A&W 2026

G-13 Timo Stucki, LFW E 13

Week 10

Sort and Select

Networks and Flows with Exercise

Long Paths

Infos

ETH zürich

Feel free to contact me:

- tistucki@student.ethz.ch
- Discord: timostucki

Material:

- timostucki.com

CodeX Tutorial next exercise session?

Quickselect

Optional Recap

Sort and Select

QuickSelect(A, ℓ, r, k)

1: $p \leftarrow \text{Uniform}(\{\ell, \ell+1, \ldots, r\})$ ▷ wähle Pivotelement zufällig
2: $t \leftarrow$ **Partition**(A, ℓ, r, p)
3: **if** $t = \ell + k - 1$ **then**
4: **return** $A[t]$ ▷ gesuchtes Element ist gefunden
5: **else if** $t > \ell + k - 1$ **then**
6: **return** **QuickSelect**$(A, \ell, t-1, k)$ ▷ gesuchtes Element ist links
7: **else**
8: **return** **QuickSelect**$(A, t+1, r, k-t)$ ▷ gesuchtes Element ist rechts

Sort and Select

QuickSelect

- **Goal**: Determine the k-th smallest value in a sequence of distinct numbers.
- **Efficiency**: Solves the selection problem in expected $O(n)$ steps, faster than the $O(n \log n)$ required for full sorting.
- **Logic**: Uses a random pivot p and `PARTITION` to split elements. It recurses only into the side containing the k-th element.

Expected Runtime Analysis

The total runtime T is the sum of comparisons $(r_i - \ell_i)$ over all random calls N.

$$T = \sum_{i=1}^{N} (r_i - \ell_i)$$

To simplify the sum, we group calls by the size of the remaining search space.

- **Definition of N_j:** N_j is the number of `QuickSelect` calls where the problem size $r_i - \ell_i + 1$ falls within the range:

$$(3/4)^j n < r_i - \ell_i + 1 \leq (3/4)^{j-1} n$$

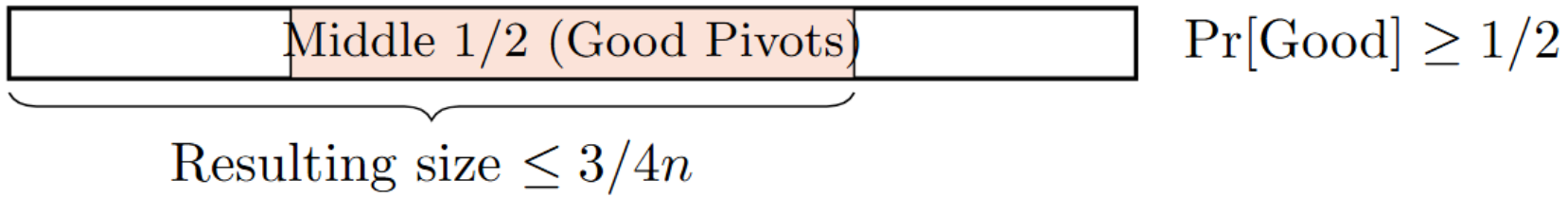
- **Bounding T:** By replacing every $r_i - \ell_i$ in a group with its upper bound $(3/4)^{j-1} n$:

$$T \leq \sum_{j=1}^{\infty} N_j \cdot (3/4)^{j-1} n$$

Sort and Select

3/4-Reduction Principle

- **Good Pivots**: Choosing one of the middle 1/2 elements as the pivot reduces the problem size to at most 3/4 of the current size.
- **Success Probability**: There is at least a 1/2 probability per call to pick such a pivot.
- **Expected** N_j: Because each attempt has a $\geq 1/2$ chance of moving to the next size range $j+1$, the expected number of calls spent in range j is $\mathbb{E}[N_j] \leq 2$.



Expected Runtime Analysis

The total runtime T is the sum of comparisons $(r_i - \ell_i)$ over all random calls N.

$$T = \sum_{i=1}^{N} (r_i - \ell_i)$$

To simplify the sum, we group calls by the size of the remaining search space.

- **Definition of N_j:** N_j is the number of `QuickSelect` calls where the problem size $r_i - \ell_i + 1$ falls within the range:

$$(3/4)^j n < r_i - \ell_i + 1 \leq (3/4)^{j-1} n$$

- **Bounding T:** By replacing every $r_i - \ell_i$ in a group with its upper bound $(3/4)^{j-1} n$:

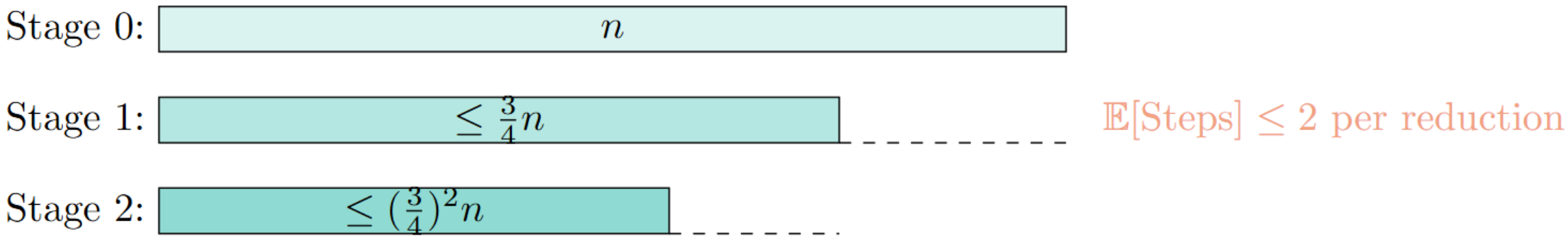
$$T \leq \sum_{j=1}^{\infty} N_j \cdot (3/4)^{j-1} n$$

Sort and Select

By linearity of expectation:

$$\mathbb{E}[T] \leq n \cdot \sum_{j=1}^{\infty} \mathbb{E}[N_j] \cdot (3/4)^{j-1} \leq 2n \sum_{j=1}^{\infty} (3/4)^{j-1} = 2n \cdot \frac{1}{1-3/4} = 8n$$

Conclusion: The expected runtime is linear, proving $\mathbb{E}[T] \in O(n)$.



Hase-Igel

“2.8.5 Finden von Duplikaten” im Skript

Algorithms – Highlights

Algorithms – Highlights

Previously:

Algorithms – Highlights

Previously:

Probability Theory

Algorithms – Highlights

Previously:

Probability Theory

Randomised (non-graph) Algorithms

Algorithms – Highlights

Algorithms – Highlights

Randomised

Deterministic

Algorithms – Highlights

Randomised

Deterministic

Networks and Flows

Networks and Flows

Definition 3.4. Ein *Netzwerk* ist ein Tupel $N = (V, A, c, s, t)$, wobei gilt:

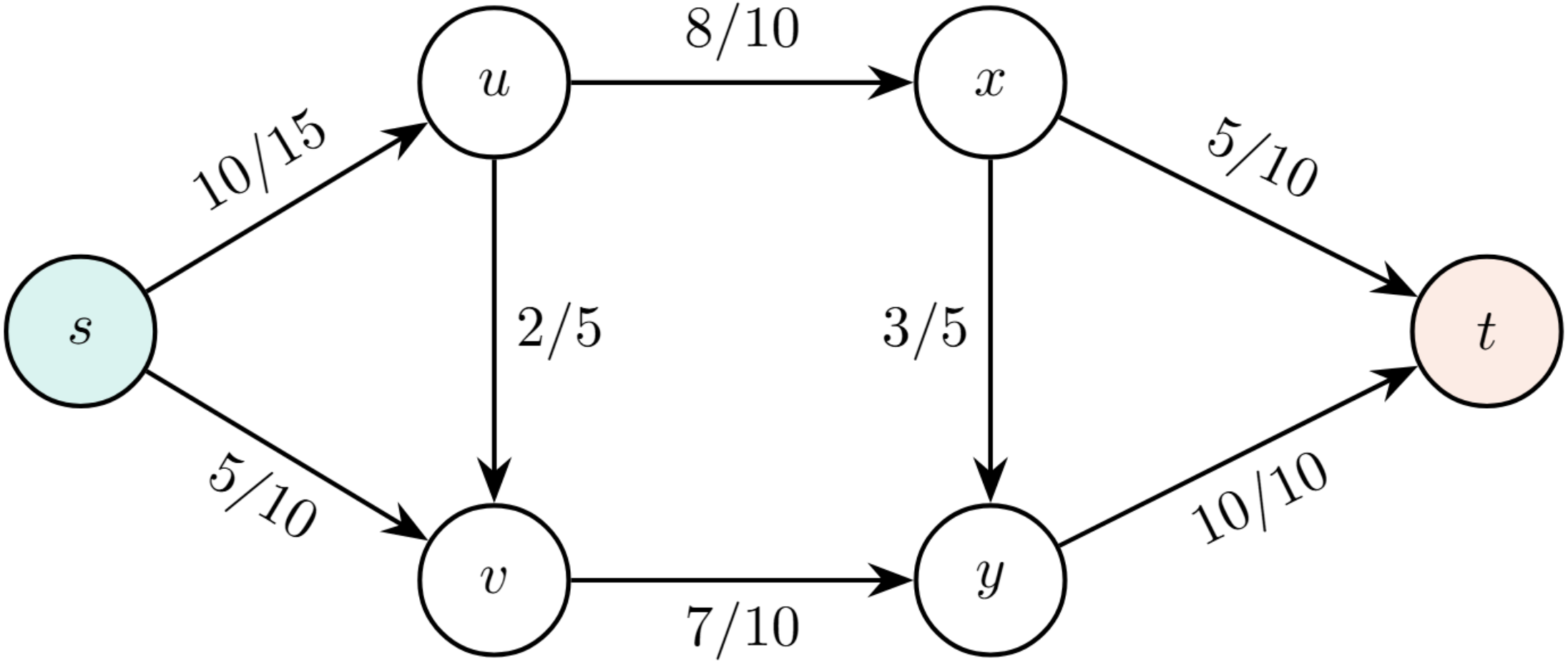
(V, A) ist ein gerichteter Graph,

$s \in V$, die *Quelle* (engl.: source),

$t \in V \setminus \{s\}$, die *Senke* (engl.: target), und

$c : A \to \mathbb{R}_0^+$, die *Kapazitätsfunktion* (engl.: capacity function).

Networks and Flows



Networks and Flows

Definition 3.5. Gegeben sei ein Netzwerk $N = (V, A, c, s, t)$. Ein Fluss in N ist eine Funktion $f : A \to \mathbb{R}$ mit den Bedingungen

$0 \leq f(e) \leq c(e)$ für alle $e \in A$, die *Zulässigkeit*, und

für alle $v \in V \setminus \{s, t\}$ gilt

$$\sum_{u \in V:\,(u,v) \in A} f(u, v) = \sum_{u \in V:\,(v,u) \in A} f(v, u)$$

die *Flusserhaltung*.

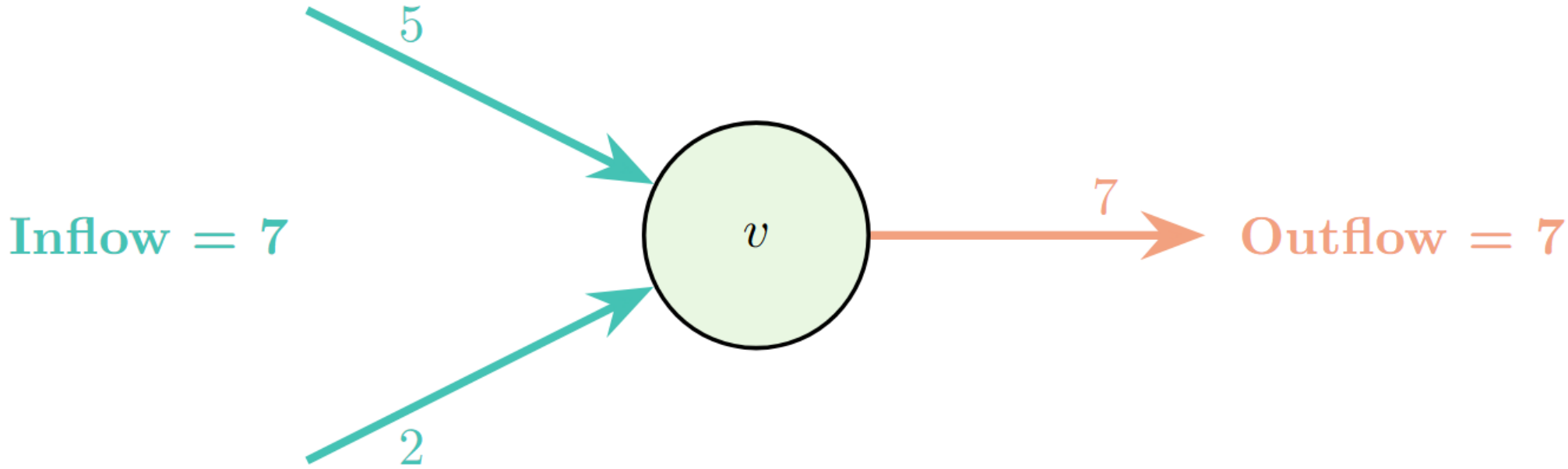
Der *Wert* (engl.: value) eines Flusses f ist durch

$$\mathsf{val}(f) := \mathsf{netoutflow}(s) := \sum_{u \in V:\,(s,u) \in A} f(s, u) - \sum_{u \in V:\,(u,s) \in A} f(u, s)$$

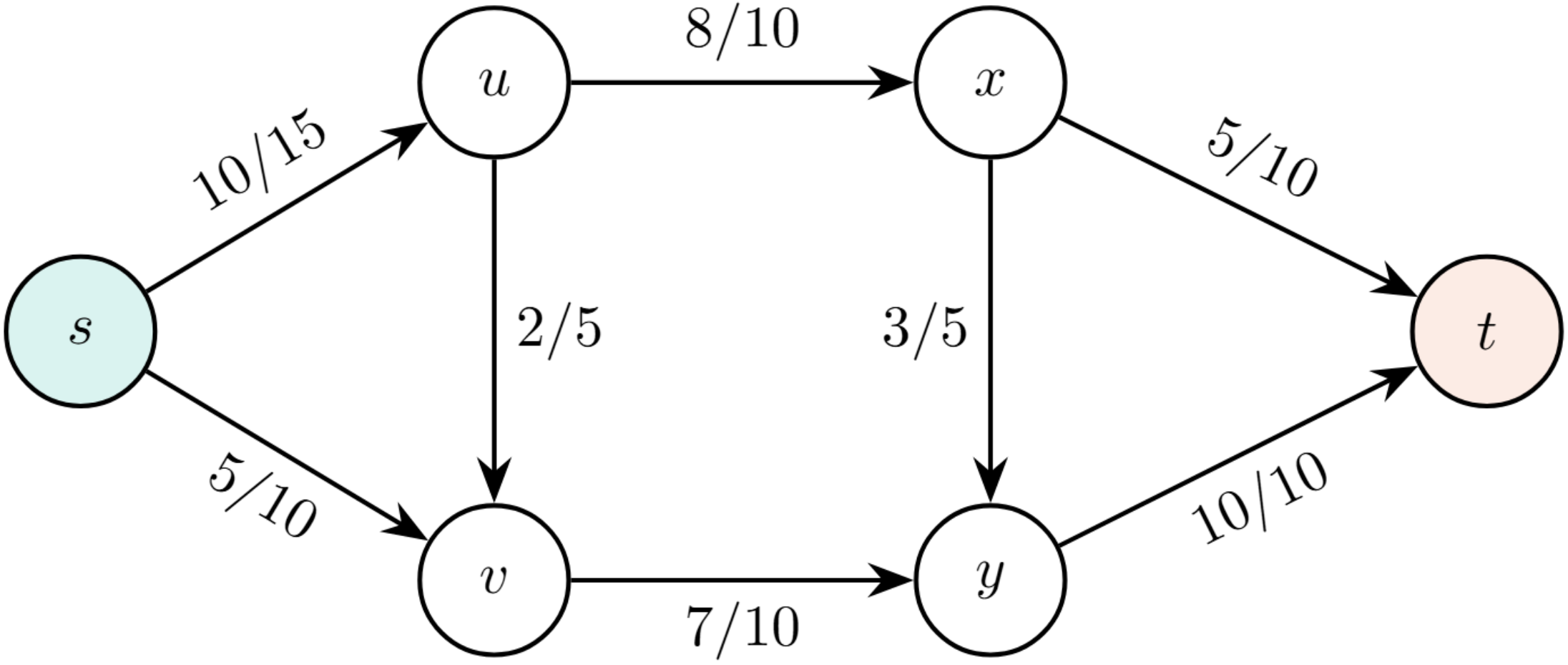
definiert. Wir nennen f *ganzzahlig*, wenn $f(e) \in \mathbb{Z}\ \forall e \in A$.

Networks and Flows

For an intermediate node $v \in V \setminus \{s, t\}$, the inflow must match the outflow.



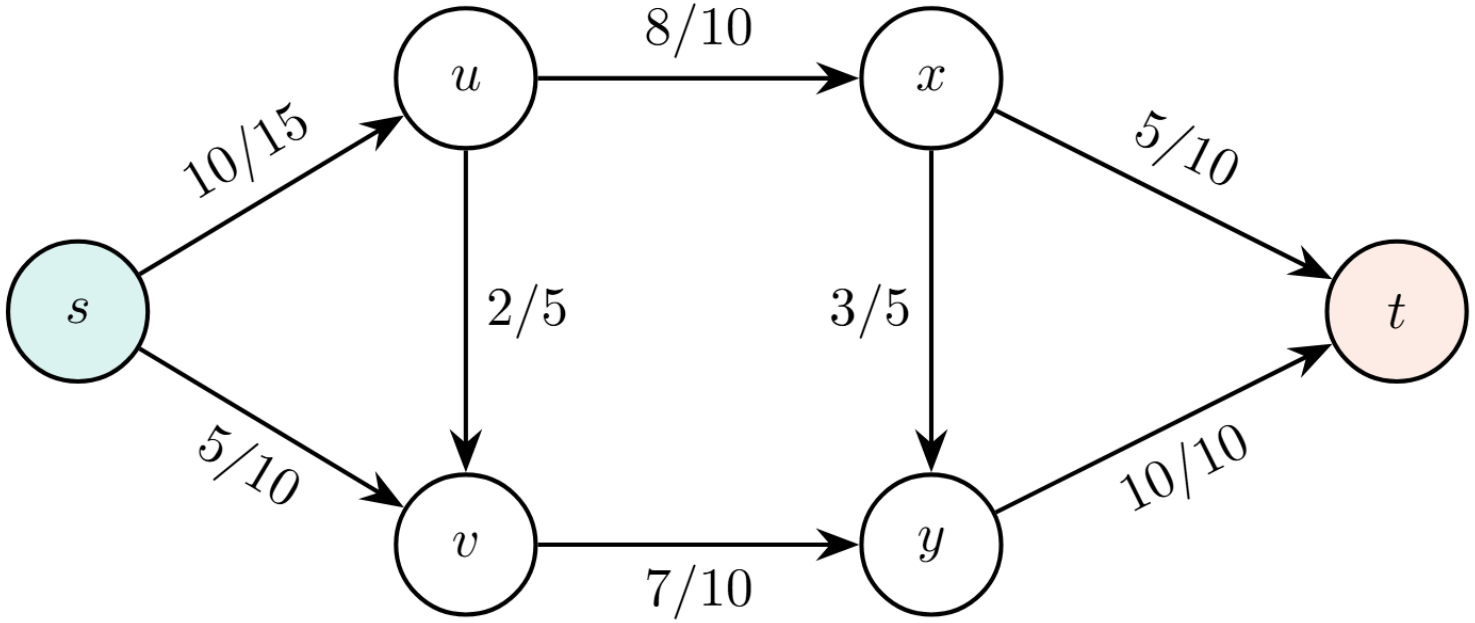
Networks and Flows



Networks and Flows

Lemma 3.6. Der Nettozufluss der Senke gleicht dem Wert des Flusses, d.h.

$$\text{netinflow}(t) := \sum_{u \in V:\,(u,t) \in A} f(u,t) - \sum_{u \in V:\,(t,u) \in A} f(t,u) \quad = \quad \text{val}(f)\ .$$



Networks and Flows

Lemma 3.8. Ist f ein Fluss und (S, T) ein s-t-Schnitt in einem Netzwerk (V, A, c, s, t), so gilt

$$\mathrm{val}(f) \leq \mathrm{cap}(S, T) \ .$$

Networks and Flows

Lemma 3.8. Ist f ein Fluss und (S, T) ein s-t-Schnitt in einem Netzwerk (V, A, c, s, t), so gilt

$$\mathrm{val}(f) \leq \mathrm{cap}(S, T) \ .$$

Beweis. Für eine Partition (U, W) von V setzen wir

$$f(U, W) := \sum_{(u,w) \in (U \times W) \cap A} f(u, w) \ .$$

Nun behaupten wir

$$\mathrm{val}(f) \overset{(i)}{=} f(S, T) - f(T, S) \overset{(ii)}{\leq} f(S, T) \overset{(iii)}{\leq} \mathrm{cap}(S, T), \quad \ldots$$

Networks and Flows

Lemma 3.8. Ist f ein Fluss und (S, T) ein s-t-Schnitt in einem Netzwerk (V, A, c, s, t), so gilt

$$\mathrm{val}(f) \leq \mathrm{cap}(S, T) \ .$$

Satz 3.9 („Maxflow-Mincut Theorem"). Jedes Netzwerk $N = (V, A, c, s, t)$ erfüllt

$$\max_{f \text{ Fluss in } N} \mathrm{val}(f) = \min_{(S,T) \ s\text{-}t\text{-Schnitt in } N} \mathrm{cap}(S, T)$$

Networks and Flows

Definition 3.10. Sei $N = (V, A, c, s, t)$ ein Netzwerk ohne entgegen gerichtete Kanten und sei f ein Fluss in N. Das Restnetzwerk $N_f := (V, A_f, r_f, s, t)$ ist wie folgt definiert:

(1) Ist $e \in A$ mit $f(e) < c(e)$, dann ist e auch eine Kante in A_f, mit $r_f(e) := c(e) - f(e)$.

(2) Ist $e \in A$ mit $f(e) > 0$, dann ist e^{opp} in A_f, mit $r_f(e^{opp}) = f(e)$.

(3) Nur Kanten wie in (1) und (2) beschrieben finden sich in A_f.

$r_f(e)$, $e \in A_f$, nennen wir die *Restkapazität* der Kante e.

Networks and Flows

Satz 3.11. Ein Fluss f in einem Netzwerk N ist ein maximaler Fluss gdw. es im Restnetzwerk N_f keinen gerichteten Pfad von der Quelle s zur Senke t gibt. Für jeden solchen maximalen Fluss gibt es einen s-t-Schnitt (S, T) mit $\mathrm{val}(f) = \mathrm{cap}(S, T)$.

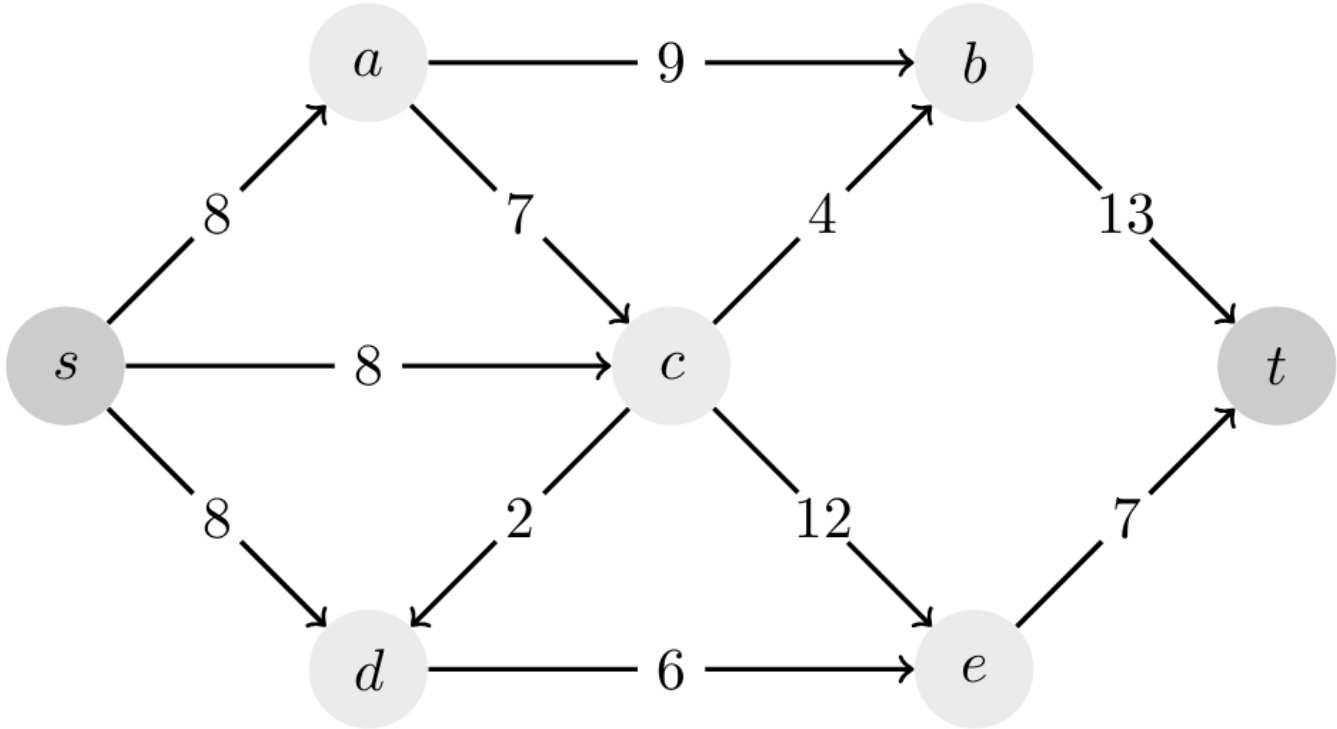
→ **Let's see this in an exercise**

Exercise S11

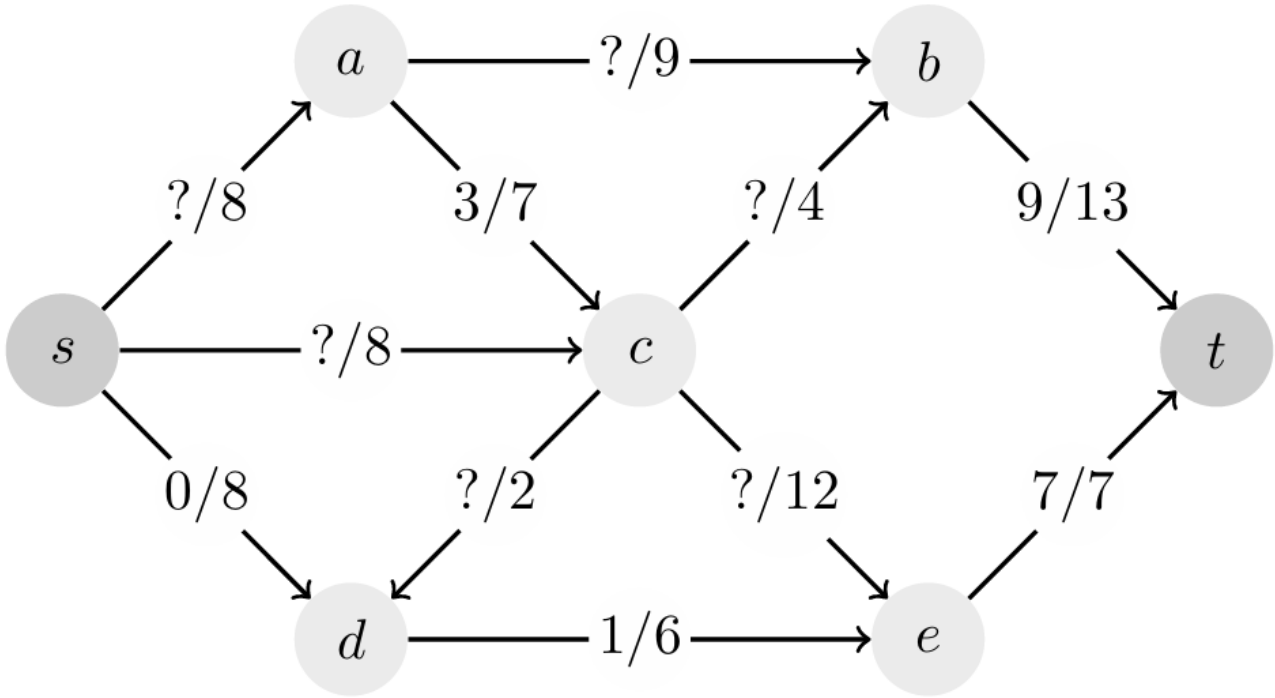
Flows

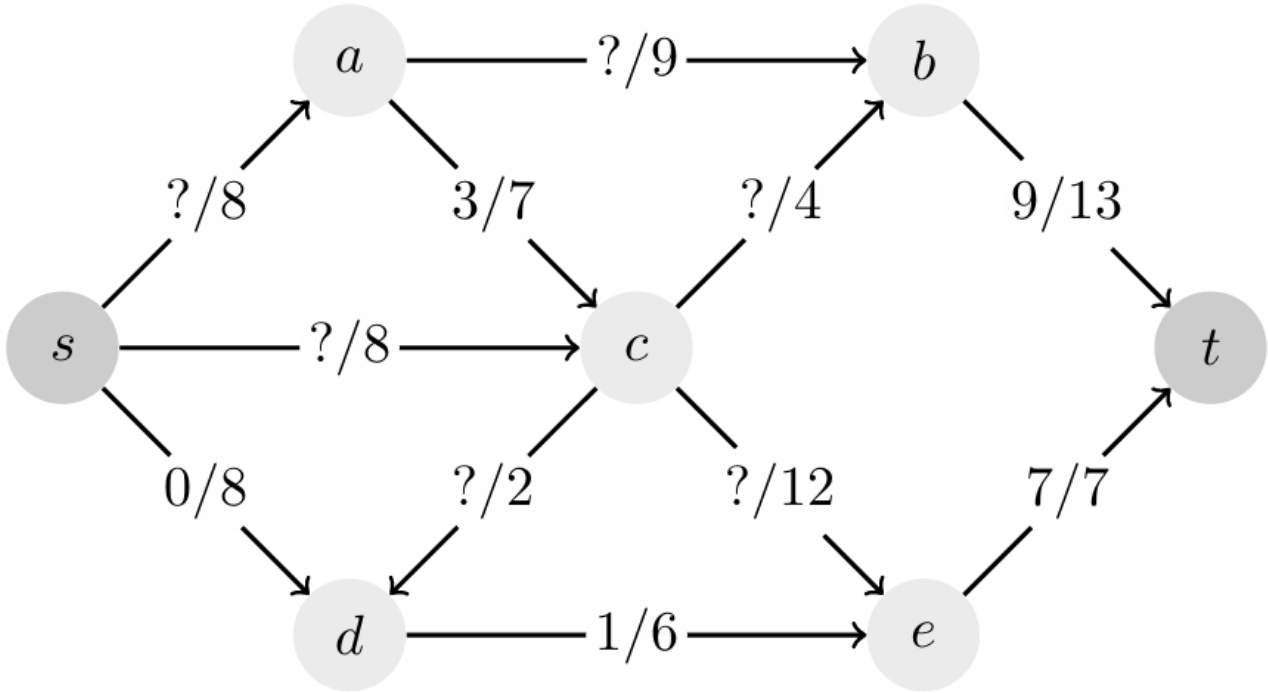
Exercise S11.1 – *Flows*

Consider the following network N. The number at each edge represents its capacity.



(a) Fill in the missing values so that they form a feasible (not necessarily maximum) flow:





(b) Construct the residual network ("Restnetzwerk").

(c) Find an augmenting s-t-path and augment the flow along this path. If necessary, repeat this step until you have found a maximum flow.

(d) Prove that your flow is a maximum flow by finding a minimum cut in N.

Networks and Flows

Networks and Flows

FORD-FULKERSON(V, A, c, s, t)

1: $f \leftarrow \mathbf{0}$ ▷ Fluss konstant 0
2: **while** $\exists$ s-t-Pfad P in (V, A_f) **do** ▷ augmentierender Pfad
3: Erhöhe den Fluss entlang P ▷ wie in Beweis zu Satz 3.11
4: **return** f ▷ maximaler Fluss

Networks and Flows

FORD-FULKERSON(V, A, c, s, t)

1: $f \leftarrow \mathbf{0}$ ▷ Fluss konstant 0
2: **while** $\exists$ s-t-Pfad P in (V, A_f) **do** ▷ augmentierender Pfad
3: Erhöhe den Fluss entlang P ▷ wie in Beweis zu Satz 3.11
4: **return** f ▷ maximaler Fluss

Satz 3.12. Sind in einem Netzwerk ohne entgegen gerichtete Kanten alle Kapazitäten ganzzahlig und höchstens U, so gibt es einen ganzzahligen maximalen Fluss, der in Zeit $O(mnU)$ berechnet werden kann (m ist die Anzahl Kanten, n die Anzahl Knoten im Netzwerk).

Networks and Flows

Number of Iterations

The value of any flow f is bounded by the capacity of the cut $(\{s\}, V \setminus \{s\})$. Since there are at most $n-1$ edges leaving the source s, each with capacity $\leq U$, the maximum flow value is bounded by:

$$val(f) \leq cap(\{s\}, V \setminus \{s\}) \leq (n-1)U < nU$$

Because the total flow value cannot exceed nU, the algorithm terminates after at most nU **iterations**.

Complexity per Iteration

In each iteration, the algorithm performs the following steps:

- Construction of the **residual network** N_f.
- Search for an **augmenting path** from source s to sink t (e.g., using BFS or DFS).
- These operations can be completed in $O(m)$ **time**.

$$\text{Total Runtime} = O(nU) \cdot O(m) = \mathbf{O(mnU)}$$

Long Paths

Summary

Long Paths

Satz 3.1. Falls wir LONG-PATH für Graphen mit n Knoten in $t(n)$ Zeit entscheiden können, dann können wir in $t(2n-2)+O(n^2)$ Zeit entscheiden, ob ein Graph mit n Knoten einen Hamiltonkreis hat.

→ LONG-PATH is **very** likely not solvable in polynomial time

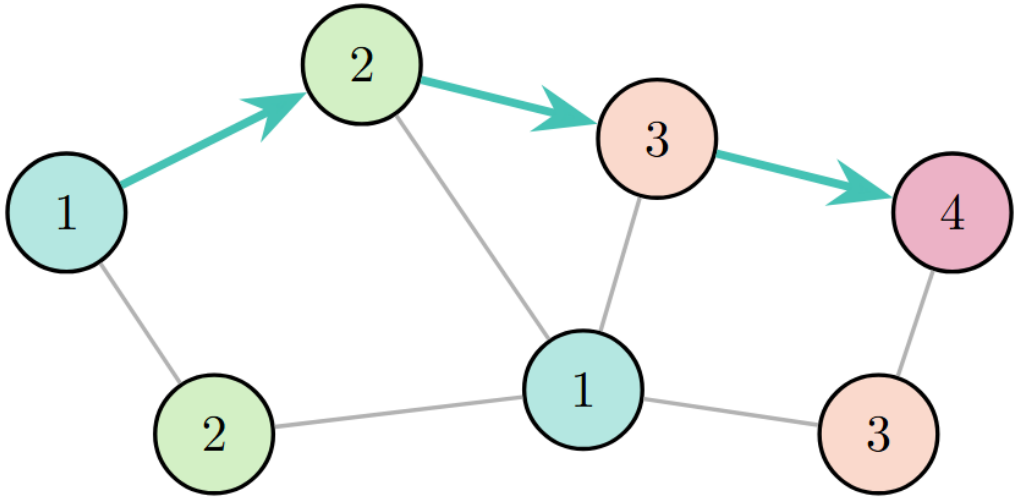
Long Paths

New problem:

"short long paths"

Now we're looking for paths of length **B = log n**

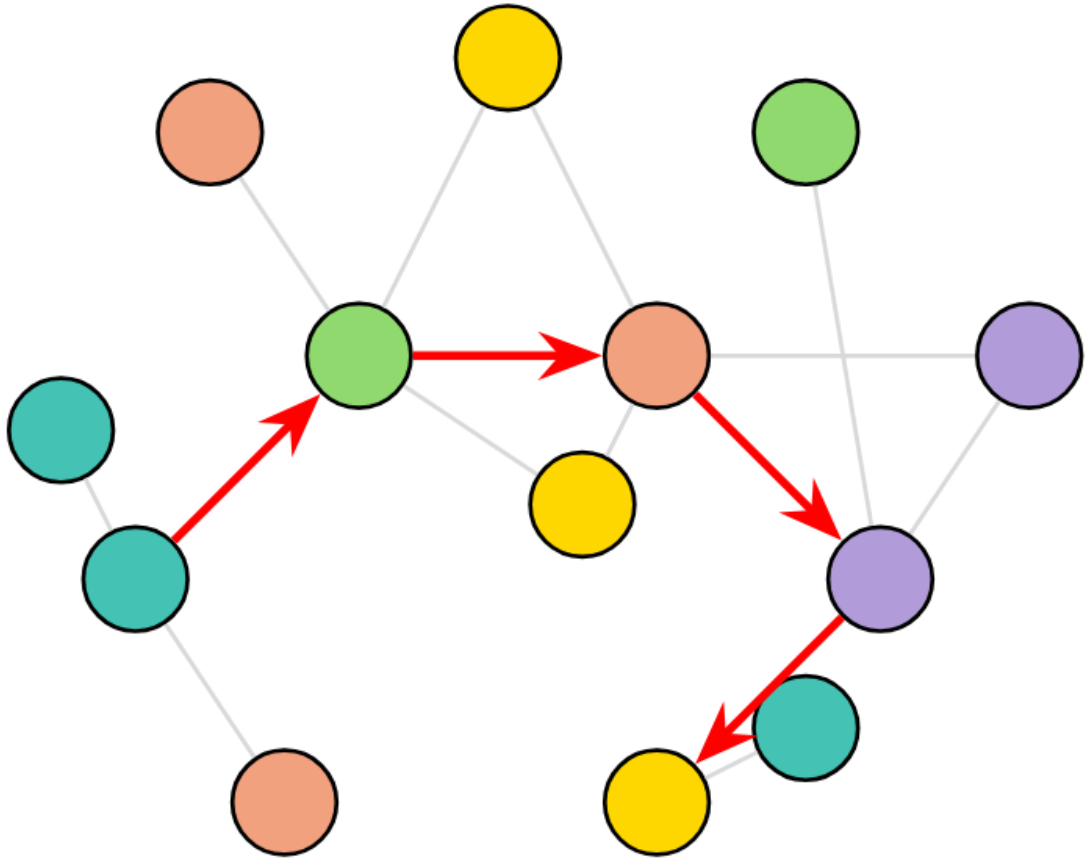
→ We use probabilistic **Colour Coding**



Colorful Path: {Blue, Green, Orange, Purple}

Long Paths

Assign each node a random color $\gamma(v) \in \{1, \ldots, k\}$.
Now we try to find a **colorful path** (all nodes distinct colors).



Long Paths

Dazu definieren wir für $v \in V$ und $i \in \mathbb{N}_0$ die Menge

$$P_i(v) := \left\{ S \in \binom{[k]}{i+1} \middle| \exists \text{ in } v \text{ endender genau mit } S \text{ gefärbter bunter Pfad} \right\};$$

$$\exists \text{ bunter Pfad der Länge } k-1 \iff \bigcup_{v \in V} P_{k-1}(v) \neq \emptyset$$

Long Paths

Dazu definieren wir für $v \in V$ und $i \in \mathbb{N}_0$ die Menge

$$P_i(v) := \left\{ S \in \binom{[k]}{i+1} \,\middle|\, \exists \text{ in } v \text{ endender genau mit } S \text{ gefärbter bunter Pfad} \right\};$$

Offensichtlich gilt $P_0(v) = \{\{\gamma(v)\}\}$ und

$$P_1(v) = \{\{\gamma(x), \gamma(v)\} \mid x \in N(v) \text{ und } \gamma(x) \neq \gamma(v)\}.$$

$$P_i(v) = \bigcup_{x \in N(v)} \left\{ R \cup \{\gamma(v)\} \mid R \in P_{i-1}(x) \text{ und } \gamma(v) \notin R \right\}$$

Long Paths

- Seems familiar?

Long Paths

- Seems familiar?

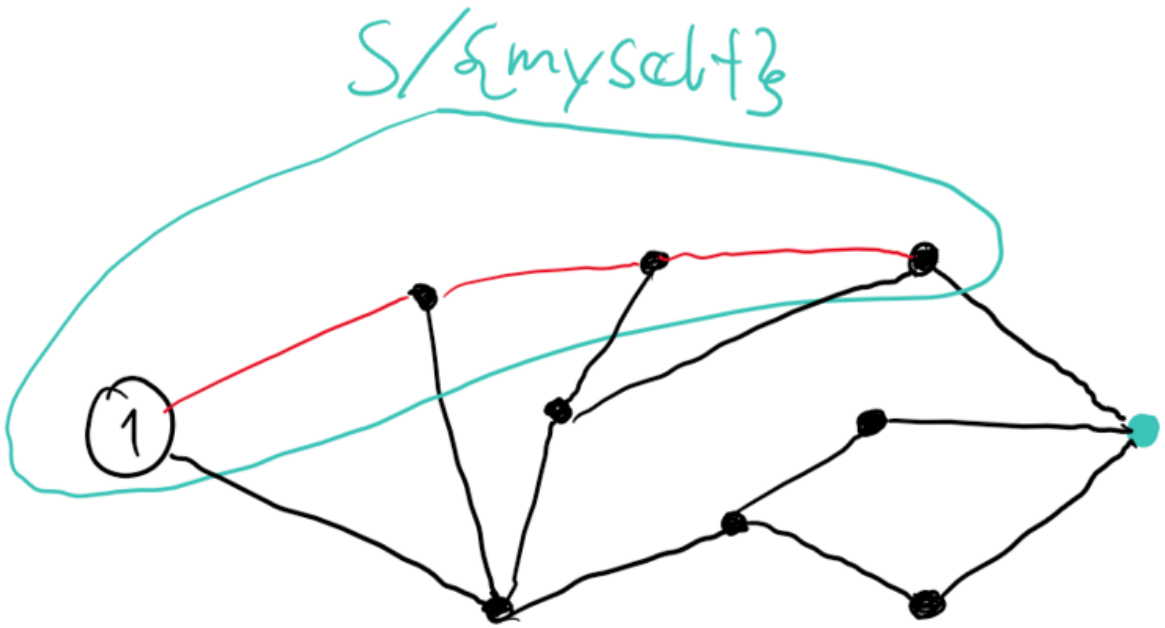
Remember exponential-time Hamiltonian cycle algorithm:

HAMILTONKREIS $(G = ([n], E))$

1: *// Initialisierung*
2: **for all** $x \in [n]$, $x \neq 1$ **do**
3: $\quad P_{\{1,x\},x} := \begin{cases} 1, & \text{falls } \{1,x\} \in E \\ 0, & \text{sonst} \end{cases}$
4: *// Rekursion*
5: **for all** $s = 3$ **to** n **do**
6: $\quad$ **for all** $S \subseteq [n]$ mit $1 \in S$ und $|S| = s$ **do**
7: $\quad\quad$ **for all** $x \in S$, $x \neq 1$ **do**
8: $\quad\quad\quad P_{S,x} = \max\{P_{S\setminus\{x\},x'} \mid x' \in S \cap N(x), x' \neq 1\}.$
9: *// Ausgabe*
10: **if** $\exists\, x \in N(1)$ mit $P_{[n],x} = 1$ **then**
11: $\quad$ **return** G enthält Hamiltonkreis
12: **else**
13: $\quad$ **return** G enthält keinen Hamiltonkreis

Long Paths

Cycles



What paths with S/{myself} are there already to my neighbours?

Long Paths

Dazu definieren wir für $v \in V$ und $i \in \mathbb{N}_0$ die Menge

$$P_i(v) := \left\{ S \in \binom{[k]}{i+1} \,\middle|\, \exists \text{ in } v \text{ endender genau mit } S \text{ gefärbter bunter Pfad} \right\};$$

Offensichtlich gilt $P_0(v) = \{\{\gamma(v)\}\}$ und

$$P_1(v) = \{\{\gamma(x), \gamma(v)\} \mid x \in N(v) \text{ und } \gamma(x) \neq \gamma(v)\}.$$

$$P_i(v) = \bigcup_{x \in N(v)} \left\{ R \cup \{\gamma(v)\} \mid R \in P_{i-1}(x) \text{ und } \gamma(v) \notin R \right\}$$

Long Paths

BUNT(G, i)

1: **for all** $v \in V$ **do**
2: $\quad P_i(v) \leftarrow \emptyset$
3: $\quad$ **for all** $x \in N(v)$ **do**
4: $\quad\quad$ **for all** $R \in P_{i-1}(x)$ **mit** $\gamma(v) \notin R$ **do**
5: $\quad\quad\quad P_i(v) \leftarrow P_i(v) \cup \{R \cup \{\gamma(v)\}\}$

Long Paths

BUNT(G, i)

1: **for all** $v \in V$ **do**
2: $\quad P_i(v) \leftarrow \emptyset$
3: $\quad$ **for all** $x \in N(v)$ **do**
4: $\quad\quad$ **for all** $R \in P_{i-1}(x)$ **mit** $\gamma(v) \notin R$ **do**
5: $\quad\quad\quad P_i(v) \leftarrow P_i(v) \cup \{R \cup \{\gamma(v)\}\}$

Here we only iterate over sets of colours i.e. of **max. size k**

→ Algorithm is only **exponential in k** and not in n or m

Long Paths

(Since the colouring is probabilistic, we need to use error prob. reduction to lower the prob. of a k-1 length path not being colourful and not found. Consider the lecture/script to recap the details and proof)

Satz 3.3.

(1) Der Algorithmus hat eine Laufzeit von $O(\lambda(2e)^k km)$.

(2) Antwortet der Algorithmus mit JA, dann hat der Graph einen Pfad der Länge $k-1$.

(3) Hat der Graph einen Pfad der Länge $k-1$, dann ist die Wahrscheinlichkeit, dass der Algorithmus mit NEIN antwortet, höchstens $e^{-\lambda}$.

Long Paths

Satz 3.3.

(1) Der Algorithmus hat eine Laufzeit von $O(\lambda(2e)^k km)$.

(2) Antwortet der Algorithmus mit JA, dann hat der Graph einen Pfad der Länge $k-1$.

(3) Hat der Graph einen Pfad der Länge $k-1$, dann ist die Wahrscheinlichkeit, dass der Algorithmus mit NEIN antwortet, höchstens $e^{-\lambda}$.

For k in O(log n) this is polynomial in n 🎉 🎉 🎉