

Informatik II

Übung 11

FS 2020

Program Today

- 1 Feedback of last exercise
- 2 Repetition of Lecture
- 3 In-Class-Exercise (practical)

1. Feedback of last exercise

2. Repetition of Lecture

Flow

A *Flow* $f : V \times V \rightarrow \mathbb{R}$ fulfills the following conditions:

- *Bounded Capacity*:

For all $u, v \in V$: $f(u, v) \leq c(u, v)$.

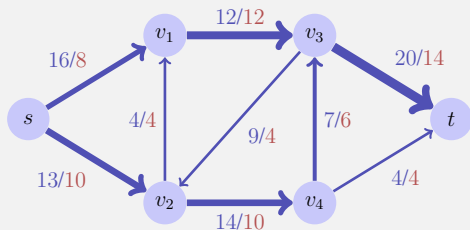
- *Skew Symmetry*:

For all $u, v \in V$: $f(u, v) = -f(v, u)$.

- *Conservation of flow*:

For all $u \in V \setminus \{s, t\}$:

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



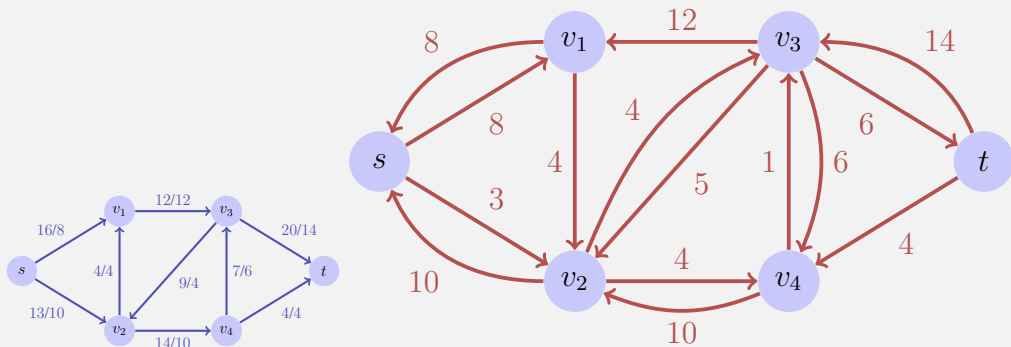
Value of the flow:

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

Here $|f| = 18$.

Rest Network

Rest network G_f provided by the edges with positive rest capacity:



Rest networks provide the same kind of properties as flow networks with the exception of permitting antiparallel capacity-edges

Augmenting Paths

expansion path p : simple path from s to t in the rest network G_f .

Rest capacity $c_f(p) = \min\{c_f(u, v) : (u, v) \text{ edge in } p\}$

Max-Flow Min-Cut Theorem

Theorem

Let f be a flow in a flow network $G = (V, E, c)$ with source s and sink t . The following statements are equivalent:

- 1 f is a maximal flow in G*
- 2 The rest network G_f does not provide any expansion paths*
- 3 It holds that $|f| = c(S, T)$ for a cut (S, T) of G .*

Algorithm Ford-Fulkerson(G, s, t)

Input: Flow network $G = (V, E, c)$

Output: Maximal flow f .

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

└ $f(u, v) \leftarrow 0$

while Exists path $p : s \rightsquigarrow t$ in rest network G_f **do**

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

└ **foreach** $(u, v) \in p$ **do**

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

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

Practical Consideration

In an implementation of the Ford-Fulkerson algorithm the negative flow edges do not necessarily have to be stored because their value always equals the negated value of the antiparallel edge.

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

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

is then transformed to

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 Algorithm

Choose in the Ford-Fulkerson-Method for finding a path in G_f the expansion path of shortest possible length (e.g. with BFS)

Edmonds-Karp Algorithm

Theorem

When the Edmonds-Karp algorithm is applied to some integer valued flow network $G = (V, E)$ with source s and sink t then the number of flow increases applied by the algorithm is in $\mathcal{O}(|V| \cdot |E|)$.

\Rightarrow Overall asymptotic runtime: $\mathcal{O}(|V| \cdot |E|^2)$

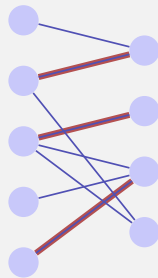
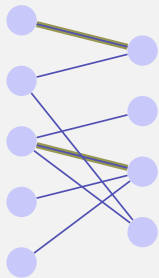
[Without proof]

Application: maximal bipartite matching

Given: bipartite undirected graph $G = (V, E)$.

Matching M : $M \subseteq E$ such that $|\{m \in M : v \in m\}| \leq 1$ for all $v \in V$.

Maximal Matching M : Matching M , such that $|M| \geq |M'|$ for each matching M' .



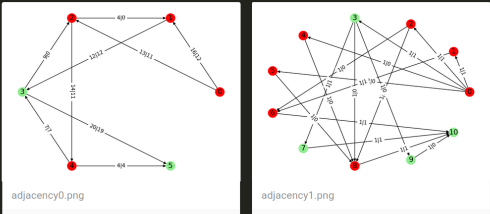
3. In-Class-Exercise (practical)

Implementation of Max-Flow

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`.

Questions?