

Algorithmic complexity and graphs: flow networks

September 29, 2023

Overview

- └ The Maximum flow problem
- └ Presentation of the problem

Max flow



Figure: Optimizing the quantity of merchandise transported from one place to another, respecting some constraints

Overview

- └ The Maximum flow problem
 - └ Presentation of the problem

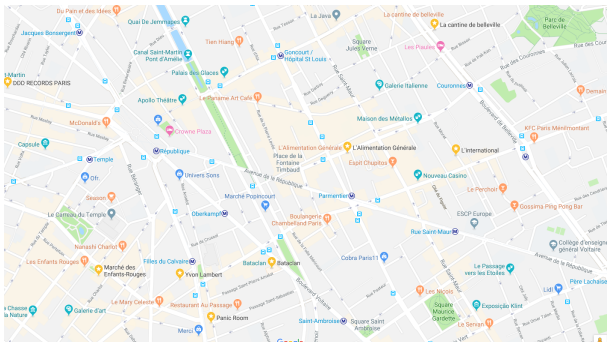


Figure: Optimizing the quantity of merchandise transported from one place to another, respecting some constraints

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** $G = (E, V)$

Formalizing the problem

Flow network (reseau de flot) :

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

Formalizing the problem

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 .

Formalizing the problem

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 .

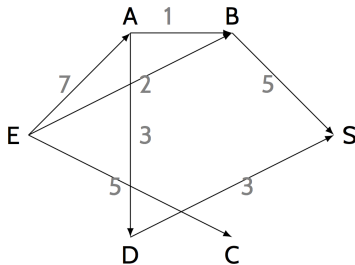


Figure: A **flow network (reseau de flot)** with capacities

Formalizing the problem

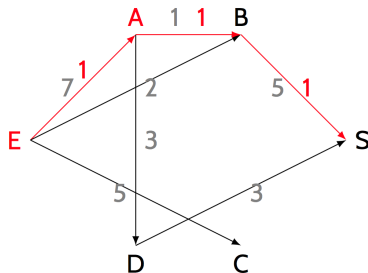
Flow network (reseau de flot) :

- ▶ 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, see below)

Formalizing the problem

Flow network (reseau de flot) :

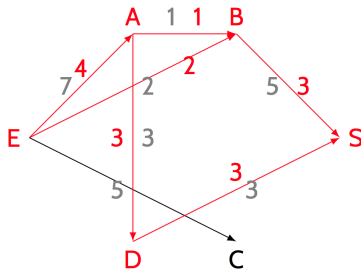
- ▶ Each edge (u, v) must have a **capacity** $c(u, v) \geq 0$
- ▶ A **flow** f is a function $f(u, v) \leq c(u, v)$ (+ additional constraints, see below)



Formalizing the problem

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, see below)



Definition of a flow

We must also have :

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

Definition of a flow

We must also have :

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

Other formulation of the flow conservation

Exercise 1 : Other formulation of the flow conservation

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

$$\sum_{f(u,v)>0} f(u,v) = \sum_{f(v,u)>0} f(v,u) \quad (1)$$

Maximum flow

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

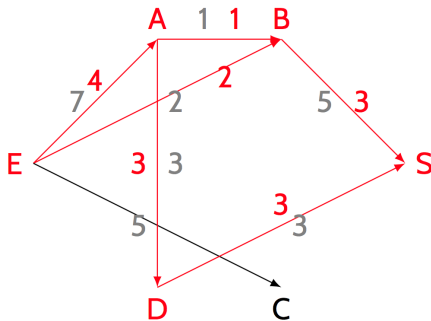


Figure: Max flow

Ford Fulkerson algorithm

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

- ▶ terminates
- ▶ is correct
- ▶ is polynomial

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.

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 $c_r(u, v)$:

$$c_r(u, v) = c(u, v) - f(u, v) \quad (2)$$

Overview

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

Example of residual graph

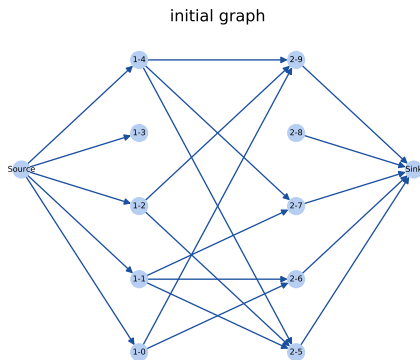


Figure: All initial capacities set to 1

Overview

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

Example of residual graph

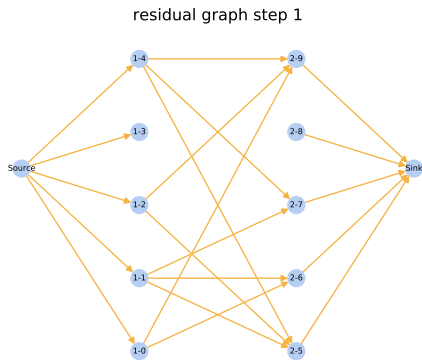


Figure: All initial capacities set to 1

Overview

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

Example of residual graph

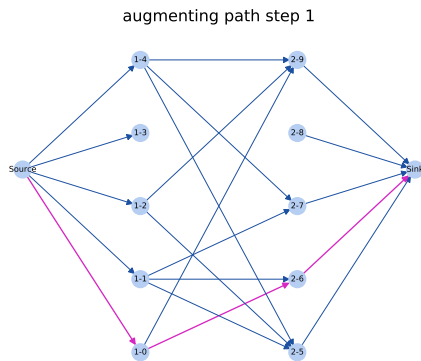


Figure: All initial capacities set to 1

Overview

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

Example of residual graph

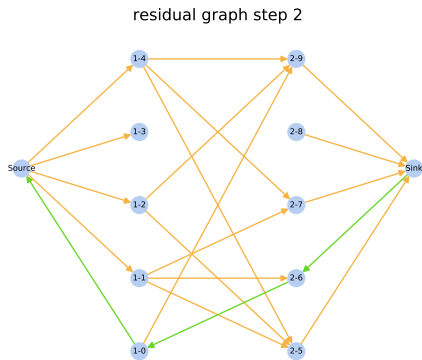


Figure: All initial capacities set to 1

Overview

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

Example of residual graph

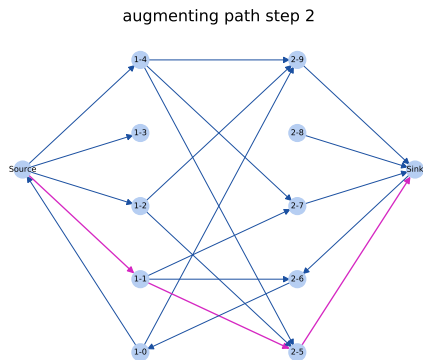


Figure: All initial capacities set to 1

Overview

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

Example of residual graph

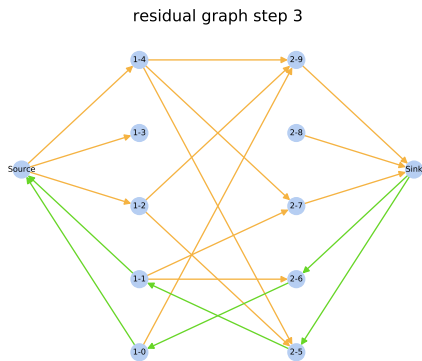


Figure: All initial capacities set to 1

Overview

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

Example of residual graph

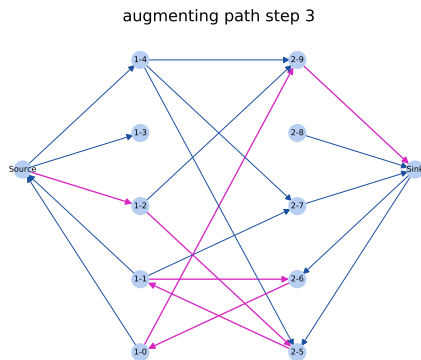


Figure: All initial capacities set to 1

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

Flow network and residual graph

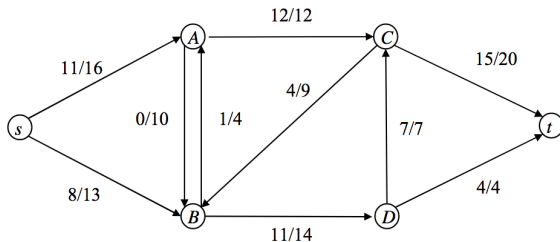


Figure: Another flow network

Overview

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

Flow network and residual graph

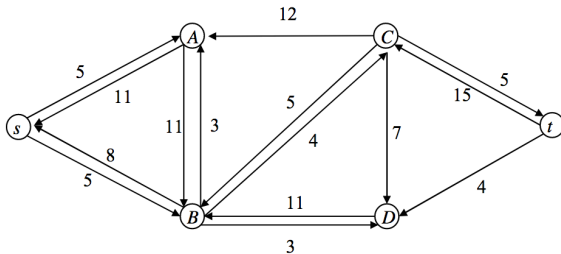


Figure: Residual graph

- └ 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 .

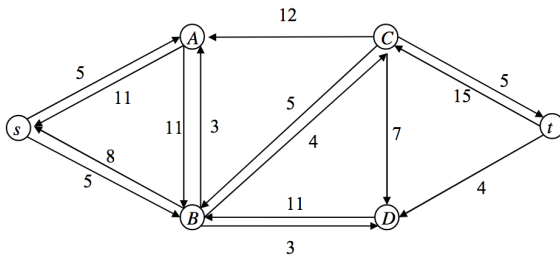


Figure: Residual graph

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

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.

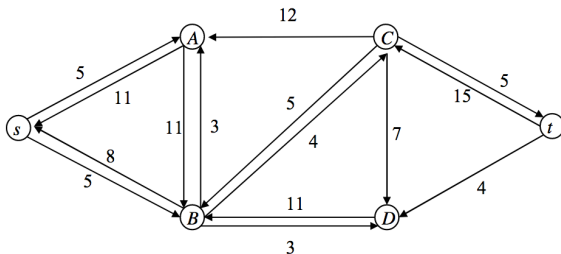


Figure: Residual graph

- └ 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

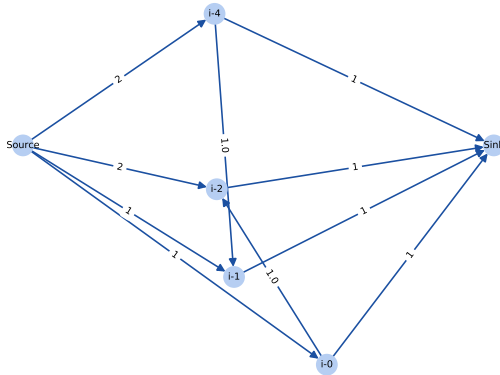
Ford-fulkerson algorithm

Let us apply the algorithm to some instances:

Overview

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

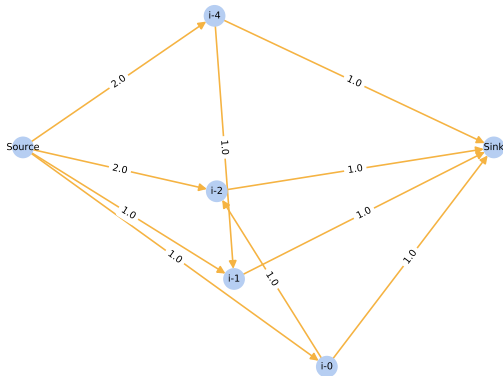
initial graph



Overview

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

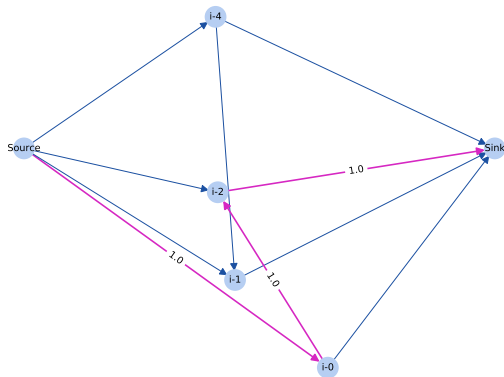
residual graph step 1



Overview

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

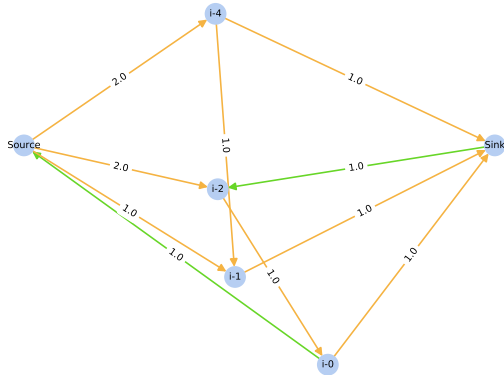
augmenting path step 1



Overview

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

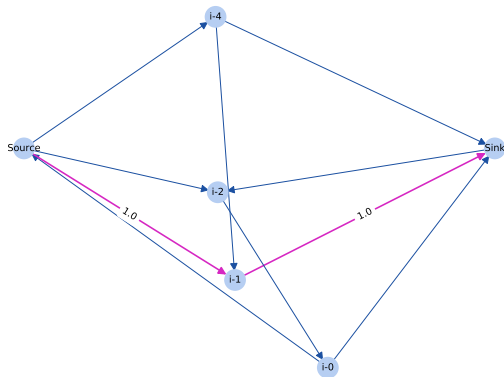
residual graph step 2



Overview

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

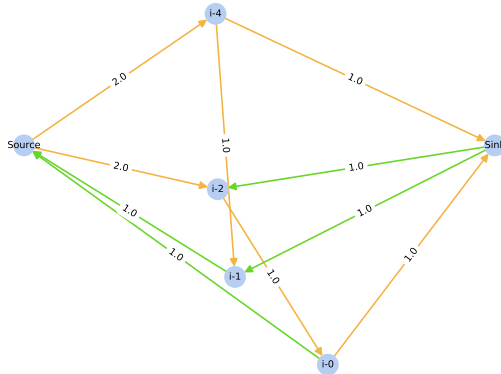
augmenting path step 2



Overview

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

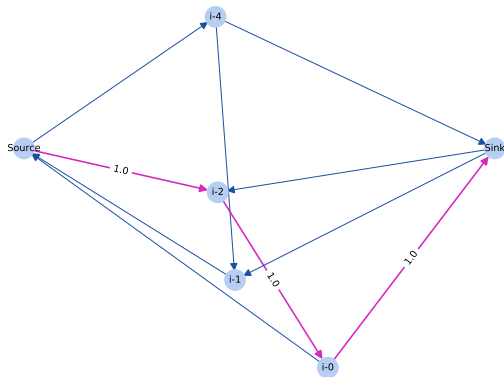
residual graph step 3



Overview

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

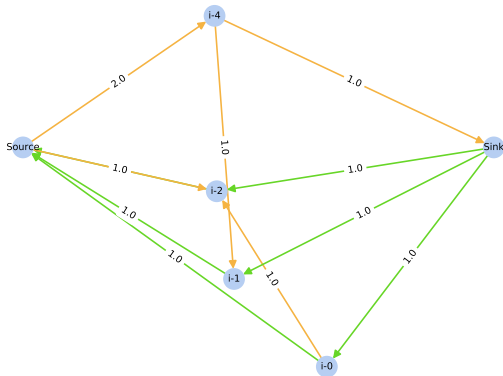
augmenting path step 3



Overview

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

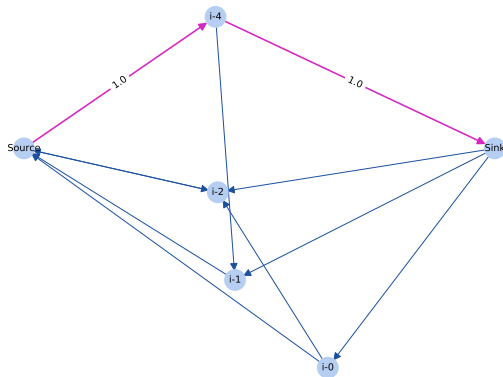
residual graph step 4



Overview

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

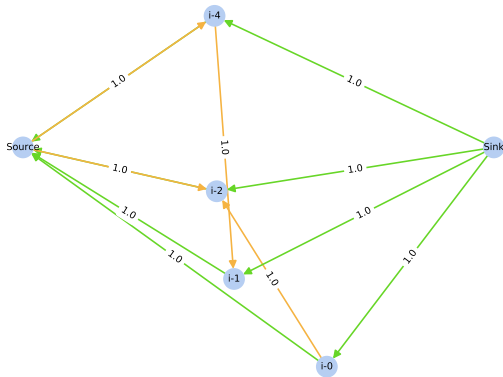
augmenting path step 4



Overview

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

residual graph step 5



Overview

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

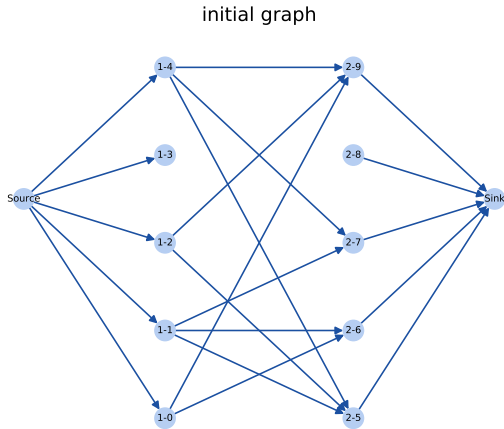
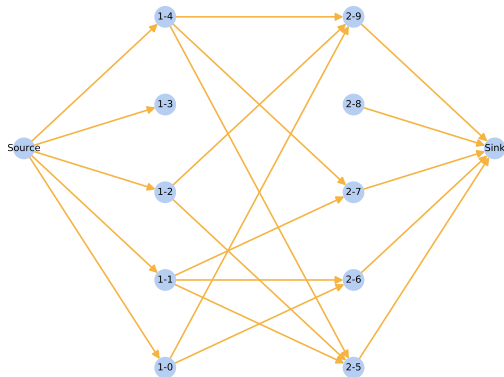


Figure: Initial capacity set to 1

Overview

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

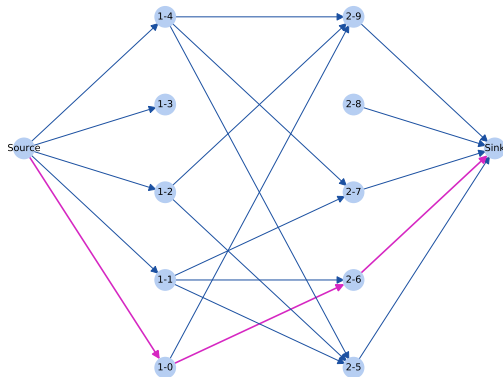
residual graph step 1



Overview

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

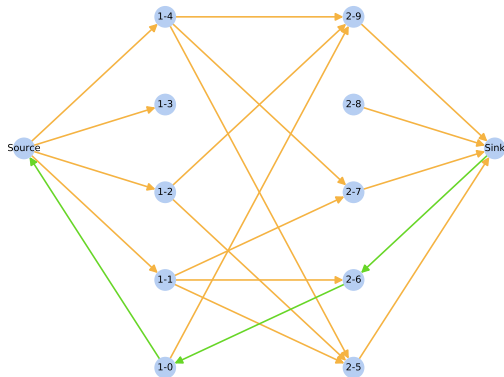
augmenting path step 1



Overview

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

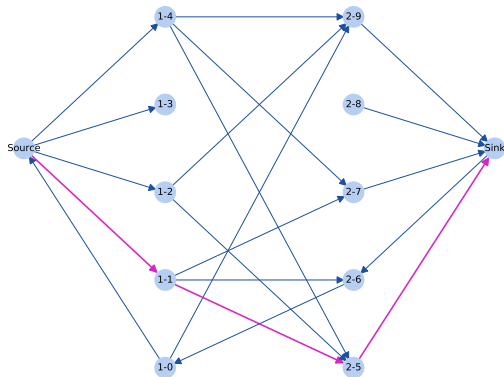
residual graph step 2



Overview

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

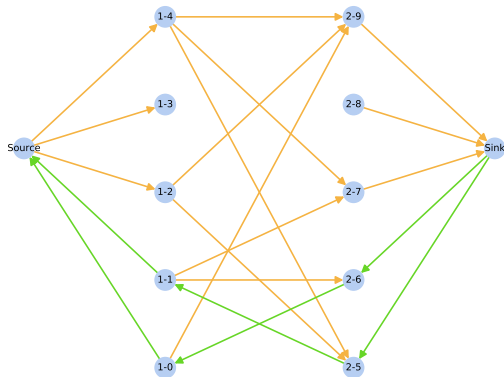
augmenting path step 2



Overview

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

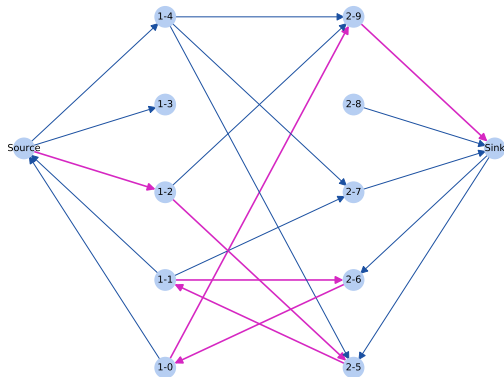
residual graph step 3



Overview

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

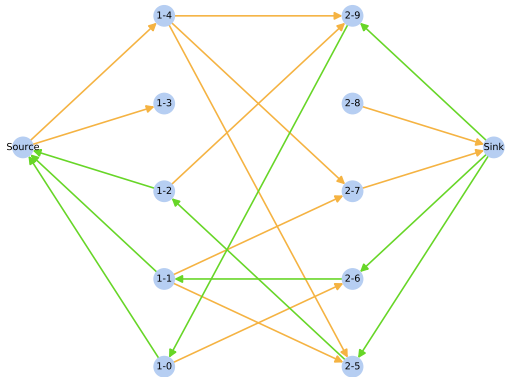
augmenting path step 3



Overview

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

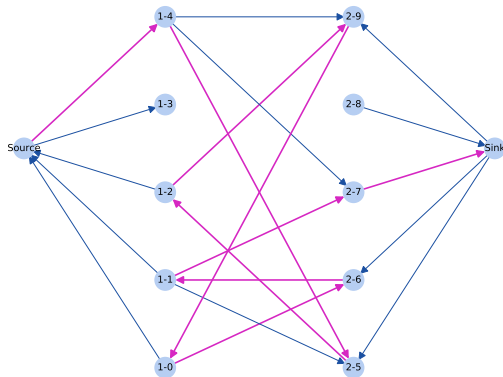
residual graph step 4



Overview

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

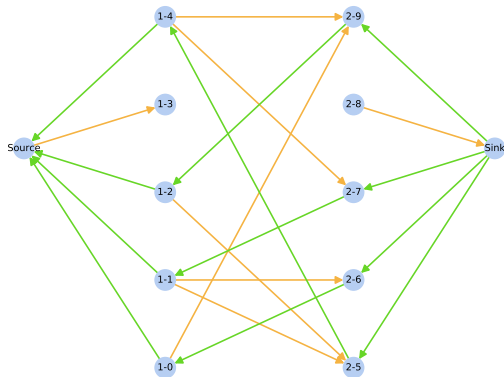
augmenting path step 4



Overview

- └ 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

Numpy exercise

Exercise 2: Numpy arrays.

First, we will do an exercise to get more familiar with numpy.

Please follow the notebook `numpy_demo/numpy_demo.py`.

Ford Fulkerson algorithm

Exercise 3 : We will implement the Ford Fulkerson algorithm (1956)

- ▶ `cd ford_fulkerson/` and use `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

Exercise 4 : step 1

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

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

Algorithm

Exercise 4 : step 2

- ▶ now edit `augment_flow()`

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

Algorithm

Exercise 4 : step 3

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

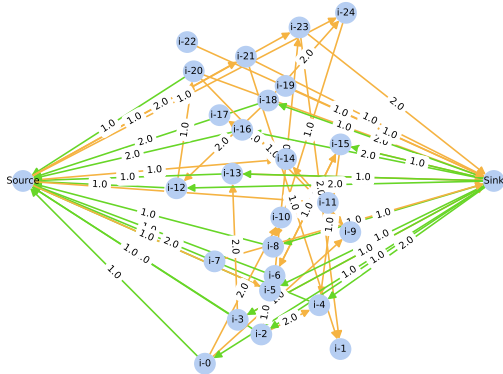
- └ The Maximum flow problem
 - └ Solution with the Ford-Fulkerson algorithm

- ▶ Now the algorithm should be able to run

Overview

- └ 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

Complexity of Ford Fulkerson:

$$\mathcal{O}(|f^*| \times |E|) \quad (3)$$

- └ 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** (that can be irrational), the algorithm might not terminate.

- └ 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 strictly positive capacity.

(Edmonds-Karp algorithm, 1972).

The time complexity is now $\mathcal{O}(|V||E|^2)$, thus **independent** on $|f^*|$.

- └ 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

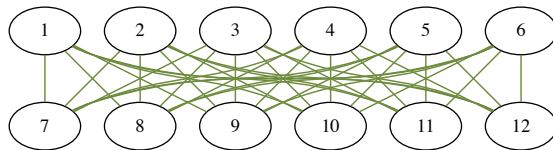


Figure: Complete bipartite graph (not all bipartite graphs are complete)

- └ The Maximum flow problem
 - └ Connection with the matching problem

Matching problem

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

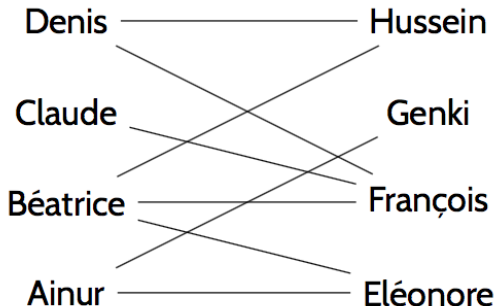


Figure: Bipartite graph

Equivalence between matching and flow

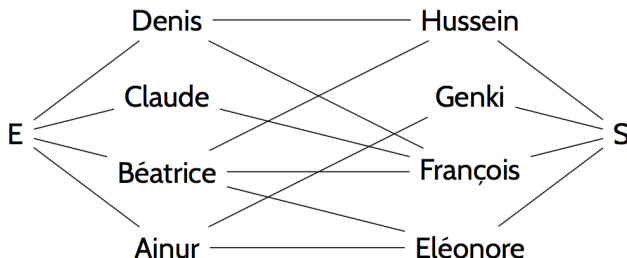


Figure: Introduce two more nodes. All edges have capacity 1. We consider **flows with integer values**

Ford Fulkerson for matching

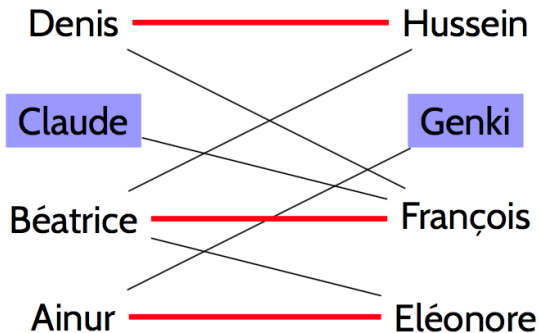


Figure: Non optimal solution

Ford Fulkerson for matching

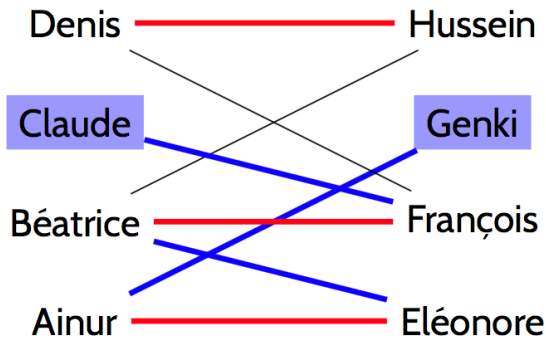


Figure: Optimal solution

Connection

Exercise 4: Find a connection between the two problems

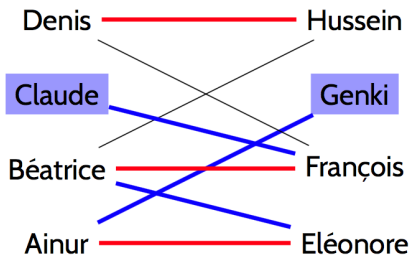


Figure: Optimal solution

- └ The Maximum flow problem
- └ Connection with the matching problem

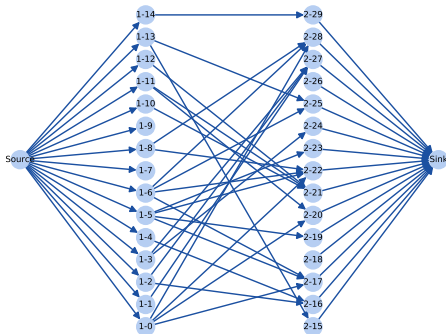
Ford Fulkerson and matching

- In the folder **ford_matching/**, the scripts apply Ford Fulkerson to a bipartite graph in order to find an optimal matching.

Overview

- └ The Maximum flow problem
 - └ Connection with the matching problem

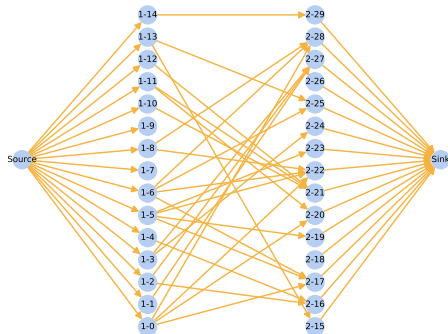
initial graph



Overview

- └ The Maximum flow problem
 - └ Connection with the matching problem

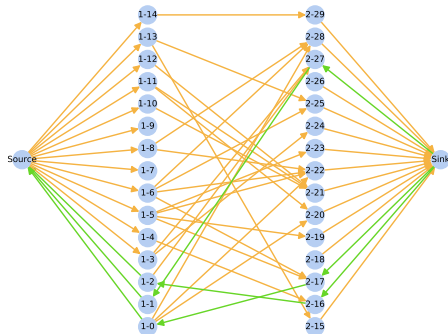
residual graph step 1



Overview

- └ The Maximum flow problem
 - └ Connection with the matching problem

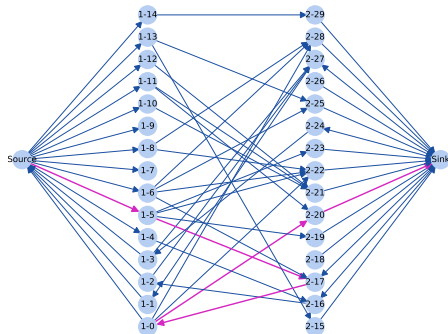
residual graph step 4



Overview

- └ The Maximum flow problem
 - └ Connection with the matching problem

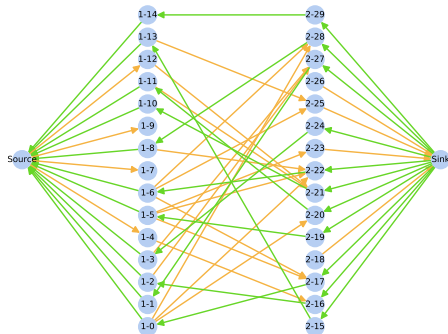
augmenting path step 5



Overview

- └ The Maximum flow problem
 - └ Connection with the matching problem

residual graph step 12



- └ The Maximum flow problem
- └ More results on the two problems

Famous theorem

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

- └ The Maximum flow problem
- └ More results on the two problems

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.

- └ The Maximum flow problem
- └ More results on the two problems

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 :

$$\forall X \subset U, |N_G(X)| \geq |X| \quad (4)$$

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

- └ The Maximum flow problem
- └ More results on the two problems

Hall's theorem

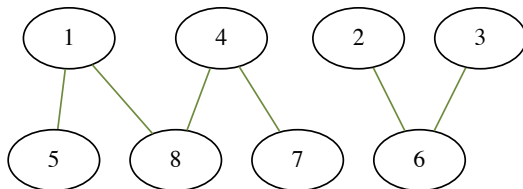
Exercise 5 : Application of the theorem.

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

- └ The Maximum flow problem
- └ More results on the two problems

Illustration of Hall's theorem

Exercise 5 : Application of the theorem



- └ The Maximum flow problem
- └ More results on the two problems

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**.

- └ The Maximum flow problem
- └ More results on the two problems

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.