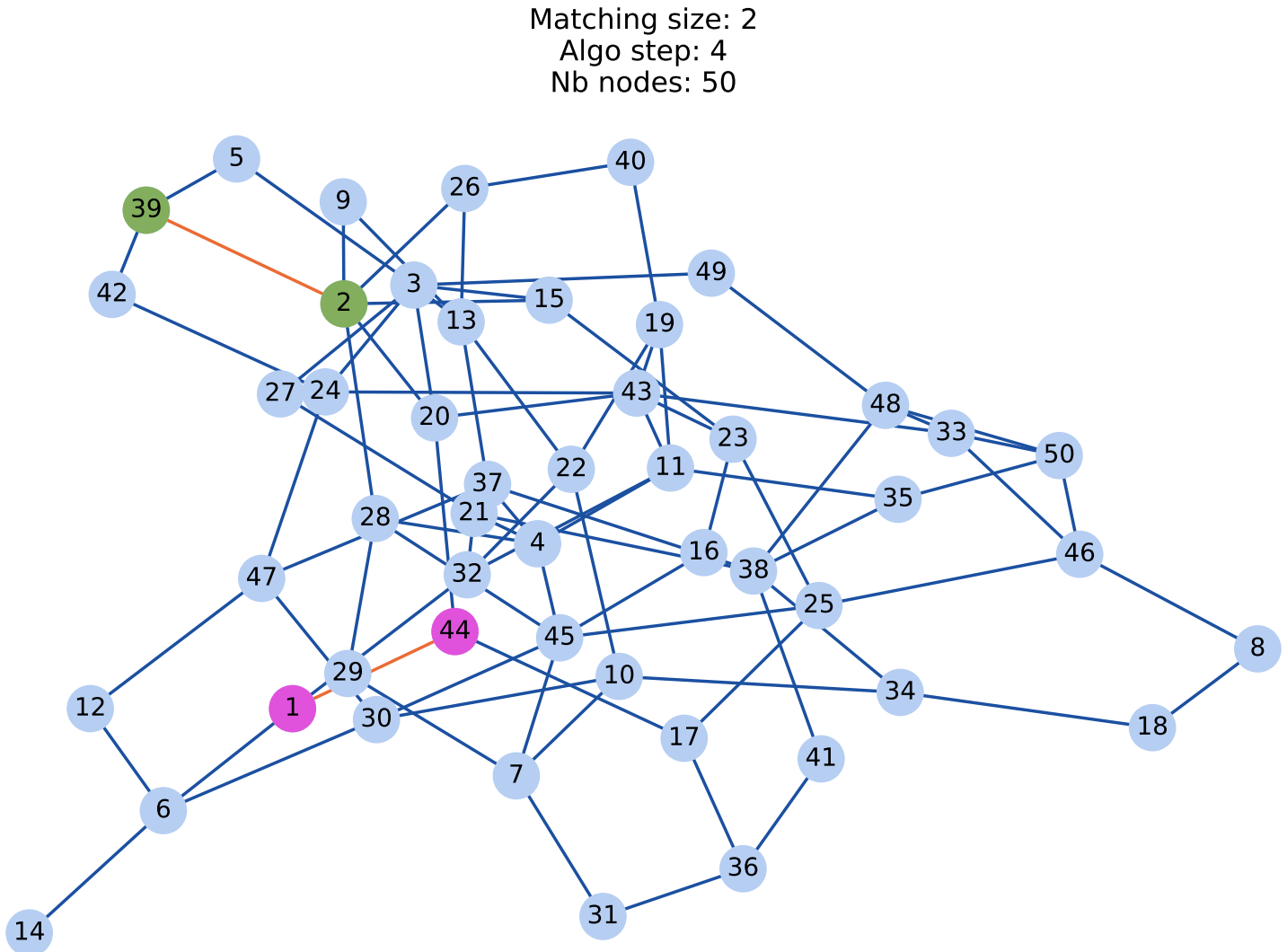
Matching size: 9
Algo step: 36
Nb nodes: 20

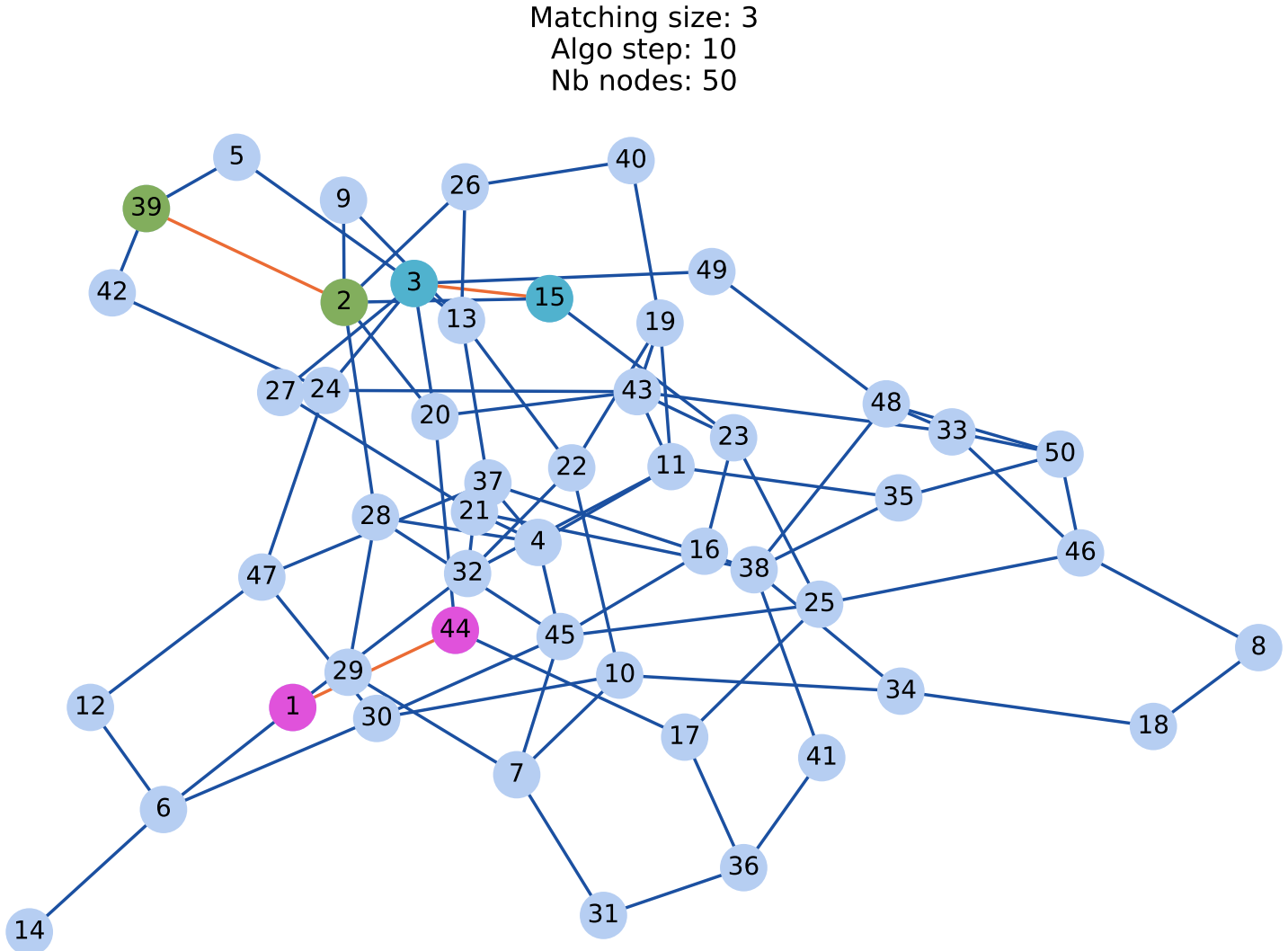


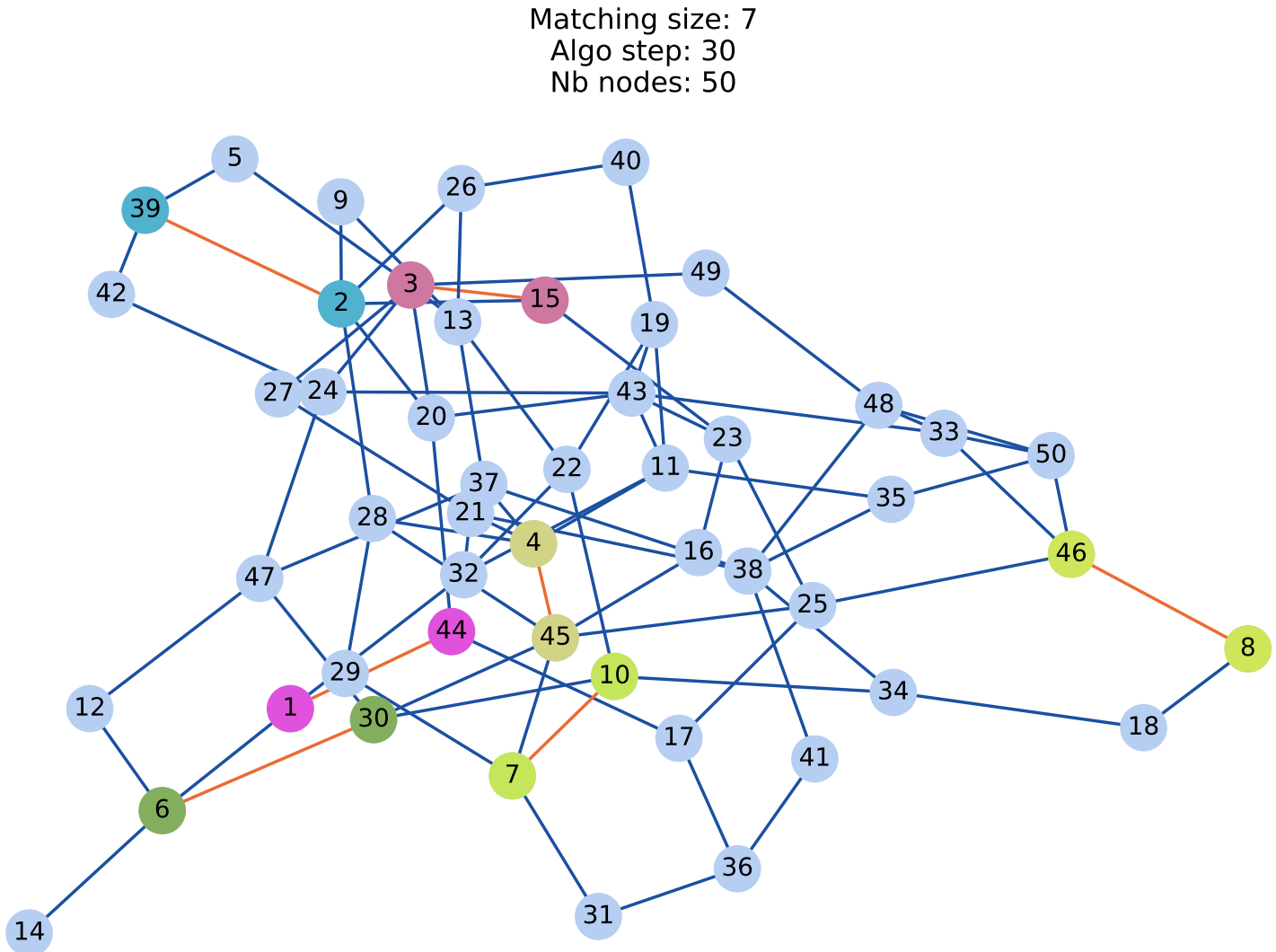
initial graph



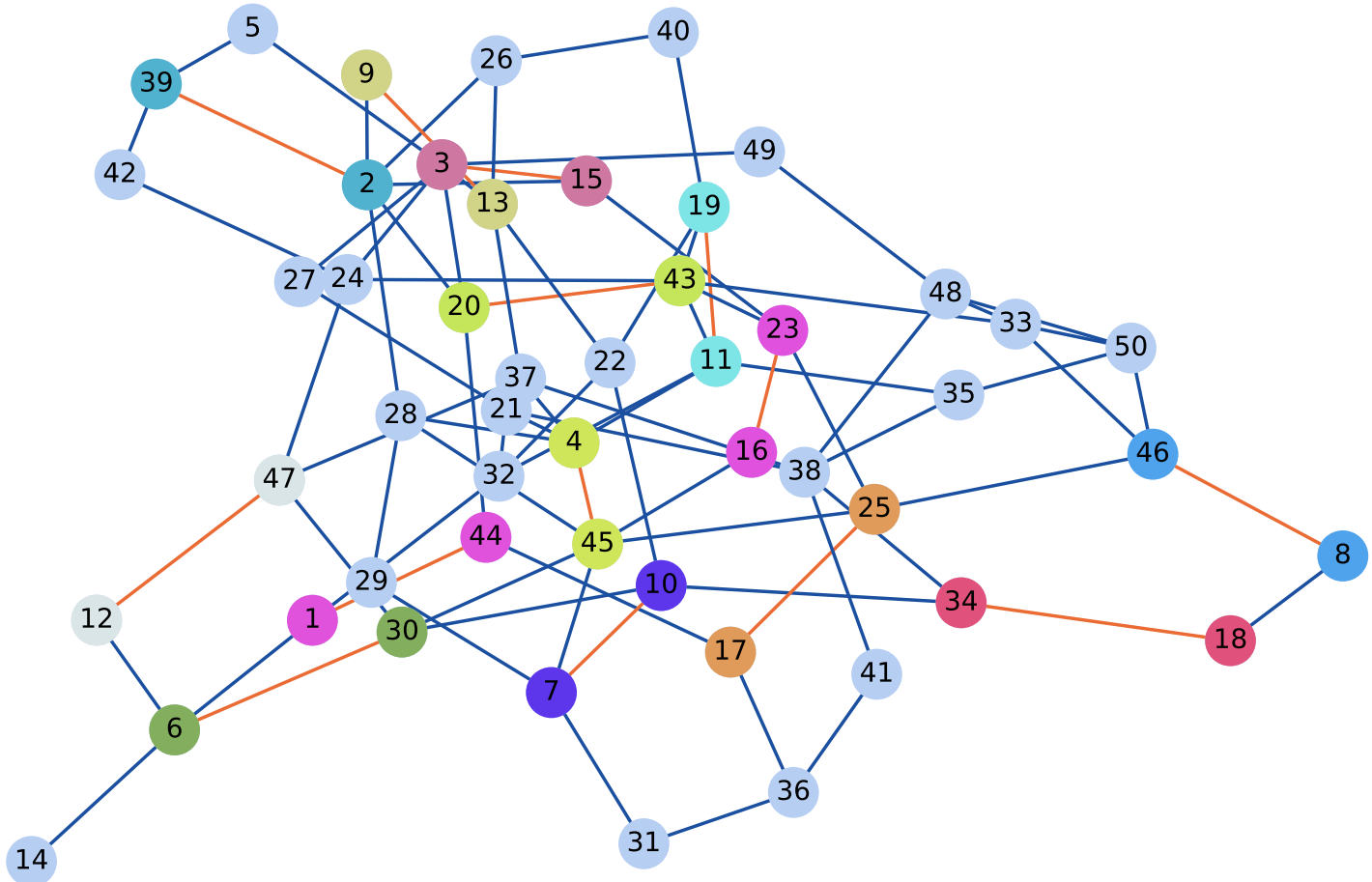

Matching size: 1
Algo step: 1
Nb nodes: 50
5
40
26
9
39
49
3
42
2
15
13
19
27
24
43
48
20
23
33
50
22
11
37
35
21
28
4
16
46
47
32
38
25
44
45
8
29
10
34
12
1
30
17
18
41
7
6
36
14
31

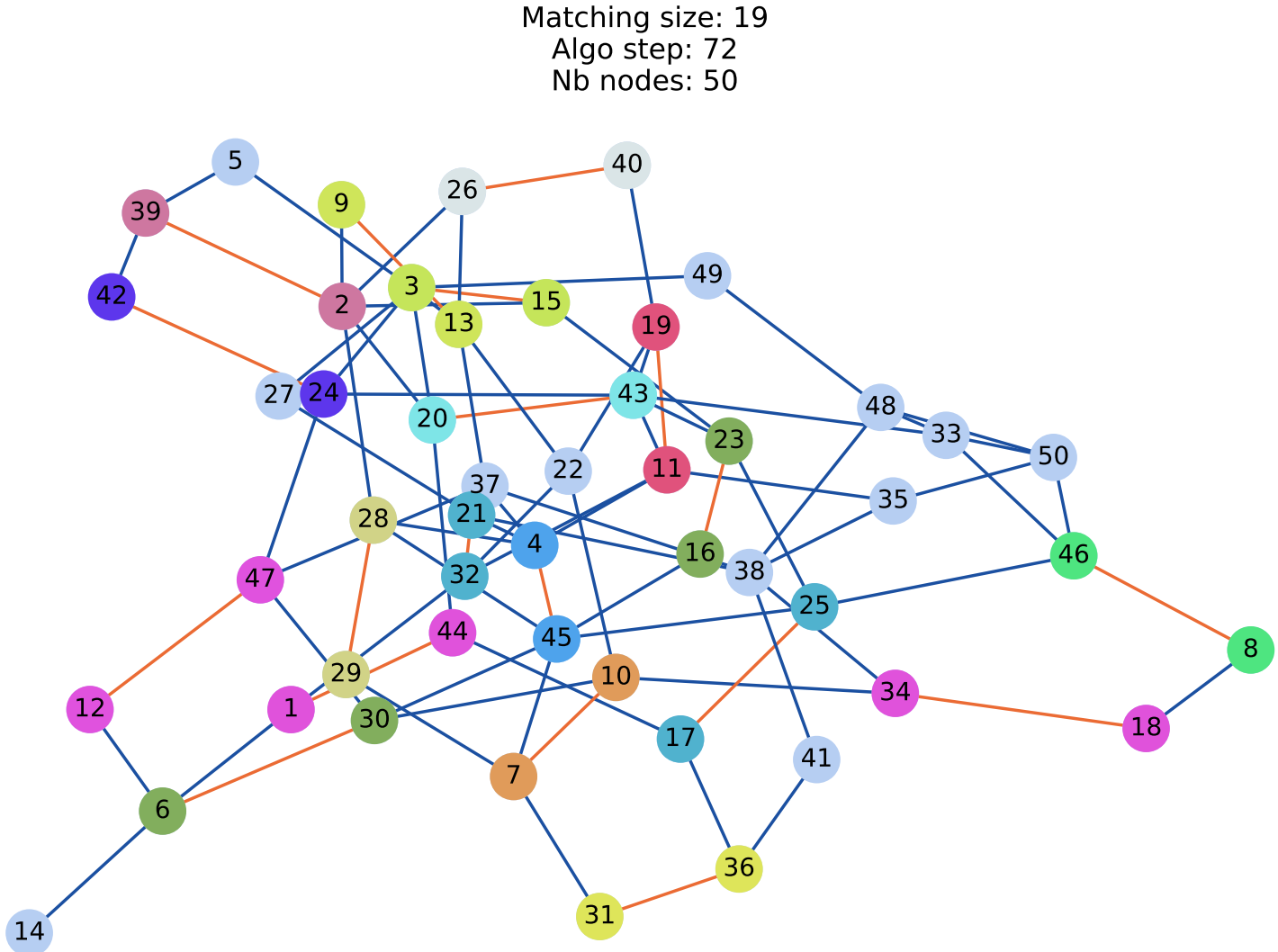

Matching size: 2
Algo step: 4
Nb nodes: 50
5
40
39
9
26
42
2
3
49
15
13
19
27
24
43
48
20
23
33
50
22
11
37
35
21
28
4
16
46
47
32
38
25
44
45
8
29
10
34
12
1
30
18
17
41
7
6
36
14
31


Matching size: 3
Algo step: 10
Nb nodes: 50
5
40
26
9
39
49
3
42
2
15
13
19
27
24
43
48
20
23
33
50
22
11
37
35
28
21
4
16
46
47
32
38
25
44
45
8
29
10
34
12
1
30
18
17
41
7
6
36
31
14

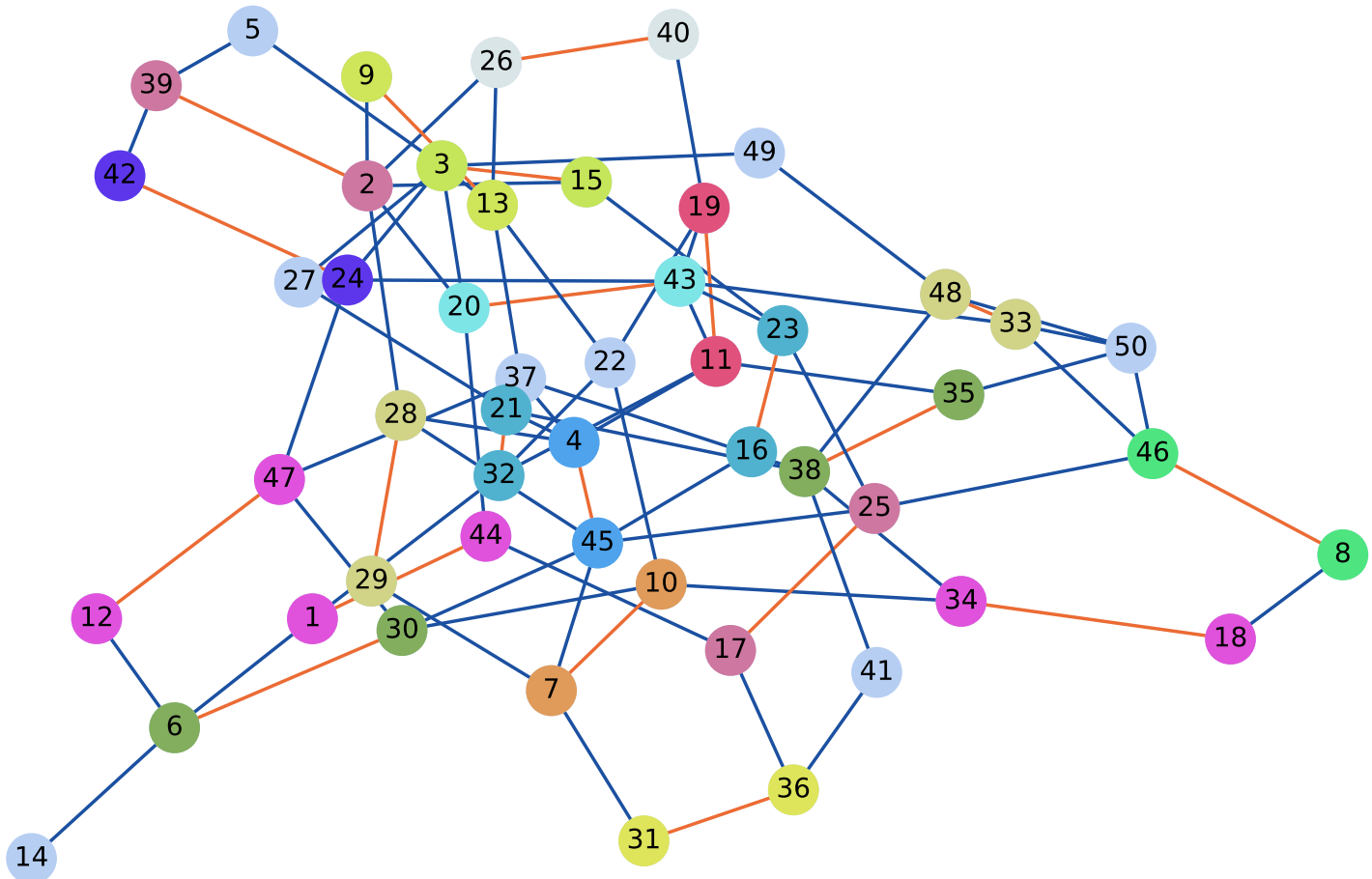

Matching size: 7
Algo step: 30
Nb nodes: 50
5
40
39
9
26
49
42
3
15
2
13
19
27
24
43
48
20
23
33
50
22
11
37
35
28
21
4
16
46
47
32
38
25
44
45
8
29
10
34
12
1
30
17
18
41
7
6
36
14
31

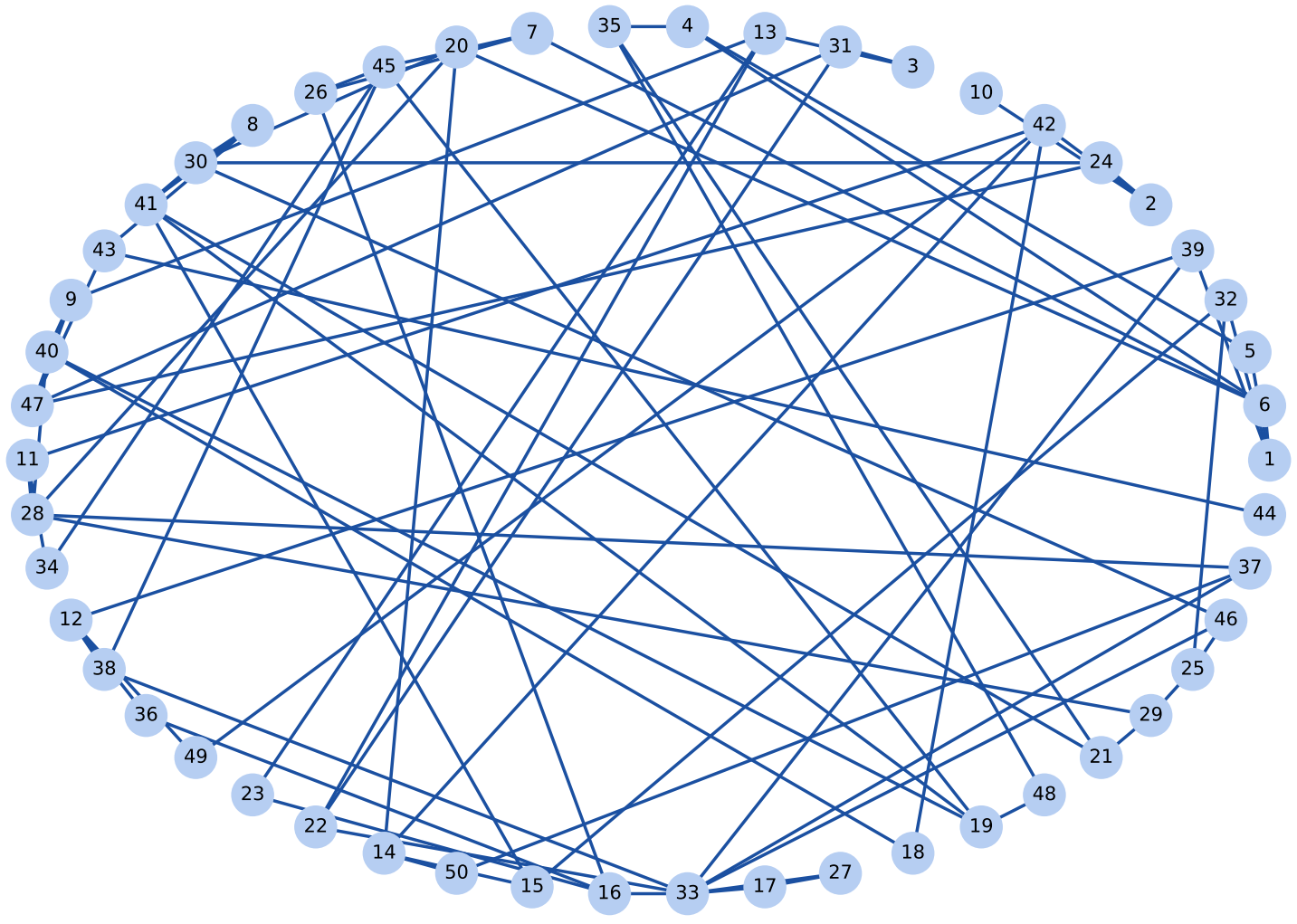
Matching size: 14
Algo step: 54
Nb nodes: 50

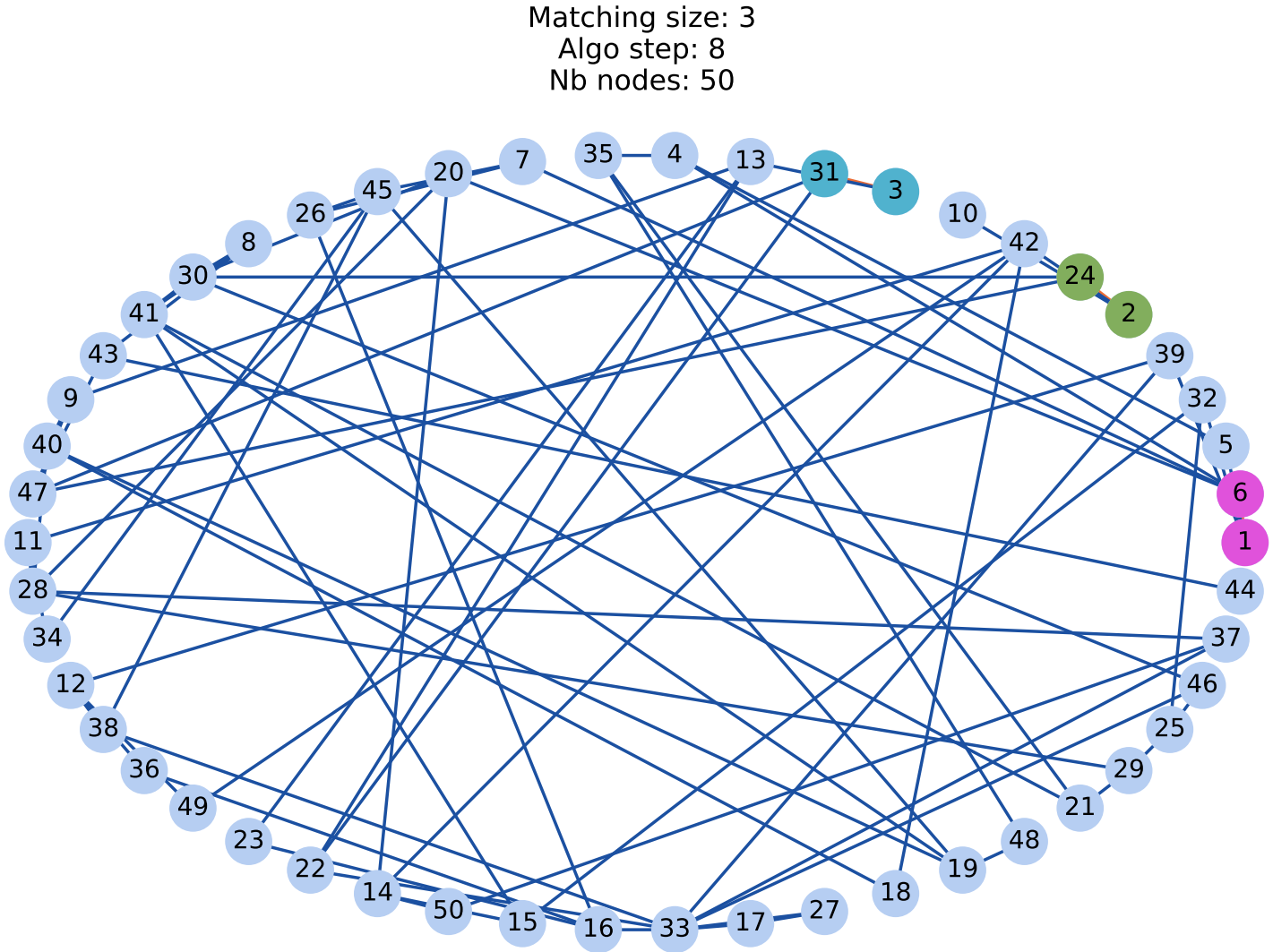


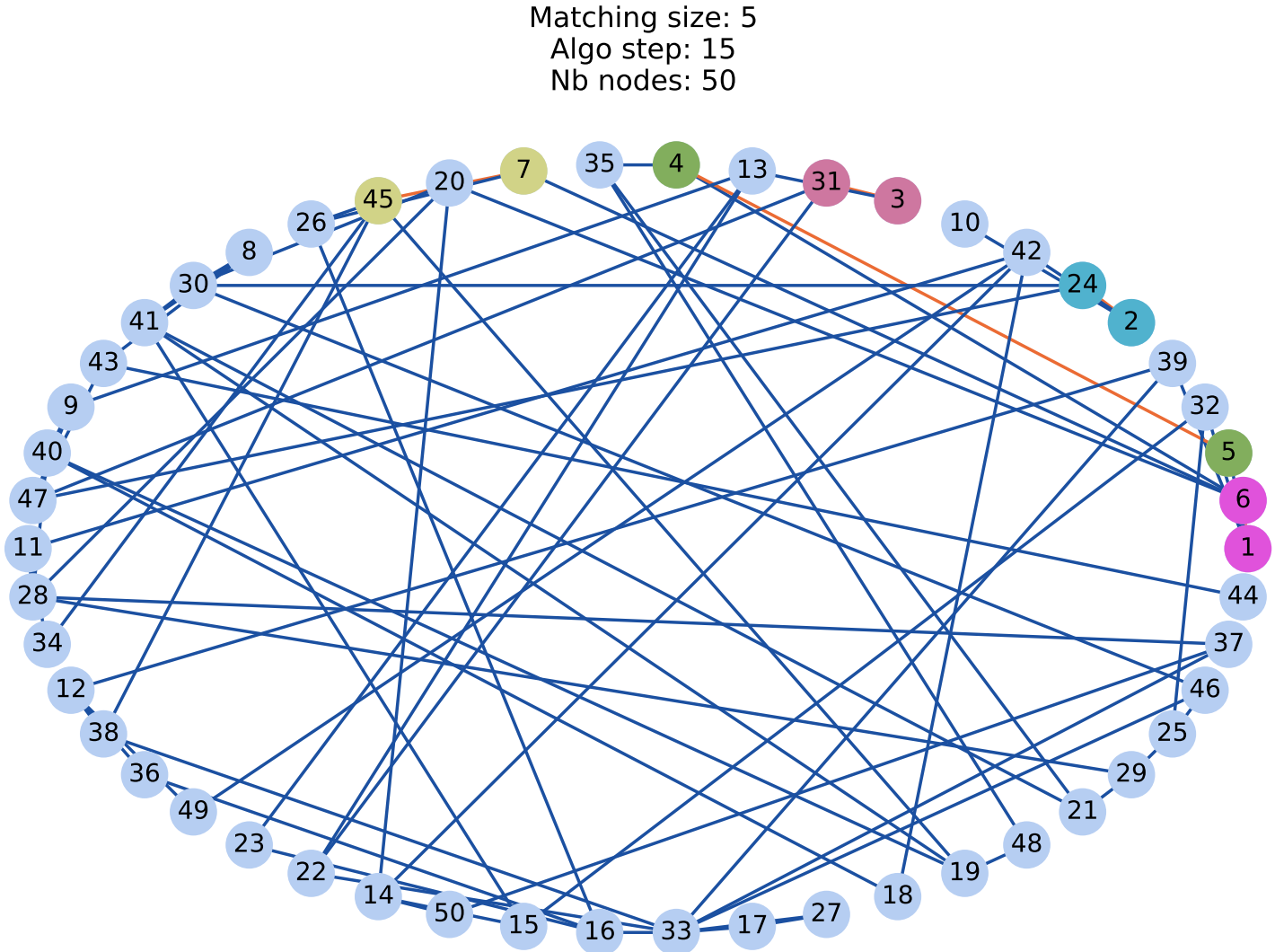

Matching size: 19
Algo step: 72
Nb nodes: 50
5
40
39
9
26
49
42
2
3
15
13
19
27
24
43
48
20
23
33
50
22
11
37
35
28
21
4
16
46
47
32
38
25
44
45
8
29
10
34
12
1
30
18
17
41
7
6
36
31
14

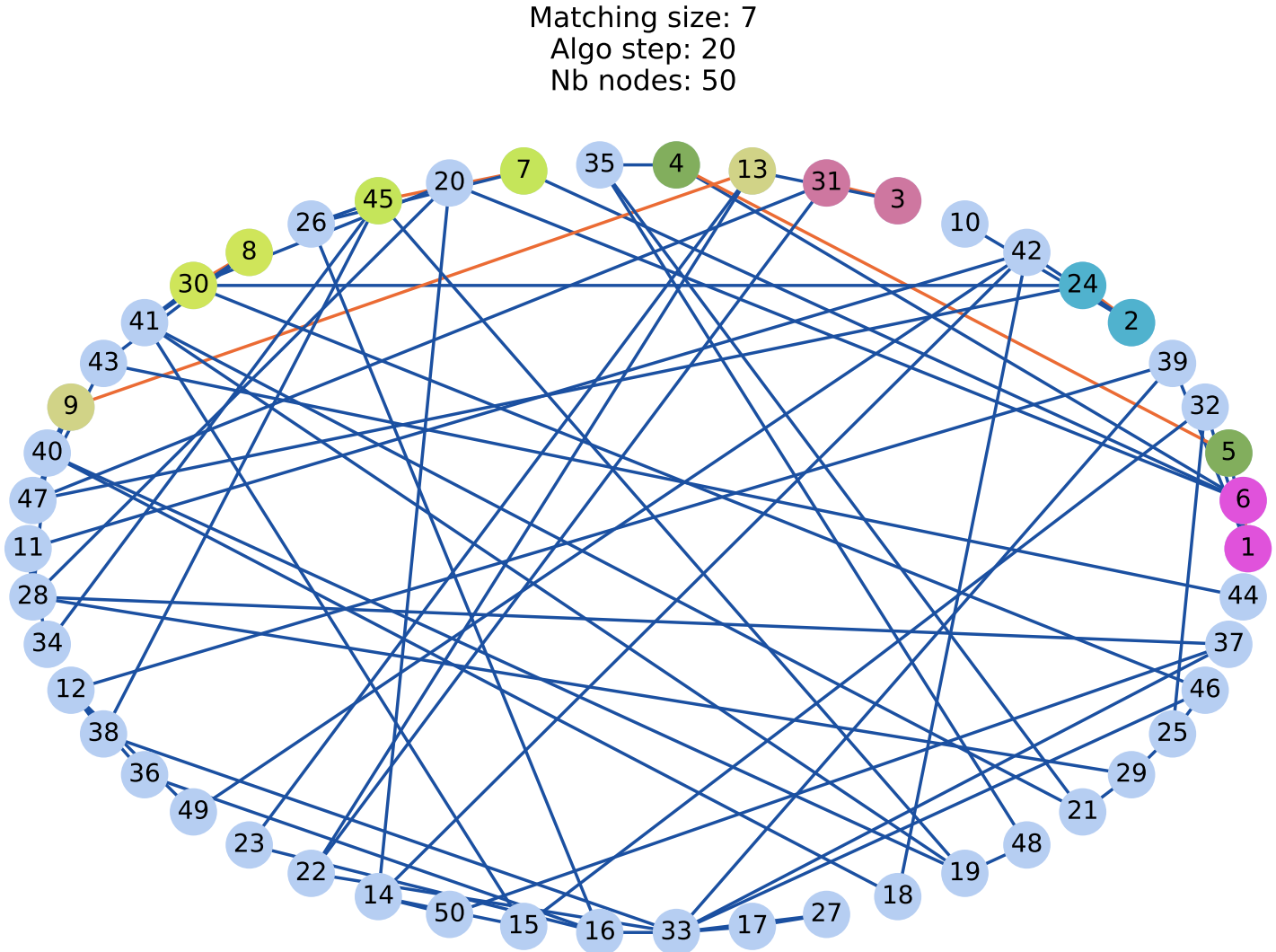
Matching size: 21
Algo step: 78
Nb nodes: 50

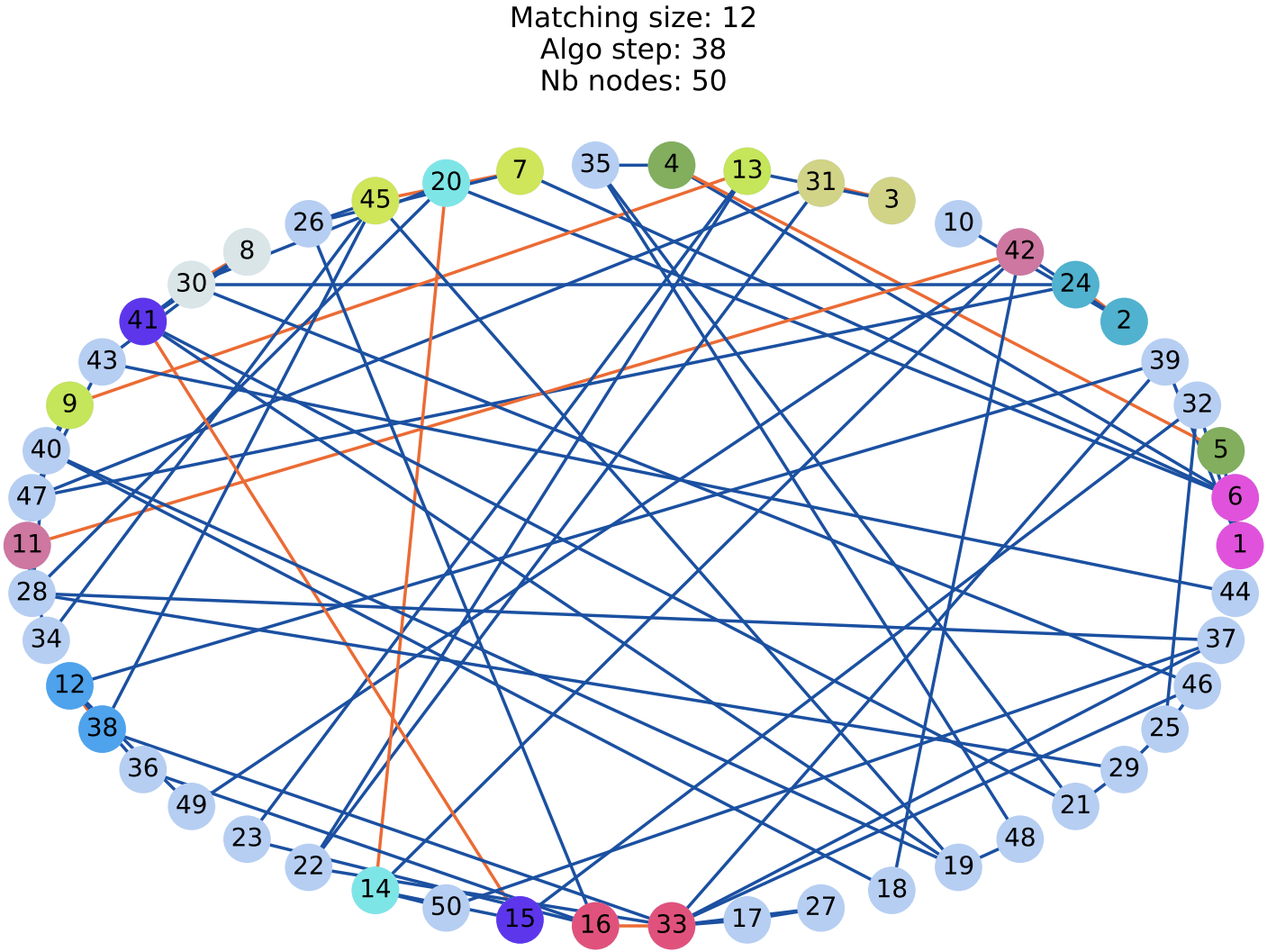


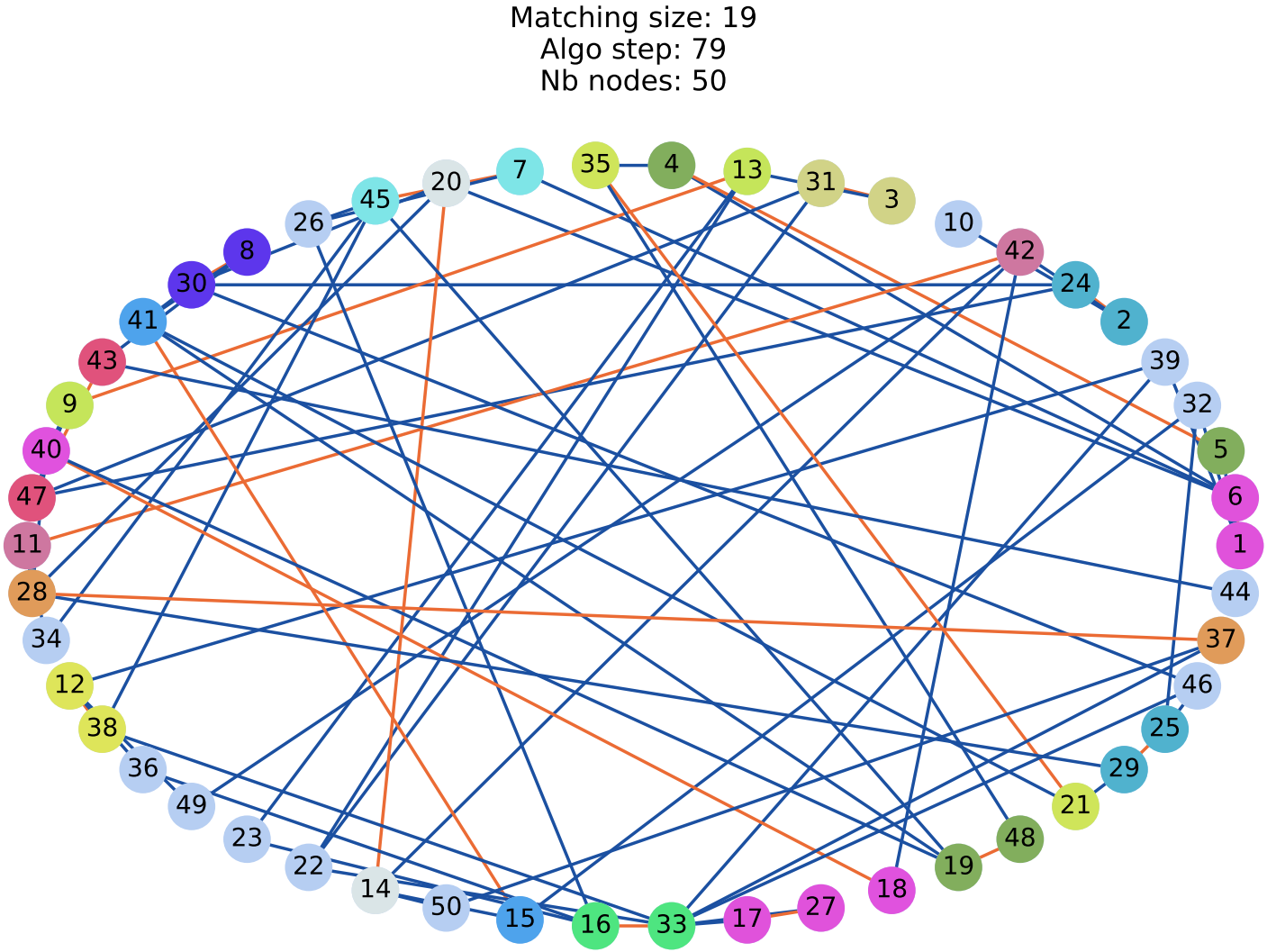

initial graph
35
4
13
31
3
10
42
24
2
39
32
5
6
1
44
37
46
25
29
21
48
19
18
27
17
33
16
15
50
14
22
23
49
36
38
12
34
28
11
47
40
9
43
41
30
8
26
45
20
7


Matching size: 3
Algo step: 8
Nb nodes: 50
35
4
13
31
3
10
42
24
2
39
32
5
6
1
44
37
46
25
29
21
48
19
18
27
17
33
16
15
50
14
22
23
49
36
38
12
34
28
11
47
40
9
43
41
30
8
26
45
20
7


Matching size: 5
Algo step: 15
Nb nodes: 50
35
4
13
31
3
10
42
24
2
39
32
5
6
1
44
37
46
25
29
21
48
19
18
27
17
33
16
15
50
14
22
23
49
36
38
12
34
28
11
47
40
9
43
41
30
8
26
45
20
7


Matching size: 7
Algo step: 20
Nb nodes: 50
35
4
13
31
3
7
20
45
26
8
30
41
43
9
40
47
11
28
34
12
38
36
49
23
22
14
50
15
16
33
17
27
18
19
48
21
29
25
46
37
44
1
6
5
32
39
2
24
42
10


Matching size: 12
Algo step: 38
Nb nodes: 50
35
4
13
31
3
7
20
45
26
8
30
41
43
9
40
47
11
28
34
12
38
36
49
23
22
14
50
15
16
33
17
27
18
19
48
21
29
25
46
37
44
1
6
5
32
39
2
24
42
10


Matching size: 19
Algo step: 79
Nb nodes: 50
35
4
13
31
3
10
42
24
2
39
32
5
6
1
44
37
46
25
29
21
48
19
18
27
17
33
16
15
50
14
22
23
49
36
38
12
34
28
11
47
40
9
43
41
30
8
26
45
20
7

# Example

Exercice 7 : Can you think of an example where the greedy algorithm gives a **bad** matching, e.g. of the size **half** the size of an optimal matching ?

# Example

Exercice 7 : Can you think of an example where the greedy algorithm gives a **bad** matching, e.g. of the size **half** the size of an optimal matching ?

# Greedy matching

However, is $|M|$ is the cardinality of a matching returned by the greedy algorithm, and if $|M^*|$ is the cardinal of the real optimal matching, we can theoretically show that :

$$|M| \geq \frac{|M^*|}{2} \tag{6}$$