

Informatik II

Übung 13

FS 2019

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

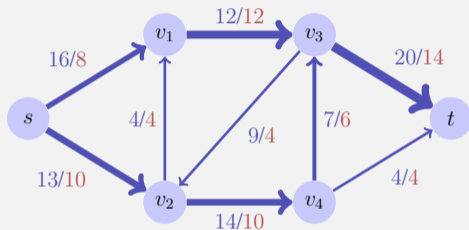
- *Skew Symmetry:*

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

- *Conservation of flow:*

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

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



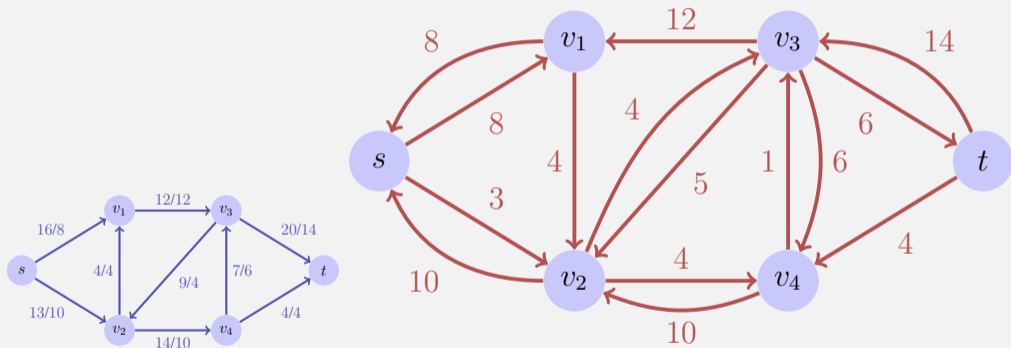
Value of the flow:

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

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 are usually not 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**

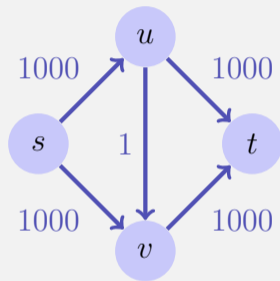
$$\quad | \quad f(u, v) \leftarrow f(u, v) + c_f(p)$$

else

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

Analysis

- For an integer flow, the algorithm requires maximally $|f_{\max}|$ iterations of the while loop (because the flow increases minimally by 1). Search a single increasing path (e.g. with DFS or BFS) $\mathcal{O}(|E|)$ Therefore overall running time in $\mathcal{O}(f_{\max}|E|)$.



With an unlucky choice the algorithm may require up to 2000 iterations here.

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: $\mathcal{O}(|V| \cdot |E|^2)$

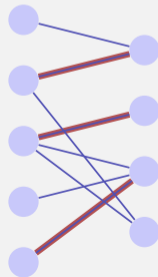
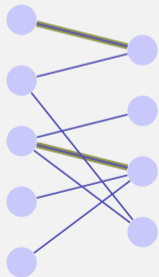
[Without proof]

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

https://expert.ethz.ch/ide2/toCfbTjgRCdMFahMQ?fileKey=pkSZLA7Hbp7qwDrBH

ETH Code Expert Service Lectures at D... Datenstrukturen und ... D-BAUG Informatik II... dict.cc | Wörterbuch E... dict.leo.org - English... Linguee | Deutsch-Eng...

Max Flow - Master Solution

```
31 Node endNode;
32
33 public Graph(int size){
34     this.size = size;
35     nodes = new Node[size];
36     capacities = new int[size][size];
37     residuals = new int[size][size];
38     flows = new int[size][size];
39     for (int i = 0; i < size; ++i)
40         nodes[i] = new Node(i);
41 }
```

Compilation successful
number nodes:6
edges [from to capacity], any neg

```
0 1 16
0 2 13
2 1 4
1 3 12
2 4 14
3 2 9
4 3 7
3 5 20
4 5 4
-1
```

flow = 0
graph: 0 1 16|0 2 13|0 1 3 12|0
flow = 12
graph: 0 1 16|12 0 2 13|0 1 3 12|
flow = 16
graph: 0 1 16|12 0 2 13|4 1 3 12|
flow = 23
graph: 0 1 16|12 0 2 13|11 1 3 12|
plot generated, please observe files

Files

graph0.png

graph1.png

maximal flow of the given network.

The generated graphs are stored in a file and are visualized by a separate Python program.

Do thus not modify method `Graph.toFile`.

The following two example scenarios are contained as tests (when you hit the test button). Networks are defined by specification of

- the number of nodes followed by
- a sequence of edges: triples of source, destination and capacity
- terminated by a negative number

Graph from the lecture:

```
6
0 1 16
0 2 13
2 1 4
1 3 12
2 4 14
3 2 9
4 3 7
3 5 20
4 5 4
-1
```

Matching example from the lecture:

```
11
0 1 1
0 2 1
0 3 1
```


Questions?