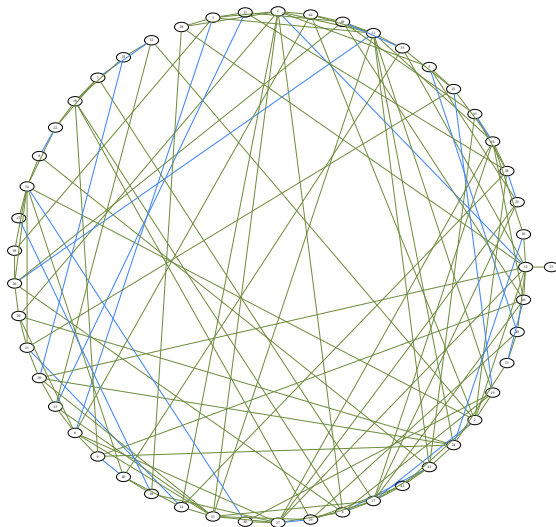


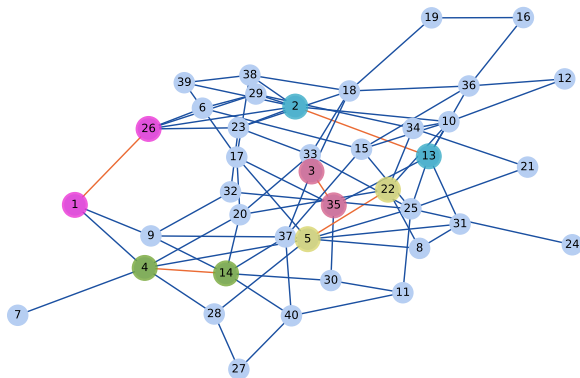
Algorithmic complexity and graphs: the matching problem

October 1, 2022

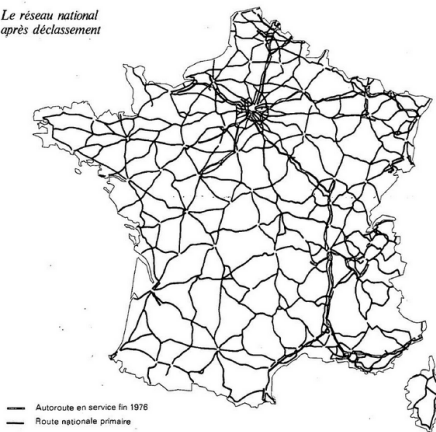


Matching size: 21
Algo step: 128
Nb nodes: 50

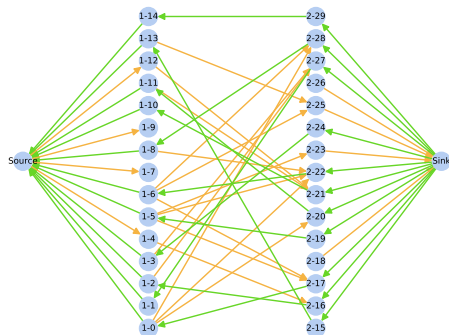
Matching size: 5
Algo step: 19
Nb nodes: 40



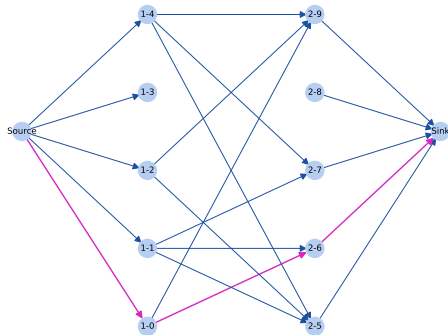
*Le réseau national
après déclassement*



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

*Le réseau national
après déclassement*

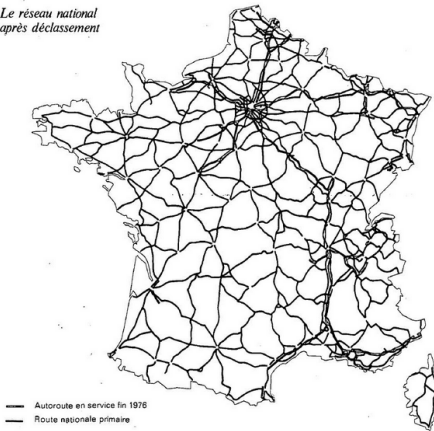


Figure: Problem 1 : transporting merchandise through a network

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

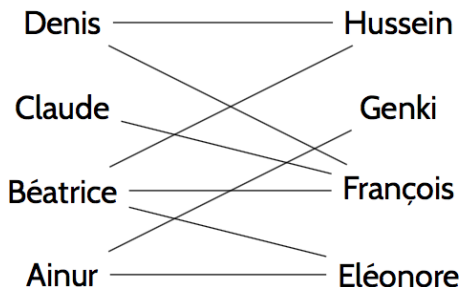


Figure: Problem 2 : Building the largest possible number of teams of 2 persons.

Introductory example 2

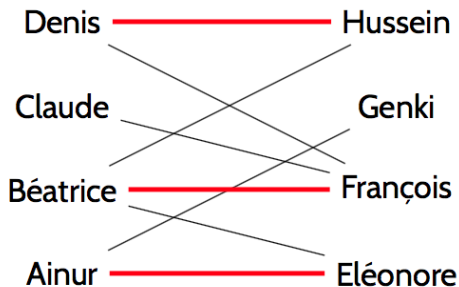


Figure: Problem 2 : not optimal assignment

Introductory example 2

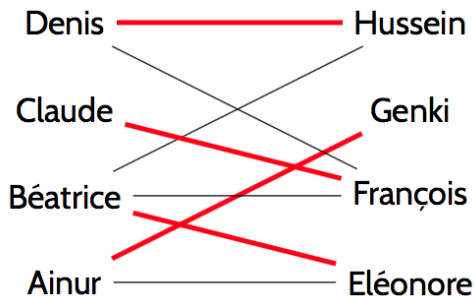


Figure: Problem 2 : optimal assignment

Other examples

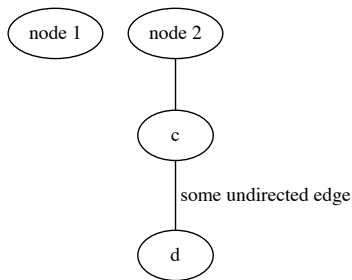
- ▶ Assigning students to internships
- ▶ Assigning machines to a task

Summary

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

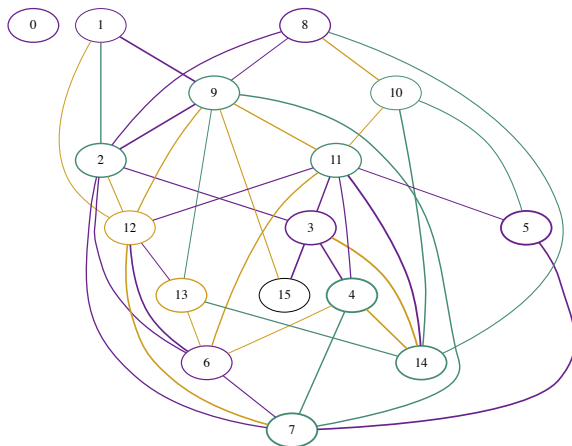
Reminders on graphs

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



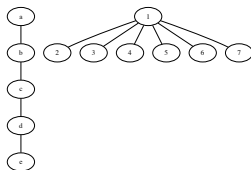
Reminders on graphs

Undirected graph

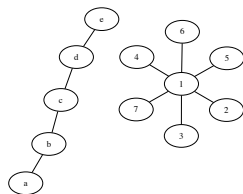


Other available tool : graphviz

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



(a) Image generated with **dot**



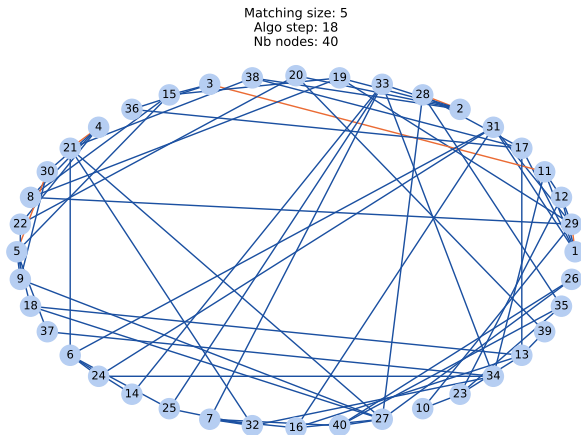
(b) Image generated with **neato**

Overview

- └ The matching problem
 - └ Definition of the problem

Networkx

We will use networkx.

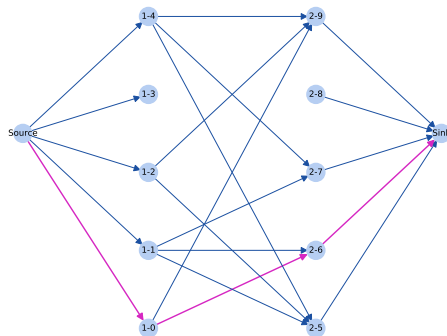


Overview

- └ The matching problem
 - └ Definition of the problem

Networkx

augmenting path step 1



Complete graph

Given a **directed** graph with n nodes, the maximum number of edges is:

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

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

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

edges.

Remark

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

$$\binom{n}{2} = \frac{n!}{2!(n-2)!} \quad (3)$$

Famous graph problem

- ▶ Dominating set

Famous graph problem

- ▶ Dominating set
- ▶ Maximum clique

Famous graph problem

- ▶ Dominating set
- ▶ Maximum clique
- ▶ Coloring

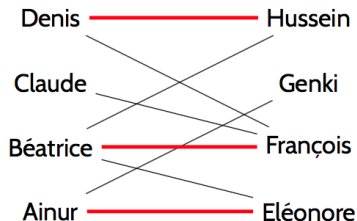
Matching problem

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:

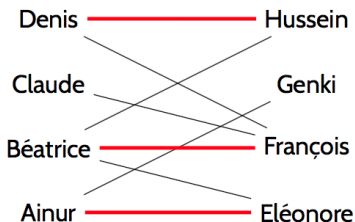
- ▶ A subset of edges $M \subset E$



Back to our problem

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

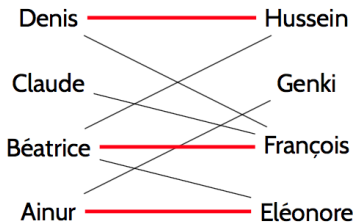
- ▶ A subset of edges $M \subset E$
- ▶ Such that no pairs of edges of M are incident
- ▶ Equivalently, each node in the graph is **at most** in one edge of M .



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

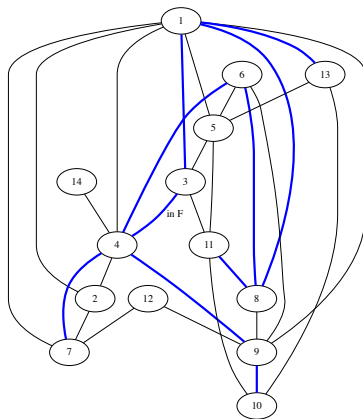


Figure: Is this a matching ?

Example 2

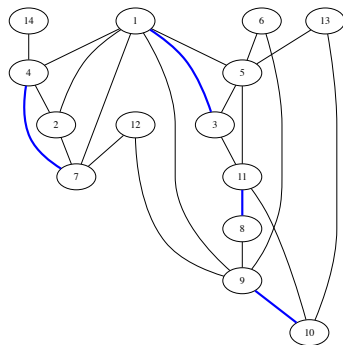


Figure: Is this a matching ?

Example 3

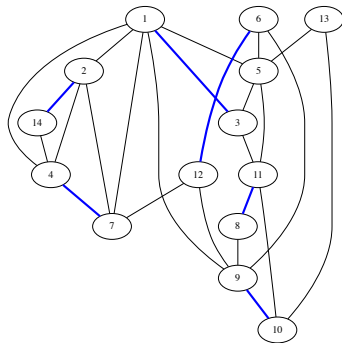


Figure: Is this an optimal matching ?

Example 4

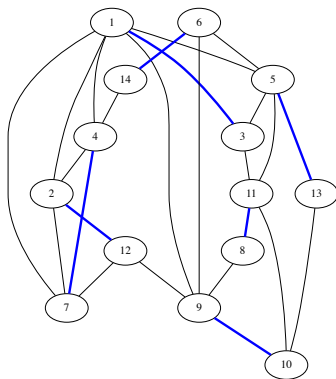


Figure: Is this an optimal matching ?

Example 5

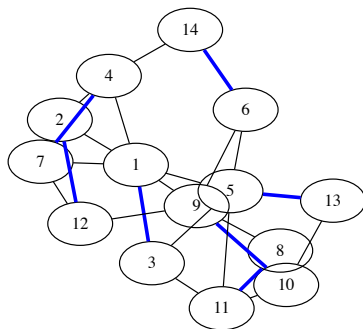


Figure: With neato

- └ The matching problem
 - └ Definition of the problem

Optimal matching

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

Optimal matching

Exercise 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

Exercise 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,

$$\left\lfloor \frac{n}{2} \right\rfloor \quad (4)$$

- └ The matching problem
 - └ Definition of the problem

Optimal matching

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

Optimal

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

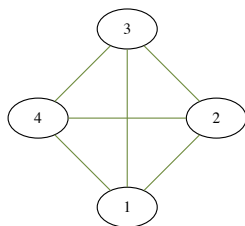


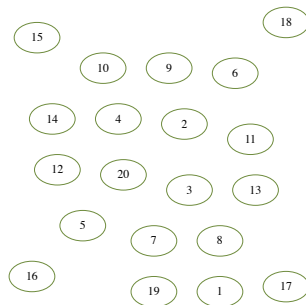
Figure: The complete graph works

Optimal matching

Exercise 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

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



- └ The matching problem
 - └ Definition of the problem

Optimal matching

Exercise 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

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

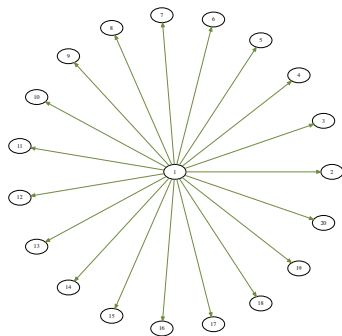


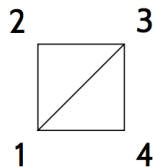
Figure: Star graph

Experiments

Possibilities to code a graph:

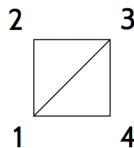
- ▶ list of sets of size 2 (for an undirected graph)
- ▶ a dictionary of successors (directed or undirected)

Coding a graph : as a list



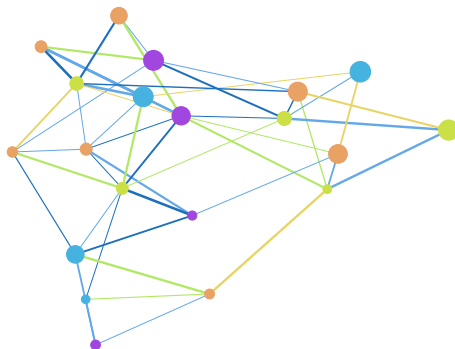
```
g1 = [{1,2},{1,3},{2,3},{3,4},{1,4}]
```

Coding a graph : as a dictionary



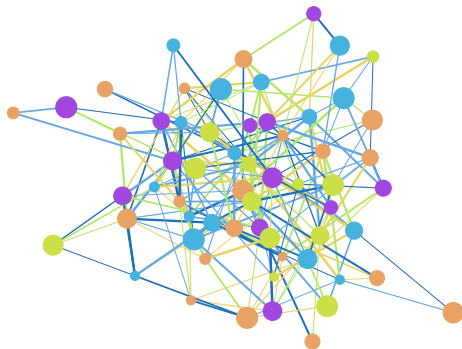
```
g1 = { 1:{2,3,4}, 2:{1,3}, 3:{1,2,4}, 4:{1,3} }
```

Generating graphs with networks.



Overview

- └ The matching problem
- └ Experimental solutions

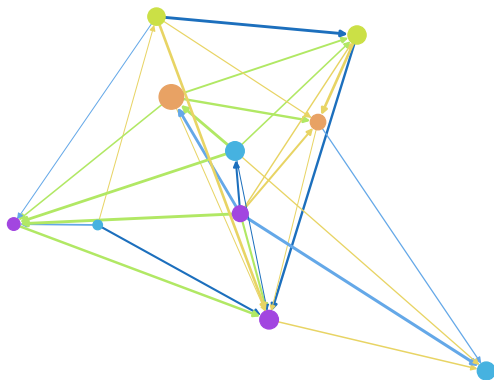


Overview

- └ The matching problem
- └ Experimental solutions



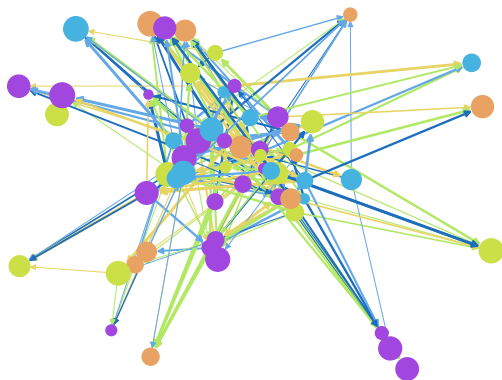
Directed graph



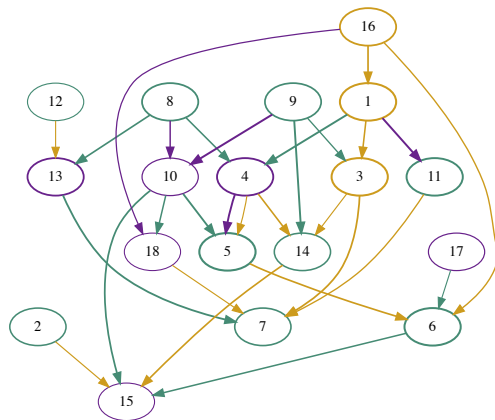
Directed graph II



Directed graph III



Example directed graph



Manual matching

Exercise 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

Exercise 4 : Enumeration

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

Brute force approach

Exercise 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

Exercise 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

Exercise 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

Exercise 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

Exercise 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

Exercise 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 number of edges, we can also write it as 2^p .

Brute force search

Exercise 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 it has the maximum possible number of edges (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 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 being a **maximum** matching ?

Maximum implies maximal

Let us show that a maximum matching is maximal.

Counter Example

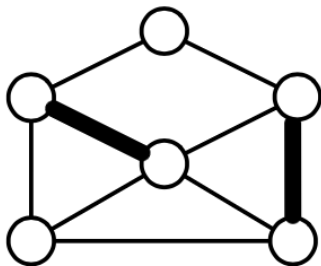
However, a matching that is maximal is **not necessary Maximum**.

Counter Example

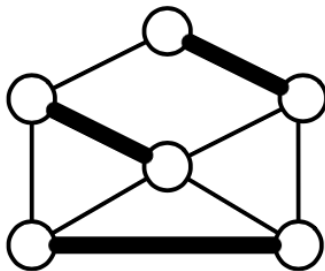
However, a matching that is maximal is **not necessary Maximum**.
Can you find an example ?

Overview

- └ The matching problem
 - └ Greedy algorithm



(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`

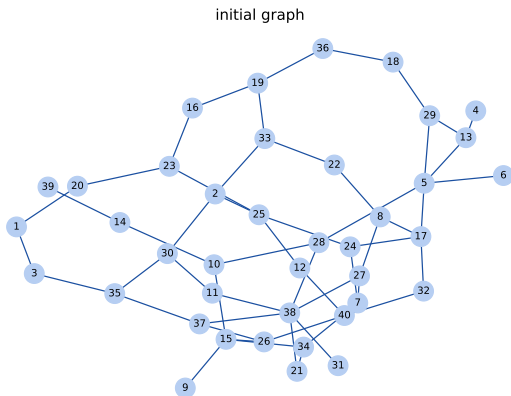
Greedy algorithm

- ▶ 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.

Exercise 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/`

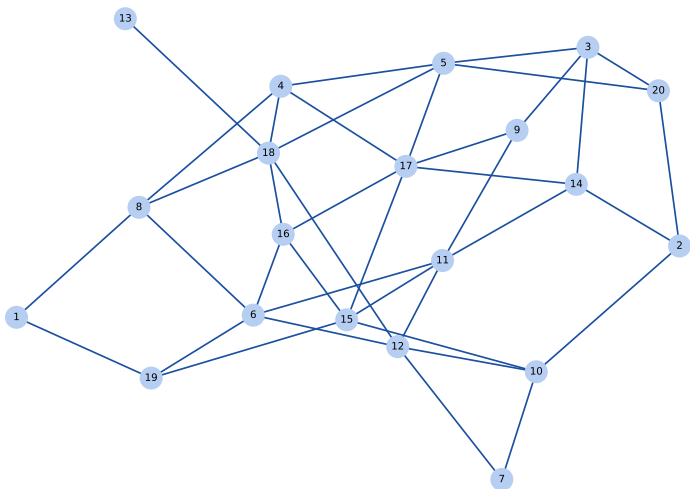


Implementing the greedy algorithm

Exercise 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.

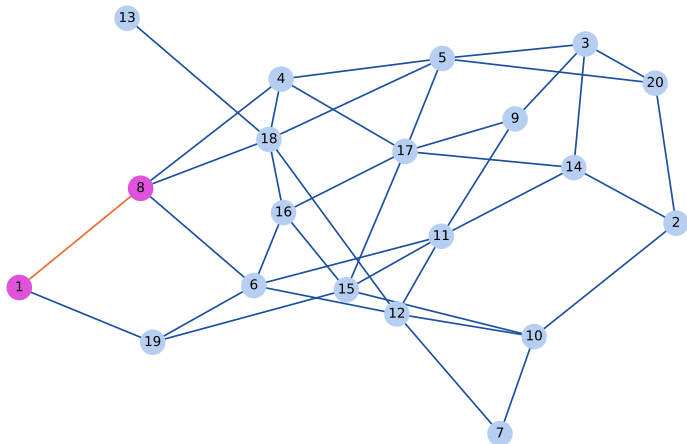
initial graph



Overview

- └ The matching problem
 - └ Greedy algorithm

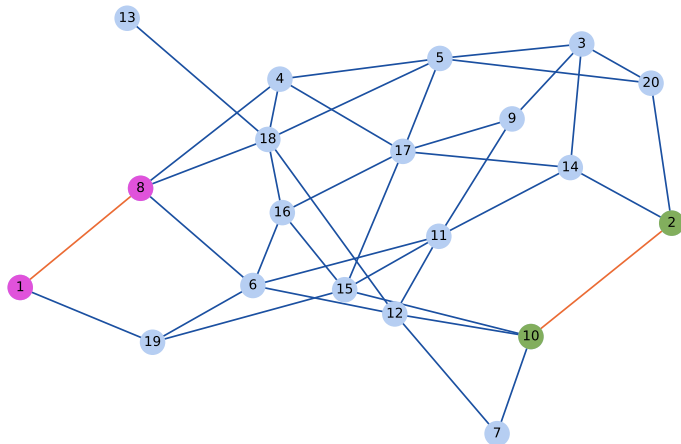
Matching size: 1
Algo step: 1
Nb nodes: 20



Overview

- └ The matching problem
 - └ Greedy algorithm

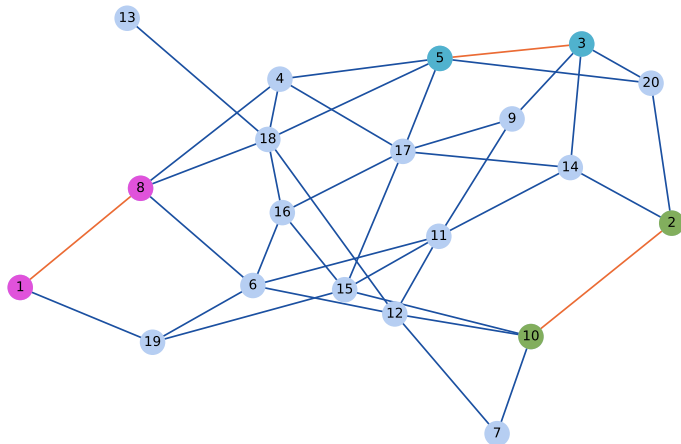
Matching size: 2
Algo step: 3
Nb nodes: 20



Overview

- └ The matching problem
 - └ Greedy algorithm

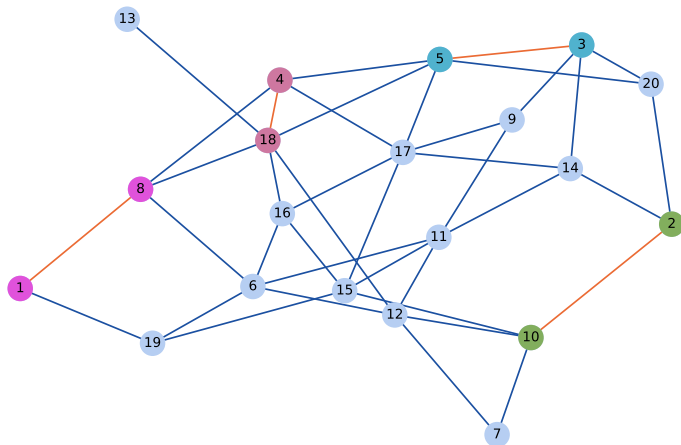
Matching size: 3
Algo step: 6
Nb nodes: 20



Overview

- └ The matching problem
 - └ Greedy algorithm

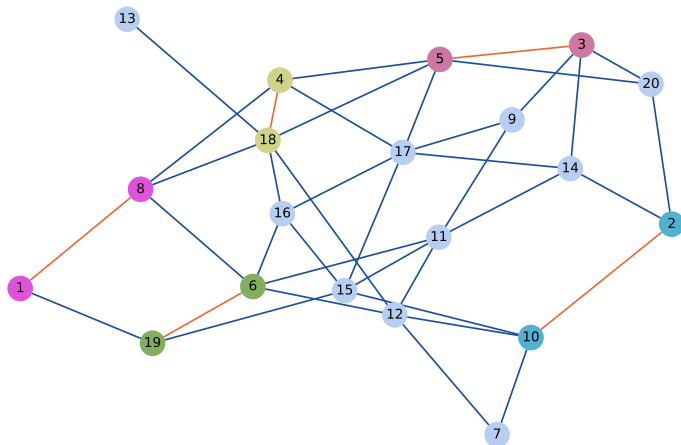
Matching size: 4
Algo step: 11
Nb nodes: 20



Overview

- └ The matching problem
 - └ Greedy algorithm

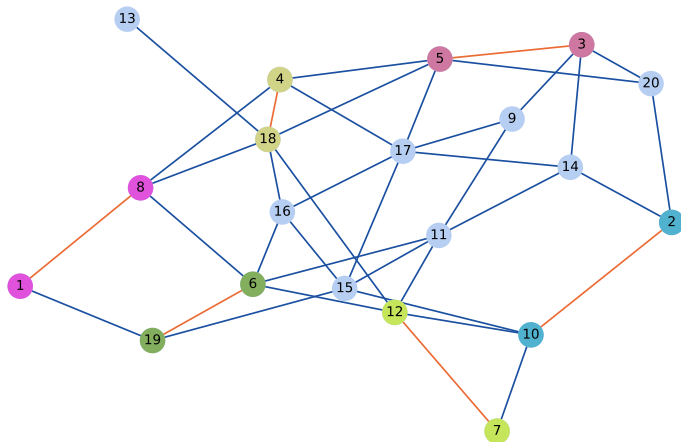
Matching size: 5
Algo step: 17
Nb nodes: 20



Overview

- └ The matching problem
 - └ Greedy algorithm

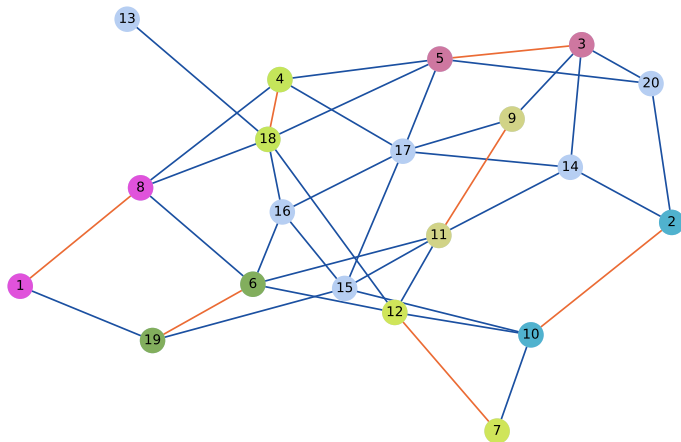
Matching size: 6
Algo step: 22
Nb nodes: 20



Overview

- └ The matching problem
 - └ Greedy algorithm

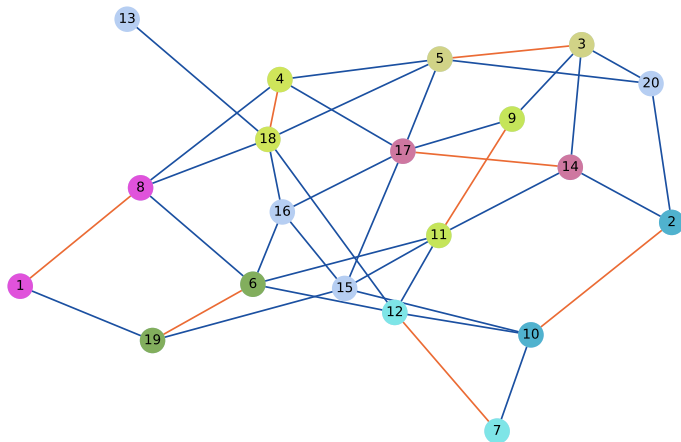
Matching size: 7
Algo step: 25
Nb nodes: 20



Overview

- └ The matching problem
 - └ Greedy algorithm

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



- Overview
 - The matching problem
 - Greedy algorithm

- Greedy algorithm

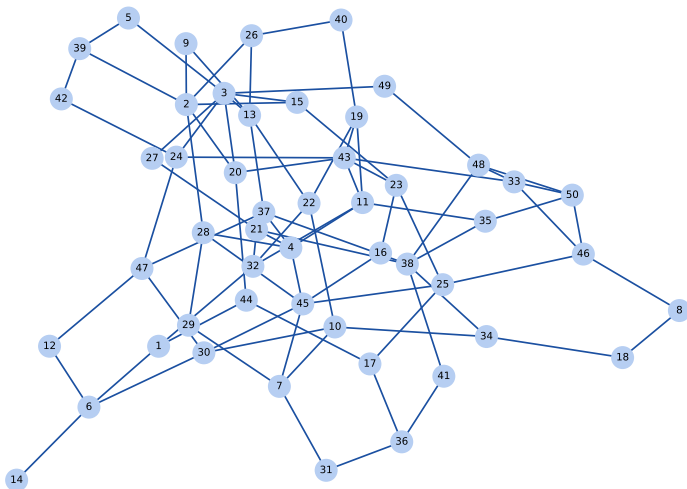
Algo step: 36

A network graph with 20 nodes, numbered 1 to 20. The nodes are colored as follows: 1 (pink), 2 (teal), 3 (light green), 4 (yellow-green), 5 (yellow), 6 (green), 7 (grey), 8 (pink), 9 (yellow-green), 10 (teal), 11 (yellow-green), 12 (grey), 13 (light blue), 14 (pink), 15 (cyan), 16 (cyan), 17 (pink), 18 (yellow-green), 19 (green), and 20 (light blue). The edges are colored as follows: blue (most edges), orange (edges 1-8, 1-19, 4-18, 6-19, 7-10, 7-12, 9-14, 11-14, 11-17, 14-17, 14-20, 17-18, 18-19, 18-17, 19-18, 19-12, 19-15, 19-16, 19-17, 19-18, 19-19, 19-20, 19-21, 19-22, 19-23, 19-24, 19-25, 19-26, 19-27, 19-28, 19-29, 19-30, 19-31, 19-32, 19-33, 19-34, 19-35, 19-36, 19-37, 19-38, 19-39, 19-40, 19-41, 19-42, 19-43, 19-44, 19-45, 19-46, 19-47, 19-48, 19-49, 19-50, 19-51, 19-52, 19-53, 19-54, 19-55, 19-56, 19-57, 19-58, 19-59, 19-60, 19-61, 19-62, 19-63, 19-64, 19-65, 19-66, 19-67, 19-68, 19-69, 19-70, 19-71, 19-72, 19-73, 19-74, 19-75, 19-76, 19-77, 19-78, 19-79, 19-80, 19-81, 19-82, 19-83, 19-84, 19-85, 19-86, 19-87, 19-88, 19-89, 19-90, 19-91, 19-92, 19-93, 19-94, 19-95, 19-96, 19-97, 19-98, 19-99, 19-100, 19-101, 19-102, 19-103, 19-104, 19-105, 19-106, 19-107, 19-108, 19-109, 19-110, 19-111, 19-112, 19-113, 19-114, 19-115, 19-116, 19-117, 19-118, 19-119, 19-120, 19-121, 19-122, 19-123, 19-124, 19-125, 19-126, 19-127, 19-128, 19-129, 19-130, 19-131, 19-132, 19-133, 19-134, 19-135, 19-136, 19-137, 19-138, 19-139, 19-140, 19-141, 19-142, 19-143, 19-144, 19-145, 19-146, 19-147, 19-148, 19-149, 19-150, 19-151, 19-152, 19-153, 19-154, 19-155, 19-156, 19-157, 19-158, 19-159, 19-160, 19-161, 19-162, 19-163, 19-164, 19-165, 19-166, 19-167, 19-168, 19-169, 19-170, 19-171, 19-172, 19-173, 19-174, 19-175, 19-176, 19-177, 19-178, 19-179, 19-180, 19-181, 19-182, 19-183, 19-184, 19-185, 19-186, 19-187, 19-188, 19-189, 19-190, 19-191, 19-192, 19-193, 19-194, 19-195, 19-196, 19-197, 19-198, 19-199, 19-200, 19-201, 19-202, 19-203, 19-204, 19-205, 19-206, 19-207, 19-208, 19-209, 19-210, 19-211, 19-212, 19-213, 19-214, 19-215, 19-216, 19-217, 19-218, 19-219, 19-220, 19-221, 19-222, 19-223, 19-224, 19-225, 19-226, 19-227, 19-228, 19-229, 19-230, 19-231, 19-232, 19-233, 19-234, 19-235, 19-236, 19-237, 19-238, 19-239, 19-240, 19-241, 19-242, 19-243, 19-244, 19-245, 19-246, 19-247, 19-248, 19-249, 19-250, 19-251, 19-252, 19-253, 19-254, 19-255, 19-256, 19-257, 19-258, 19-259, 19-260, 19-261, 19-262, 19-263, 19-264, 19-265, 19-266, 19-267, 19-268, 19-269, 19-270, 19-271, 19-272, 19-273, 19-274, 19-275, 19-276, 19-277, 19-278, 19-279, 19-280, 19-281, 19-282, 19-283, 19-284, 19-285, 19-286, 19-287, 19-288, 19-289, 19-290, 19-291, 19-292, 19-293, 19-294, 19-295, 19-296, 19-297, 19-298, 19-299, 19-300, 19-301, 19-302, 19-303, 19-304, 19-305, 19-306, 19-307, 19-308, 19-309, 19-310, 19-311, 19-312, 19-313, 19-314, 19-315, 19-316, 19-317, 19-318, 19-319, 19-320, 19-321, 19-322, 19-323, 19-324, 19-325, 19-326, 19-327, 19-328, 19-329, 19-330, 19-331, 19-332, 19-333, 19-334, 19-335, 19-336, 19-337, 19-338, 19-339, 19-340, 19-341, 19-342, 19-343, 19-344, 19-345, 19-346, 19-347, 19-348, 19-349, 19-350, 19-351, 19-352, 19-353, 19-354, 19-355, 19-356, 19-357, 19-358, 19-359, 19-360, 19-361, 19-362, 19-363, 19-364, 19-365, 19-366, 19-367, 19-368, 19-369, 19-370, 19-371, 19-372, 19-373, 19-374, 19-375, 19-376, 19-377, 19-378, 19-379, 19-380, 19-381, 19-382, 19-383, 19-384, 19-385, 19-386, 19-387, 19-388, 19-389, 19-390, 19-391, 19-392, 19-393, 19-394, 19-395, 19-396, 19-397, 19-398, 19-399, 19-400, 19-401, 19-402, 19-403, 19-404, 19-405, 19-406, 19-407, 19-408, 19-409, 19-410, 19-411, 19-412, 19-413, 19-414, 19-415, 19-416, 19-417, 19-418, 19-419, 19-420, 19-421, 19-422, 19-423, 19-424, 19-425, 19-426, 19-427, 19-428, 19-429, 19-430, 19-431, 19-432, 19-433, 19-434, 19-435, 19-436, 19-437, 19-438, 19-439, 19-440, 19-441, 19-442, 19-443, 19-444, 19-445, 19-446, 19-447, 19-448, 19-449, 19-450, 19-451, 19-452, 19-453, 19-454, 19-455, 19-456, 19-457, 19-458, 19-459, 19-460, 19-461, 19-462, 19-463, 19-464, 19-465, 19-466, 19-467, 19-468, 19-469, 19-470, 19-471, 19-472, 19-473, 19-474, 19-475, 19-476, 19-477, 19-478, 19-479, 19-480, 19-481, 19-482, 19-483, 19-484, 19-485, 19-486, 19-487, 19-488, 19-489, 19-490, 19-491, 19-492, 19-493, 19-494, 19-495, 19-496, 19-497, 19-498, 19-499, 19-500,

Overview

- └ The matching problem
 - └ Greedy algorithm

initial graph



Overview

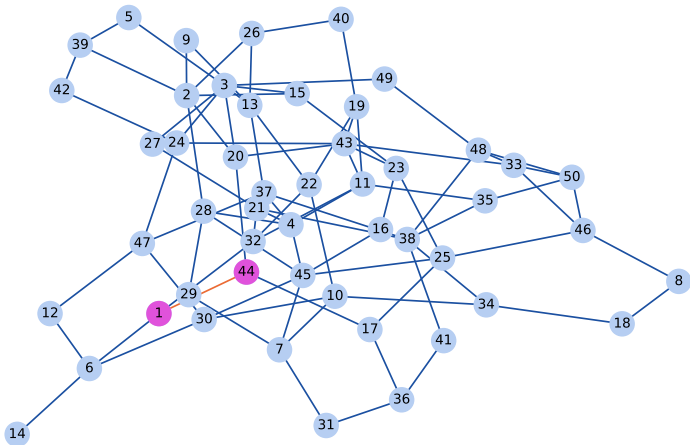
└ The matching problem

└ Greedy algorithm

Matching size: 1

Algo step: 1

Nb nodes: 50



Overview

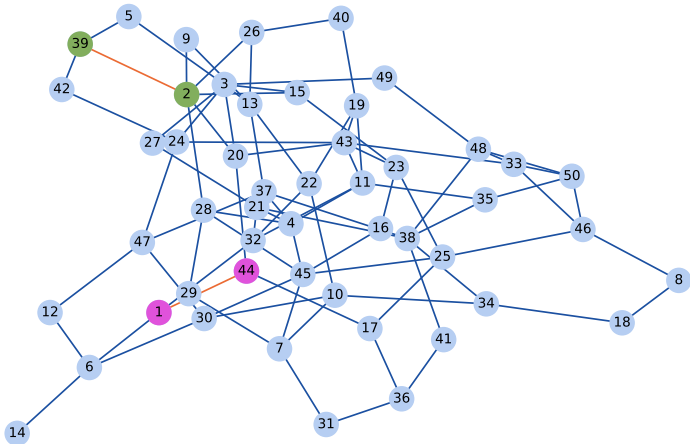
└ The matching problem

└ Greedy algorithm

Matching size: 2

Algo step: 4

Nb nodes: 50



Overview

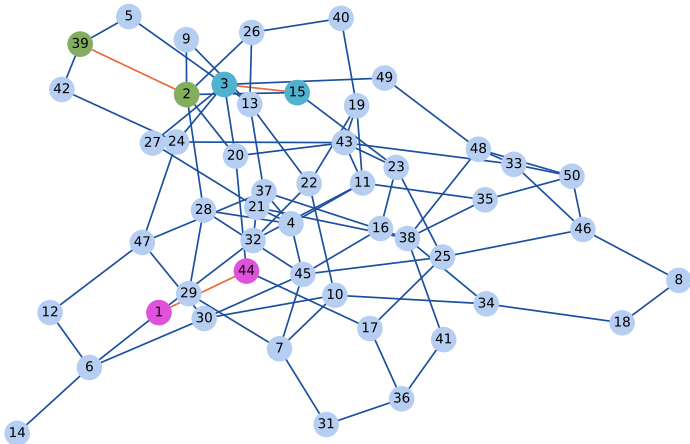
└ The matching problem

└ Greedy algorithm

Matching size: 3

Algo step: 10

Nb nodes: 50



Overview

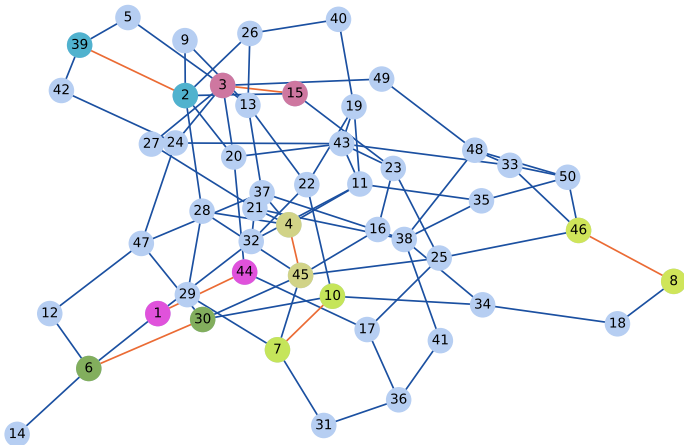
└ The matching problem

└ Greedy algorithm

Matching size: 7

Algo step: 30

Nb nodes: 50



Overview

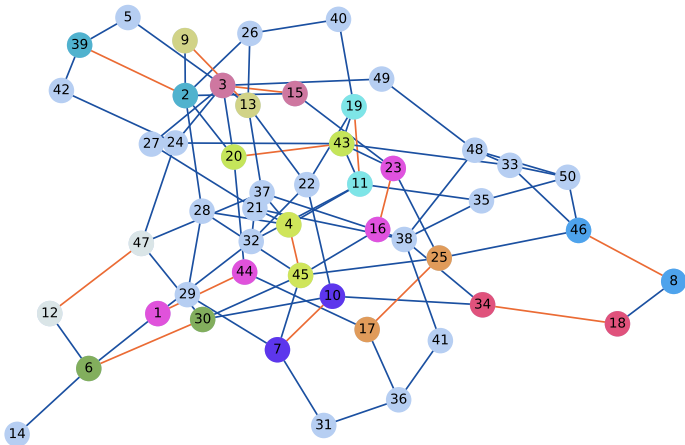
└ The matching problem

└ Greedy algorithm

Matching size: 14

Algo step: 54

Nb nodes: 50



Overview

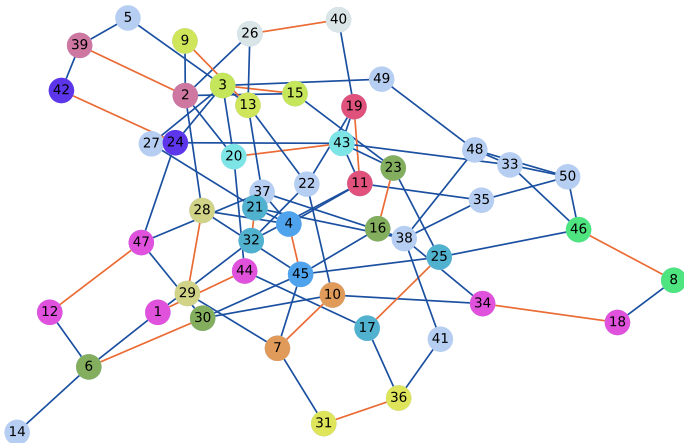
└ The matching problem

└ Greedy algorithm

Matching size: 19

Algo step: 72

Nb nodes: 50



Overview

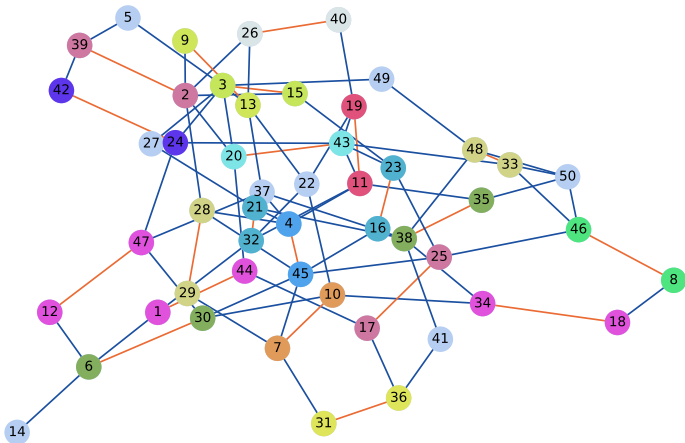
└ The matching problem

└ Greedy algorithm

Matching size: 21

Algo step: 78

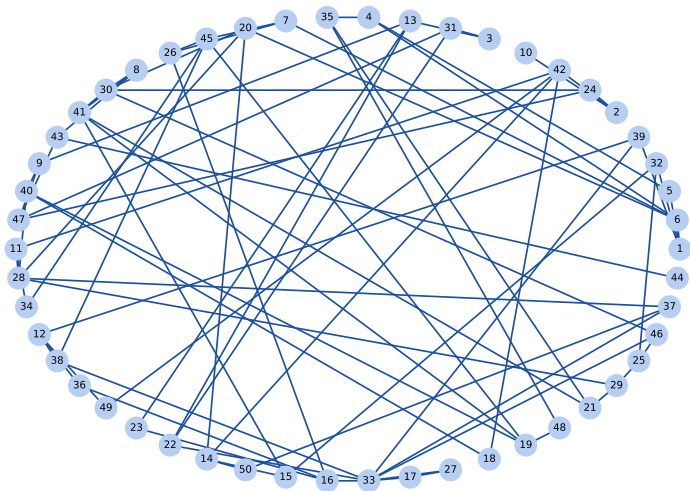
Nb nodes: 50



Overview

- └ The matching problem
 - └ Greedy algorithm

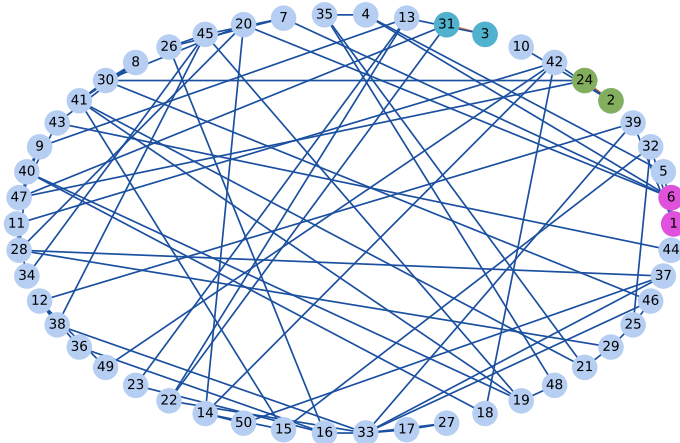
initial graph



Overview

- └ The matching problem
 - └ Greedy algorithm

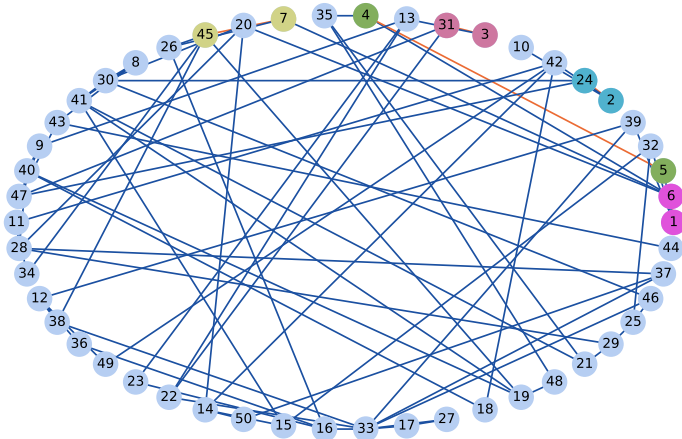
Matching size: 3
Algo step: 8
Nb nodes: 50



Overview

- └ The matching problem
 - └ Greedy algorithm

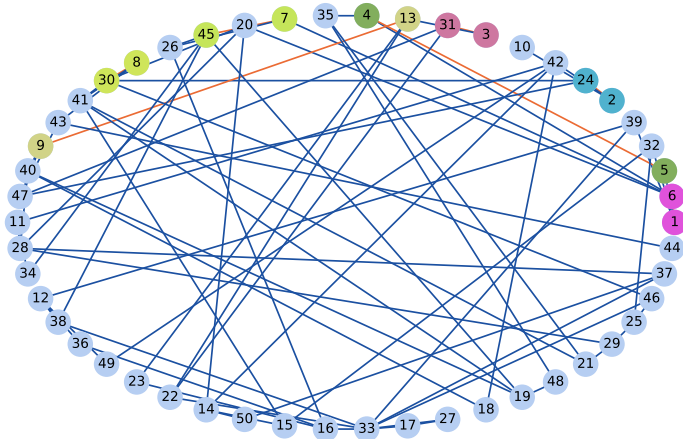
Matching size: 5
Algo step: 15
Nb nodes: 50



Overview

- └ The matching problem
 - └ Greedy algorithm

Matching size: 7
Algo step: 20
Nb nodes: 50



Overview

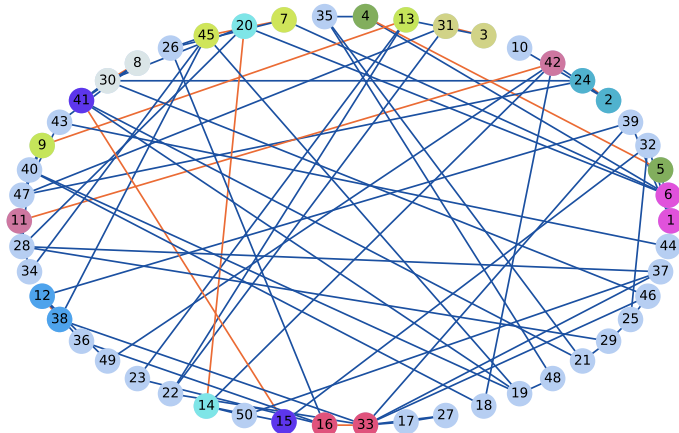
- The matching problem

- Greedy algorithm

Matching size: 12

Algo step: 38

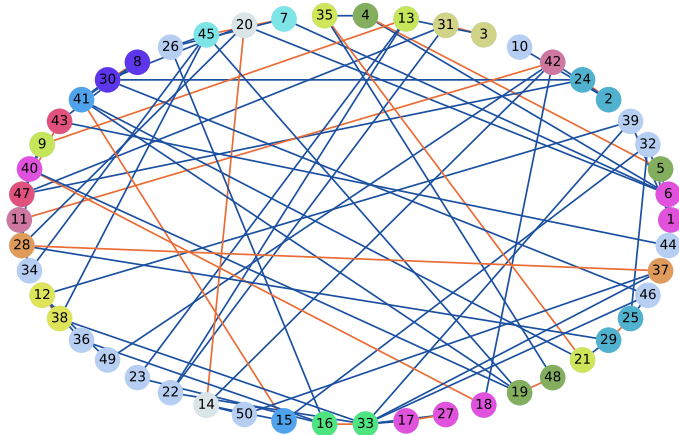
Nb nodes: 50



Overview

- └ The matching problem
 - └ Greedy algorithm

Matching size: 19
Algo step: 79
Nb nodes: 50



Example

Exercise 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

Exercise 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, if $|M|$ 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 :

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