# Algorithm Theory

## Chapter 6
## Graph Algorithms

### Part II:
### Basic Ford Fulkerson Analysis

Fabian Kuhn

# Ford-Fulkerson Algorithm

- Improve flow using an augmenting path as long as possible:

1. Initially, $f(e) = 0$ for all edges $e \in E$, $G_f = G$
2. **while** there is an augmenting $s$-$t$-path $P$ in $G_f$ **do**
3. Let $P$ be an augmenting $s$-$t$-path in $G_f$;
4. $f' \coloneqq \text{augment}(f, P)$;
5. update $f$ to be $f'$;
6. update the residual graph $G_f$
7. **end**;

# Ford-Fulkerson Running Time

**Theorem:** If all edge *capacities* are *integers*, the Ford-Fulkerson algorithm terminates after at most $C$ iterations, where

$$C = \text{"max flow value"} \leq \sum_{e \text{ out of } s} c_e \,.$$

**Proof:**

1. At all times, for all $e \in E$, $f(e)$ is an integer
   - Initially: $f(e) = 0$
   - In one iteration:
     - augmenting path $P$: all residual capacities are integers
     - $\text{bottleneck}(P, f) > 0$ and also $\text{bottleneck}(P, f)$ is an integer
     - $f'(e) = f(e)$ or $f'(e) = f(e) \pm \text{bottleneck}(P, f)$

2. New flow value $|f'| = |f| + \text{bottleneck}(P, f) \geq |f| + 1$

   $\Longrightarrow$ **#iterations $\leq C$**

# Ford-Fulkerson Running Time

**Theorem:** If all edge *capacities* are *integers*, the Ford-Fulkerson algorithm can be implemented to run in $O(mC)$ time.

$m$: #edges

**Proof:**

Show that each of the $\leq C$ iterations requires $O(m)$ time.

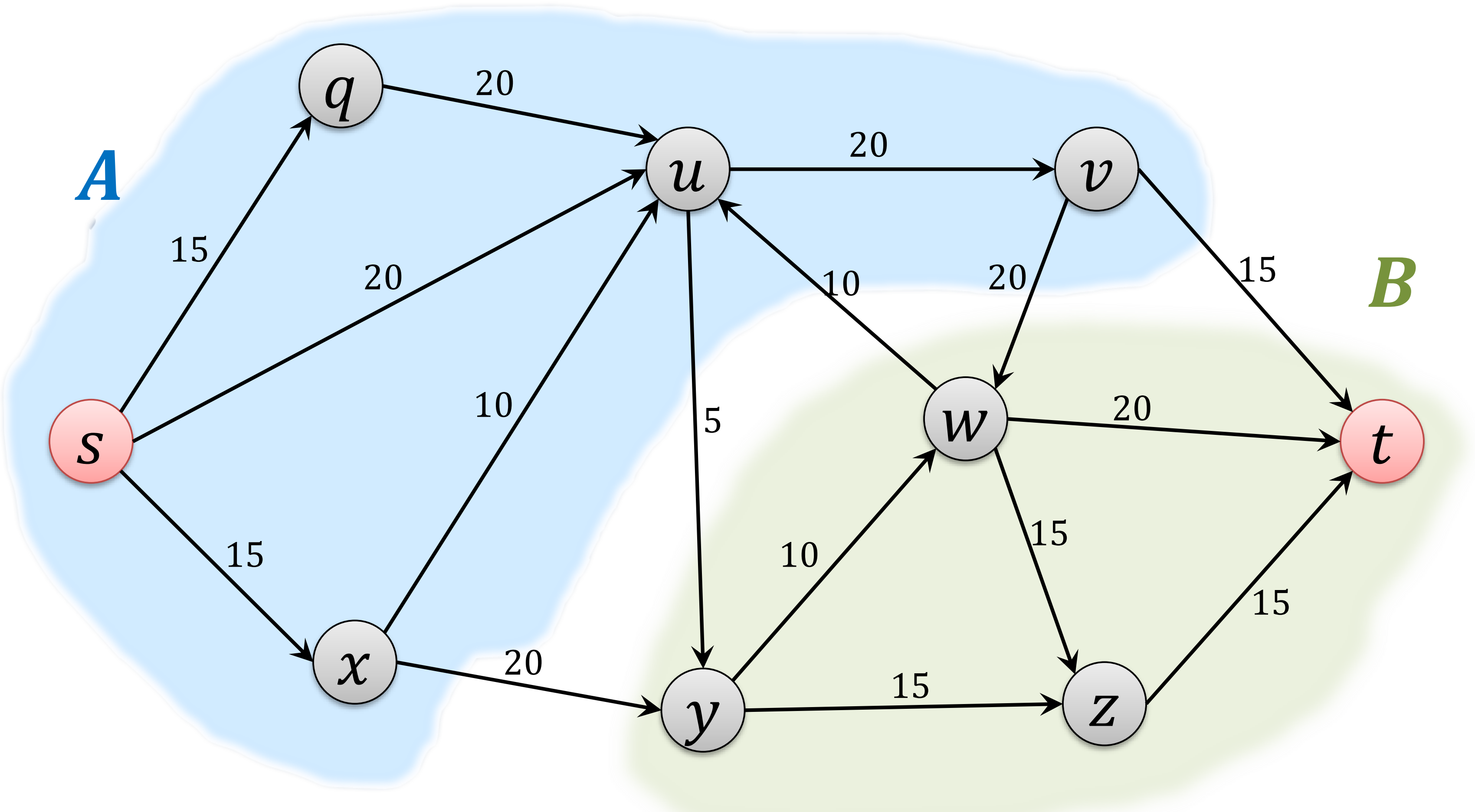
1. Compute / update residual graph:
   - 1st iteration: $O(m)$
   - Later iterations: $O(n)$
2. Find augmenting path / conclude that no augm. path exists

   find positive $s$-$t$ path in residual graph $G_f$

   $\Longrightarrow$ Graph traversal: using DFS or BFS: $O(m)$
3. Update flow values: $O(n)$

# $s$-$t$ Cuts

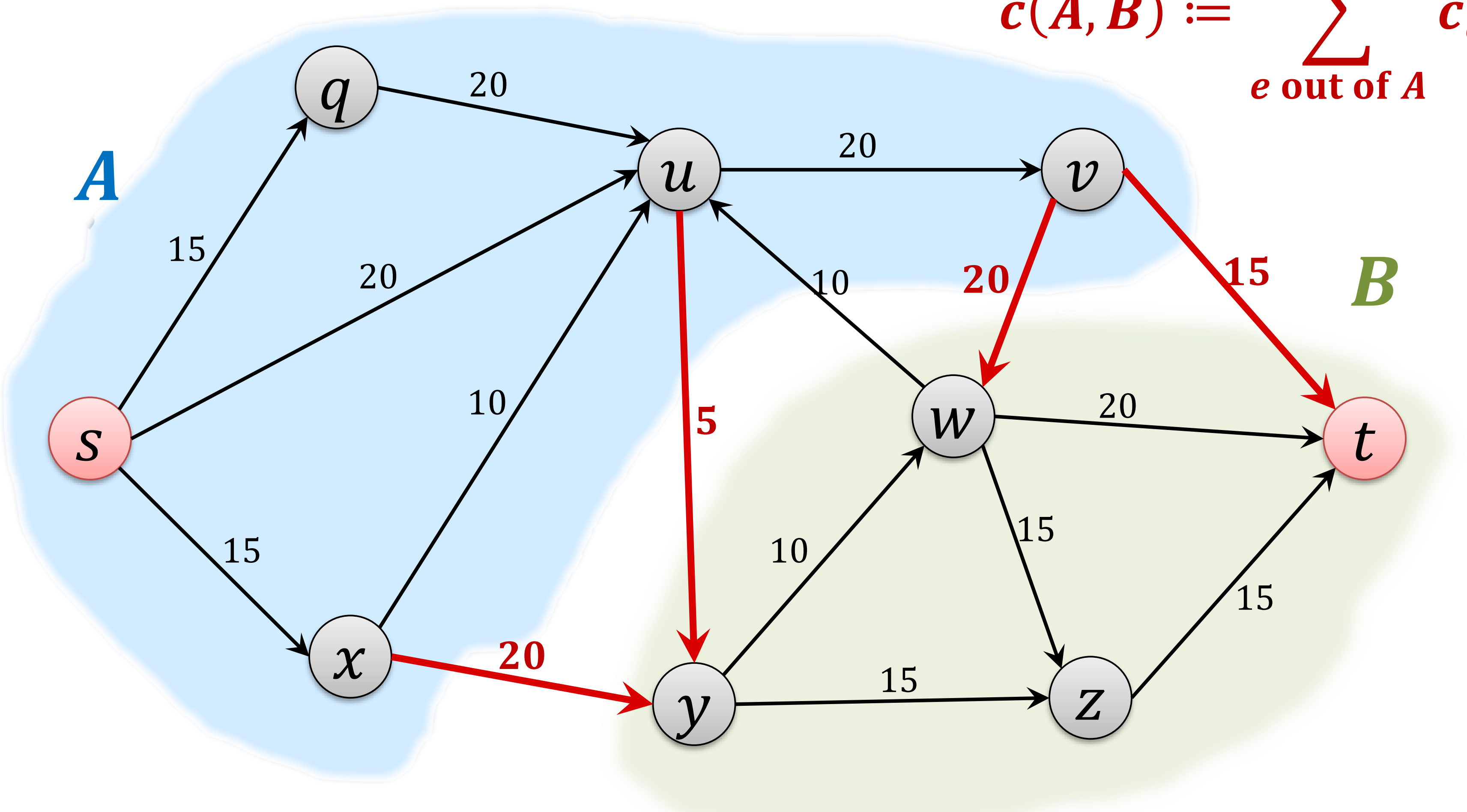
**Definition:**
An $s$-$t$ cut is a partition $(A, B)$ of the vertex set such that $s \in A$ and $t \in B$



# Cut Capacity

**Definition:**
The capacity $c(A, B)$ of an $s$-$t$-cut $(A, B)$ is defined as

$$c(A,B) := \sum_{e \text{ out of } A} c_e \, .$$



# Cuts and Flow Value

**Lemma:** Let $f$ be any $s$-$t$ flow, and $(A, B)$ any $s$-$t$ cut. Then,

$$|\boldsymbol{f}| = \boldsymbol{f}^{\text{out}}(\boldsymbol{A}) - \boldsymbol{f}^{\text{in}}(\boldsymbol{A}).$$

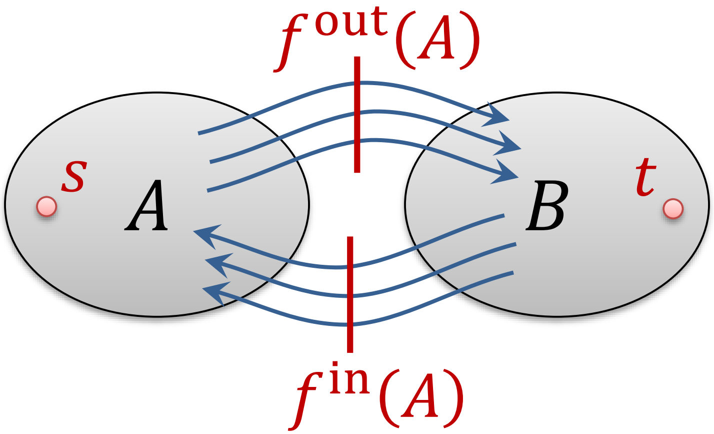
**Proof:**

$$|f| = f^{\text{out}}(s), \qquad \left(= f^{\text{in}}(t)\right)$$

$$|f| = f^{\text{out}}(s) - \underbrace{f^{\text{in}}(s)}_{=0}$$

$$= \sum_{v \in A} \Big(\underbrace{f^{\text{out}}(v) - f^{\text{in}}(v)}_{=0,\ \text{except for } v = s}\Big)$$

$$= f^{\text{out}}(A) - f^{\text{in}}(A)$$



# Cuts and Flow Value

**Lemma:** Let $f$ be any $s$-$t$ flow, and $(A, B)$ any $s$-$t$ cut. Then,

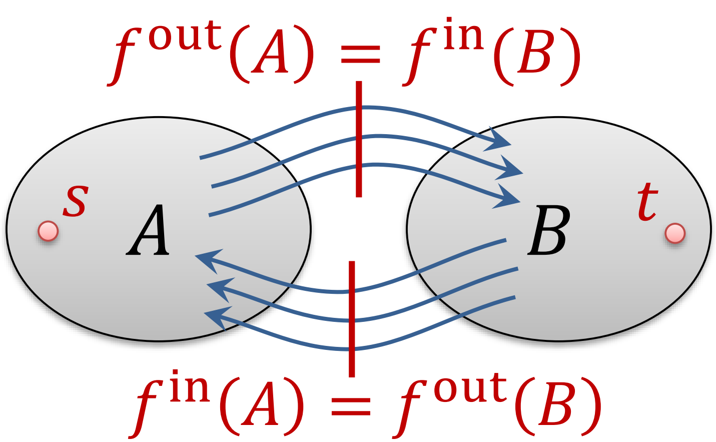
$$|\boldsymbol{f}| = \boldsymbol{f}^{\text{out}}(\boldsymbol{A}) - \boldsymbol{f}^{\text{in}}(\boldsymbol{A}).$$

**Lemma:** Let $f$ be any $s$-$t$ flow, and $(A, B)$ any $s$-$t$ cut. Then,

$$|\boldsymbol{f}| = \boldsymbol{f}^{\text{in}}(\boldsymbol{B}) - \boldsymbol{f}^{\text{out}}(\boldsymbol{B}).$$

**Proof:**

- Either do the same argument as before, symmetrically
- Or, use that $f^{\text{out}}(A) = f^{\text{in}}(B)$ and $f^{\text{in}}(A) = f^{\text{out}}(B)$



# Upper Bound on Flow Value

**Lemma:**

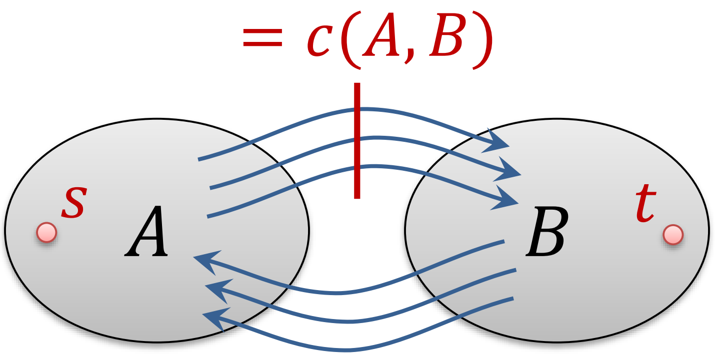
Let $f$ be any $s$-$t$ flow and $(A, B)$ any $s$-$t$ cut. Then $|\boldsymbol{f}| \leq \boldsymbol{c}(\boldsymbol{A}, \boldsymbol{B})$.

**Proof:**

$$|f| = f^{\text{out}}(A) - f^{\text{in}}(A) \leq c(A, B)$$

$$f^{\text{out}}(A) \leq c(A, B)$$

$$f^{\text{in}}(A) \geq 0$$



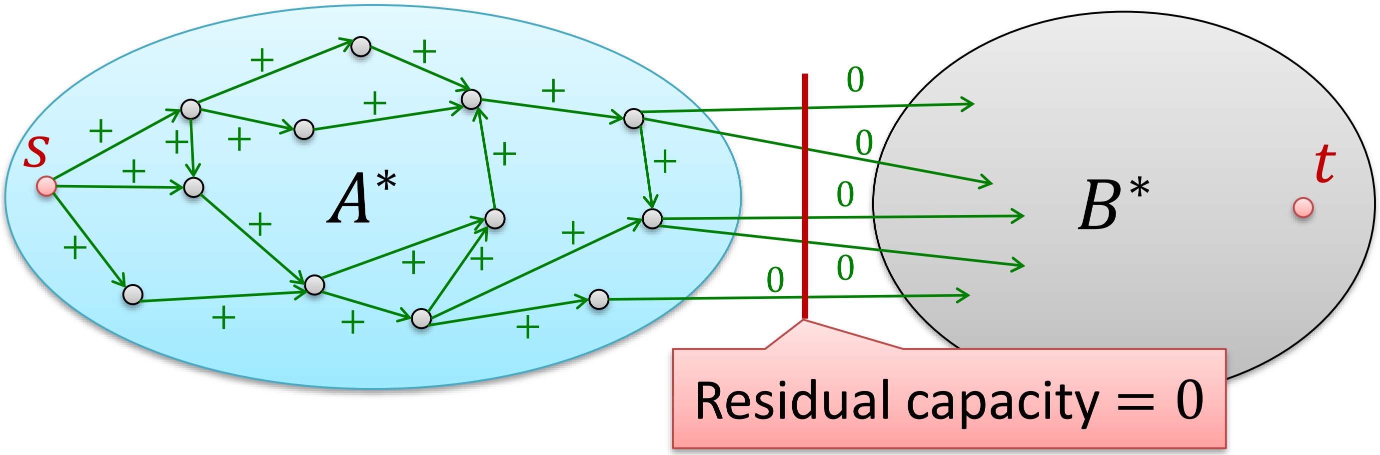
# Ford-Fulkerson Gives Optimal Solution

**Lemma:** If $f$ is an $s$-$t$ flow such that there is no augmenting path in $G_f$, then there is an $s$-$t$ cut $(A^*, B^*)$ in $G$ for which

$$|f| = c(A^*, B^*).$$

**Proof:**

- Define $A^*$: set of nodes that can be reached from $s$ on a path with positive residual capacities in $G_f$:



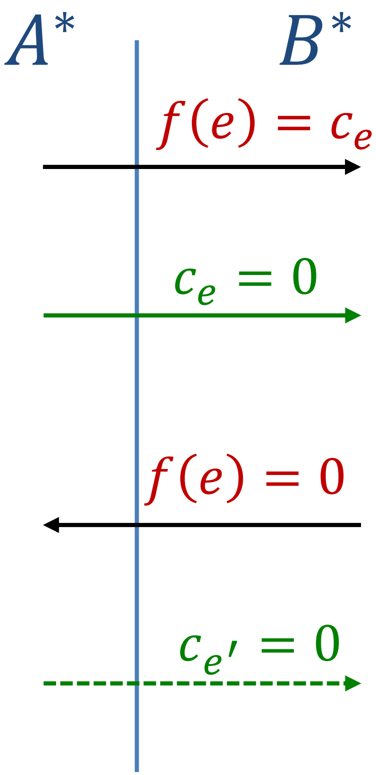

- For $B^* = V \setminus A^*$, $(A^*, B^*)$ is an $s$-$t$ cut
  - By definition $s \in A^*$ and $t \notin A^*$

# Ford-Fulkerson Gives Optimal Solution

**Lemma:** If $f$ is an $s$-$t$ flow such that there is no augmenting path in $G_f$, then there is an $s$-$t$ cut $(A^*, B^*)$ in $G$ for which

$$|\boldsymbol{f}| = \boldsymbol{c}(\boldsymbol{A}^*, \boldsymbol{B}^*).$$

**Proof:**

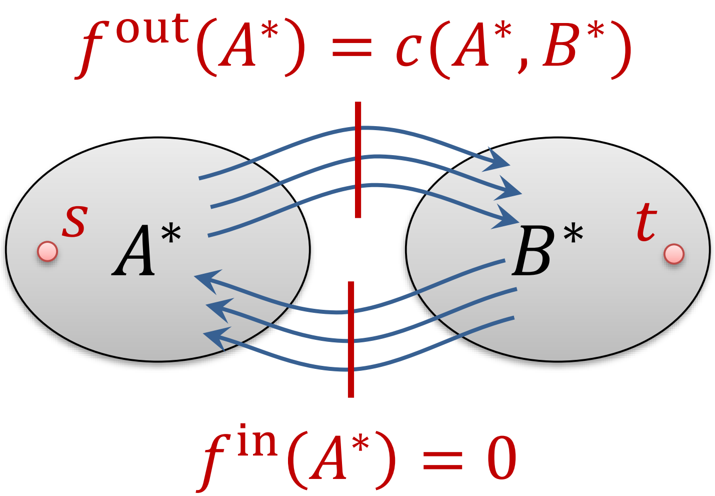



Edge $e$ from $A^*$ to $B^*$ in $G$

Forward edge $e$ in $G_f$

Edge $e$ from $B^*$ to $A^*$ in $G$

Backward edge $e'$ in $G_f$



# Ford-Fulkerson Gives Optimal Solution

**Theorem:** The flow returned by the Ford-Fulkerson algorithm is a maximum flow.

**Proof:**

- Ford-Fulkerson algorithm gives a flow $f^*$ and a cut $(A^*, B^*)$

$$\text{s.t.}\ \ |f^*| = c(A^*, B^*).$$

- We saw that $|f| \leq c(A, B)$ for every valid flow $f$ and every $s$-$t$ cut $(A, B)$.
  - And thus in particular also $|f| \leq c(A^*, B^*)$.

# Min-Cut Algorithm

Ford-Fulkerson also gives a minimum $s$-$t$ cut algorithm:

**Theorem:** Given a flow $f$ of maximum value, we can compute an $s$-$t$ cut of minimum capacity in $O(m)$ time.

**Proof:**

- $f$ maximum $\Longrightarrow$ no augmenting path
- We can therefore construct cut $(A^*, B^*)$ as before
  - By using DFS/BFS on the positive res. cap. edges of $G_f$ in time $O(m)$.

- $(A^*, B^*)$ is a cut of minimum capacity:
  - For every other $s$-$t$ cut $(A, B)$, we know that $|f| \leq c(A, B)$
  - Because $|f| = c(A^*, B^*)$, we therefore have

$$c(A^*, B^*) \leq c(A, B).$$

# Max-Flow Min-Cut Theorem

**Theorem: (Max-Flow Min-Cut Theorem)**

In every flow network, the maximum value of an $s$-$t$ flow is equal to the minimum capacity of an $s$-$t$ cut.

**Proof:**

- Ford-Fulkerson gives a maximum flow $f^*$ and a minimum cut $(A^*, B^*)$ s.t.

$$|f^*| = c(A^*, B^*).$$

# Integer Capacities

**Theorem: (Integer-Valued Flows)**

If all capacities in the flow network are integers, then there is a maximum flow $f$ for which the flow $f(e)$ of every edge $e$ is an integer.

**Proof:**

- If all the capacities are integers, the Ford-Fulkerson algorithm gives an integer solution.
  - By induction on the steps of the algorithm, all flow values are always integers and all residual capacities of $G_f$ are always integers.