

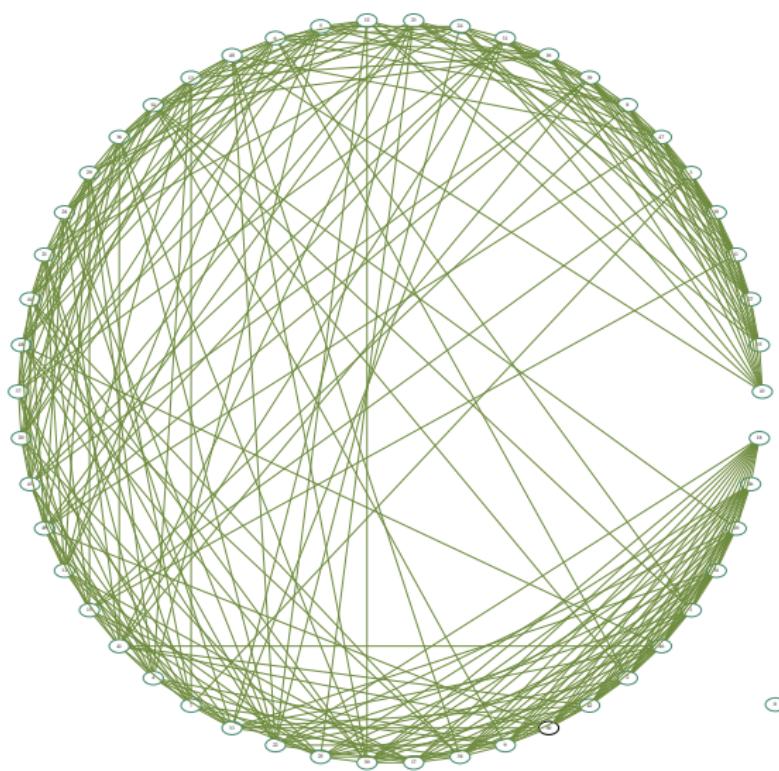


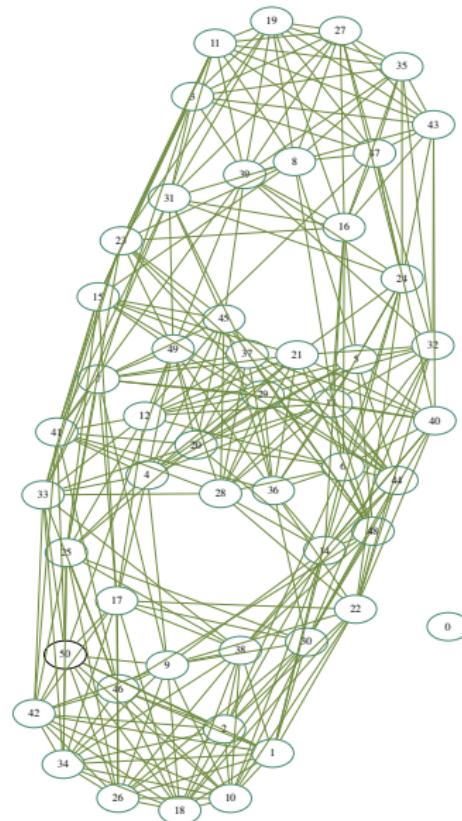
# Algorithms, Matching

Part 1. Networks and Matchings

B9 - Algorithms Matching

M-ALG-102





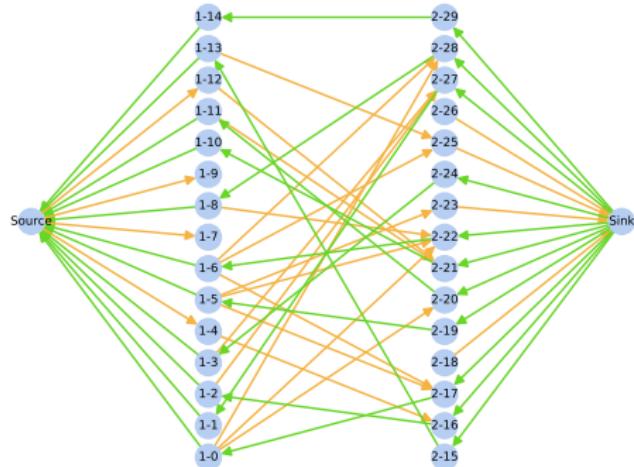
*Le réseau national  
après déclassement*



...

## └ Introduction

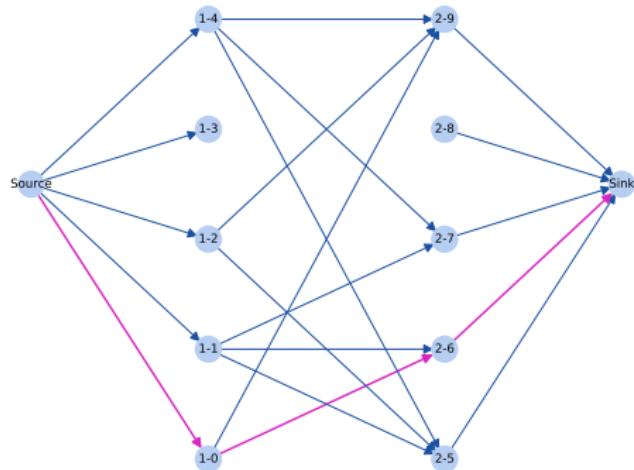
residual graph step 12

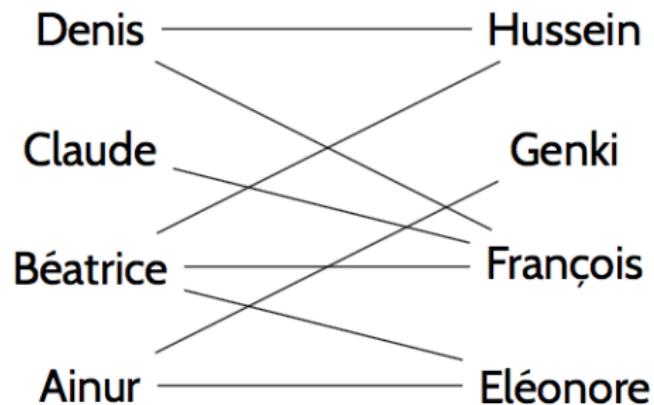


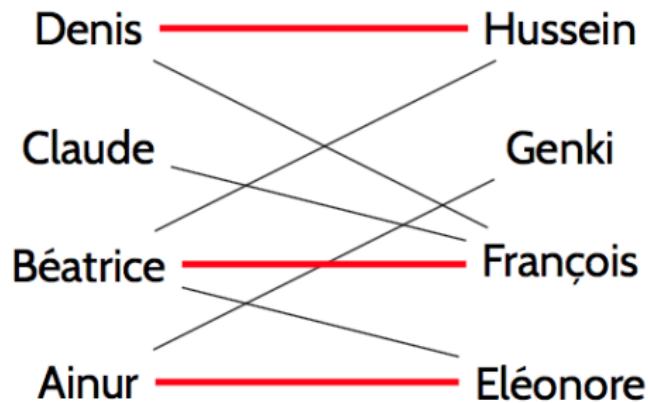
...

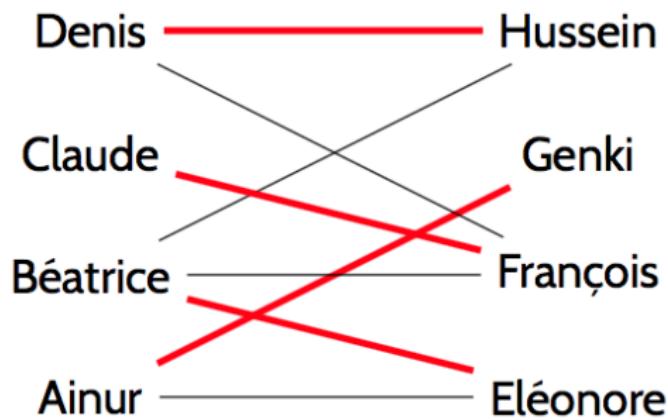
## └ Introduction

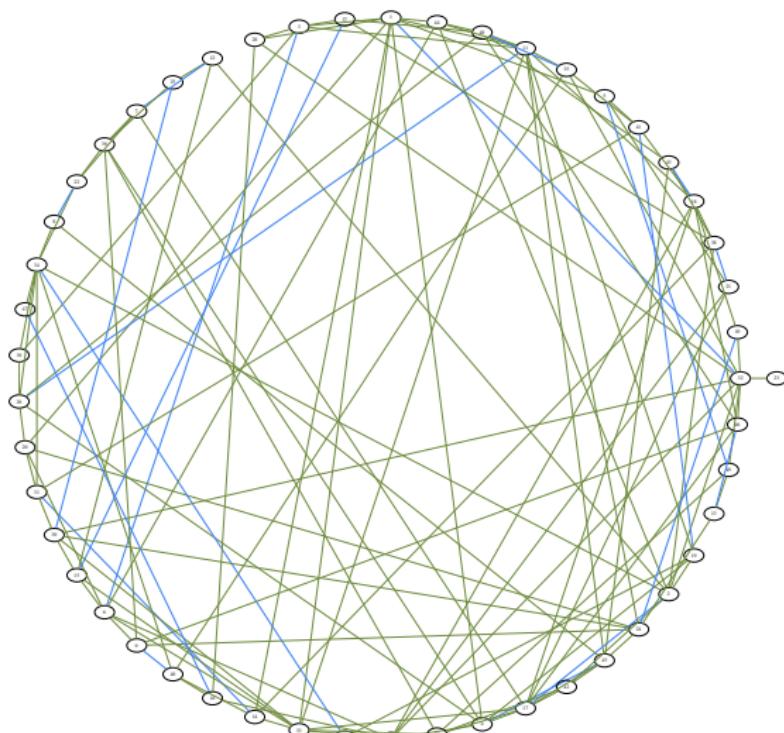
augmenting path step 1



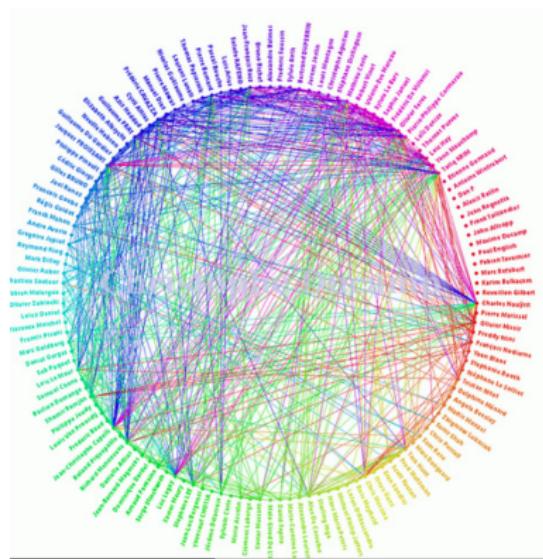






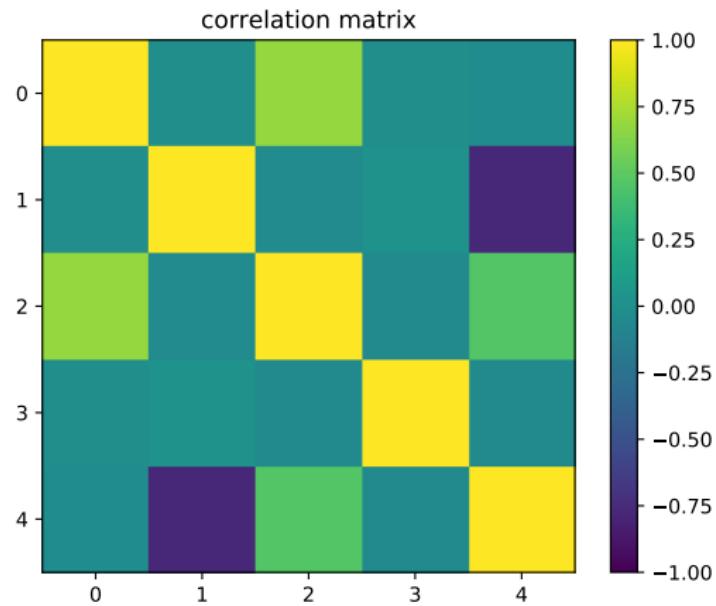


Matching size: 21  
Algo step: 128  
Nb nodes: 50



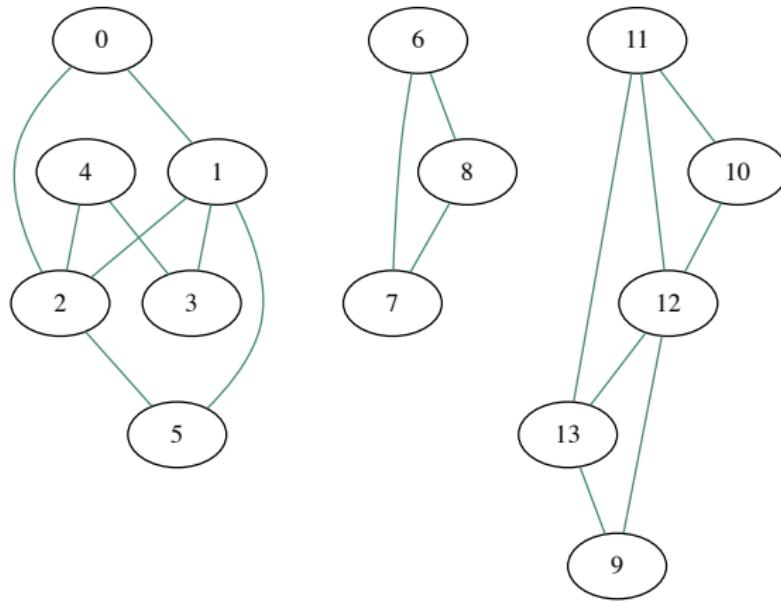
...

## └ Introduction



...

## └ Introduction



...

└ Introduction

---

- ▶ The content of the course will sometimes be mathematical.
- ▶ Don't hesitate to ask if you need more reminders.

# Overview of the module

**Day 1** Networks, the matching problem and the maximum flow problem

**Day 2** Data clustering and representation

## Organisation of the module

- ▶ Course and exercises in python 3
- ▶ Small coding exercises, also paper + pen
- ▶ Project : explained tomorrow
- ▶ Please clone the following repository  
<https://github.com/nlehir/ALG02>

...

└ Introduction

# Libs

Day 1 networkx, numpy, matplotlib

Day 2 numpy, matplotlib, pandas, sklearn

# Day 1

## The matching problem

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

## The Maximum flow problem

- Presentation of the problem
- Solution with the Ford-Fulkerson algorithm
- Connection with the matching problem
- More results on the two problems

## Introductory example 1 : Max Flow

*Le réseau national  
après déclassement*



**Figure:** Problem 1 : transporting merchandise through a network

## Introductory example 2 : Maximum matching (Optimal allocation)

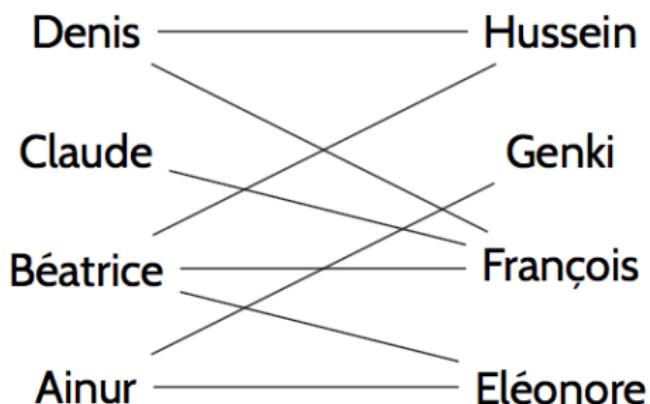


Figure: Problem 2 : Building the largest possible number of teams of 2 persons.

## Introductory example 2

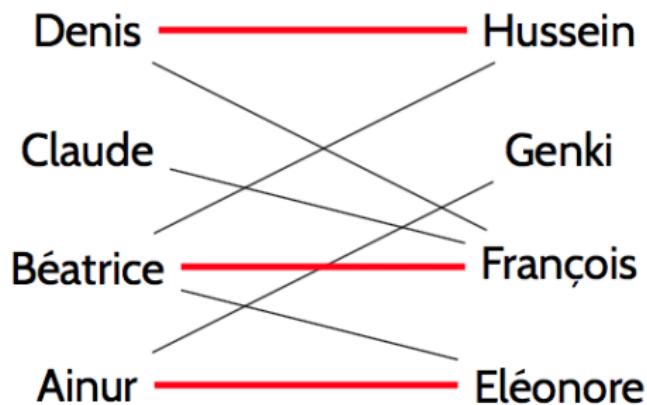


Figure: Problem 2 : not optimal allocation

## Introductory example 2

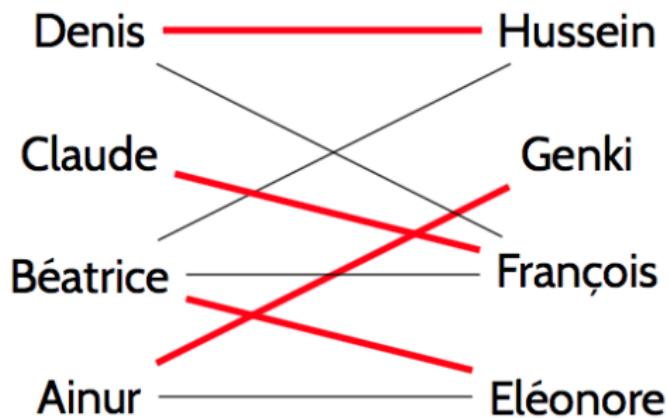


Figure: Problem 2 : optimal allocation

...

└ Introduction

## Other examples

- ▶ Assigning students to internships

...

└ Introduction

## Other examples

- ▶ Assigning students to internships
- ▶ Assigning machines to a task

...

└ Introduction

## Summary

- ▶ Today we will work on **connecting the two problems**.

...

└ Introduction

## Summary

- ▶ Today we will work on **connecting the two problems**.
- ▶ Under some restrictions, the two problems **equivalent**.

...

└ The matching problem

└ Definition of the problem

## Reminders on graphs

- ▶ A graph is defined by ?

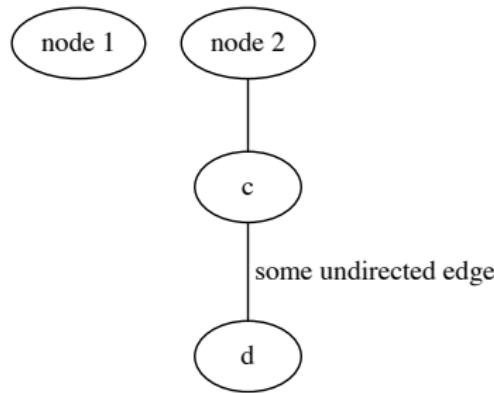
...

- └ The matching problem

- └ Definition of the problem

## Reminders on graphs

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



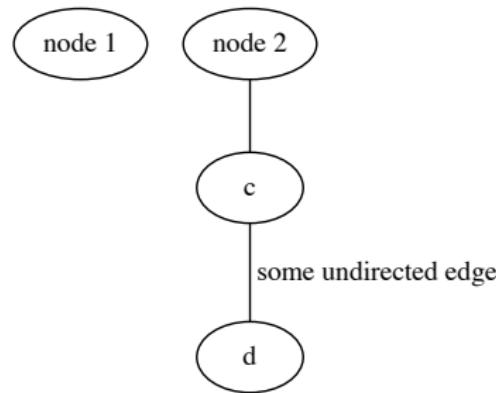
...

- └ The matching problem

- └ Definition of the problem

## Reminders on graphs

- ▶ It can be **undirected**, as this one :



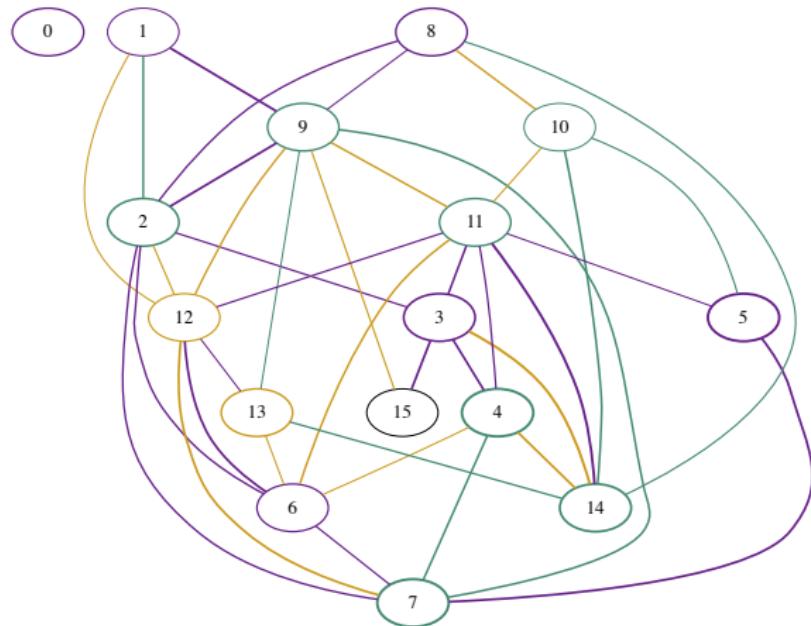
...

- └ The matching problem

- └ Definition of the problem

## Reminders on graphs

### Undirected graph



...

- └ The matching problem

- └ Definition of the problem

## Reminders on graphs

- ▶ Or **directed**, as this one. (it is then called a **digraph**)

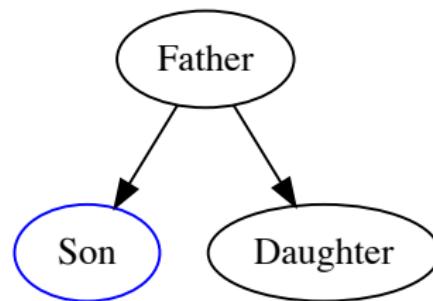


Figure: Digraph (graphviz demo)

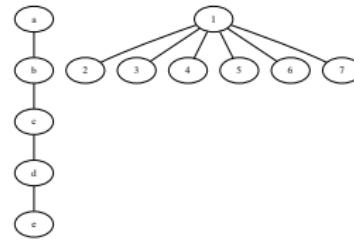
...

└ The matching problem

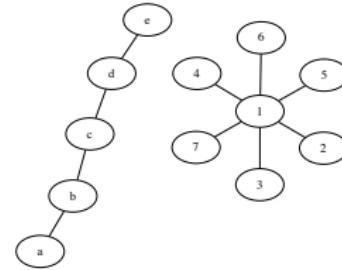
  └ Definition of the problem

## Useful tool : graphviz

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



(a) Image generated with **dot**



(b) Image generated with **neato**

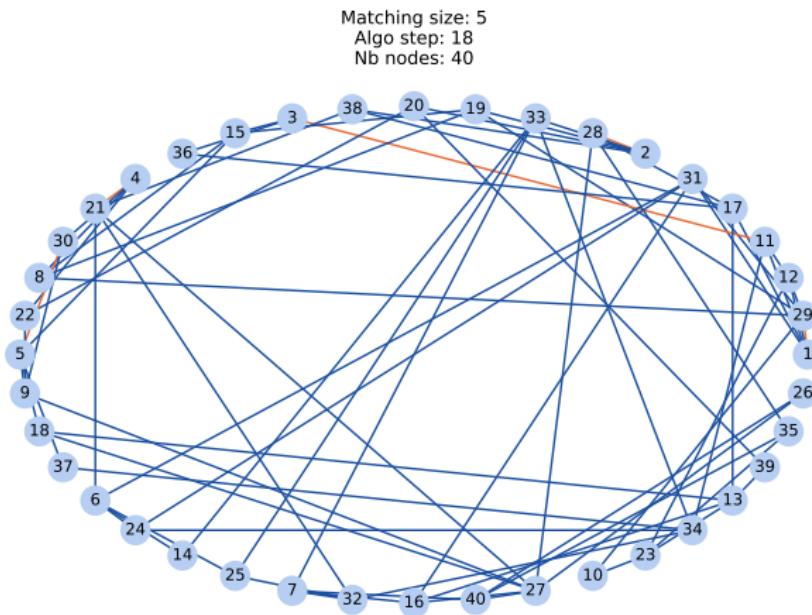
...

- └ The matching problem

- └ Definition of the problem

# Networkx

We will use networkx.



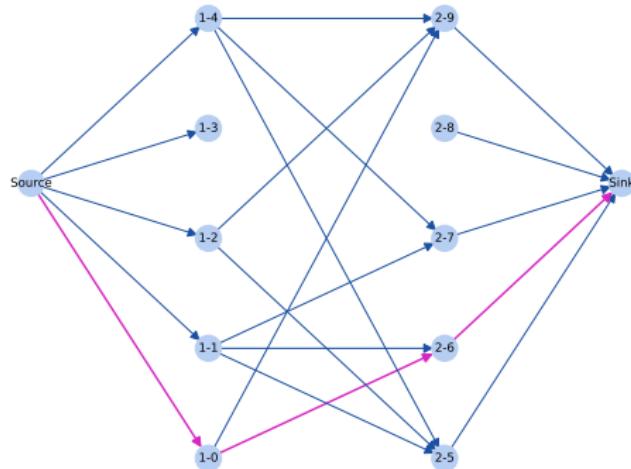
...

- └ The matching problem

- └ Definition of the problem

# Networkx

augmenting path step 1



...

- └ The matching problem

- └ Definition of the problem

## 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

...

└ The matching problem

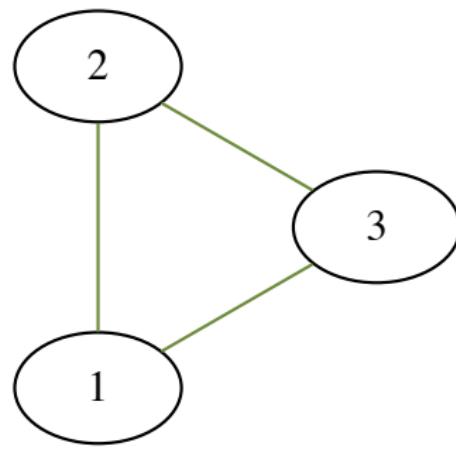
└ Definition of the problem



...

- └ The matching problem

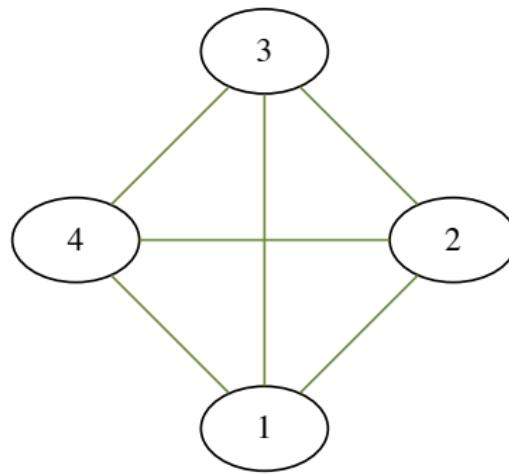
- └ Definition of the problem



...

- └ The matching problem

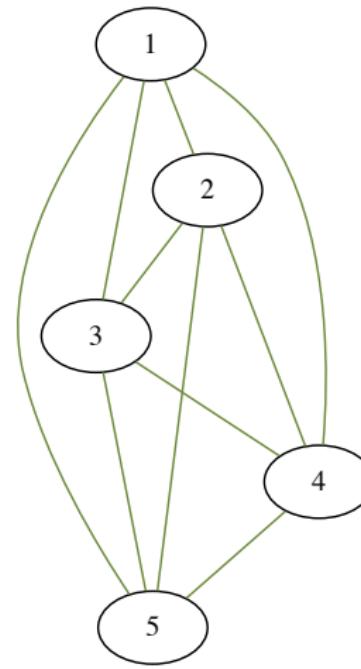
- └ Definition of the problem



...

- └ The matching problem

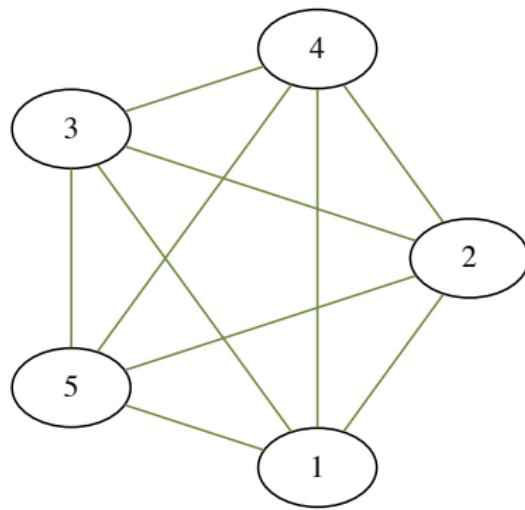
- └ Definition of the problem



...

- └ The matching problem

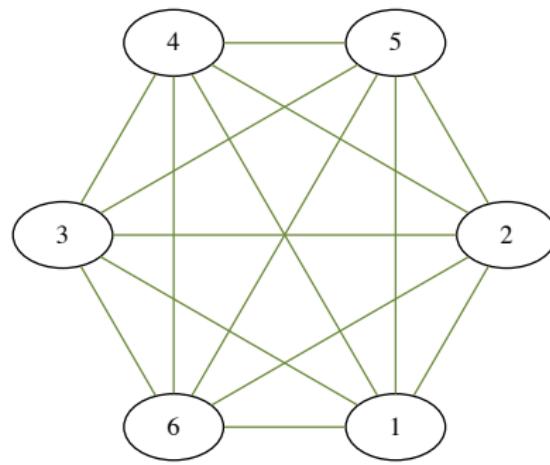
- └ Definition of the problem



...

- └ The matching problem

- └ Definition of the problem



...

## The matching problem

### Definition of the problem

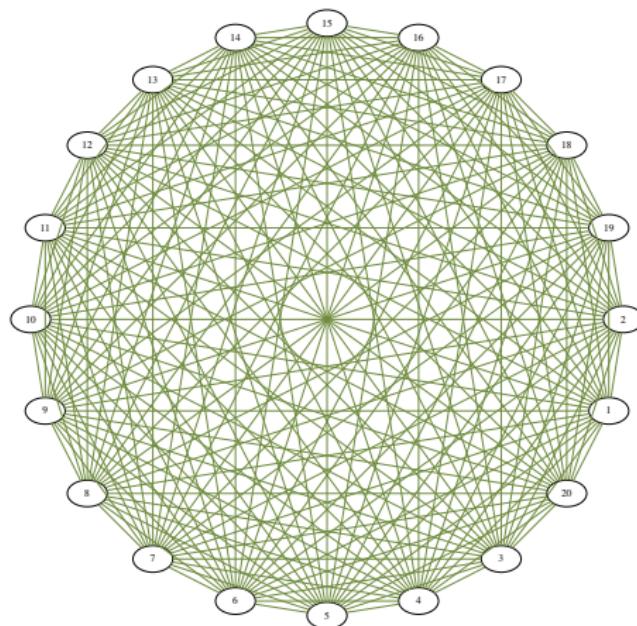


Figure: We cannot count anymore

...

## The matching problem

### Definition of the problem

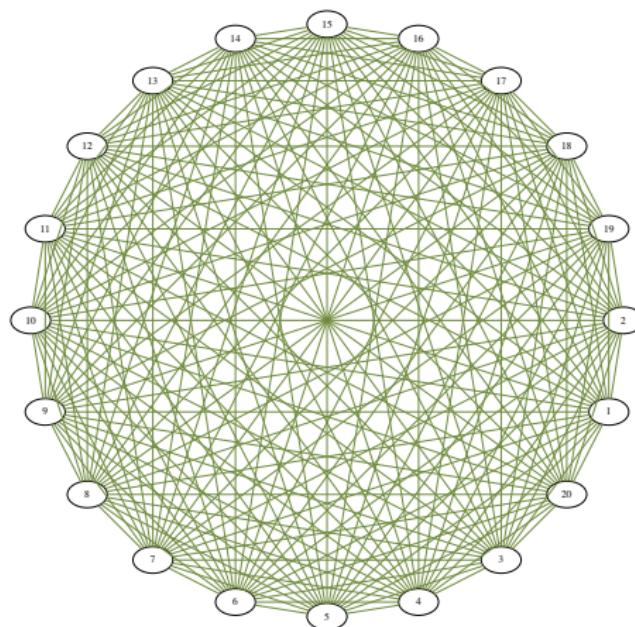


Figure: We cannot count anymore

...

- └ The matching problem

- └ Definition of the problem

What if the graph is directed ?

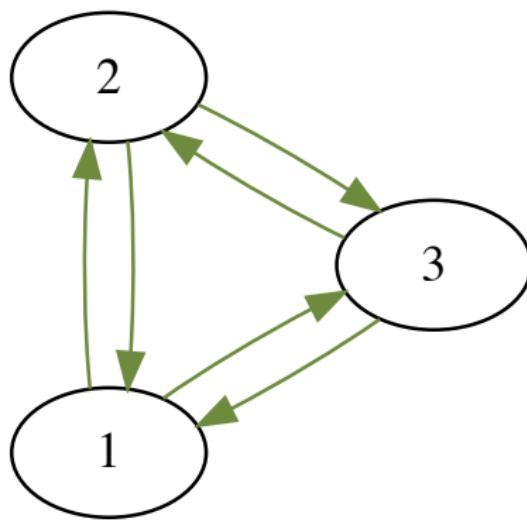


...

- └ The matching problem

- └ Definition of the problem

What if the graph is directed ?

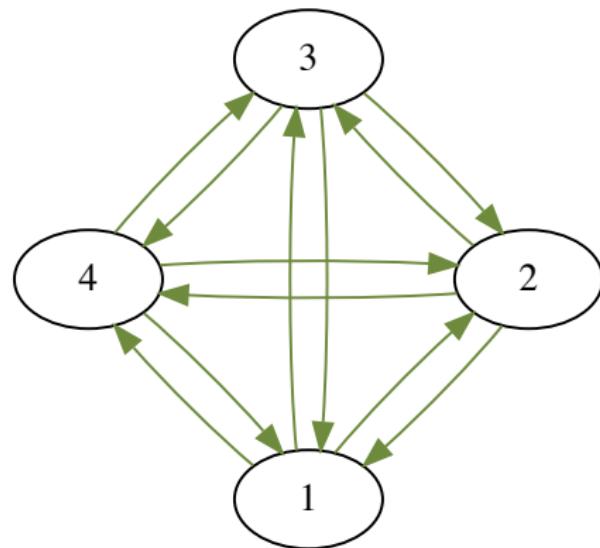


...

- └ The matching problem

- └ Definition of the problem

## What if the graph is directed ?

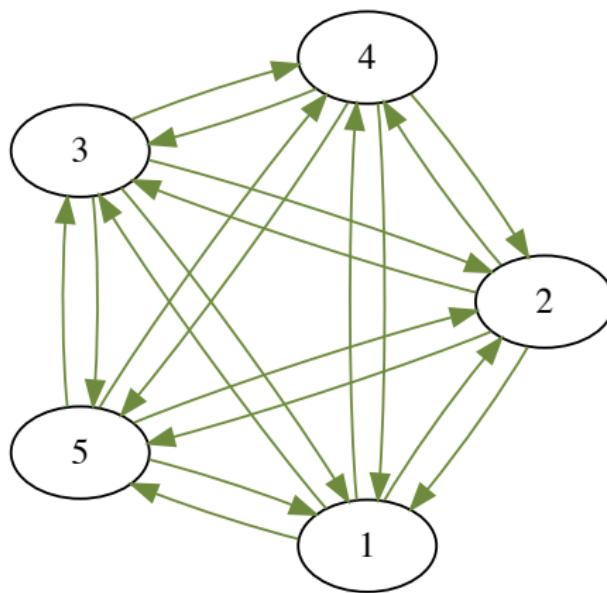


...

- └ The matching problem

- └ Definition of the problem

## What if the graph is directed ?

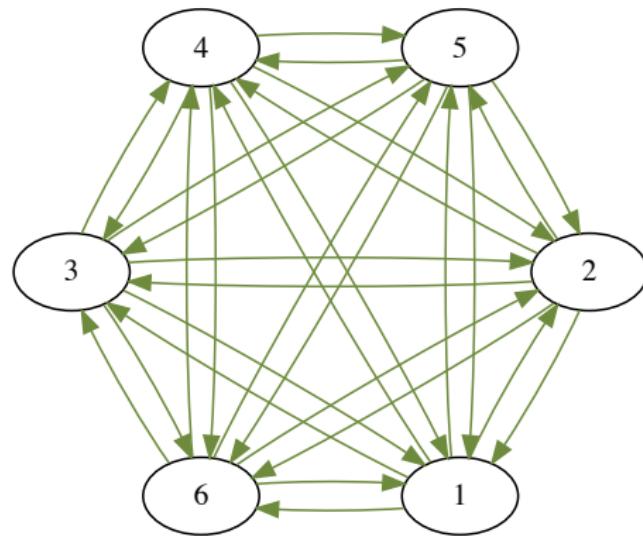


...

- └ The matching problem

- └ Definition of the problem

## What if the graph is directed ?

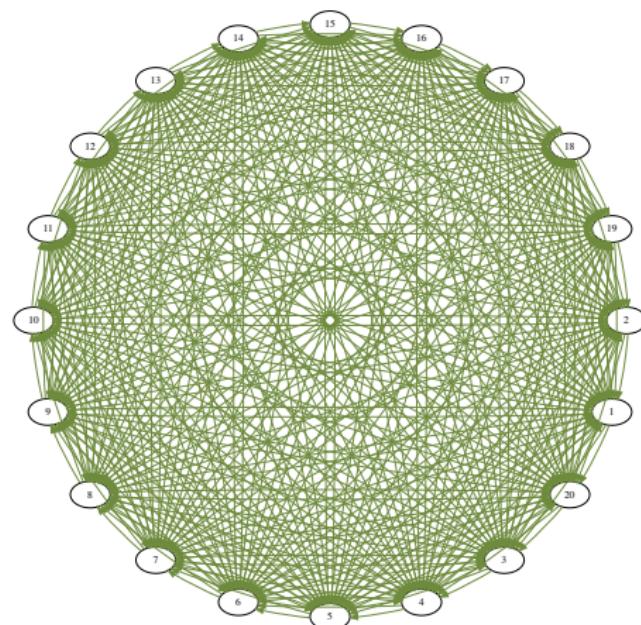


...

- └ The matching problem

- └ Definition of the problem

# What if the graph is directed ?



...

└ The matching problem

└ Definition of the problem

## Warm up question

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

...

└ The matching problem

└ Definition of the problem

## Warm up question

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

$$n(n - 1) \quad (1)$$

...

- └ The matching problem

- └ Definition of the problem

## Warm up question

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

$$n(n - 1) \quad (2)$$

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

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

edges.

...

- └ The matching problem

- └ Definition of the problem

## Remark

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

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

...

└ The matching problem

└ Definition of the problem

## Famous graph problem

- ▶ Do you know some famous **graph problems** ?

...

└ The matching problem

└ Definition of the problem

## Famous graph problem

- ▶ Do you know some famous **graph problems** ?
- ▶ Dominating set

...

└ The matching problem

└ Definition of the problem

## Famous graph problem

- ▶ Do you know some famous **graph problems** ?
- ▶ Dominating set
- ▶ Maximum clique

...

└ The matching problem

  └ Definition of the problem

## Famous graph problem

- ▶ Do you know some famous **graph problems** ?
- ▶ Dominating set
- ▶ Maximum clique
- ▶ Coloring

...

└ The matching problem

  └ Definition of the problem

## Matching problem

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

...

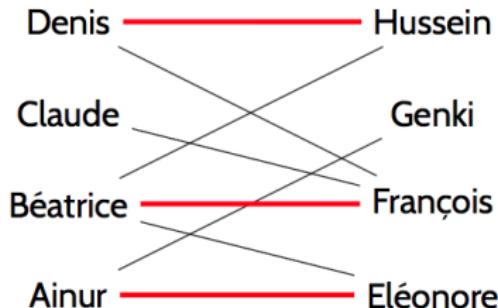
- └ The matching problem

- └ Definition of the problem

## Back to our problem

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

- ▶ A subset of edges  $M \subset E$



...

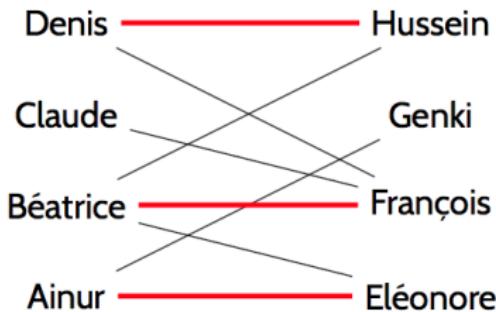
- The matching problem

- Definition of the problem

## 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 has **at most** one edge connected



...

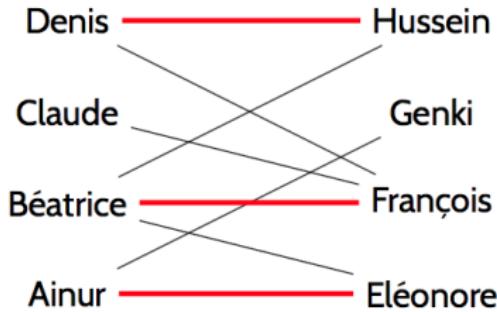
- The matching problem

- Definition of the problem

## 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 has **at most** one edge connected
- ▶ Such that no pairs of edges of  $M$  are incident



...

└ The matching problem

  └ Definition of the problem

## Maximum matching

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

...

└ The matching problem

  └ Definition of the problem

## 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.

...

- └ The matching problem

- └ Definition of the problem

## Example 1

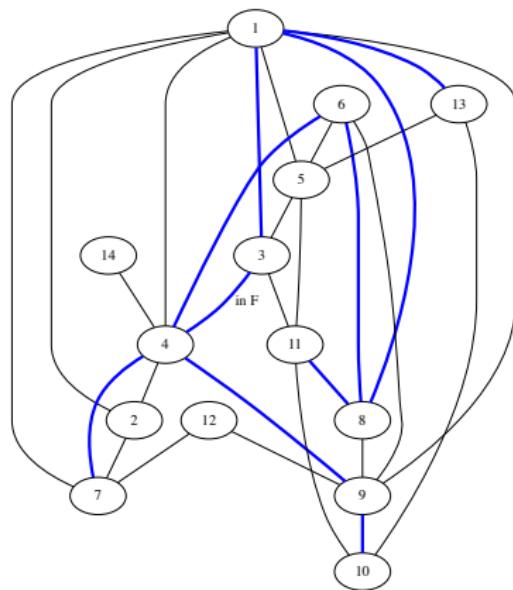


Figure: Is this a matching ?

...

- └ The matching problem

- └ Definition of the problem

## Example 2

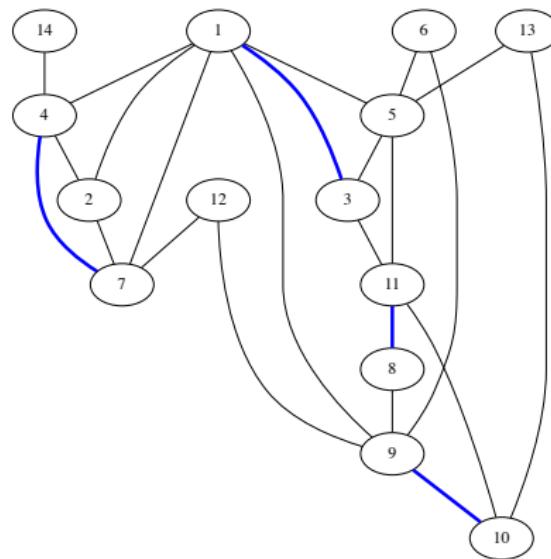


Figure: Is this a matching ?

...

- └ The matching problem

- └ Definition of the problem

## Example 3

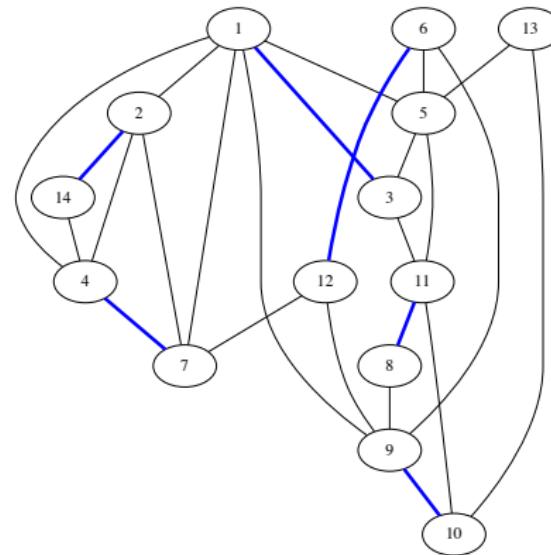


Figure: Is this an optimal matching ?

...

- └ The matching problem

- └ Definition of the problem

## Example 4

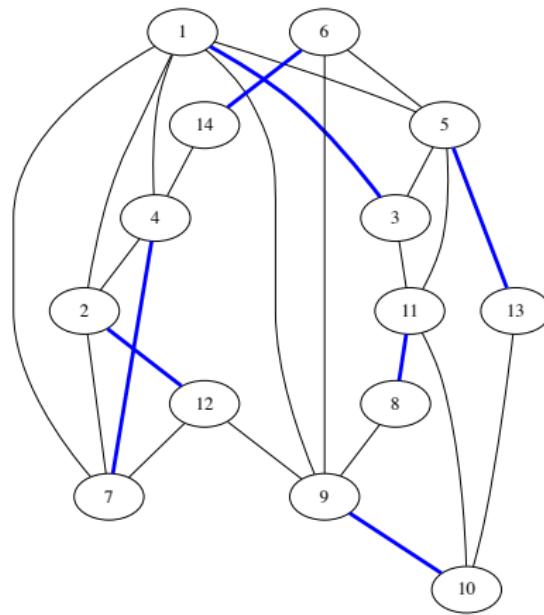


Figure: Is this an optimal matching ?

...

- └ The matching problem

- └ Definition of the problem

## Example 5

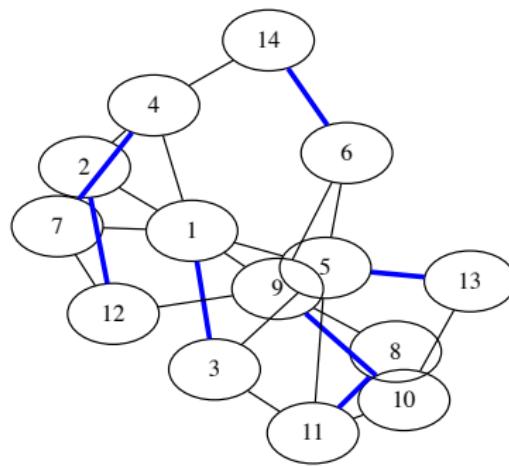


Figure: With neato

...

└ The matching problem

  └ Definition of the problem

## Optimal matching

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

...

- └ The matching problem

- └ Definition of the problem

## 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 :  $\frac{n-1}{2}$

...

└ The matching problem

  └ Definition of the problem

## 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)

...

- └ The matching problem

- └ Definition of the problem

## 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)

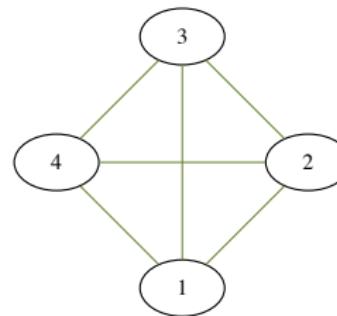


Figure: The complete graph works

...

└ The matching problem

  └ Definition of the problem

## 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)

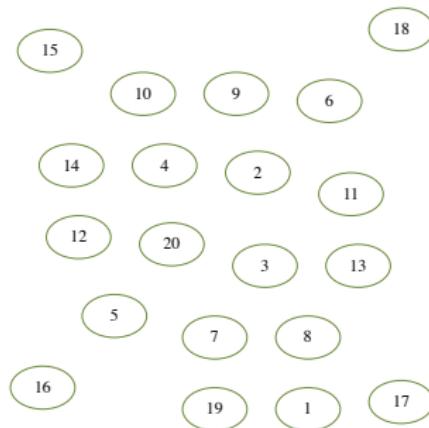
...

- └ The matching problem

- └ Definition of the problem

## 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)



...

└ The matching problem

  └ Definition of the problem

## Optimal matching

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

...

- └ The matching problem

- └ Definition of the problem

## Optimal matching

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

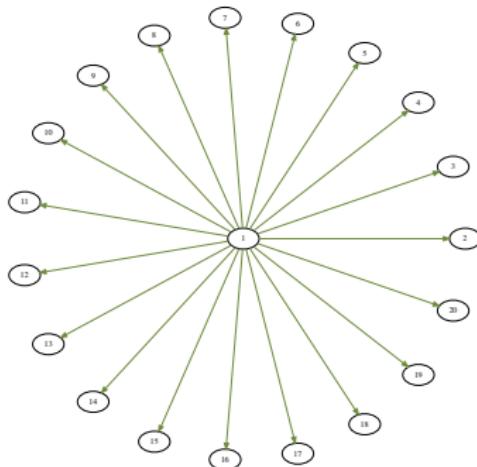


Figure: Star graph

...

- └ The matching problem
- └ Experimental solutions

## Experiments

How would you code a graph ?

...

- └ The matching problem
- └ Experimental solutions

## Experiments

How would you code a graph ?

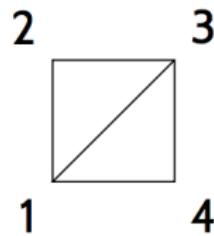
- ▶ list of sets of size 2 (for an undirected graph)
- ▶ a dictionary of successors (directed or undirected)

...

- └ The matching problem

- └ Experimental solutions

## Coding a graph : as a list



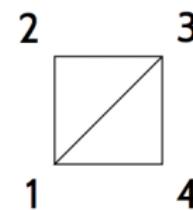
```
g1 = [{1,2},{1,3},{2,3},{3,4},{1,4}]
```

...

- └ The matching problem

- └ Experimental solutions

## Coding a graph : as a dictionary



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

...

- └ The matching problem
- └ Experimental solutions

## Random graph

**Exercice 3:** `cd other_graphs/` and please use  
`random_undirected_graph.py` to build a graph with 20 vertices  
and 50 edges.

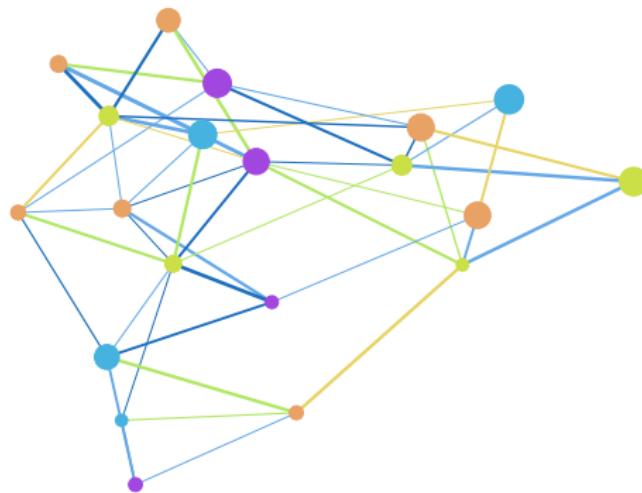
- ▶ You will need to install `networkx`

...

- └ The matching problem

- └ Experimental solutions

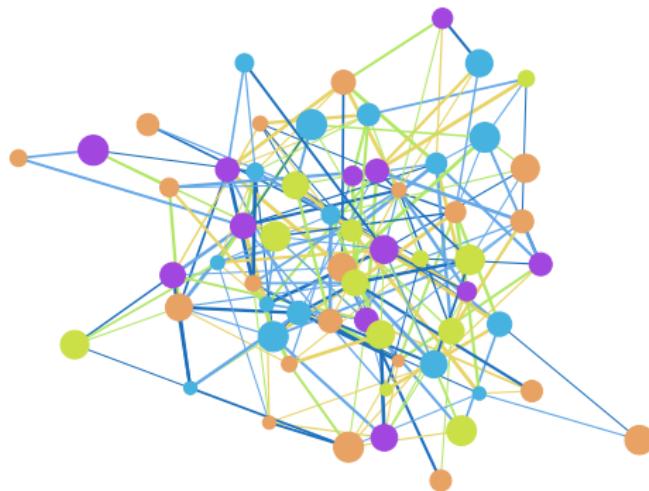
## Example undirected graph obtained



...

- └ The matching problem
- └ Experimental solutions

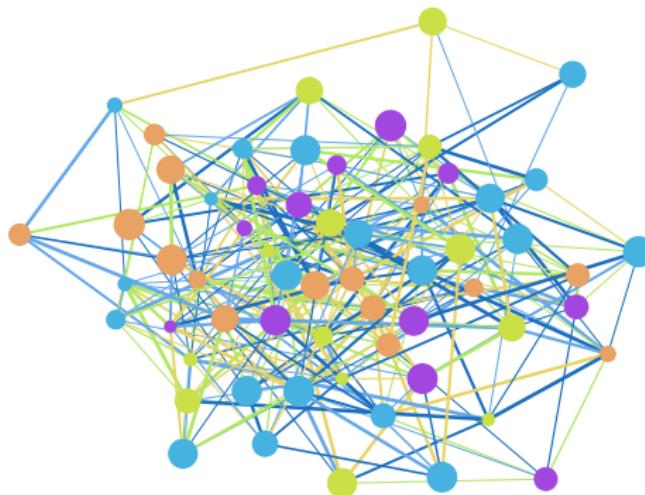
## Example undirected graph obtained



...

- └ The matching problem
- └ Experimental solutions

## Example undirected graph obtained



...

└ The matching problem

└ Experimental solutions

Exercice 4 : Random directed graph.

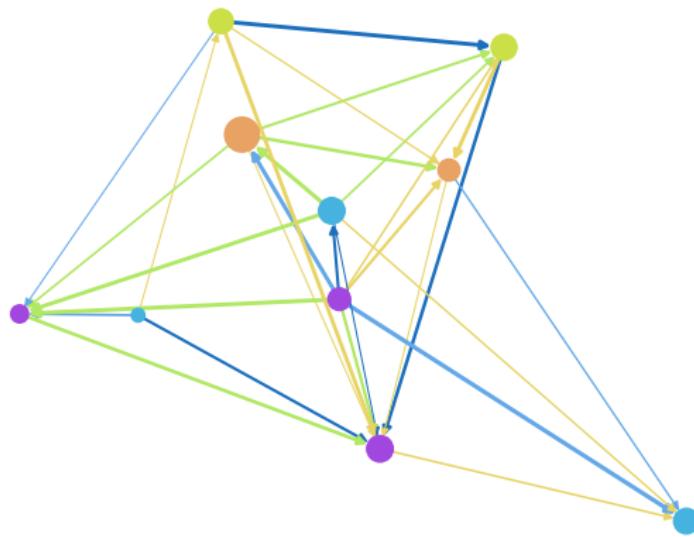
- ▶ Please use **random\_directed\_graph.py** to build a **directed** graph with a chosen number of vertices and **directed edges**.

...

- └ The matching problem

- └ Experimental solutions

## Example directed graph

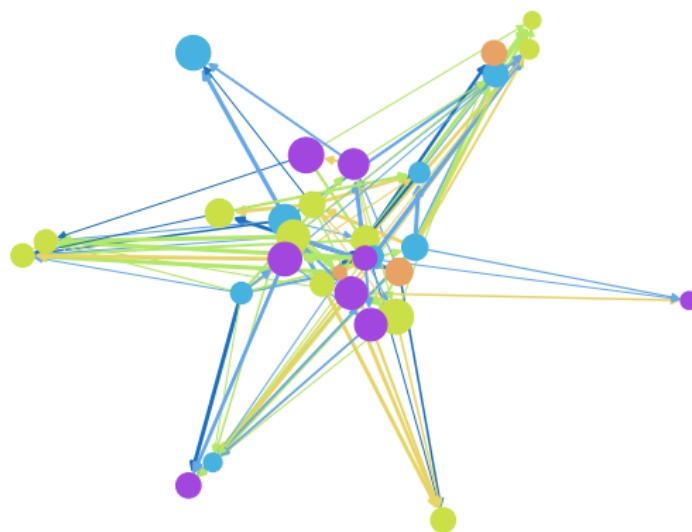


...

- └ The matching problem

- └ Experimental solutions

## Example directed graph

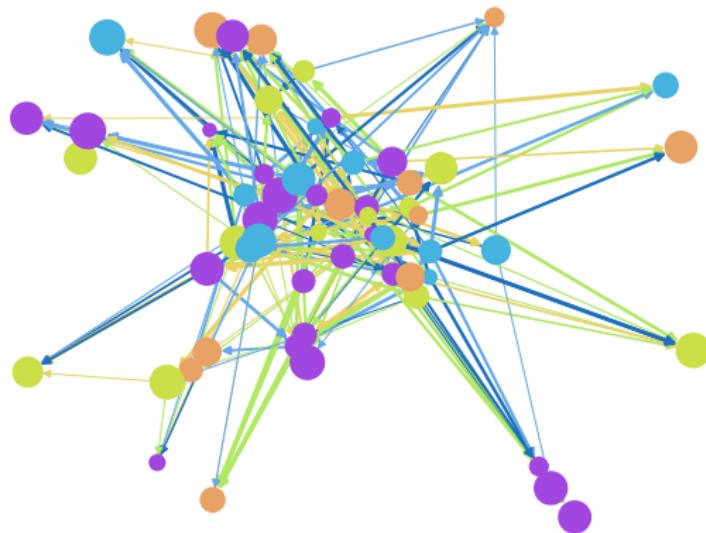


...

- └ The matching problem

- └ Experimental solutions

## Example directed graph

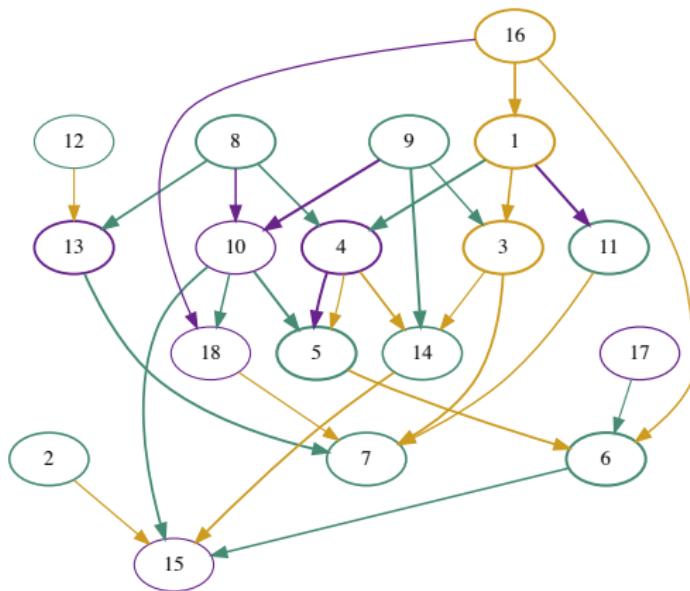


...

- └ The matching problem

- └ Experimental solutions

## Example directed graph



...

- └ The matching problem
- └ Experimental solutions

## Manual matching

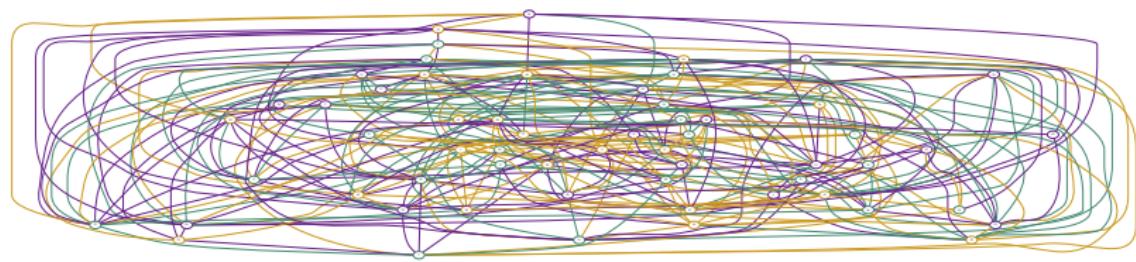
**Exercice 5 :** Please manually find an **optimal matching** in your **undirected** graph.

...

- └ The matching problem
- └ Experimental solutions

## Big graph

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



...

- └ The matching problem
  - └ Brute force algorithm

## Summary

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

...

└ The matching problem

  └ Brute force algorithm

## Brute force approach

### Exercice 6 : Enumeration

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

...

- └ The matching problem
- └ Brute force algorithm

## Brute force approach

### Exercice 6 : 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.

...

- └ The matching problem
- └ Brute force algorithm

## Brute force approach

### Exercice 6 : 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 ?

...

- └ The matching problem
- └ Brute force algorithm

## Brute force approach

### Exercice 6 : 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.

...

- └ The matching problem
- └ Brute force algorithm

## Brute force approach

### Exercice 6 : 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.

...

└ The matching problem

  └ Brute force algorithm

## Notion of complexity

- ▶ The **time complexity** of an algorithm is a measure of the **number of elementary** operations needed for the algorithm to terminate with respect to the input size.

...

└ The matching problem

  └ Brute force algorithm

## Brute force search

Exercice 7 : 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 matching problem
- └ Brute force algorithm

## Brute force search

Exercice 7 : 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.

...

└ The matching problem

  └ Brute force algorithm

## Brute force search

**Exercice 7 :** Complexity of brute force Assume that checking a subset requires 1 microsecond. How long should we wait in order to check all possible matching in a graph with 100 nodes ?

...

- └ The matching problem
- └ Brute force algorithm

## Other example of complexities

- ▶ linear search
- ▶ dichotomic search

...

- └ The matching problem
  - └ Greedy algorithm

## 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)

...

- └ The matching problem
- └ Greedy algorithm

## Algorithms

- ▶ Hence, we need different algorithms to solve the problem.

...

- └ The matching problem
- └ Greedy algorithm

## Algorithms

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

...

- └ The matching problem
  - └ Greedy algorithm

## Notion of maximal and maximum matching

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

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

...

- └ The matching problem
- └ Greedy algorithm

## 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 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.

...

└ The matching problem

└ Greedy algorithm

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

...

- └ The matching problem
- └ Greedy algorithm

## Maximum implies maximal

Let us show that a maximum matching is maximal.

...

- └ The matching problem
- └ Greedy algorithm

## Counter Example

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

...

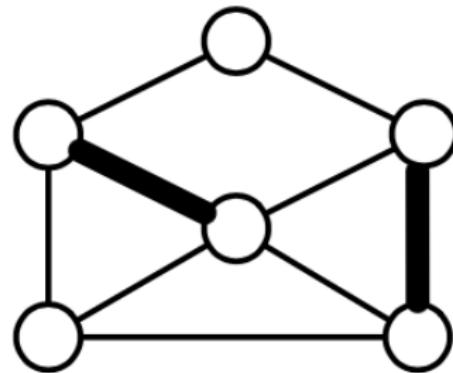
- └ The matching problem
- └ Greedy algorithm

## Counter Example

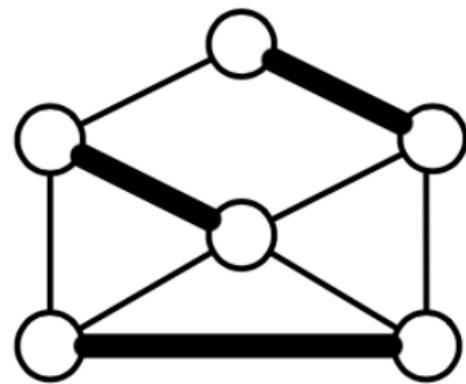
However, a matching that is maximal is **not necessary Maximum**.  
Can you find an example ?

...

- └ The matching problem
- └ Greedy algorithm



(a) A maximal matching not maximum



(b) A maximum matching

...

- └ The matching problem
- └ Greedy algorithm

## Greedy algorithm

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

...

- └ The matching problem

- └ Greedy algorithm

## 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

...

- └ The matching problem
- └ Greedy algorithm

## 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)

...

- └ The matching problem
  - └ Greedy algorithm

## Greedy algorithm

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

...

- └ The matching problem
  - └ Greedy algorithm

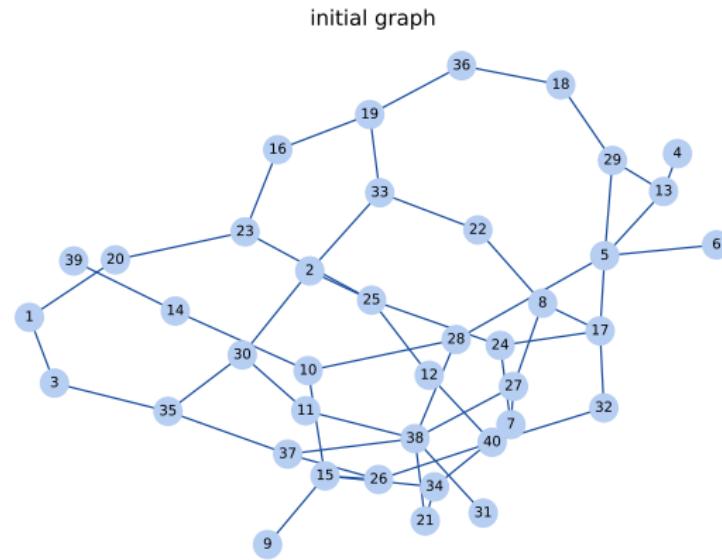
## Greedy algorithm

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

...

- └ The matching problem
- └ Greedy algorithm

**Exercice 8:** cd matching\_greedy/ and use generate\_graph.py to build a graph with at least 30 nodes. The images are stored in images/, data stored in data/



...

- └ The matching problem

- └ Greedy algorithm

## Implementing the greedy algorithm

Exercice 8 : Implement the greedy algorithm on this graph.

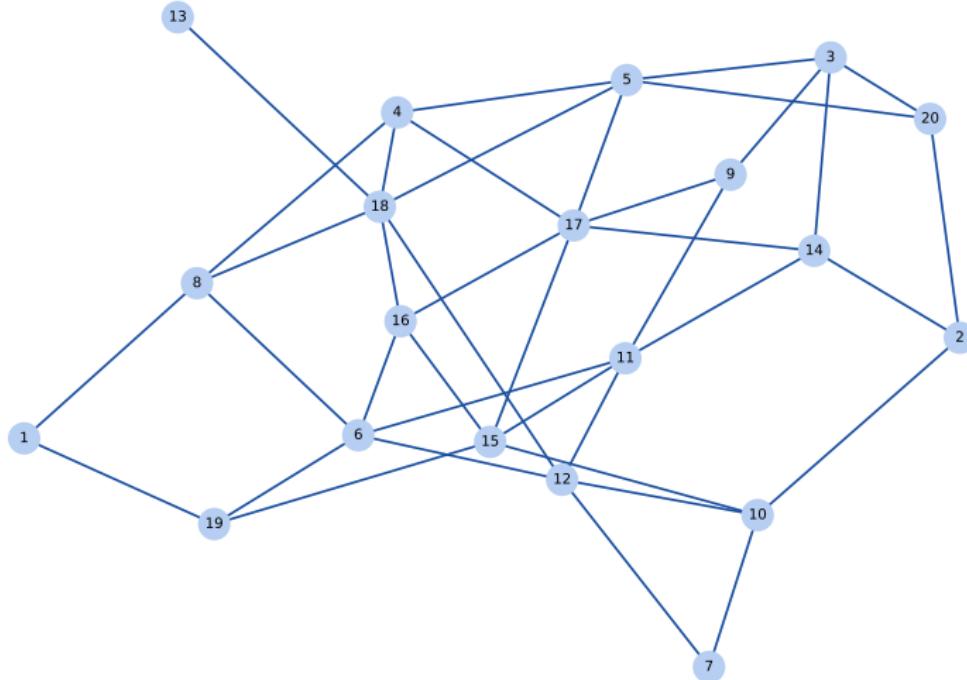
- ▶ Use the functions in **matching\_functions.py** and call them from **apply\_matching\_algorithm.py**
- ▶ More details in the file.

...

- └ The matching problem

- └ Greedy algorithm

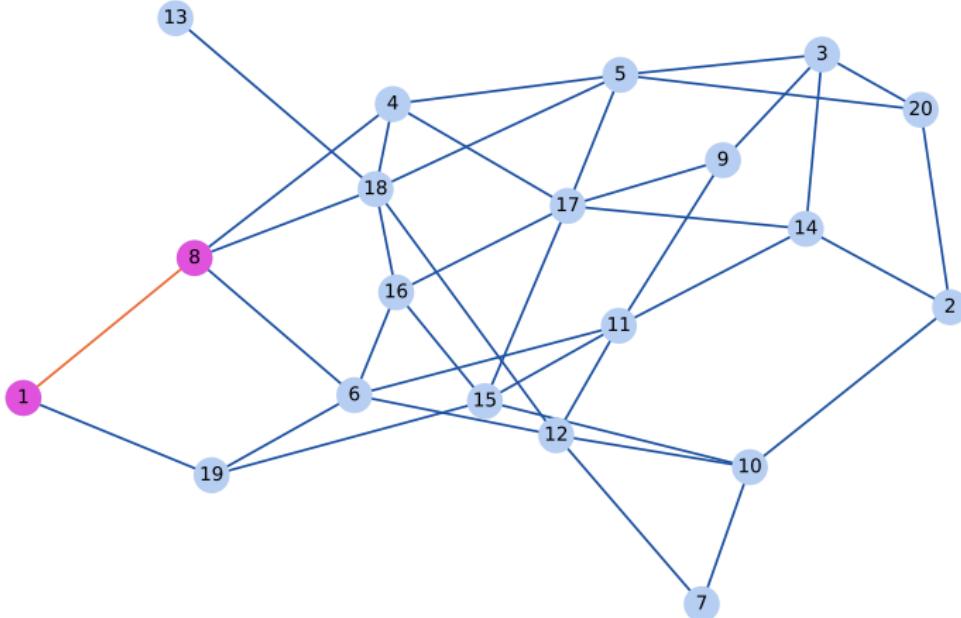
initial graph



...

- └ The matching problem
- └ Greedy algorithm

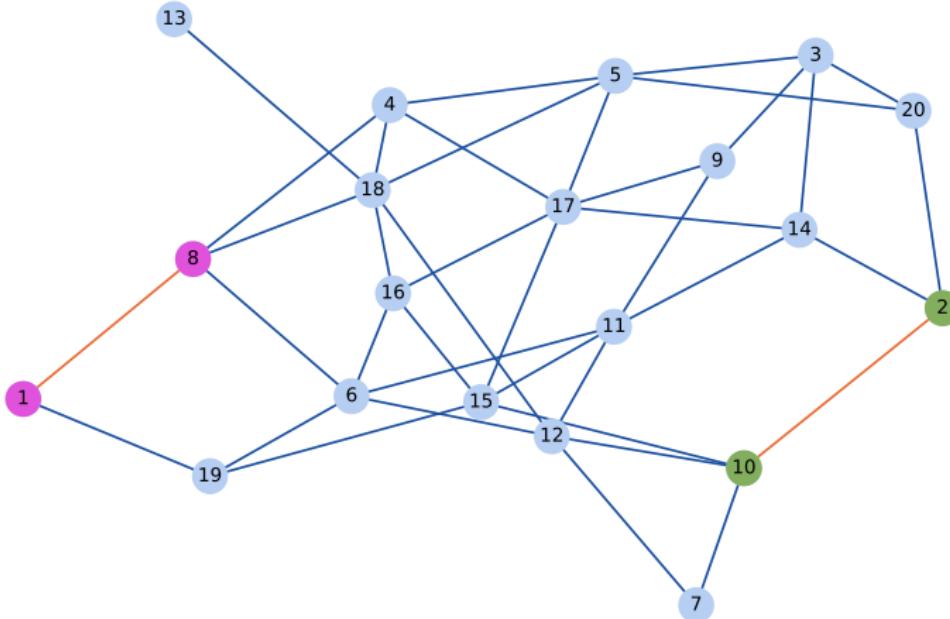
Matching size: 1  
Algo step: 1  
Nb nodes: 20



...

- └ The matching problem
- └ Greedy algorithm

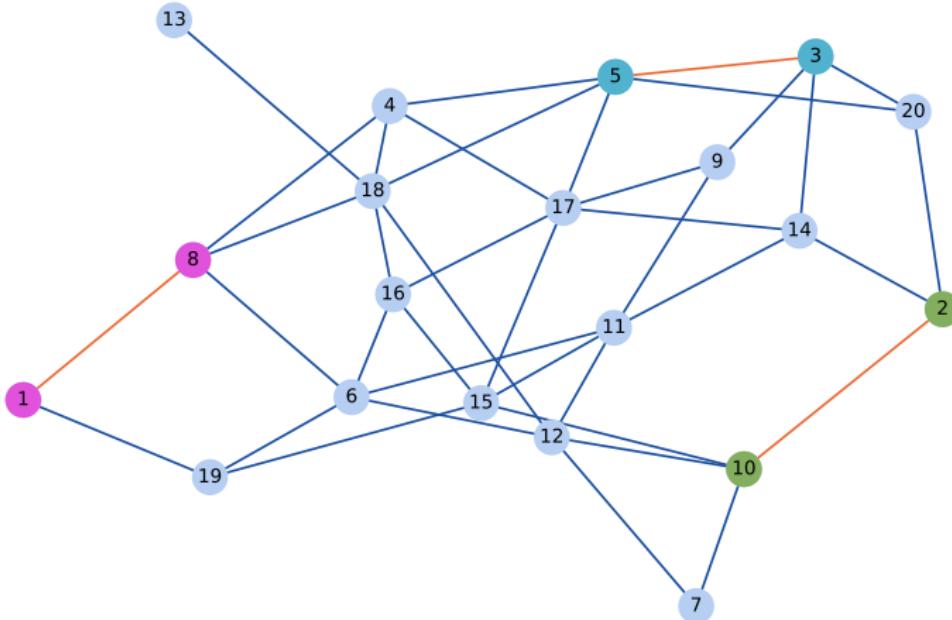
Matching size: 2  
Algo step: 3  
Nb nodes: 20



...

- └ The matching problem
- └ Greedy algorithm

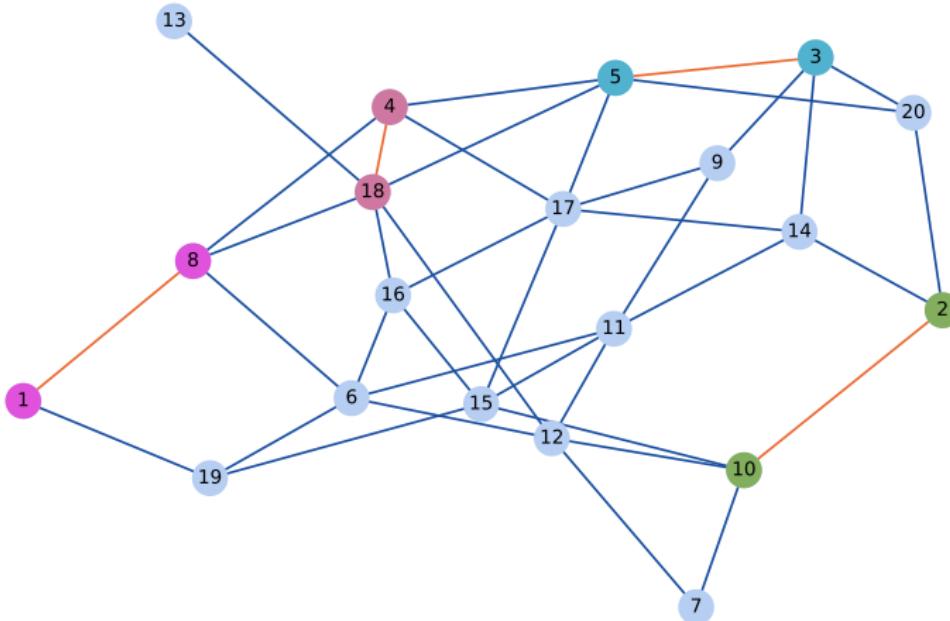
Matching size: 3  
Algo step: 6  
Nb nodes: 20



...

- └ The matching problem
- └ Greedy algorithm

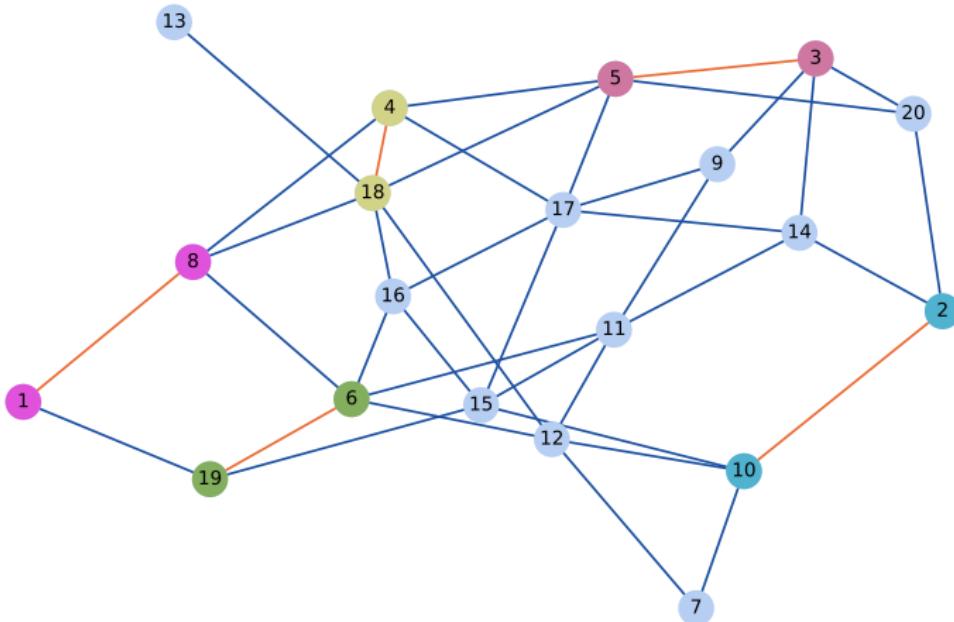
Matching size: 4  
Algo step: 11  
Nb nodes: 20



...

- └ The matching problem
- └ Greedy algorithm

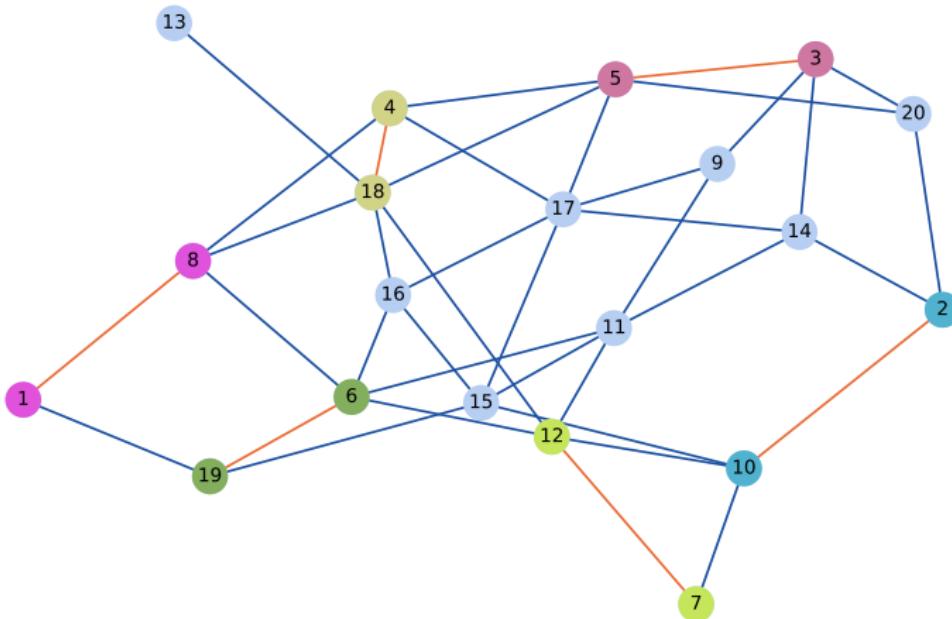
Matching size: 5  
Algo step: 17  
Nb nodes: 20



...

- └ The matching problem
- └ Greedy algorithm

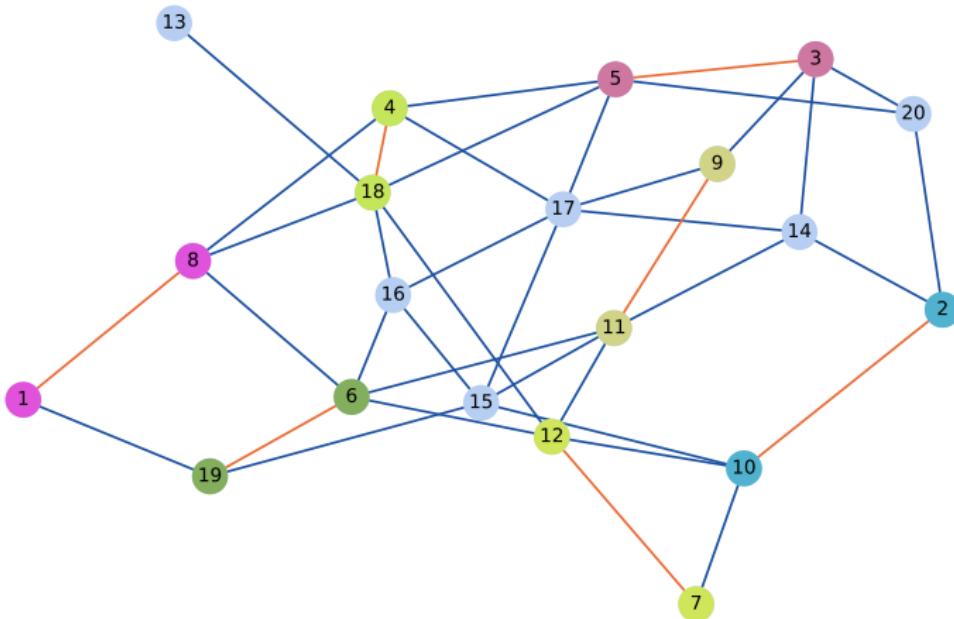
Matching size: 6  
Algo step: 22  
Nb nodes: 20



...

- └ The matching problem
- └ Greedy algorithm

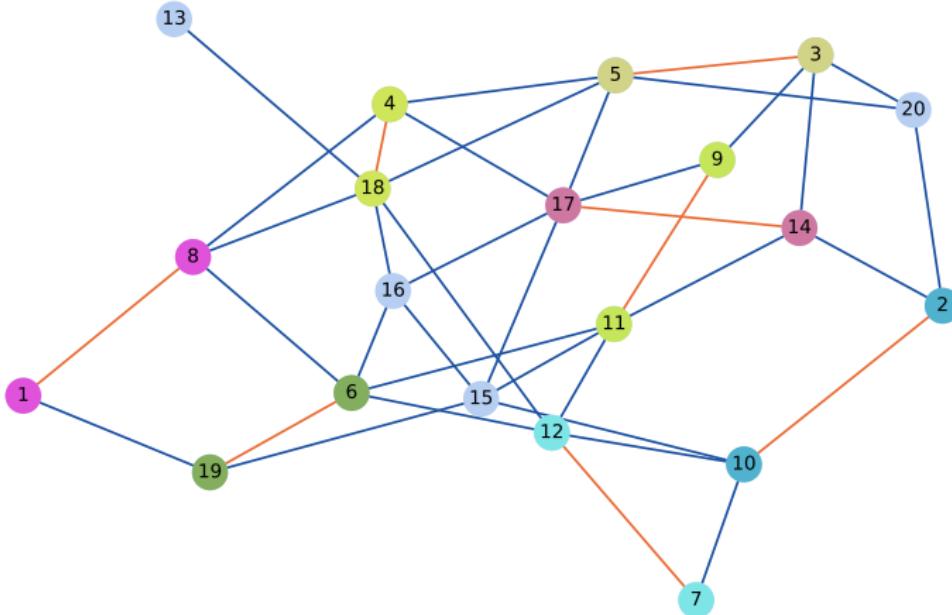
Matching size: 7  
Algo step: 25  
Nb nodes: 20



...

- └ The matching problem
- └ Greedy algorithm

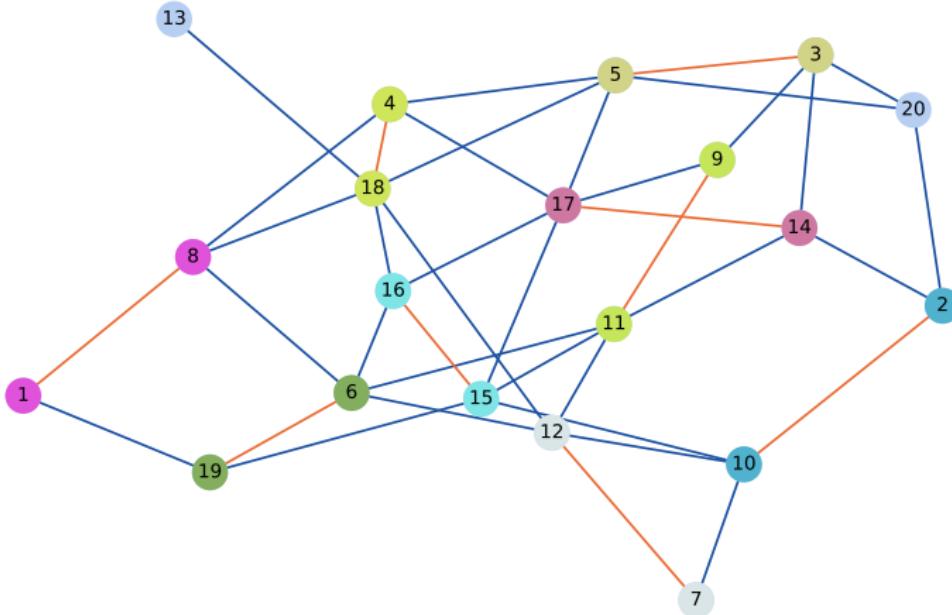
Matching size: 8  
Algo step: 34  
Nb nodes: 20



...

- └ The matching problem
- └ Greedy algorithm

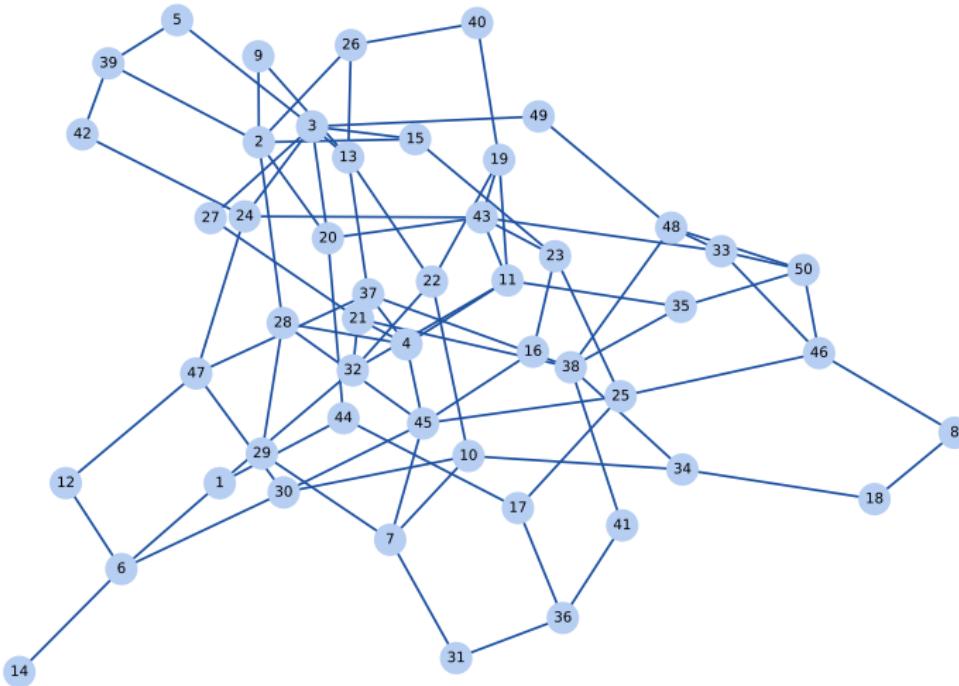
Matching size: 9  
Algo step: 36  
Nb nodes: 20



...

- └ The matching problem
- └ Greedy algorithm

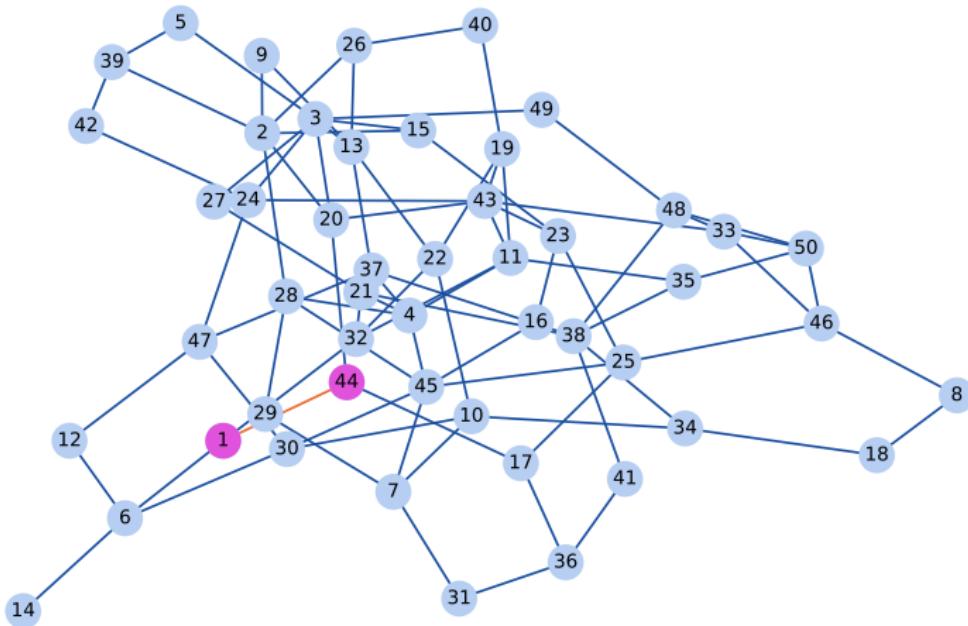
initial graph



...

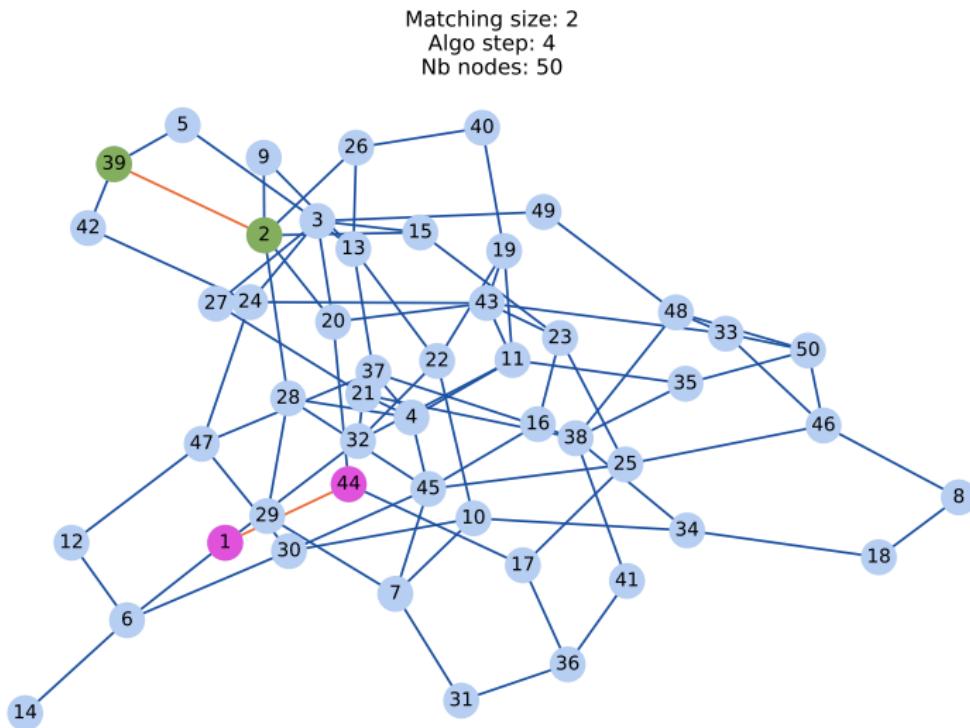
- └ The matching problem
- └ Greedy algorithm

Matching size: 1  
Algo step: 1  
Nb nodes: 50



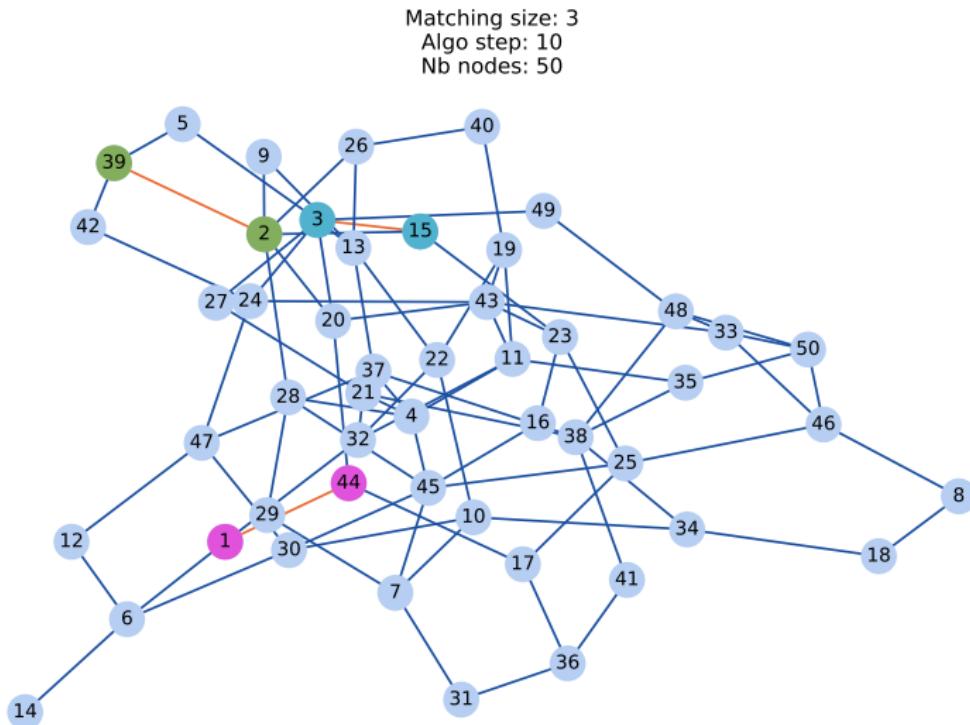
...

- └ The matching problem
- └ Greedy algorithm



...

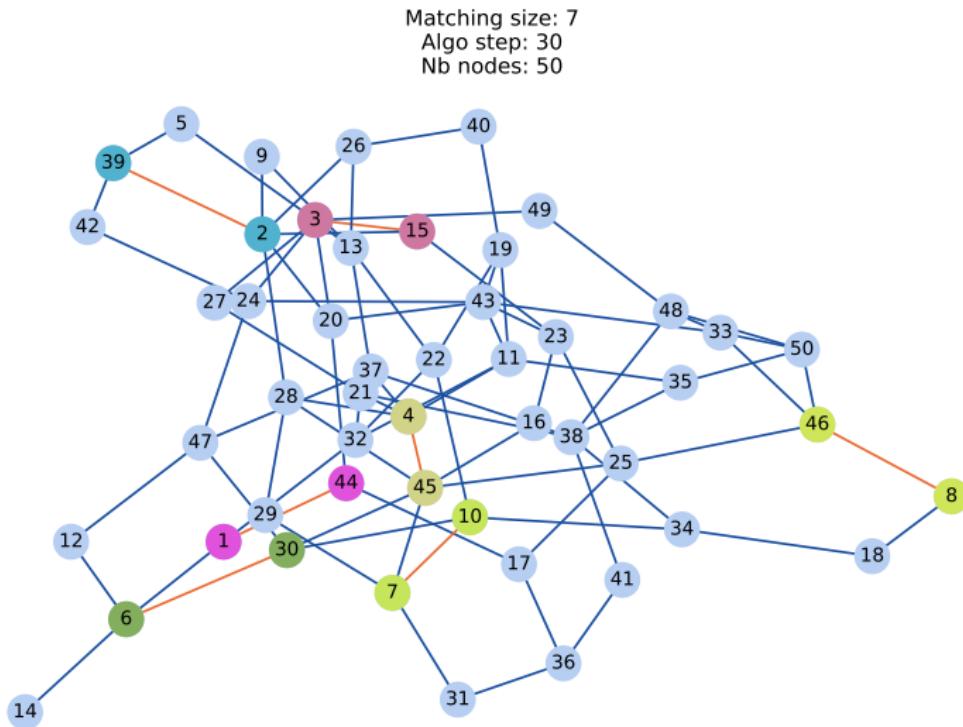
- └ The matching problem
- └ Greedy algorithm



...

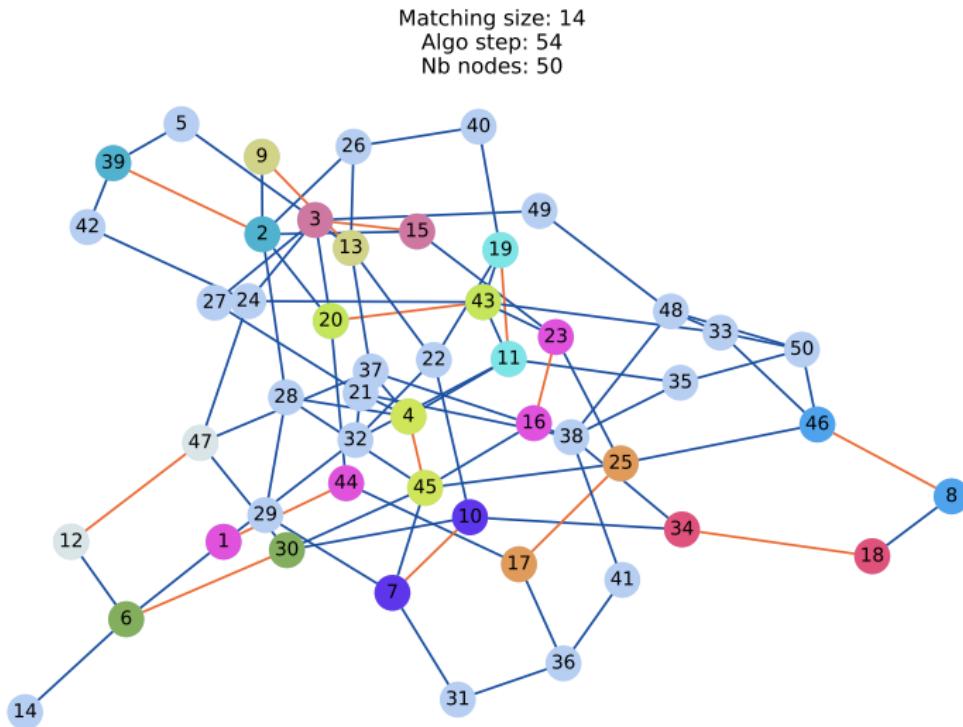
## The matching problem

### Greedy algorithm



...

- └ The matching problem
- └ Greedy algorithm



...

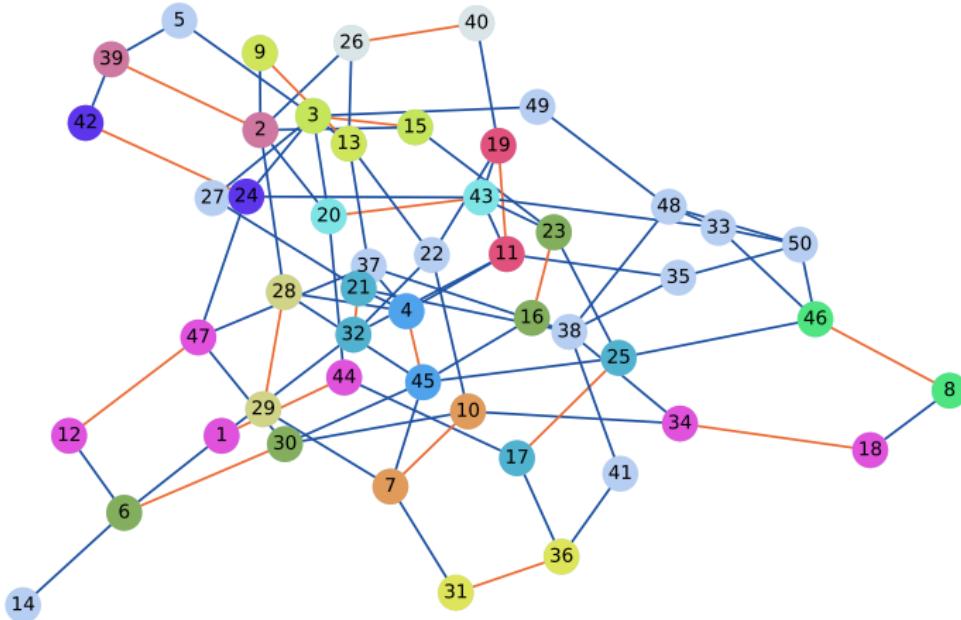
## The matching problem

### Greedy algorithm

Matching size: 19

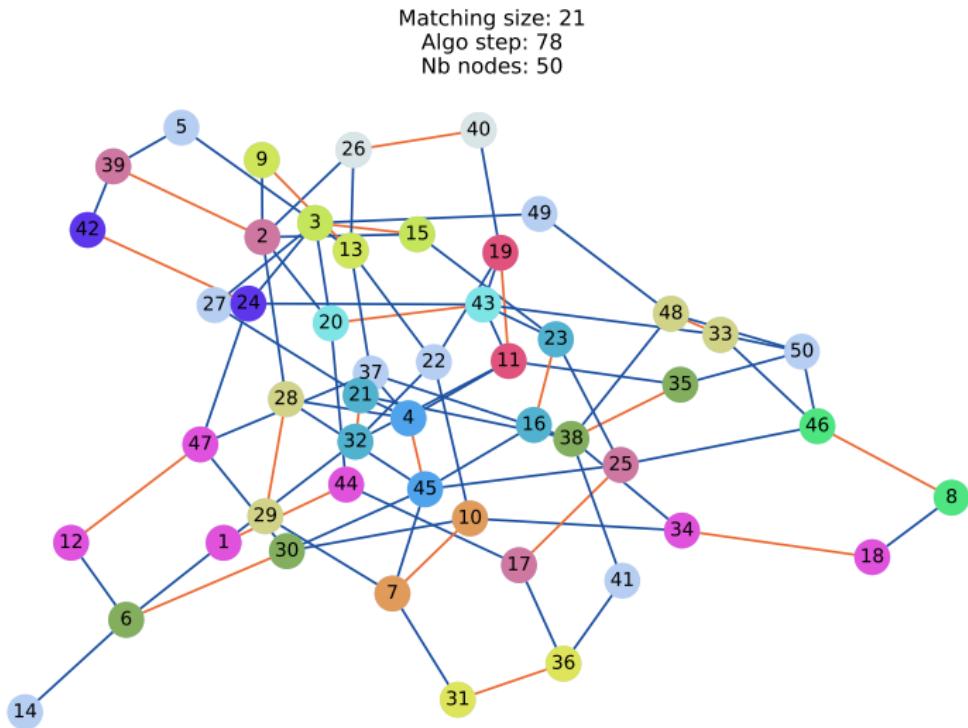
Algo step: 72

Nb nodes: 50



...

- └ The matching problem
- └ Greedy algorithm

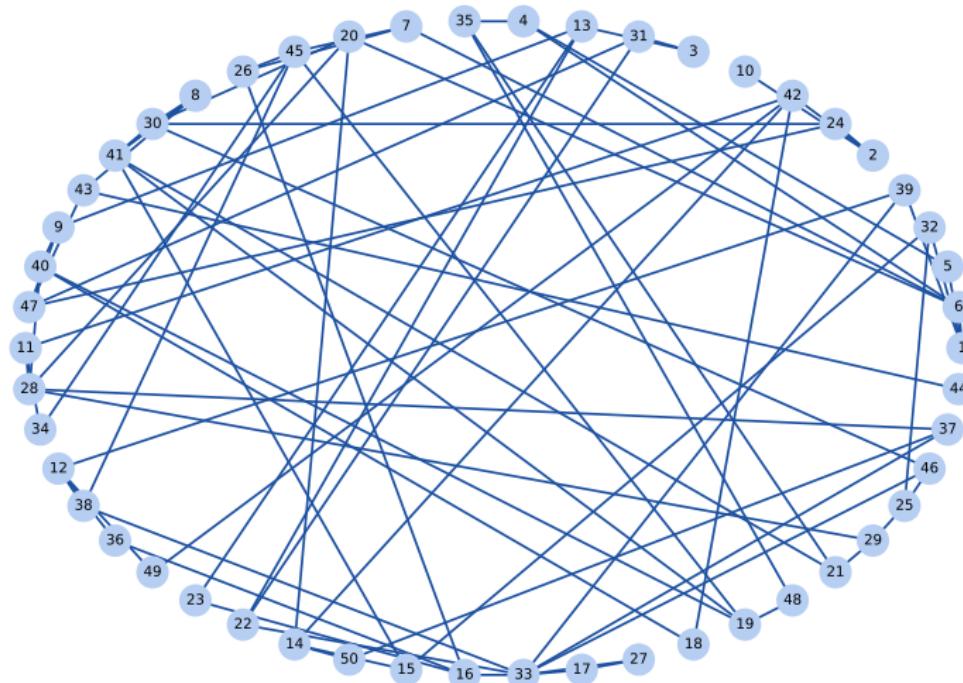


...

- The matching problem

- Greedy algorithm

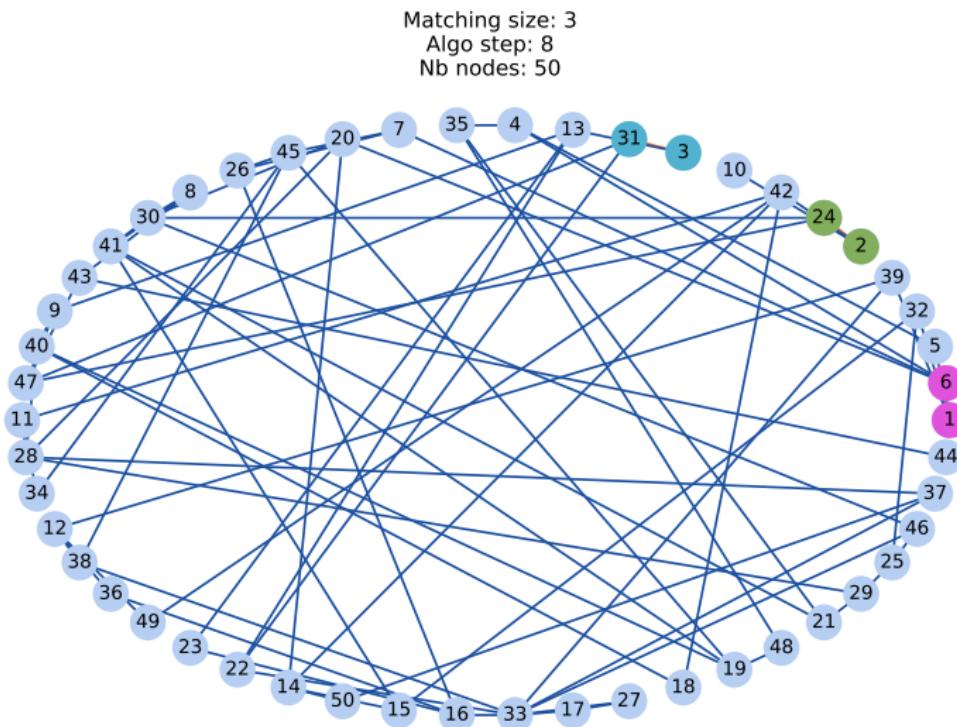
initial graph



...

## The matching problem

### Greedy algorithm



...

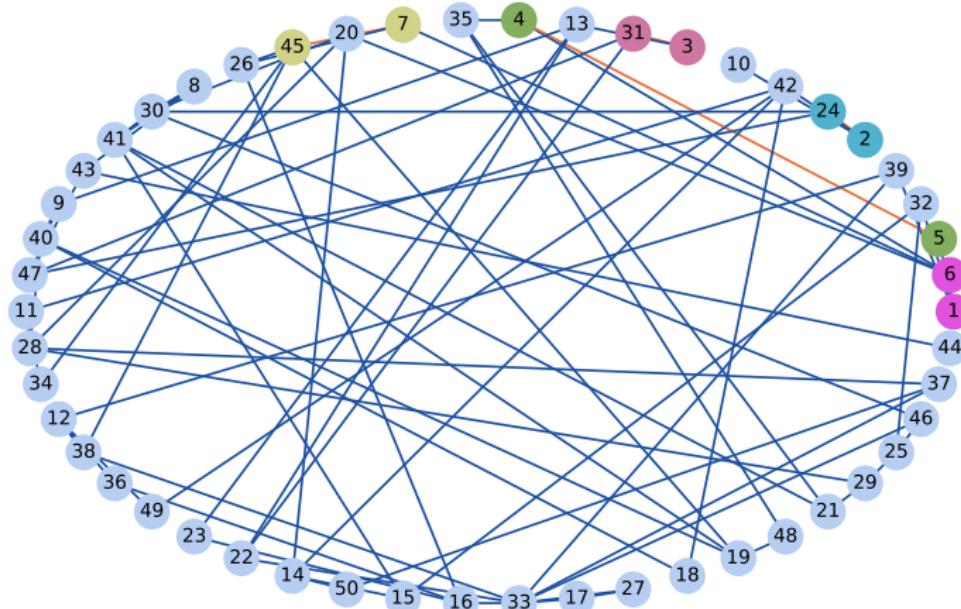
## The matching problem

### Greedy algorithm

Matching size: 5

Algo step: 15

Nb nodes: 50



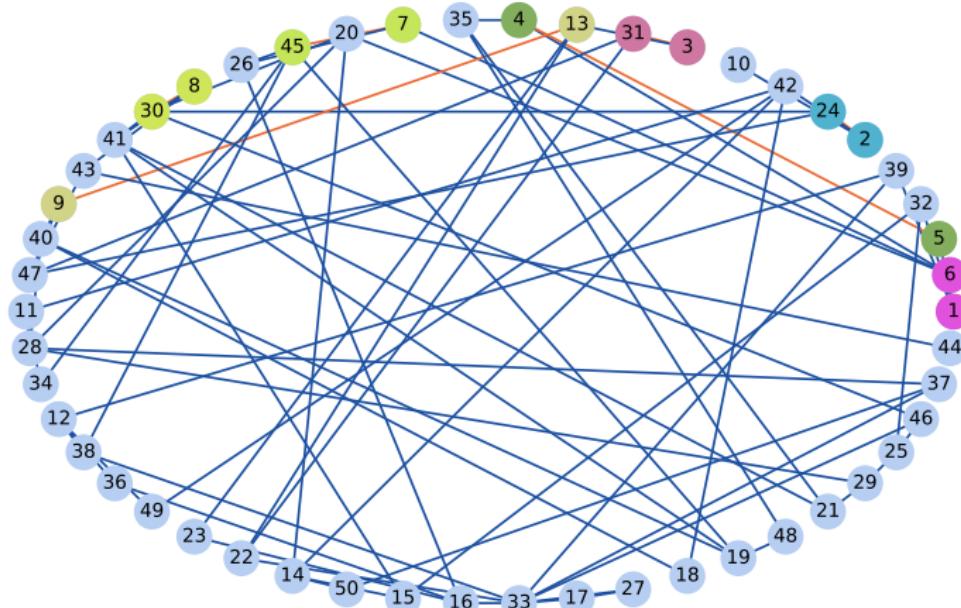
...

- └ The matching problem
- └ Greedy algorithm

Matching size: 7

Algo step: 20

Nb nodes: 50



...

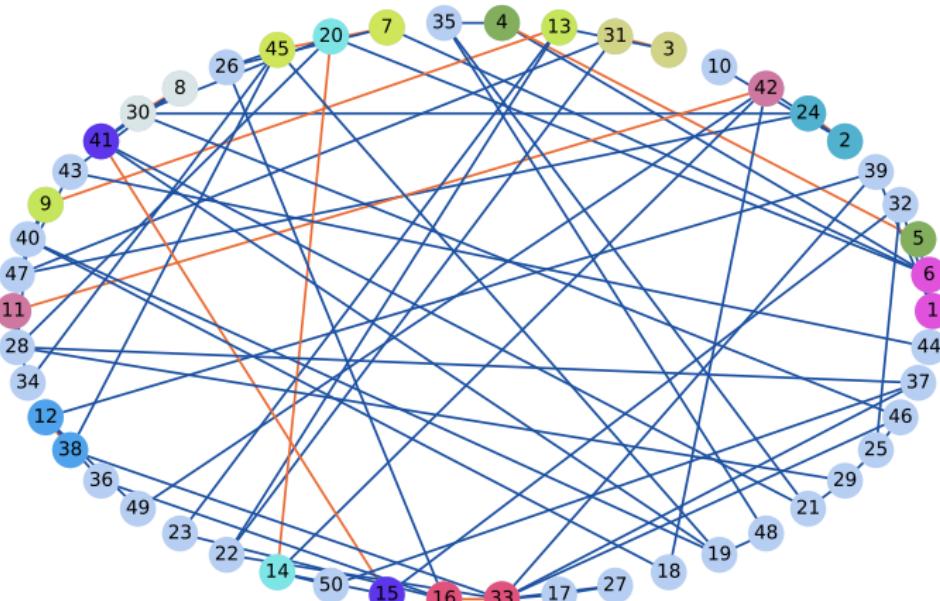
## The matching problem

### Greedy algorithm

Matching size: 12

Algo step: 38

Nb nodes: 50



...

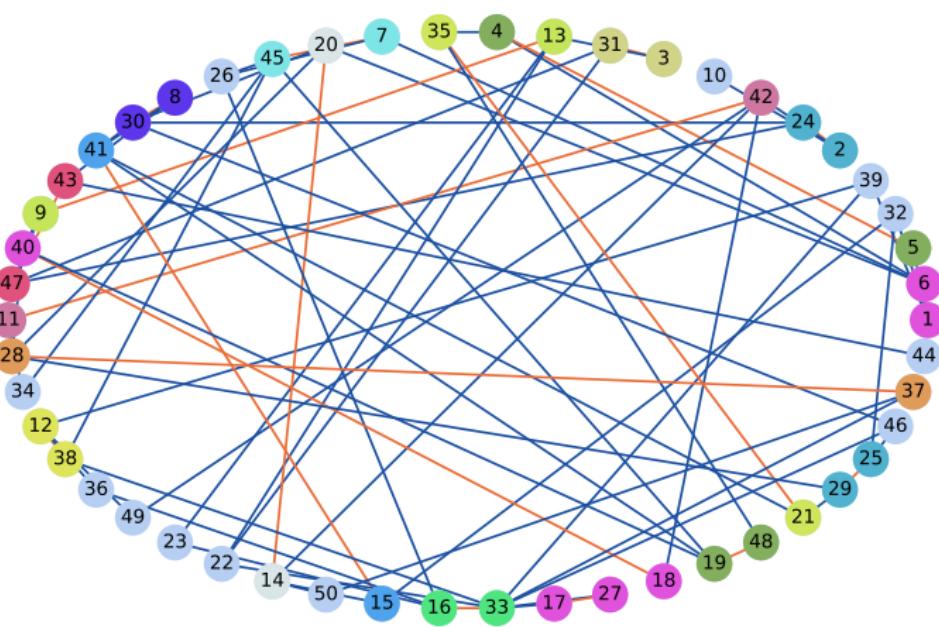
## The matching problem

### Greedy algorithm

Matching size: 19

Algo step: 79

Nb nodes: 50



...

- └ The matching problem
- └ Greedy algorithm

## Example

**Exercice 9:** 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 ?

...

- └ The matching problem
- └ Greedy algorithm

## Example

**Exercice 10:** 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 ?



...

- └ The matching problem
- └ Greedy algorithm

## 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 have :

$$|M| \geq \frac{|M^*|}{2} \quad (5)$$

...

└ The Maximum flow problem

## Changing the problem (for now)

We temporarily leave the maximum matching problem to focus on another problem : the **Maximum flow problem**

...

- └ The Maximum flow problem

- └ Presentation of the problem

# Max flow



Figure: Optimizing the quantity of something transported from one place to another, under constraints

...

## The Maximum flow problem

### Presentation of the problem

# Example

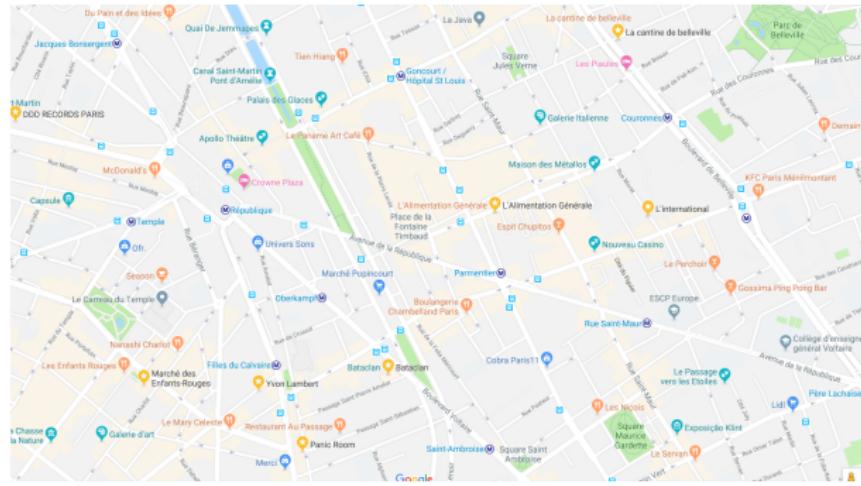


Figure: Optimizing the quantity of something transported from one place to another, under constraints

...

└ The Maximum flow problem

  └ Presentation of the problem

## Formalizing the problem

We introduce the concept of **flow network** (*reseau de flot*).

...

└ The Maximum flow problem

└ Presentation of the problem

## Formalizing the problem

- ▶ A **Directed graph**  $G = (E, V)$

...

- └ The Maximum flow problem

- └ Presentation of the problem

## Formalizing the problem

We introduce the concept of **flow network** (reseau de flot).

- ▶ A **Directed graph**  $G = (E, V)$
- ▶ Each edge  $(u, v)$  must have a **capacity**  $c(u, v) \geq 0$

...

- └ The Maximum flow problem

- └ Presentation of the problem

## Formalizing the problem

We introduce the concept of **flow network** (reseau de flot).

- ▶ A **Directed graph**  $G = (E, V)$
- ▶ Each edge  $(u, v)$  must have a **capacity**  $c(u, v) \geq 0$
- ▶ We define two special nodes : a **source**  $E$  and a **sink**  $S$ .

...

└ The Maximum flow problem

  └ Presentation of the problem

## Formalizing the problem

We introduce the concept of **flow network (reseau de flot)**.

- ▶ A **Directed graph**  $G = (E, V)$
- ▶ Each edge  $(u, v)$  must have a **capacity**  $c(u, v) \geq 0$
- ▶ We define two special nodes : a **source**  $E$  and a **sink**  $S$ .

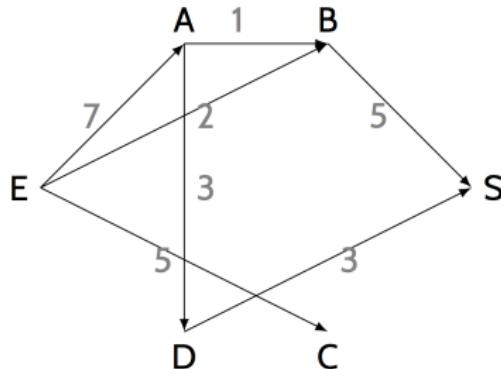


Figure: A **flow network (reseau de flot)** with capacities

...

- └ The Maximum flow problem

- └ Presentation of the problem

## Formalizing the problem

We introduce the concept of **flow network** (reseau de flot).

- ▶ A **Directed graph**  $G = (E, V)$
- ▶ Each edge  $(u, v)$  must have a **capacity**  $c(u, v) \geq 0$
- ▶ We define two special nodes : a **source**  $E$  and a **sink**  $S$ .
- ▶ A **flow**  $f$  is a function  $f(u, v) \leq c(u, v)$  (+ additional constraints)

...

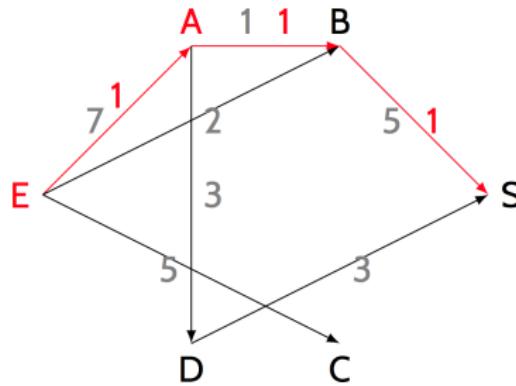
- The Maximum flow problem

- Presentation of the problem

## Formalizing the problem

We introduce the concept of **flow network** (reseau de flot).

- ▶ Each edge  $(u, v)$  must have a **capacity**  $c(u, v) \geq 0$
- ▶ A **flow**  $f$  is a function  $f(u, v) \leq c(u, v)$  (+ additional constraints)



...

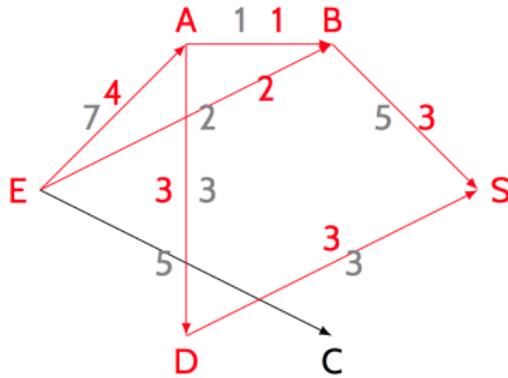
- The Maximum flow problem

- Presentation of the problem

## Formalizing the problem

We introduce the concept of **flow network** (reseau de flot).

- ▶ Each edge  $(u, v)$  must have a **capacity**  $c(u, v) \geq 0$
- ▶ A flow  $f$  is a function  $f(u, v) \leq c(u, v)$  (+ additional constraints)



...

└ The Maximum flow problem

  └ Presentation of the problem

## Conservation of the flow

We must have :

- ▶ antisymmetry :  $f(v, u) = -f(u, v)$
- ▶ flow conservation

...

- └ The Maximum flow problem

- └ Presentation of the problem

## conservation of the flow

we must have :

- ▶ antisymmetry :  $f(v, u) = -f(u, v)$
- ▶ flow conservation :  $\sum_{w \in V} f(u, w) = 0$  for any  $u \notin \{e, s\}$

...

- └ The Maximum flow problem

- └ Presentation of the problem

## Other formulation of the flow conservation

**Exercice 10:** Other formulation of the flow conservation

Let us show that for a flow  $f$ , we have for any node  $u \notin \{e, s\}$ :

$$\sum_{f(u,v)>0} f(u,v) = \sum_{f(v,u)>0} f(v,u) \quad (6)$$

...

- The Maximum flow problem

- Presentation of the problem

## Maximum flow

- The **value of the flow**, noted  $|f|$ , is  $\sum_{v \in S} f(E, v)$
- The problem is that of finding a flow with **maximum value**

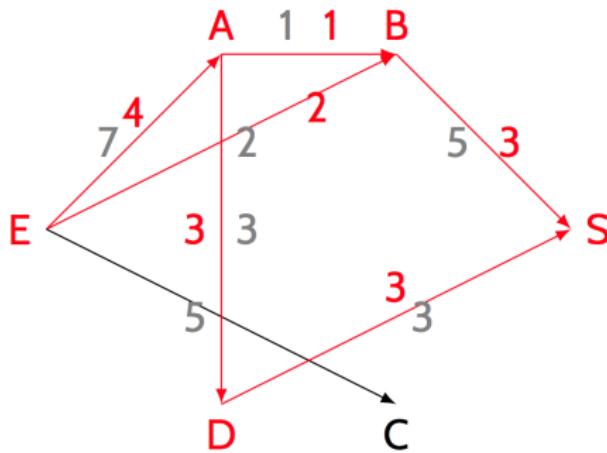


Figure: Max flow

...

└ The Maximum flow problem

└ Solution with the Ford-Fulkerson algorithm

## Ford Fulkerson algorithm

We will introduce an algorithm to solve the problem. This algorithm :

- ▶ terminates
- ▶ is correct
- ▶ is polynomial

...

└ The Maximum flow problem

└ Solution with the Ford-Fulkerson algorithm

## Ford Fulkerson algorithm

We will introduce an algorithm to solve the problem. This algorithm :

- ▶ terminates
- ▶ is correct
- ▶ is polynomial

So it is a good algorithm.

...

└ The Maximum flow problem

└ Solution with the Ford-Fulkerson algorithm

## Ford Fulkerson algorithm

We will introduce an algorithm to solve the problem. This algorithm :

- ▶ terminates
- ▶ is correct
- ▶ is polynomial

So it a good algorithm. **This section is going to be a little bit technical.**

...

- └ The Maximum flow problem

- └ Solution with the Ford-Fulkerson algorithm

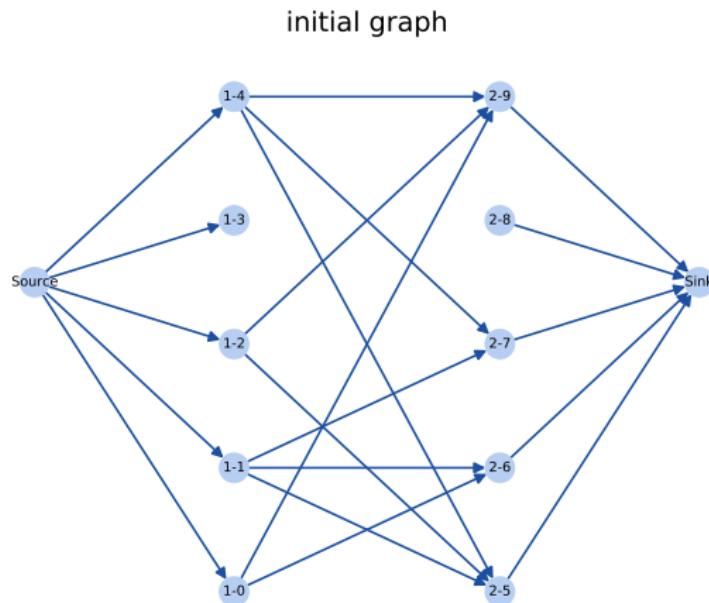
## Residual graph

- ▶ Given a graph with capacities  $c(u, v)$  and a flow  $f(u, v)$ , we will define its **residual graph** that has a capacity  $c_r(u, v)$  :

$$c_r(u, v) = c(u, v) - f(u, v) \quad (7)$$

- ...  
└ The Maximum flow problem  
  └ Solution with the Ford-Fulkerson algorithm

## Example of residual graph



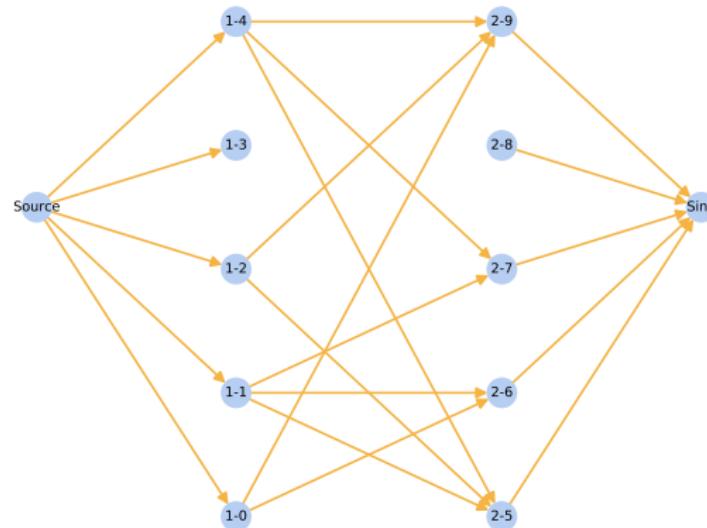
...

- └ The Maximum flow problem

- └ Solution with the Ford-Fulkerson algorithm

## Example of residual graph

residual graph step 1



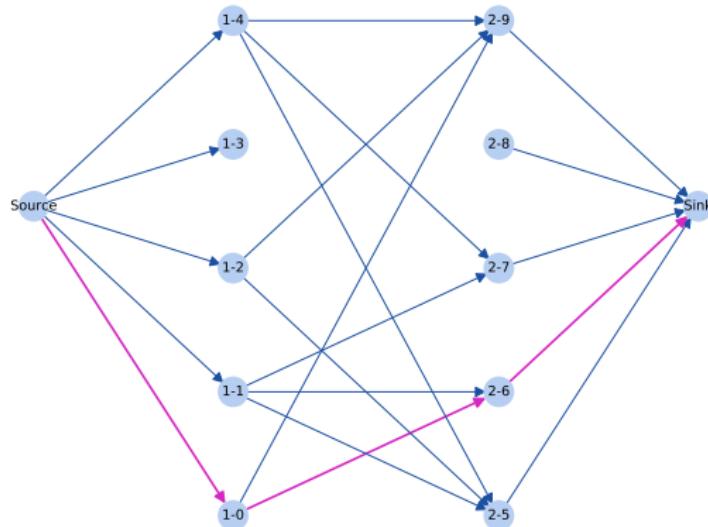
...

- └ The Maximum flow problem

- └ Solution with the Ford-Fulkerson algorithm

## Example of residual graph

augmenting path step 1

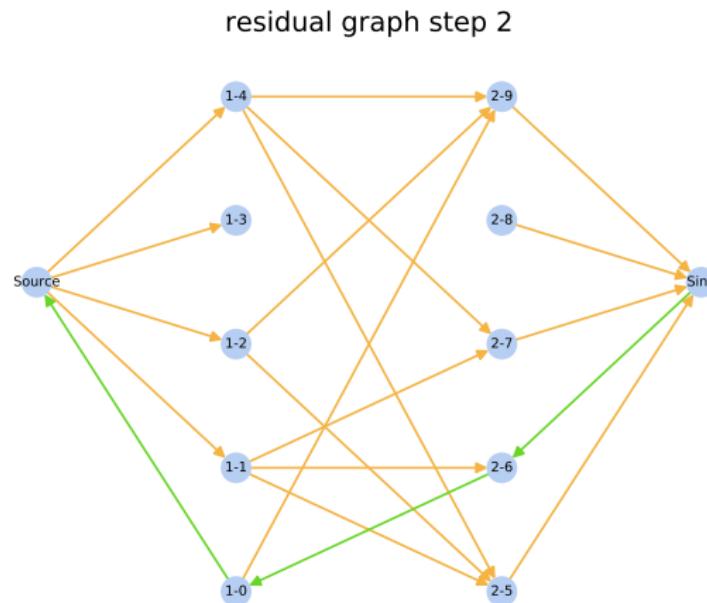


...

- └ The Maximum flow problem

- └ Solution with the Ford-Fulkerson algorithm

## Example of residual graph



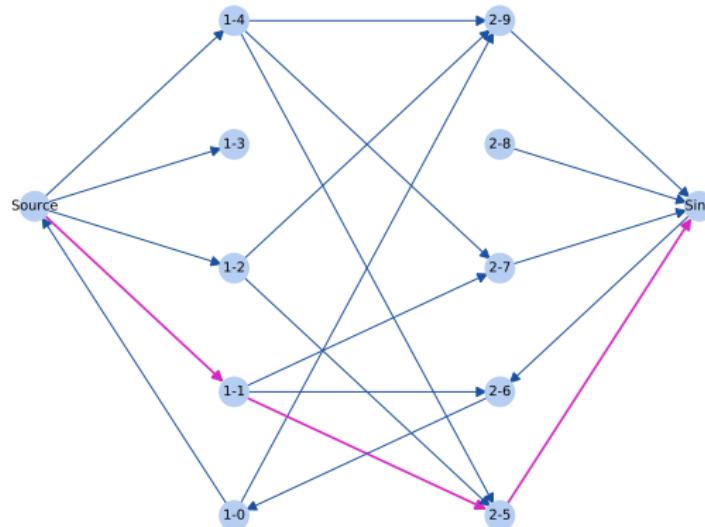
...

- └ The Maximum flow problem

- └ Solution with the Ford-Fulkerson algorithm

## Example of residual graph

augmenting path step 2

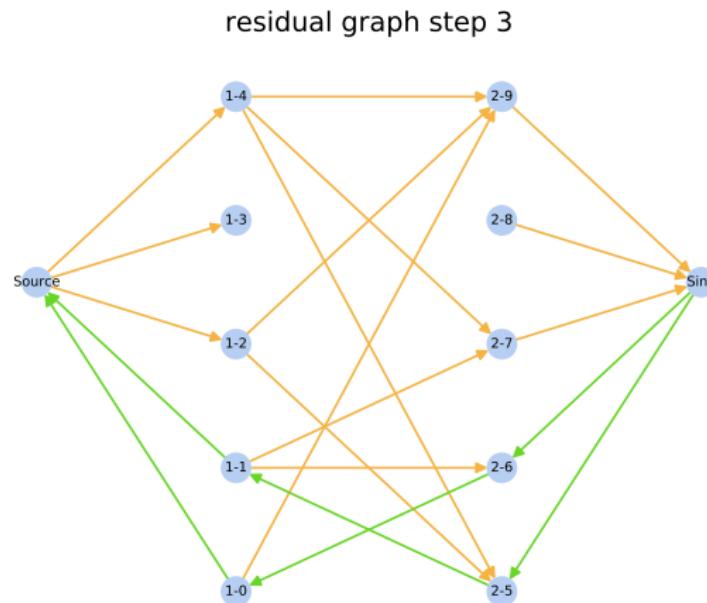


...

- └ The Maximum flow problem

- └ Solution with the Ford-Fulkerson algorithm

## Example of residual graph



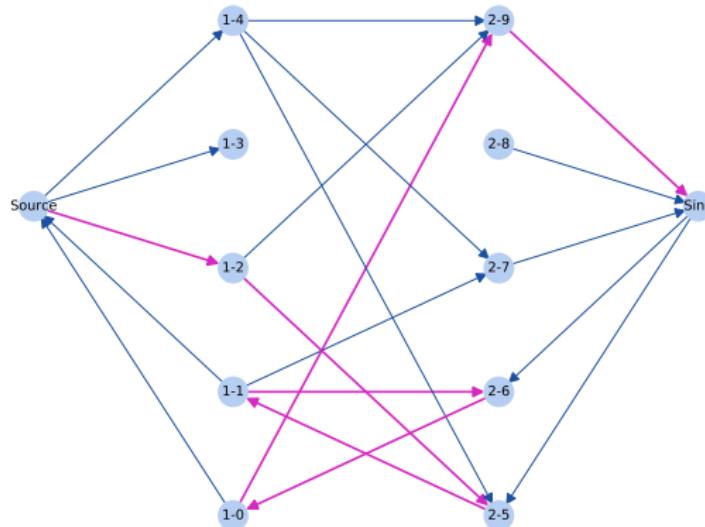
...

- └ The Maximum flow problem

- └ Solution with the Ford-Fulkerson algorithm

## Example of residual graph

augmenting path step 3



...

- └ The Maximum flow problem

- └ Solution with the Ford-Fulkerson algorithm

## Residual graph

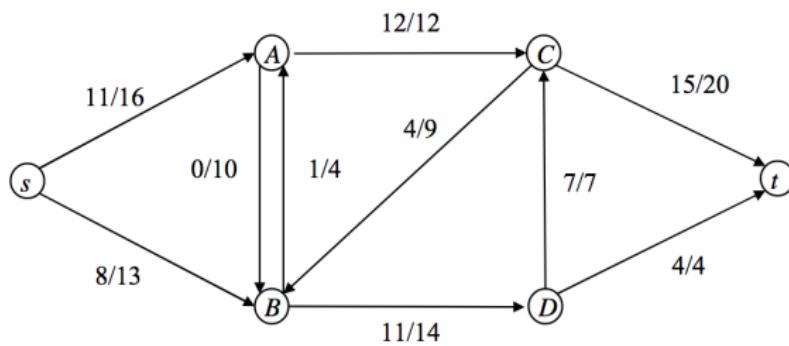


Figure: Another flow network

...

- └ The Maximum flow problem

- └ Solution with the Ford-Fulkerson algorithm

## Residual graph

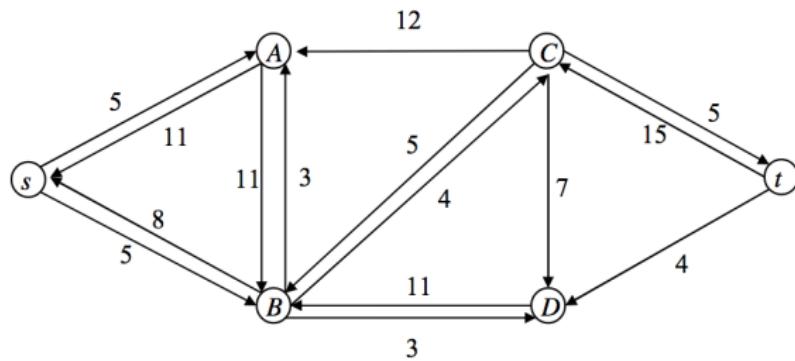


Figure: Residual graph

...

└ The Maximum flow problem

└ Solution with the Ford-Fulkerson algorithm

## Augmenting path

An augmenting path is a path in the **residual graph** from the source to the sink with capacities  $> 0$ .

...

- The Maximum flow problem

- Solution with the Ford-Fulkerson algorithm

## Augmenting path

An augmenting path is a path in the **residual graph** from the source to the sink with capacities  $> 0$ .

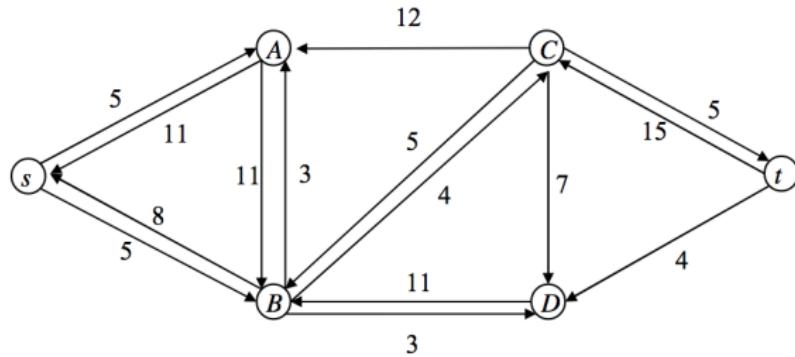


Figure: Residual graph

## Augmenting path

An augmenting path is a path from the source to the sink with capacities  $> 0$ .

The Ford-Fulkerson algorithm uses augmenting paths until there are no more augmenting paths.

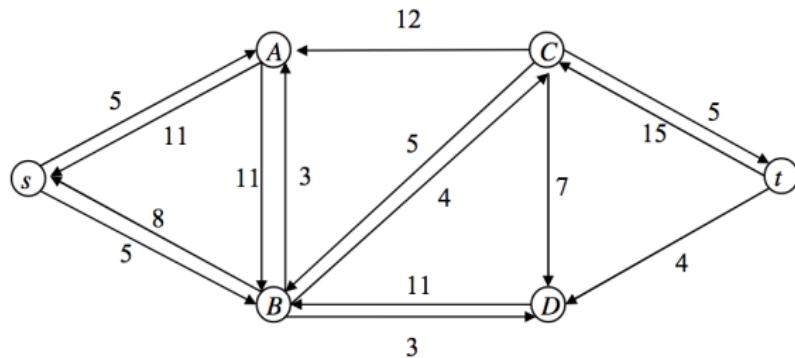


Figure: Residual graph

...

└ The Maximum flow problem

└ Solution with the Ford-Fulkerson algorithm

## Ford Fulkerson algorithm

Can you deduce the algorithm from the previous remarks ?

...

- └ The Maximum flow problem

- └ Solution with the Ford-Fulkerson algorithm

## Ford Fulkerson algorithm

**Result :** Flow  $f$

**for**  $(u, v) \in E$  **do**

  |  $f(u, v) = 0$

**end**

**while**  $\exists \rho$  augmenting path **do**

  | augment  $f$  with  $\rho$

**end**

return  $f$

**Algorithme 1 :** Ford Fulkerson algorithm

...

└ The Maximum flow problem

└ Solution with the Ford-Fulkerson algorithm

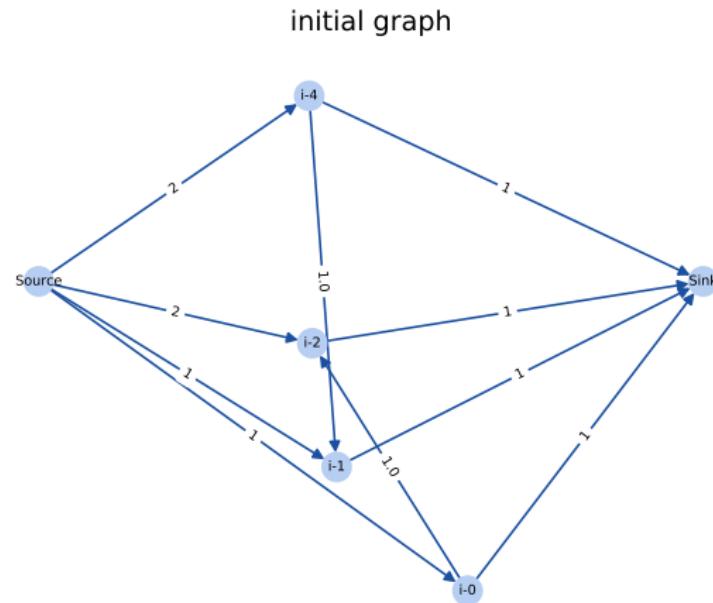
## Ford-Fulkerson algorithm

Let us some complete instances of the algorithm:

...

- └ The Maximum flow problem

- └ Solution with the Ford-Fulkerson algorithm

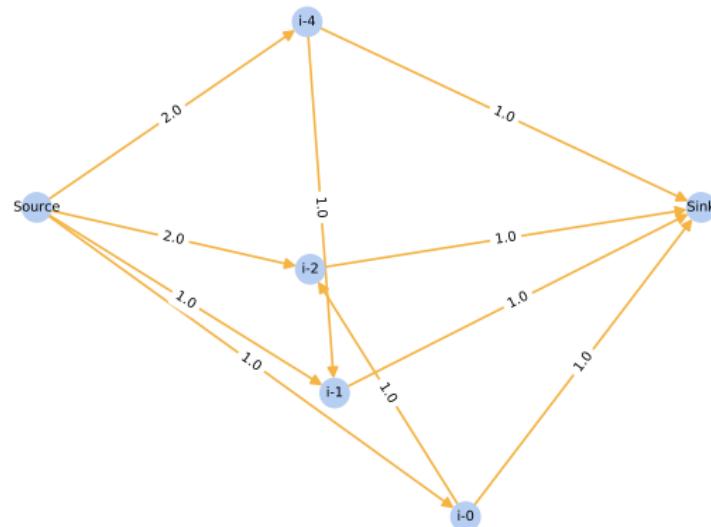


...

- └ The Maximum flow problem

- └ Solution with the Ford-Fulkerson algorithm

residual graph step 1

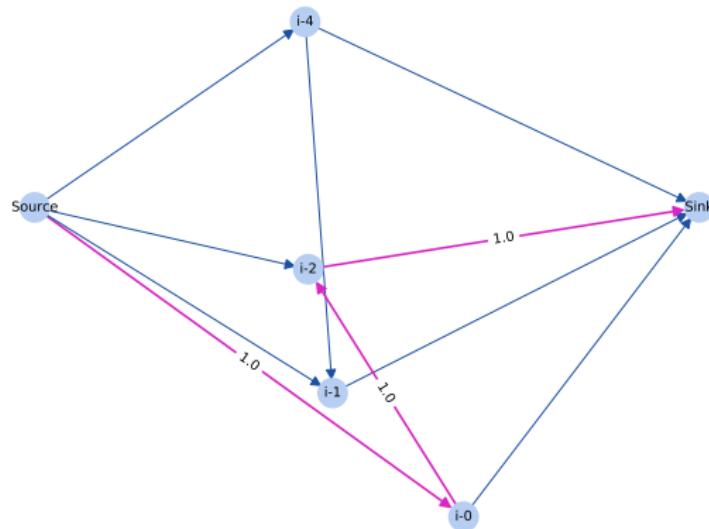


...

- └ The Maximum flow problem

- └ Solution with the Ford-Fulkerson algorithm

augmenting path step 1

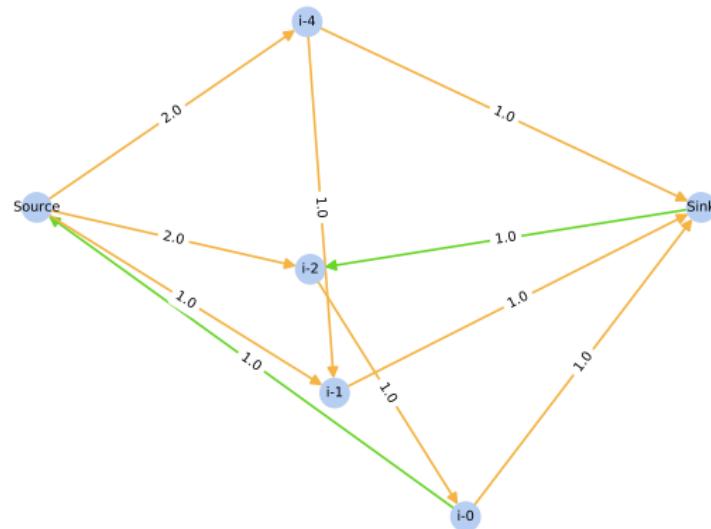


...

- └ The Maximum flow problem

- └ Solution with the Ford-Fulkerson algorithm

residual graph step 2

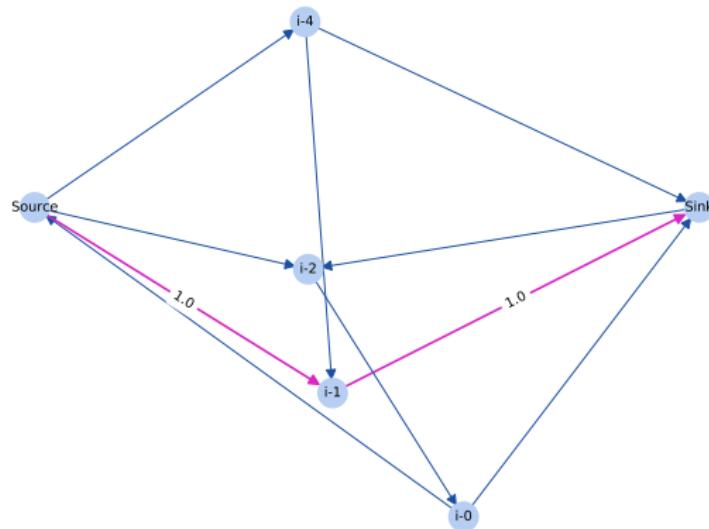


...

- └ The Maximum flow problem

- └ Solution with the Ford-Fulkerson algorithm

augmenting path step 2

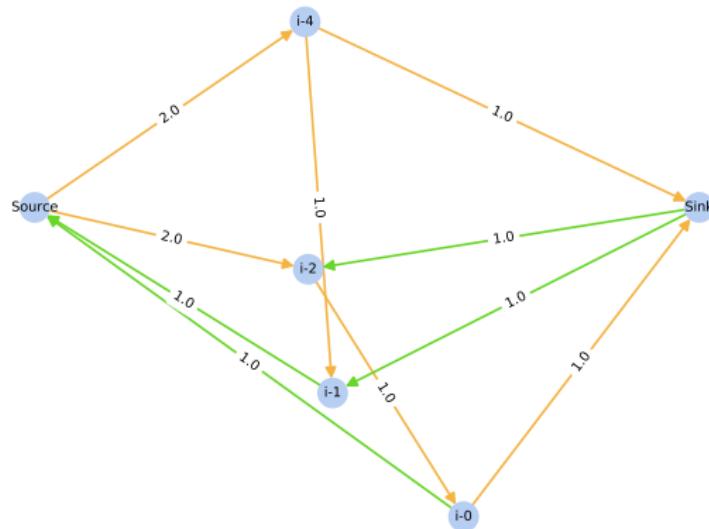


...

## The Maximum flow problem

### Solution with the Ford-Fulkerson algorithm

residual graph step 3

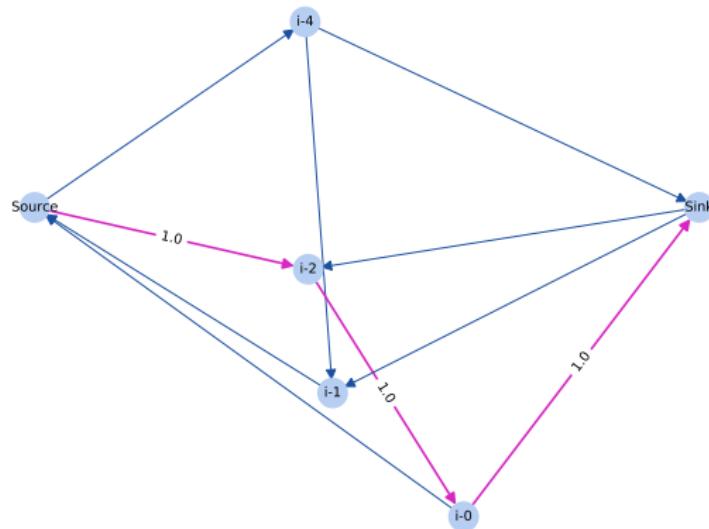


...

## The Maximum flow problem

### Solution with the Ford-Fulkerson algorithm

augmenting path step 3

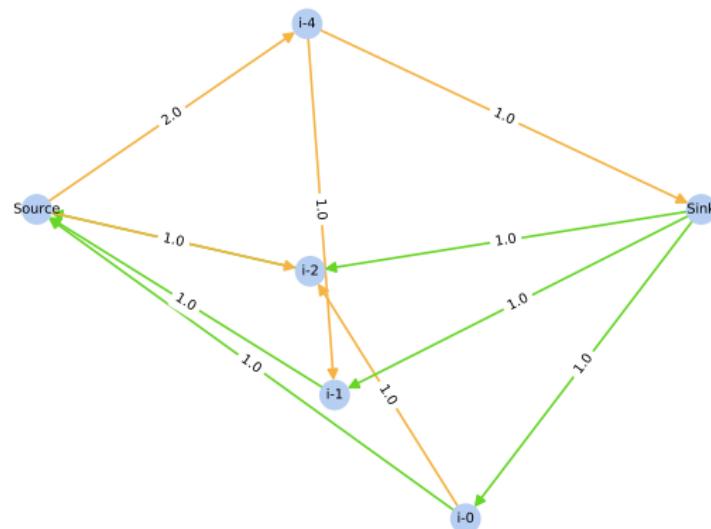


...

## The Maximum flow problem

### Solution with the Ford-Fulkerson algorithm

residual graph step 4

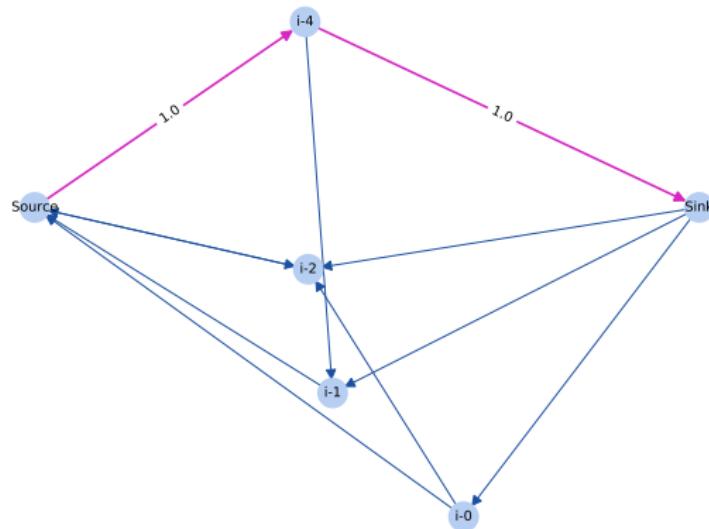


...

- └ The Maximum flow problem

- └ Solution with the Ford-Fulkerson algorithm

augmenting path step 4

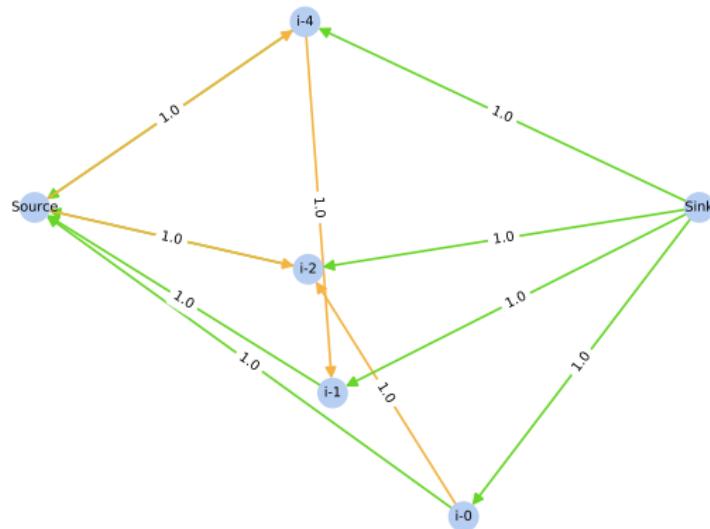


...

- └ The Maximum flow problem

- └ Solution with the Ford-Fulkerson algorithm

residual graph step 5



...

- └ The Maximum flow problem

- └ Solution with the Ford-Fulkerson algorithm

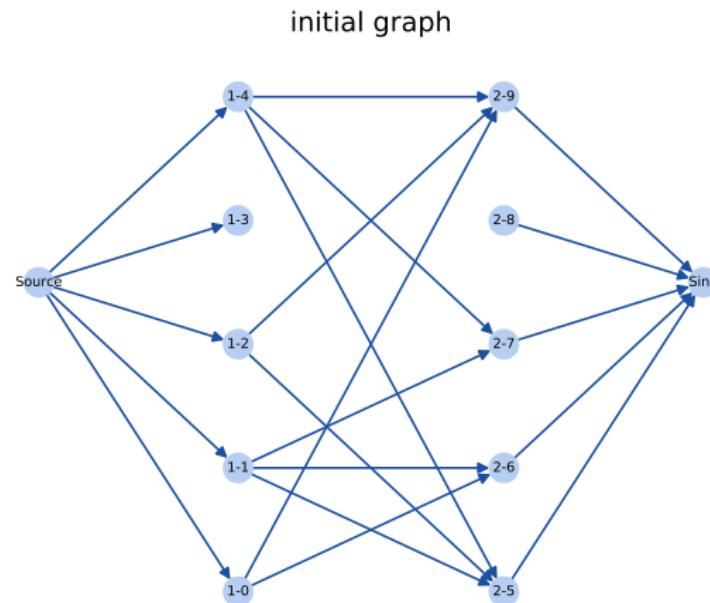


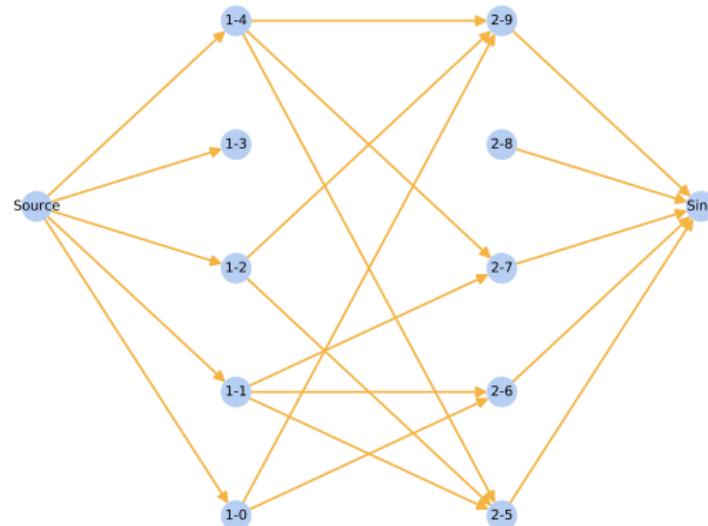
Figure: simple/initial

...

- └ The Maximum flow problem

- └ Solution with the Ford-Fulkerson algorithm

residual graph step 1

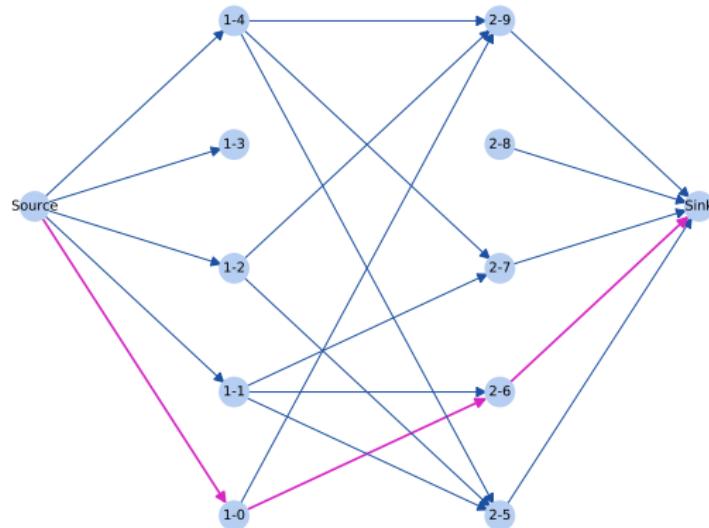


...

- └ The Maximum flow problem

- └ Solution with the Ford-Fulkerson algorithm

augmenting path step 1

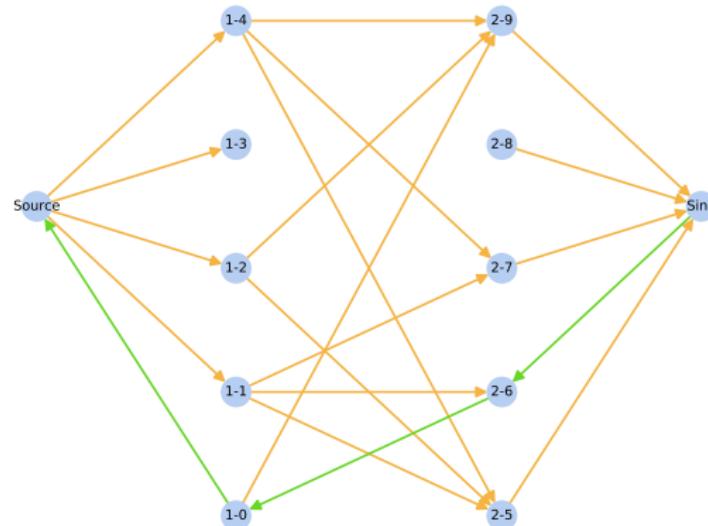


...

- └ The Maximum flow problem

- └ Solution with the Ford-Fulkerson algorithm

residual graph step 2

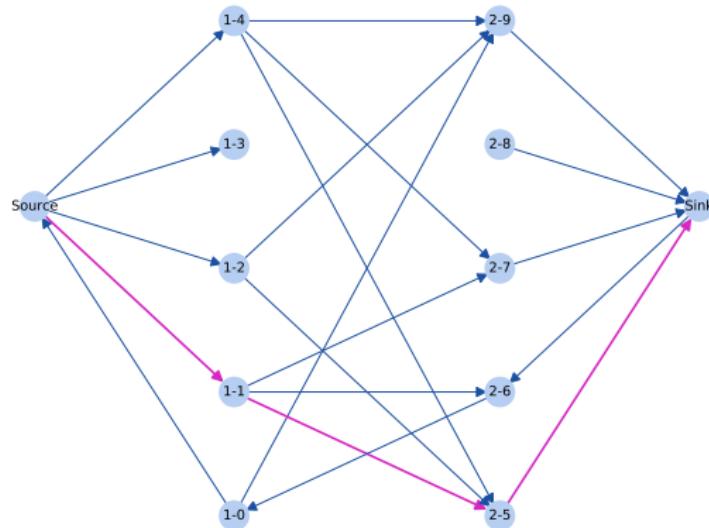


...

- └ The Maximum flow problem

- └ Solution with the Ford-Fulkerson algorithm

augmenting path step 2

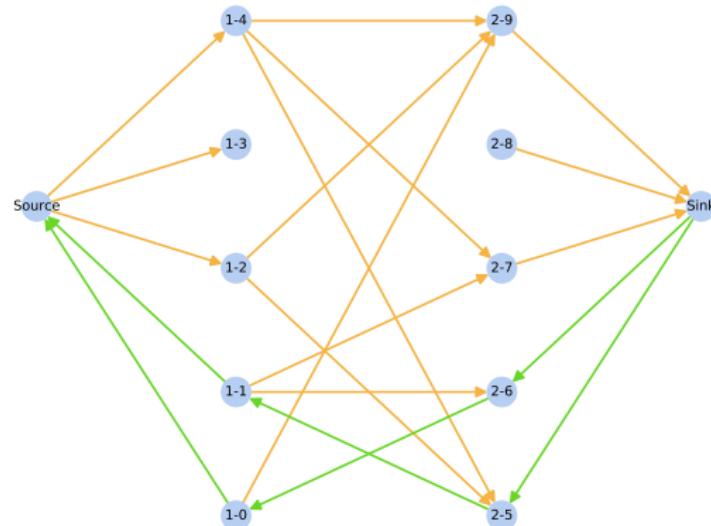


...

- └ The Maximum flow problem

- └ Solution with the Ford-Fulkerson algorithm

residual graph step 3

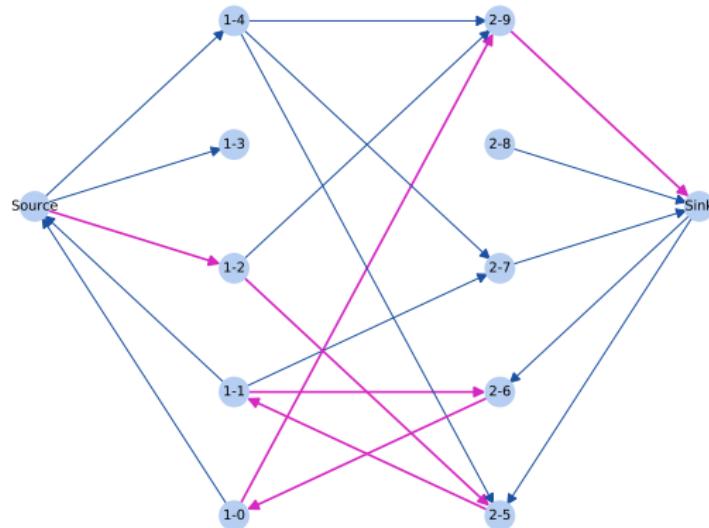


...

- └ The Maximum flow problem

- └ Solution with the Ford-Fulkerson algorithm

augmenting path step 3

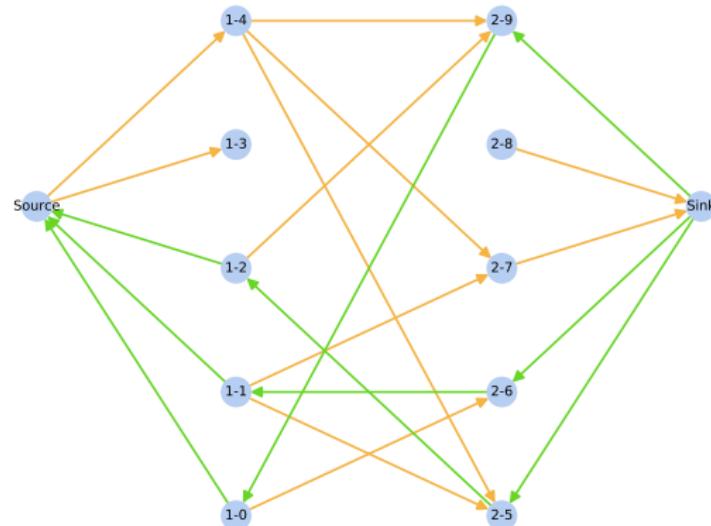


...

- └ The Maximum flow problem

- └ Solution with the Ford-Fulkerson algorithm

residual graph step 4

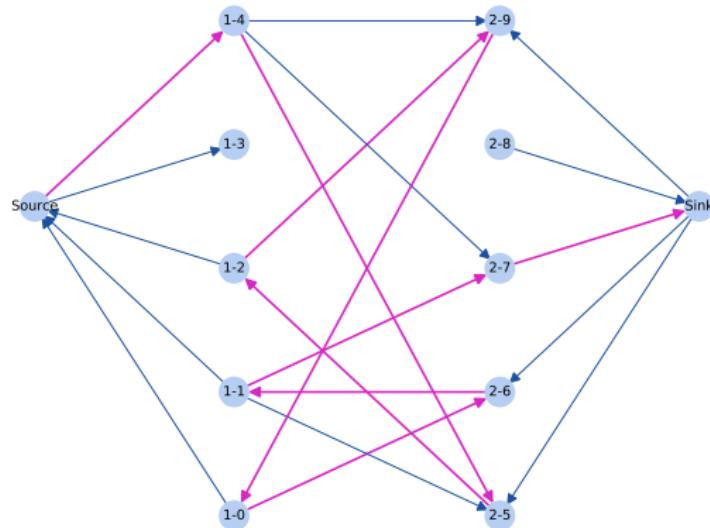


...

- └ The Maximum flow problem

- └ Solution with the Ford-Fulkerson algorithm

augmenting path step 4

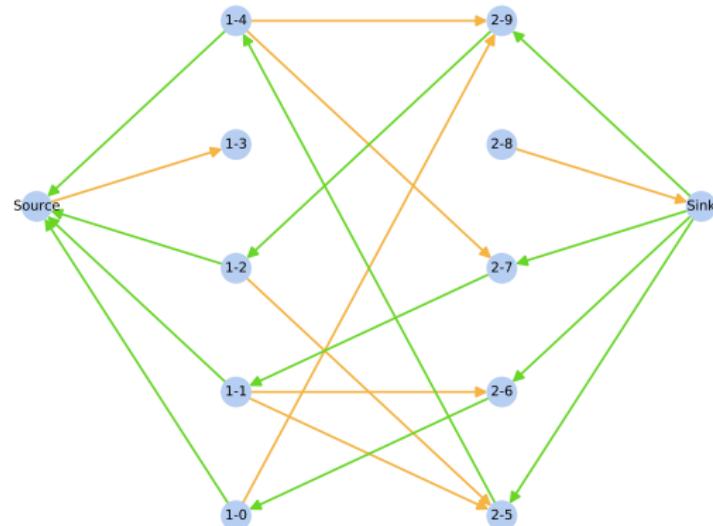


...

- └ The Maximum flow problem

- └ Solution with the Ford-Fulkerson algorithm

residual graph step 5



...

└ The Maximum flow problem

  └ Solution with the Ford-Fulkerson algorithm

## Ford Fulkerson algorithm

- ▶ We will implement the Ford Fulkerson algorithm (1956) on a general graph.

...

└ The Maximum flow problem

└ Solution with the Ford-Fulkerson algorithm

## Ford Fulkerson algorithm

**Exercice 11:** We will implement the Ford Fulkerson algorithm (1956)

- ▶ `cd ford_fulkerson/` and edit `generate_flow_network.py` to generate a flow network.

...

└ The Maximum flow problem

└ Solution with the Ford-Fulkerson algorithm

## Algorithm

- ▶ We will now use the functions contained in `ford_functions.py` and call them from `apply_ford_fulkerson.py`

...

└ The Maximum flow problem

└ Solution with the Ford-Fulkerson algorithm

# Algorithm

Exercice 12 : step 1

- ▶ Modify `find_augmenting_paths()` in order to find the augmenting paths.

...

└ The Maximum flow problem

└ Solution with the Ford-Fulkerson algorithm

# Algorithm

Exercice 12 : step 2

- ▶ now edit **augment\_flow()**

...

└ The Maximum flow problem

└ Solution with the Ford-Fulkerson algorithm

# Algorithm

Exercice 12 : step 3

- ▶ finally, edit the computation of the value of the flow

...

└ The Maximum flow problem

└ Solution with the Ford-Fulkerson algorithm

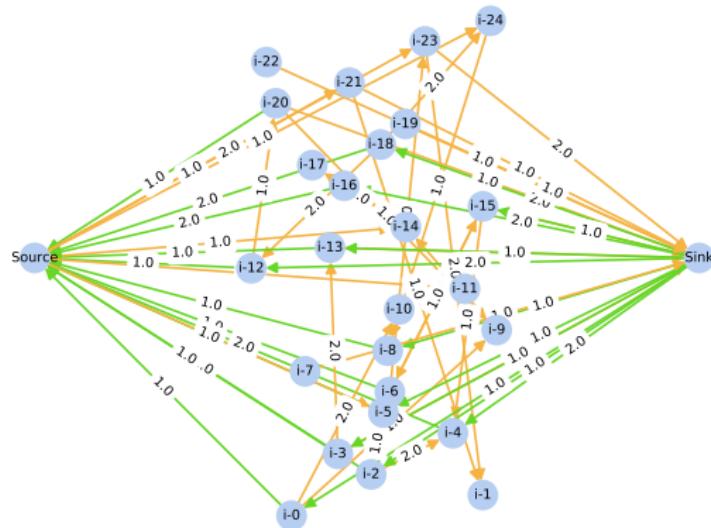
- ▶ Now the algorithm should be able to run

...

## The Maximum flow problem

### Solution with the Ford-Fulkerson algorithm

residual graph step 15



...

└ The Maximum flow problem

└ Solution with the Ford-Fulkerson algorithm

## Complexity

What is the complexity of Ford Fulkerson ?

...

└ The Maximum flow problem

└ Solution with the Ford-Fulkerson algorithm

## Complexity

What is the complexity of Ford Fulkerson ?

$$\mathcal{O}(|f^*| \times |E|) \tag{8}$$

...

└ The Maximum flow problem

└ Solution with the Ford-Fulkerson algorithm

## Modification of Ford Fulkerson

What would we an intuitive and potentially faster modification of the algorithm ?

...

└ The Maximum flow problem

└ Solution with the Ford-Fulkerson algorithm

## Modification of the algorithm

What would we an intuitive and potentially faster modification of the algorithm ?

Use the shortest augmenting path with positive capacity.

...

└ The Maximum flow problem

└ Solution with the Ford-Fulkerson algorithm

## Termination

- ▶ When the capacities are **integer numbers** or **rational numbers** Ford Fulkerson terminates.

...

- └ The Maximum flow problem

- └ Solution with the Ford-Fulkerson algorithm

## Termination

- ▶ When the capacities are **integer numbers** or **rational numbers** Ford Fulkerson terminates.
- ▶ However, when the capacities are general **real numbers**, the algorithm might not terminate.

...

└ The Maximum flow problem

└ Connection with the matching problem

## Link with the matching problem

- ▶ We now go back to the matching problem, in the case of a **bipartite graph** ("problème d'affectation")
- ▶ We will show that in that case, we can connect the two problems.

...

- └ The Maximum flow problem

- └ Connection with the matching problem

## Bipartite graph

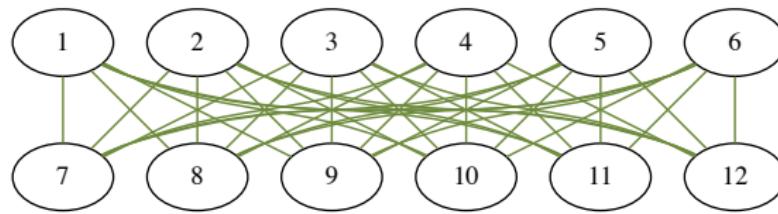


Figure: Complete bipartite graph (not all bipartite graphs are complete)

## Matching problem

We now go back to the matching problem, in the case of a bipartite graph.

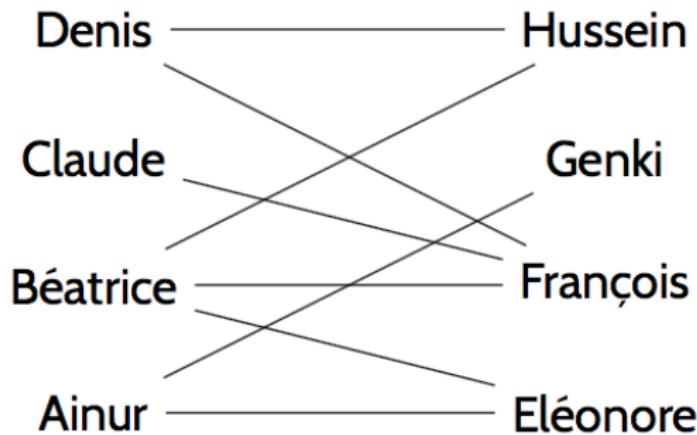


Figure: Bipartite graph

## Equivalence between matching and flow

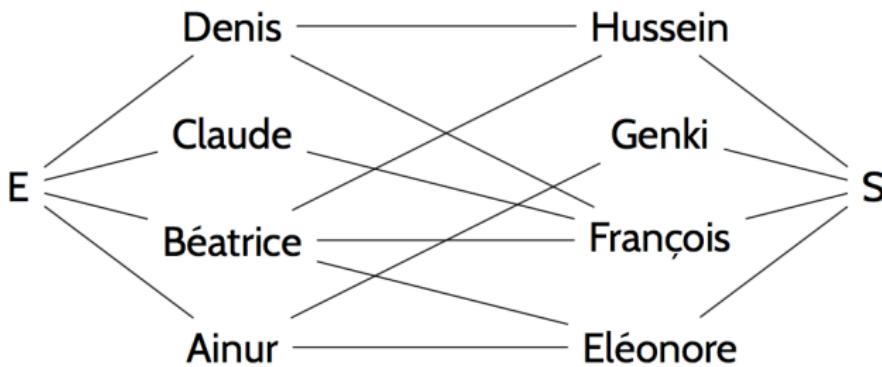


Figure: Introduce two more nodes. All edges have capacity 1. We consider **flows with integer values**

## Ford Fulkerson for matching

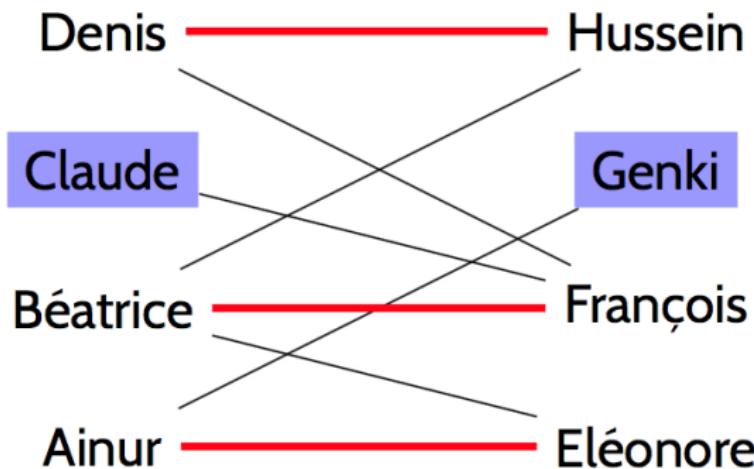


Figure: Non optimal solution

## Ford Fulkerson for matching

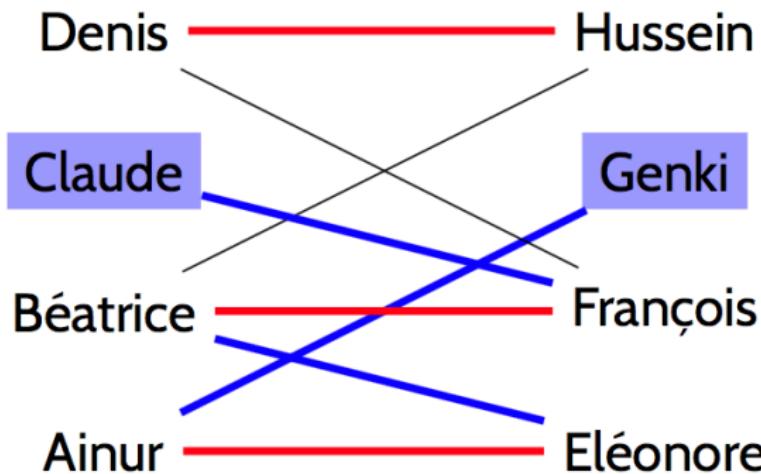


Figure: Optimal solution

...

- └ The Maximum flow problem

- └ Connection with the matching problem

## Connection

Exercice 12: Find a connection between the two problems

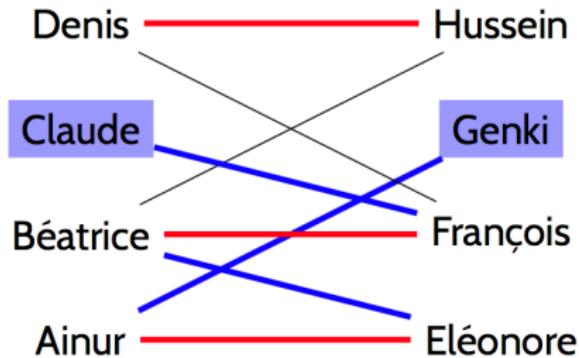


Figure: Optimal solution

### Exercice 13 : Ford Fulkerson and matching

- ▶ We will transpose Ford Fulkerson to a bipartite graph in order to find an optimal matching.
- ▶ `cd ford_matching/`
- ▶ `edit generate_matching_problem.py` in order to generate an instance of the problem.

...

└ The Maximum flow problem

└ Connection with the matching problem

### Exercice 13 : Ford Fulkerson and matching

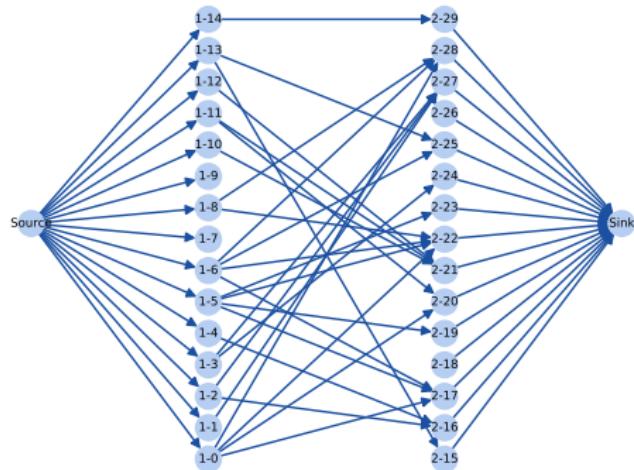
- ▶ Apply the algorithm on an example generated by the previous function.
- ▶ Apply the algorithm to as an instance of your choice.

...

## The Maximum flow problem

### Connection with the matching problem

initial graph

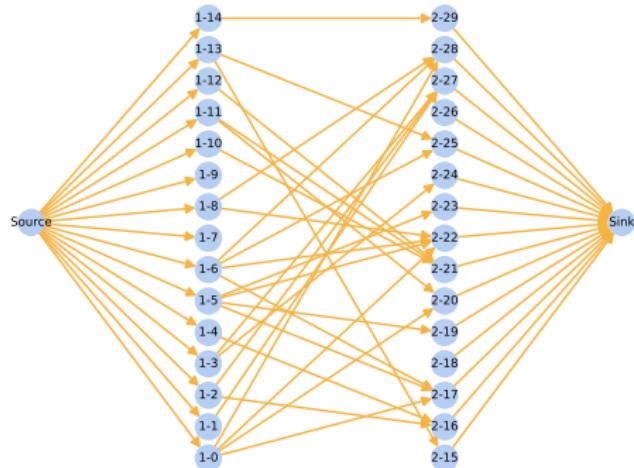


...

- └ The Maximum flow problem

- └ Connection with the matching problem

residual graph step 1

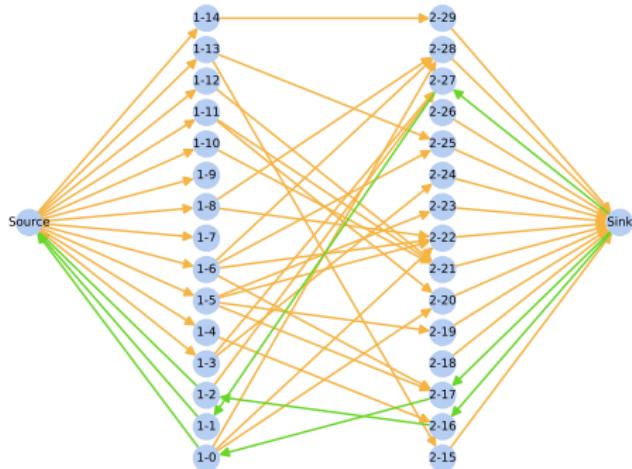


...

- └ The Maximum flow problem

- └ Connection with the matching problem

residual graph step 4

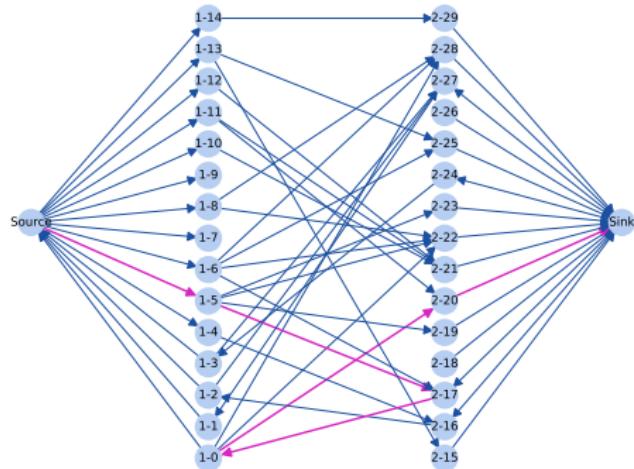


...

## The Maximum flow problem

### Connection with the matching problem

augmenting path step 5

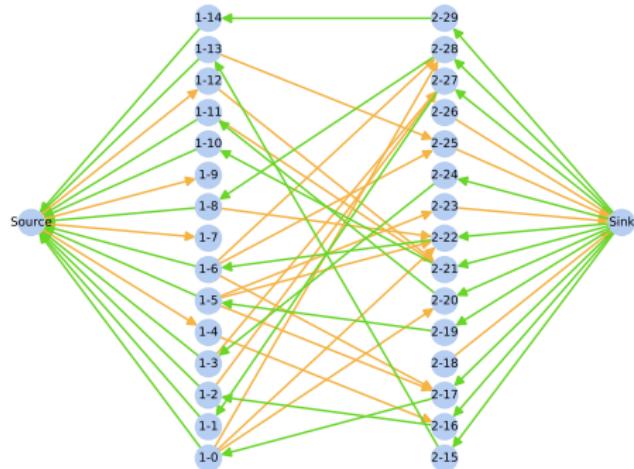


...

## The Maximum flow problem

### Connection with the matching problem

residual graph step 12



## Famous theorem

The maximum flow theorem is equivalent to another famous problem, the **minimum cut** theorem.

## Perfect matching

In the case of a bipartite graph, what is the best matching possible ?

## Perfect matching

In the case of a bipartite graph, what is the best matching possible ?

A matching where **all nodes are allocated**. It is called a **perfect matching**.

We must have that the two parts of the graph are of same cardinality in order to have a perfect matching.

## Hall's marriage theorem

This theorem gives a condition that is necessary and sufficient for the existence of a perfect matching in a bipartite graph : the "marriage condition".

If  $G = (U, V, E)$  is bipartite, the condition means that :

$$\forall X \subset U, |N_G(X)| \geq |X| \quad (9)$$

where  $N_G(X)$  is the set of neighbors of  $X$  in  $G$ .

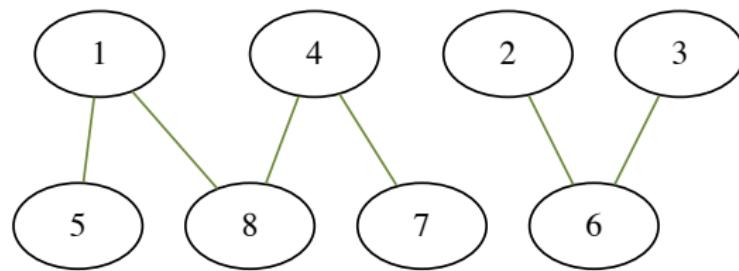
## Hall's theorem

**Exercice 14:** Application of the theorem.

Can you think of a graph that does not abide by the marriage condition and thus has **no perfect matching** ?

## Illustration of Hall's theorem

Exercice 14 : Application of the theorem



## Case of a non bipartite graph

In the case of a **non-bipartite**, we can not use the Ford-Fulkerson algorithm.

Other methods exist such as the **Blossom algorithm**.  
(Edmonds-Karp)

## Conclusion

Ford Fulkerson and its variants (Edmonds-Karp) are polynomial.  
As a result they can run on datasets that are way bigger than  
exhaustive search algorithms.

...

└ The Maximum flow problem

└ More results on the two problems

See you tomorrow