

# Informatik II

## Übung 11

FS 2020

# Heutiges Programm

- 1 Feedback letzte Übung
- 2 Wiederholung Vorlesung
- 3 In-Class-Exercise (praktisch)

# 1. Feedback letzte Übung

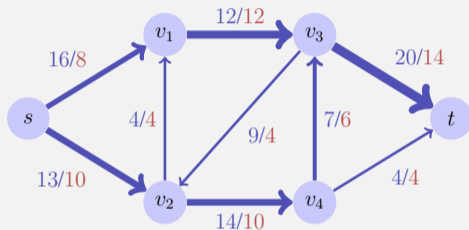
## **2. Wiederholung Vorlesung**

# Fluss

Ein *Fluss*  $f : V \times V \rightarrow \mathbb{R}$  erfüllt folgende Bedingungen:

- **Kapazitätsbeschränkung:**  
Für alle  $u, v \in V$ :  $f(u, v) \leq c(u, v)$ .
- **Schiefsymmetrie:**  
Für alle  $u, v \in V$ :  $f(u, v) = -f(v, u)$ .
- **Flusserhaltung:**  
Für alle  $u \in V \setminus \{s, t\}$ :

$$\sum_{v \in V} f(u, v) = 0.$$



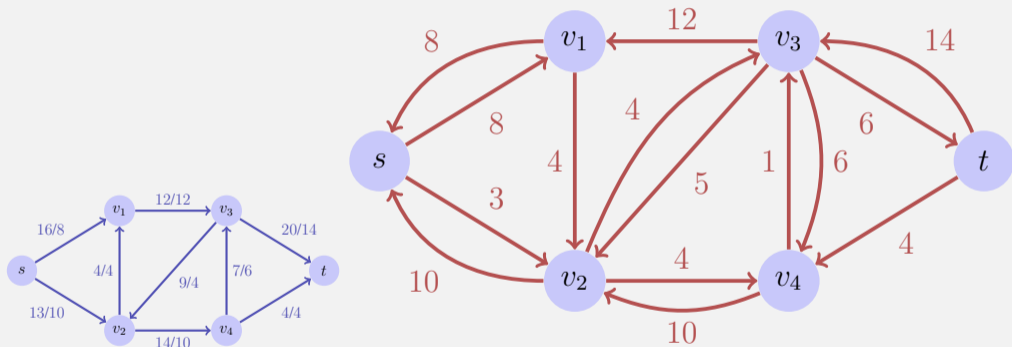
**Wert**  $w$  des Flusses:

$$|f| = \sum_{v \in V} f(s, v).$$

Hier  $|f| = 18$ .

# Restnetzwerk

*Restnetzwerk*  $G_f$  gegeben durch alle Kanten mit Restkapazität:



Restnetzwerke haben dieselben Eigenschaften wie Flussnetzwerke, ausser dass antiparallele Kapazitäten-Kanten zugelassen sind.

# Erweiterungspfade

*Erweiterungspfad*  $p$ : einfacher Pfad von  $s$  nach  $t$  im Restnetzwerk  $G_f$ .

*Restkapazität*  $c_f(p) = \min\{c_f(u, v) : (u, v) \text{ Kante in } p\}$

# Max-Flow Min-Cut Theorem

## Theorem

*Wenn  $f$  ein Fluss in einem Flussnetzwerk  $G = (V, E, c)$  mit Quelle  $s$  und Senke  $t$  ist, dann sind folgende Aussagen äquivalent:*

- 1  $f$  ist ein maximaler Fluss in  $G$*
- 2 Das Restnetzwerk  $G_f$  enthält keine Erweiterungspfade*
- 3 Es gilt  $|f| = c(S, T)$  für einen Schnitt  $(S, T)$  von  $G$ .*



# Algorithmus Ford-Fulkerson( $G, s, t$ )

**Input:** Flussnetzwerk  $G = (V, E, c)$

**Output:** Maximaler Fluss  $f$ .

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

└  $f(u, v) \leftarrow 0$

**while** Existiert Pfad  $p : s \rightsquigarrow t$  im Restnetzwerk  $G_f$  **do**

└  $c_f(p) \leftarrow \min\{c_f(u, v) : (u, v) \in p\}$

└ **foreach**  $(u, v) \in p$  **do**

└└  $f(u, v) \leftarrow f(u, v) + c_f(p)$

└└  $f(v, u) \leftarrow f(v, u) - c_f(p)$

# Praktische Anmerkung

In einer Implementation des Ford-Fulkerson Algorithmus müssen die negativen Flusskanten nicht unbedingt gespeichert werden, da ihr Wert sich stets als der negierter Wert der Gegenkante ergibt.

$$f(u, v) \leftarrow f(u, v) + c_f(p)$$

$$f(v, u) \leftarrow f(v, u) - c_f(p)$$

wird dann zu

**if**  $(u, v) \in E$  **then**

$$\quad | \quad f(u, v) \leftarrow f(u, v) + c_f(p)$$

**else**

$$\quad | \quad f(v, u) \leftarrow f(v, u) - c_f(p)$$

# Edmonds-Karp Algorithmus

Wähle in der Ford-Fulkerson-Methode zum Finden eines Pfades in  $G_f$  jeweils einen Erweiterungspfad kürzester Länge (z.B. durch Breitensuche).

# Edmonds-Karp Algorithmus

## Theorem

*Wenn der Edmonds-Karp Algorithmus auf ein ganzzahliges Flussnetzwerk  $G = (V, E)$  mit Quelle  $s$  und Senke  $t$  angewendet wird, dann ist die Gesamtanzahl der durch den Algorithmus angewendete Flusserhöhungen in  $\mathcal{O}(|V| \cdot |E|)$ .*

*$\Rightarrow$  Gesamte asymptotische Laufzeit:  $\mathcal{O}(|V| \cdot |E|^2)$*

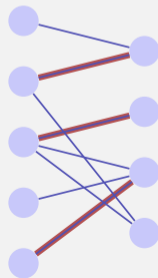
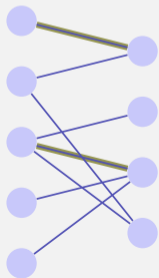
[Ohne Beweis]

# Anwendung: Maximales bipartites Matching

Gegeben: bipartiter ungerichteter Graph  $G = (V, E)$ .

**Matching**  $M$ :  $M \subseteq E$  so dass  $|\{m \in M : v \in m\}| \leq 1$  für alle  $v \in V$ .

**Maximales Matching**  $M$ : Matching  $M$ , so dass  $|M| \geq |M'|$  für jedes Matching  $M'$ .



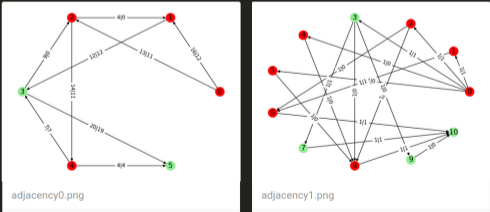
# 3. In-Class-Exercise (praktisch)

Maximale Flüsse – Implementation

# Max-Flow Implementation

Maximum Flow - Master Solution

```
21 for x in self.capacities:
22     print(x)
23
24 # Breadth first search on the graph
25 # return predecessor vector
26 def BFS(self):
27     n = len(self.capacities)
28     s = 0
29     predecessor = {s:s}
30
31     queue = Queue();
32     queue.put(s);
33
34     while not queue.empty():
35         u = queue.get()
36         for v in range(0,n):
37             if self.has_edge(u,v):
38
```



adjacency0.png

adjacency1.png

Console HTML Files

Felix Oliver Friedrich

Already published

Publish Solution & Template

## Maximum Flow

The building blocks of the Edmonds Karp (Ford Fulkerson) algorithm are the computation of the residual network, Breadth First Search (for finding the augmented graph in the residual network), and augmentation of the flow by the path value.

In this exercise, the graph  $G = (V, E, c)$  is represented as adjacency matrix that stores the capacities. Here,  $V$  is represented by numbers  $V = \{0, \dots, n-1\}$ ,  $c$  is the adjacency matrix and  $(u, v) \in E$  whenever  $c[u, v] > 0$ .

### Task

Complement methods `Graph.augment` and `Graph.residual` in `graph.py` such that the Ford Fulkerson (Edmonds Karp) algorithm provided in `main.py` computes the maximal flow of the given network.

Make use of `graph.print()` when you want to see the capacities, and of `plot_graph.show()` when you want to see a visualisation. Everything is prepared in `main.py`.

Fragen?