Long Path Problem

Gegeben: ein Graph G und eine Länge B

Gesucht: gibt es einen Pfad der Länge B ?

hat also B Kanten und B+1 verschiedene Knoten

B = 4 Kanten und B+1 = 5 Knoten

Schwierigkeit: Falls das Hamiltonkreisproblem NP-schwer ist, dann ist auch Long-Path NP-schwer

Beweis: mittels Kontraposition ($A \to B \equiv \neg B \to \neg A$)

Annahme: Long-Path ist in polynomieller Zeit Lösbar

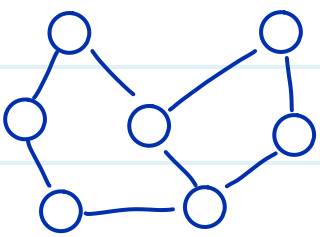
Zu zeigen: Hamiltonkreisproblem ist in polynomieller Zeit Lösbar

Sei G ein beliebiger Graph mit n Knoten

Wir konstruieren G' wie folgt:

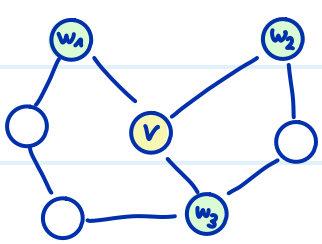
- Sei G' eine Kopie von G, ausser dass wir einen beliebigen Knoten v entfernen
- Dann erstelle für alle Kanten $\{v, w_i\}$ aus G einen neuen Knoten $\hat{w}_i$ und eine neue Kante $\{w_i, \hat{w}_i\}$ in G'
- G' hat also $\underbrace{(n-1)}_{\text{alle ausser } v} + \underbrace{\deg(v)}_{\text{für jede Kante von } v \text{ gibt es einen neuen Knoten } \hat{w}_i \text{ und } \deg(v) \leq n-1} \leq 2n-2$ viele Knoten

G:

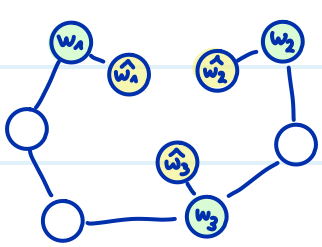


(G hat Hamiltonkreis)

wähle v

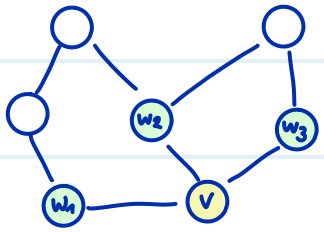


G':

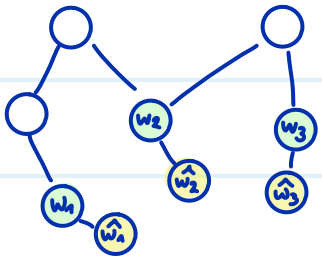


(G' hat Pfad der Länge n)

wähle v (andere Möglichkeit)



G':



(G' hat Pfad der Länge n)

Zu zeigen: G hat einen Hamiltonkreis <=> G' hat einen Pfad der Länge n

"=>" Annahme: $(v_1, v_2, .., v_n, v_1)$ ein Hamiltonkreis in G

Sei $v = v_1$ w.l.o.g (wir können den Kreis rotieren sodass $v = v_1$)

=> $(\hat{v}_2, v_2, v_3, .., v_n, \hat{v}_n)$ ist ein Pfad der Länge n in G'

v_2 war Nachbar von v, deshalb gibt es $\hat{v}_2$

v_n war Nachbar von v und, deshalb gibt es $\hat{v}_n$

"<=" Annahme: $(u_0, u_1, \dots, u_n)$ ist ein Pfad der Länge n in G'

=> die Knoten $u_1, .., u_{n-1}$ haben alle Grad ≥ 2 (weil sie innere Knoten im Pfad sind)

=> die Knoten $u_1, .., u_{n-1}$ sind genau die n-1 überlebenden Knoten aus G

(weil alle anderen Knoten $\hat{w}_i$ Grad 1 haben)

=> $u_0 = \hat{w}_i$ und $u_n = \hat{w}_j$ sind zwei der neuen Knoten in G' (einzige übrige Möglichkeit)

=> u_1 und u_{n-1} haben eine Kante zu v in G (weil sie eine Kante zu $\hat{w}_i$ / $\hat{w}_j$ in G' haben)

=> $(v, u_1, u_2, \dots, u_{n-1}, v)$ ist ein Hamiltonkreis in G

Schlussfolgerung: Falls Long-Path in polynomieller Zeit $O(t(n))$ entscheidbar ist, (t ist irgendeine Funktion)

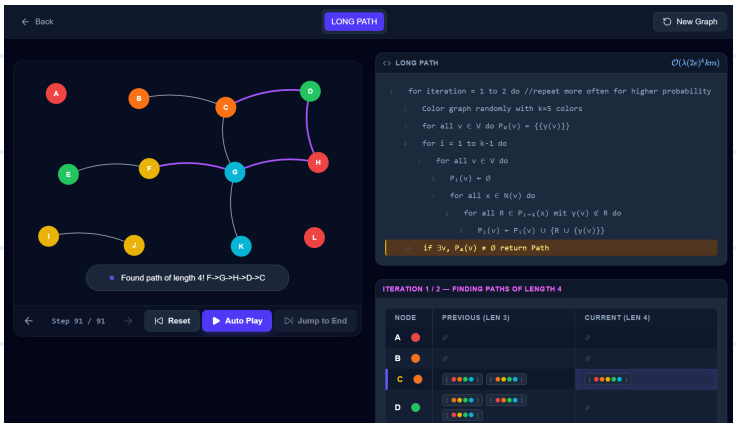
dann ist auch das Hamiltonkreisproblem in polynomieller Zeit $O(t(2n-2) + n^2)$ entscheidbar.

$t(2n-2)$: Long-Path auf G'

n^2: Konstruktion von G'

Monte Carlo Long Path

Gesucht: gibt es einen Pfad der Länge k-1 ? (also bestehend aus k Knoten)



1.) Färbe die Knoten mit einer zufällig gleichverteilten Färbung $\gamma: V \to \underbrace{[k]}_{k \text{ viele Farben}}$

- ein Pfad heisst bunt wenn alle Farben seiner Knoten paarweise verschieden sind
- Falls ein Pfad P der Länge k-1 existiert, dann existiert mit Wahrscheinlichkeit $\geq e^{-k}$ auch ein bunter Pfad der Länge k-1

Beweis: $\Pr[\exists \text{ bunter Pfad der Länge } k-1] \geq \Pr[\text{Pfad P ist bunt}]$

$= \frac{k!}{k^k}$ → Anzahl bunte Färbungen → ziehen ohne zurücklegen
→ Anzahl aller Färbungen → ziehen mit zurücklegen

$\geq e^{-k}$

weil $e^k = \sum_{n=0}^{\infty} \frac{k^n}{n!} = 1 + k + \frac{k^2}{2!} + \frac{k^3}{3!} + \dots$

$\Rightarrow e^k \geq \frac{k^k}{k!}$ alles weglassen ausser k-ten Term

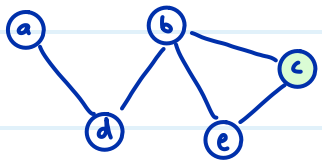
$\Rightarrow e^{-k} \leq \frac{k!}{k^k}$ beide Seiten hoch -1, Vergleichzeichen dreht sich um

2.) DP Algorithmus bestimmt (mit absoluter Sicherheit) ob es einen Pfad der Länge k-1 gibt

Teilproblem: $P_i(v) = \{ S \subseteq [k] \mid |S| = i+1$ und $\exists$ ein in v endender, genau mit den Farben aus S gefärbter bunter Pfad der Länge $i \}$

S ist eine Menge von $i+1$ Farben

Bsp



Farblegende: 1 = gelb, 2 = grün, 3 = rot, 4 = violett

Bestimme $P_1(c) = \{\{2,3\},\{2,1\}\}$

$P_2(a) = \{\{4,2,3\}\}$

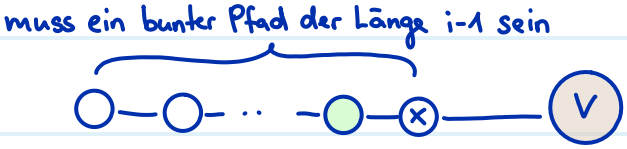
$P_2(b) = \{\{3,2,4\},\{3,2,1\}\}$

$P_0(e) = \{\{1\}\}$

Base Case: $P_0(v) = \{\{\gamma(v)\}\}$ für alle Knoten $v \in V$

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

- $x \in N(v)$: x muss ein Nachbar von v sein, damit wir den Pfad verlängern können
- $R \cup \{\gamma(v)\}$: Farbset des verlängerten Pfades
- $R \in P_{i-1}(x)$: $\exists$ ein bunter in x endender Pfad mit genau den Farben aus R der Länge $i-1$
- $\gamma(v) \notin R$: die Farbe von v wurde vorher noch nicht verwendet



$R = \{ \ , \ , .. , \ , \ \}$ und $\notin R$

$R \cup \{\gamma(v)\} = \{ \ , \ , .. , \ , \ , \ \}$

Lösung: $\exists$ bunter Pfad der Länge $k-1$ $\Leftrightarrow$ $\bigcup_{v \in V} P_{k-1}(v) \neq \emptyset$ $\left(\Leftrightarrow \exists v \in V: P_{k-1}(v) \neq \emptyset\right)$

Laufzeit: Base case in $O(n)$, danach wird Bunt(G, i) $k-1$ mal ausgeführt

BUNT(G, i) macht den nächsten Rekursionsschritt

```
1: for all v ∈ V do
2:     P_i(v) ← ∅
3:     for all x ∈ N(v) do          es gibt genau deg(v) viele Nachbarn
4:         for all R ∈ P_{i-1}(x) mit γ(v) ∉ R do
5:             P_i(v) ← P_i(v) ∪ {R ∪ {γ(v)}}        |R| = i viele Elemente überprüfen und kopieren
```

$$O(\text{Bunt}(G,i)) = O\left(\sum_{v \in V} \deg(v) \cdot |P_{i-1}(v)| \cdot i\right)$$

↳ $|P_{i-1}(v)| \leq \binom{k}{i}$ weil $P_{i-1}(v) \leq \binom{[k]}{i}$

$$\leq O\left(\sum_{v \in V} \deg(v) \cdot \binom{k}{i} \cdot i\right)$$

↳ Handschlag-Lemma $\sum_{v \in V} \deg(v) = 2m$

$$\leq O\left(2m \cdot \binom{k}{i} \cdot i\right)$$

↳ 2 weglassen

$$\leq O\left(\binom{k}{i} \cdot m \cdot i\right)$$

Total: $O\left(\sum_{i=1}^{k-1} \binom{k}{i} m \cdot i\right)$

↳ $i \leq k$

$$\leq O\left(\sum_{i=1}^{k-1} \binom{k}{i} m k\right)$$

↳ Summe von 0 bis k statt 1 bis k-1

$$\leq O\left(\sum_{i=0}^{k} \binom{k}{i} m k\right)$$

↳ $\sum_{i=0}^{k} \binom{k}{i} = 2^k$ folgt aus dem Binomialsatz, der aber für diesen Kurs nicht weiter relevant ist

$$= O(2^k m k)$$

Bisher: Monte-Carlo Algorithmus mit einseitigem Fehler

- Immer korrekt für NEIN-Instanzen:

$$\Pr[\text{kein Pfad der Länge } k-1 \text{ wird gefunden} \mid \nexists \text{ Pfad der Länge } k-1] = 1$$

- Mit Wahrscheinlichkeit $\geq e^{-k}$ korrekt für JA-Instanzen:

$$\Pr[\text{Pfad der Länge } k-1 \text{ wird gefunden} \mid \exists \text{ Pfad der Länge } k-1] \geq e^{-k}$$

3.) Fehlerreduktion: Sei $\delta = e^{-\lambda}$ unser Zielfehler.

$\varepsilon = e^{-k}$ ist der aktuelle Fehler

Anzahl Wiederholungen: $N = \lceil \varepsilon^{-1} \ln(\delta^{-1}) \rceil = \lceil e^{k} \ln(e^{\lambda}) \rceil = \lceil \lambda e^{k} \rceil$

(ε und δ einsetzen; $\ln(e^{\lambda}) = \lambda$)

Laufzeit insgesamt $O(\underbrace{2^k mk}_{\text{DP}} \cdot \underbrace{\lambda e^k}_{\text{Wiederholungen}}) = O(\lambda (2e)^k km)$

Für $k \leq O(\log n)$ ist die Laufzeit polynomiell:

$$O(\lambda (2e)^{\log(n)} \log(n) m) = O(\lambda n^{\log(2e)} \log(n) m)$$

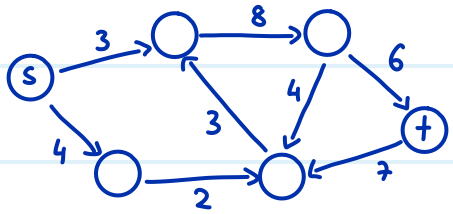
weil $a^{\log b} = b^{\log a}$

$\log(2e)$ ist konstant

Flows

Definition Netzwerk: $N = (V, A, c, s, t)$ wobei

- $G = (V, A)$ ist ein gerichteter Graph (A für Arcs)
- $s \in V$ ist die source/Quelle
- $t \in V$ ist das target/Senke
- $c : A \to \mathbb{R}_0^+$ ist die Kapazitätsfunktion, die jeder Kante $e \in A$ eine Kapazität $c(e)$ zuweist

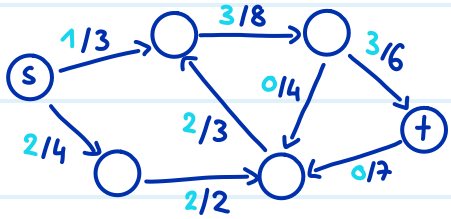


Definition Fluss: eine Funktion $f : A \to \mathbb{R}_0^+$, die jeder Kante $e \in A$ einen Flusswert $f(e)$ zuweist mit folgenden Bedingungen:

Zulässigkeit: $\forall e \in A : 0 \leq f(e) \leq c(e)$

Flusserhaltung: $\forall v \in V \setminus \{s,t\} :$ $\sum_{\substack{u \in V \\ (u,v) \in A}} f(u,v) = \sum_{\substack{u \in V \\ (v,u) \in A}} f(v,u)$

alles was rein fliesst = alles was raus fliesst



hat Flusswert = 3

ganzzahliger Fluss: falls $\forall e \in A: f(e) \in \mathbb{Z}$

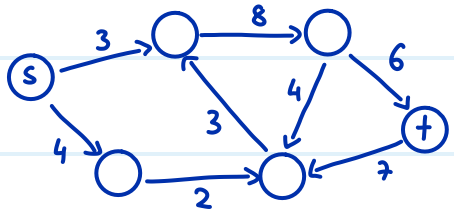
Wert des Flusses: $val(f) = netoutflow(s) = \sum_{\substack{u \in V \\ (s,u) \in A}} f(s,u) - \sum_{\substack{u \in V \\ (u,s) \in A}} f(u,s)$

was aus s raus fliesst − was in s hinein fliesst

Lemma Skript $= netinflow(t) = \sum_{\substack{u \in V \\ (u,t) \in A}} f(u,t) - \sum_{\substack{u \in V \\ (t,u) \in A}} f(t,u)$

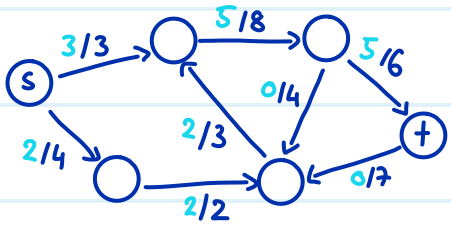
was in t hinein fliesst − was aus t raus fliesst

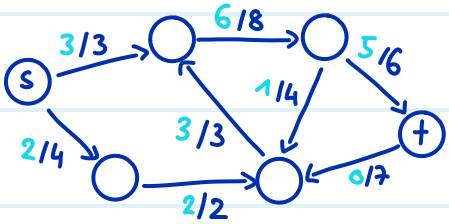
Aufgabe: bestimme den maximalen Flusswert für folgendes Netzwerk:

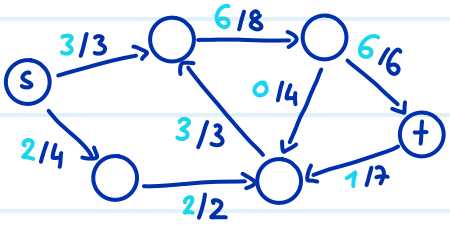


MaxFlow = 5

Finde drei verschiedene Flüsse in diesem Netzwerk, die alle den maximalen Wert haben







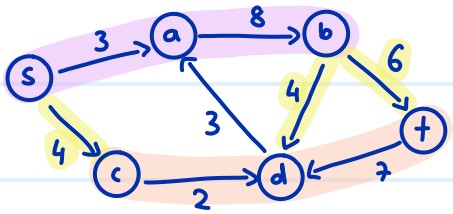
s-t-Schnitt: eine Partition $(S, \bar{T})$ von V sodass $s \in S$ und $t \in \bar{T}$

(das heisst $S \cup \bar{T} = V$ und $S \cap \bar{T} = \emptyset$)

Kapazität eines s-t-Schnitts: $cap(S, \bar{T}) = \sum_{(u,v) \in (S \times \bar{T}) \cap A} c(u,v)$

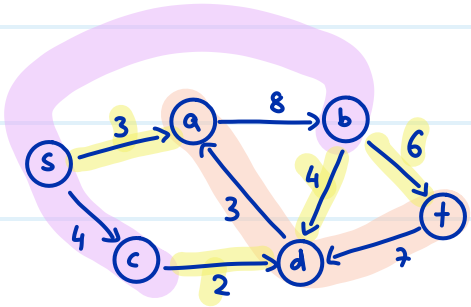
(alle Kanten die von S zu $\bar{T}$ gehen)

Bsp



Sei $S = \{s, a, b\}$ und $\bar{T} = \{t, c, d\}$

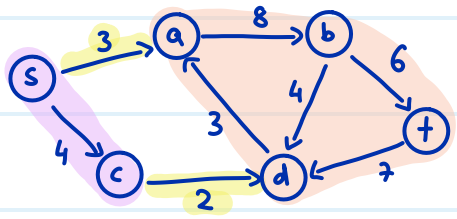
dann ist $cap(S, \bar{T}) = 4 + 4 + 6 = 14$



Sei $S = \{s, b, c\}$ und $\bar{T} = \{t, a, d\}$

Berechne $cap(S, \bar{T}) = 3 + 2 + 4 + 6 = 15$

Finde einen s-t-Schnitt mit $cap(S, \bar{T}) = 5$



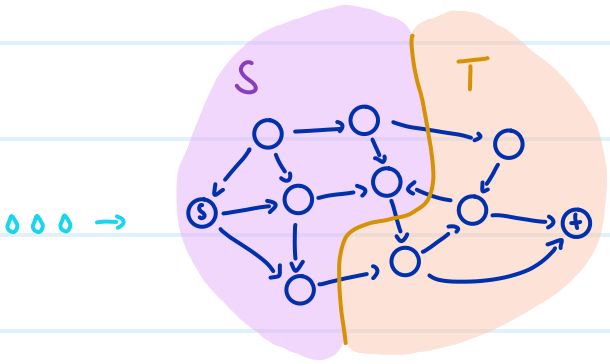
Sei $S = \{s, c\}$ und $\bar{T} = \{t, a, b, d\}$

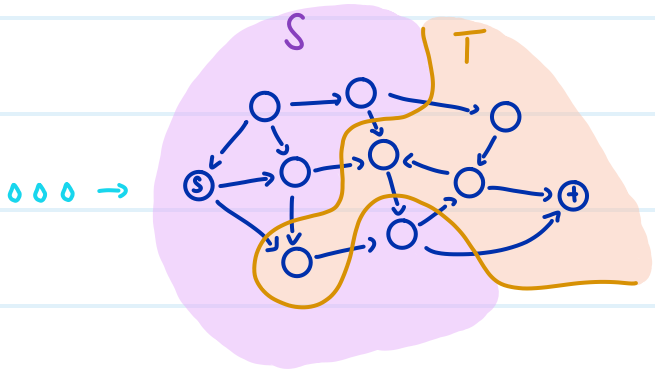
dann ist $cap(S, \bar{T}) = 3 + 2 = 5$

Lemma 3.8. Für einen beliebigen Fluss und einen beliebigen s-t-Schnitt $(S, \bar{T})$ gilt:

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

Intuition: Jeder einzelne "Tropfen" muss irgendwann die Grenzlinie von S nach T überqueren





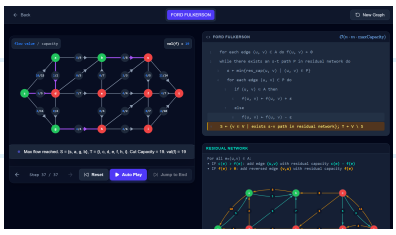
Maxflow-Mincut Theorem: $\max_{\text{Fluss } f} \text{val}(f) = \min_{(S,T)\ s\text{-}t\text{-Schnitt}} \text{cap}(S,T)$

=> um zu beweisen, dass ein Fluss f maximal ist, reicht es, einen s-t-Schnitt zu finden sodass $\text{val}(f) = \text{cap}(S,T)$

Beweis mittels Ford-Fulkerson Algorithmus:

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

1: $f \leftarrow 0$ ▷ 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



- Laufzeit $O(nmU)$ falls alle Kapazitäten ganzzahlig sind und U die maximale Kapazität ist (ähnliche Idee wie bei Hopcroft-Karp)

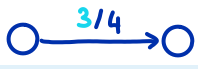
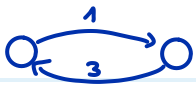
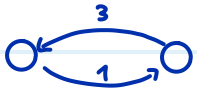
Restnetzwerk: Sei $N = (V, A, c, s, t)$ ein Netzwerk und f ein Fluss.

Dann ist $N_f = (V, A_f, r_f, s, t)$ das Restnetzwerk, wobei

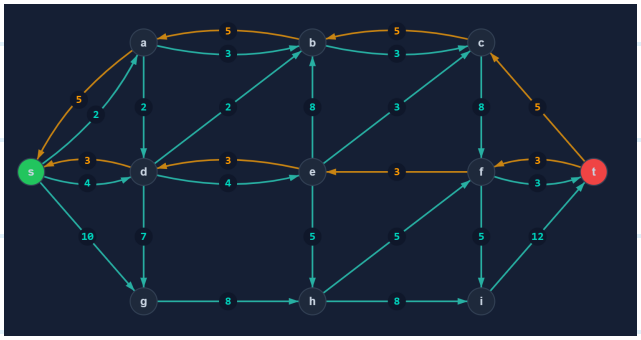
- für jede noch nicht vollständig ausgelastete (d h $f(e) < c(e)$) Kante $e \in A$ fügen wir die Kante e zu A_f hinzu mit der noch übrigen Restkapazität $r_f(e) := c(e) - f(e)$
- für jede Kante $e \in A$ auf der etwas fliesst (d h $f(e) > 0$) fügen wir eine umgekehrte Kante e^{opp} zu A_f hinzu mit dem fliessenden Wert $r_f(e^{opp}) := f(e)$
- Wir nehmen an, dass es in N keine einander entgegen gerichtete Kanten gibt

Intuitiv sagt uns das Restnetzwerk, wie wir den aktuellen Fluss verändern können

Aufgabe:

Kante in N	Kante(n) in N_f
	
	
	
	
	

Augmentierende Pfade: gerichtete Pfade von s nach t im Restnetzwerk



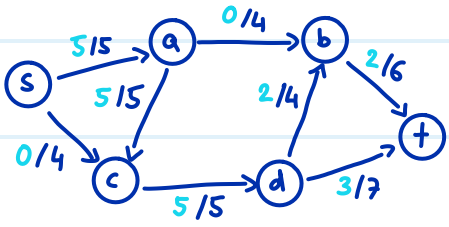
$\delta :=$ minimale Restkapazität einer Kante des augmentierenden Pfades

Für eine Kante e des augmentierenden Pfades P ändern wir f wie folgt.

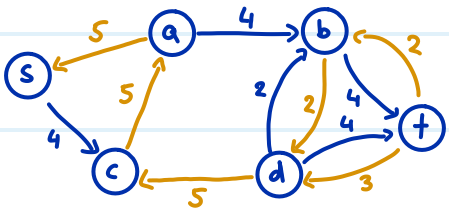
$$f(e) = \begin{cases} f(e) + \delta & \text{falls die Kante in P in dieselbe Richtung zeigt wie in N} \\ f(e) - \delta & \text{sonst} \end{cases}$$

Pfad in N_f	
Veränderung von f in N	

Aufgabe: Gegeben sind das folgende Netzwerk und der Fluss:



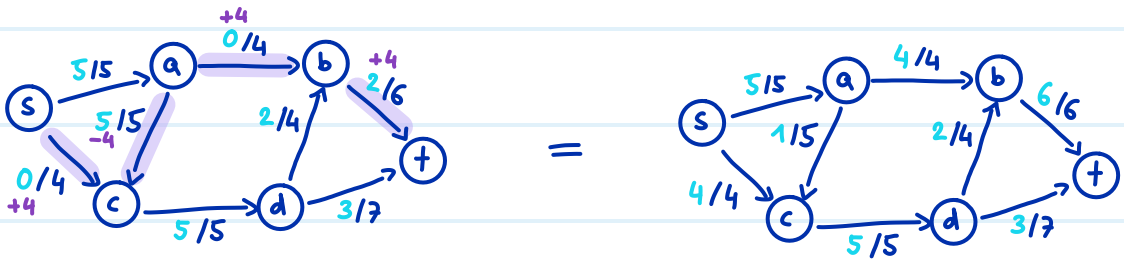
1) Zeichne das Restnetzwerk ein.



2) Finde einen augmentierenden Pfad: (s, c, a, b, t)

3) Bestimme die minimale Restkapazität einer Kante des augmentierenden Pfades: $\delta = 4$

4) Augmentiere den Fluss mit diesem Pfad:



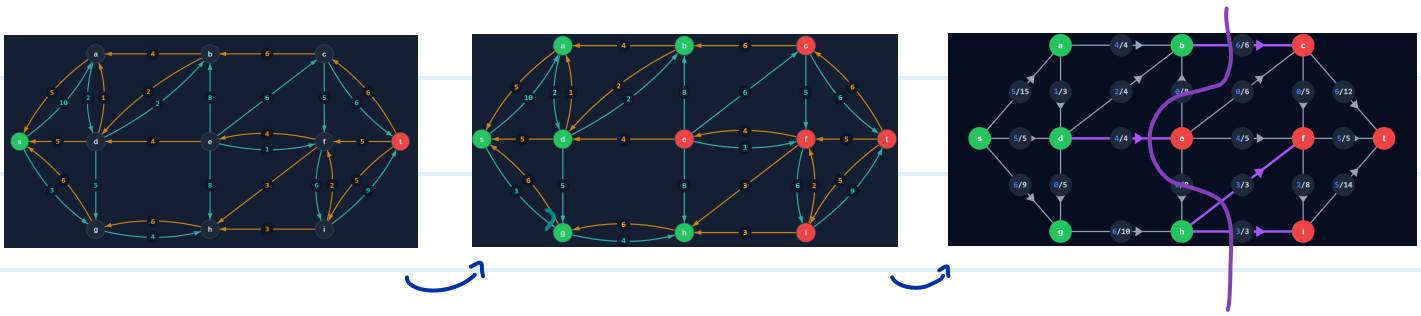
Satz 3.11: f ist maximal <=> es gibt keinen gerichteten s-t-Pfad in N_f

(analog zu Satz von Berge)

Wenn f maximal ist, dann gilt $cap(S,T) = val(f)$

für $S = \{v \in V \mid v \text{ ist von } s \text{ aus erreichbar in } N_f\}$

und $T = V \setminus S$



Anwendungen: Verkehr, Strom-/ Wasserversorgung, Internet, Truppenverschiebung, ... (sehr viele)

Für die Prüfung: höchstwahrscheinlich keine Netzwerk-Beweise

höchstwahrscheinlich eine Code Expert Modellierungsaufgabe

Aufgaben:

N Familien nehmen an einem Abendessen teil. Die i-te Familie besteht aus m_i Mitgliedern. Es stehen T Tische zur Verfügung, wobei der j-te Tisch genau s_j viele Stühle hat. Um den sozialen Austausch zu fördern, dürfen keine zwei Mitglieder derselben Familie am selben Tisch sitzen.
Erstelle ein Netzwerk, mit dem sich bestimmen lässt, ob alle Familienmitglieder unter Einhaltung dieser Regel untergebracht werden können.

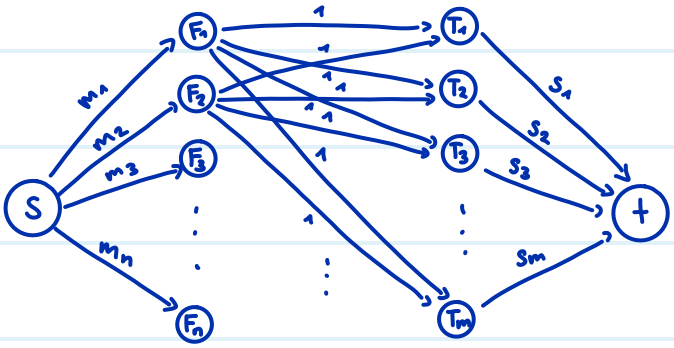
Sei $N = (V, A, c, s, t)$ ein Netzwerk, wobei

- $V = \{ s, t, \underbrace{F_1, F_2, \dots, F_n}_{\text{ein Knoten pro Familie}}, \underbrace{T_1, T_2, \dots, T_m}_{\text{ein Knoten pro Tisch}} \}$
- A hat folgende Kanten

  (s, F_i) mit Kapazität m_i $\quad \forall 1 \le i \le n$

  (F_i, T_j) mit Kapazität 1 $\quad \forall 1 \le i \le n \; \forall 1 \le j \le m$

  (T_j, t) mit Kapazität s_j $\quad \forall 1 \le j \le m$



Zuteilung möglich $\Leftrightarrow \max_{\text{Fluss } f} \text{val}(f) = \underbrace{\sum_{i=1}^{n} m_i}_{\text{alle Menschen}}$

In einer Fabrik müssen n Aufträge bis in F Tagen abgeschlossen werden. Jeder Auftrag benötigt genau einen Arbeitstag. Der i-te Auftrag kann jedoch nur zwischen einem Starttag s_i und einem Endtag e_i ausgeführt werden. Ihnen stehen m Maschinen zur Verfügung, und jede Maschine kann höchstens einen Auftrag pro Tag bearbeiten. Erstelle ein Netzwerk, um zu prüfen, ob alle n Aufträge bis zur Frist abgeschlossen werden können.

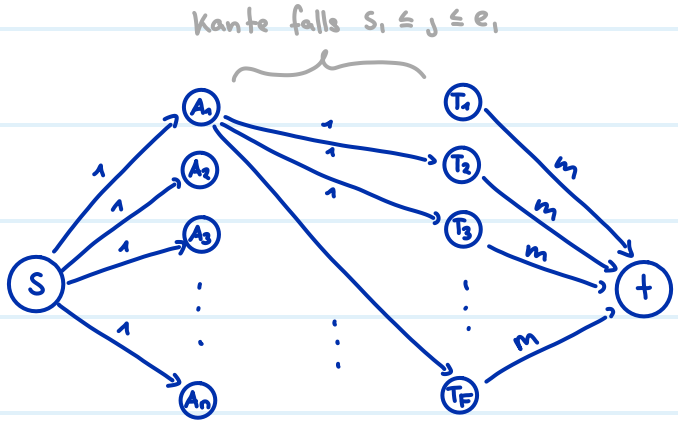
Sei $N = (V, A, c, s, t)$ ein Netzwerk, wobei

- $V = \{ s, t, \underbrace{A_1, A_2, \dots, A_n}_{\text{ein Knoten pro Auftrag}}, \underbrace{T_1, T_2, \dots, T_F}_{\text{ein Knoten pro Tag}} \}$
- A hat folgende Kanten

  (s, A_i) mit Kapazität 1 $\quad \forall 1 \leq i \leq n$

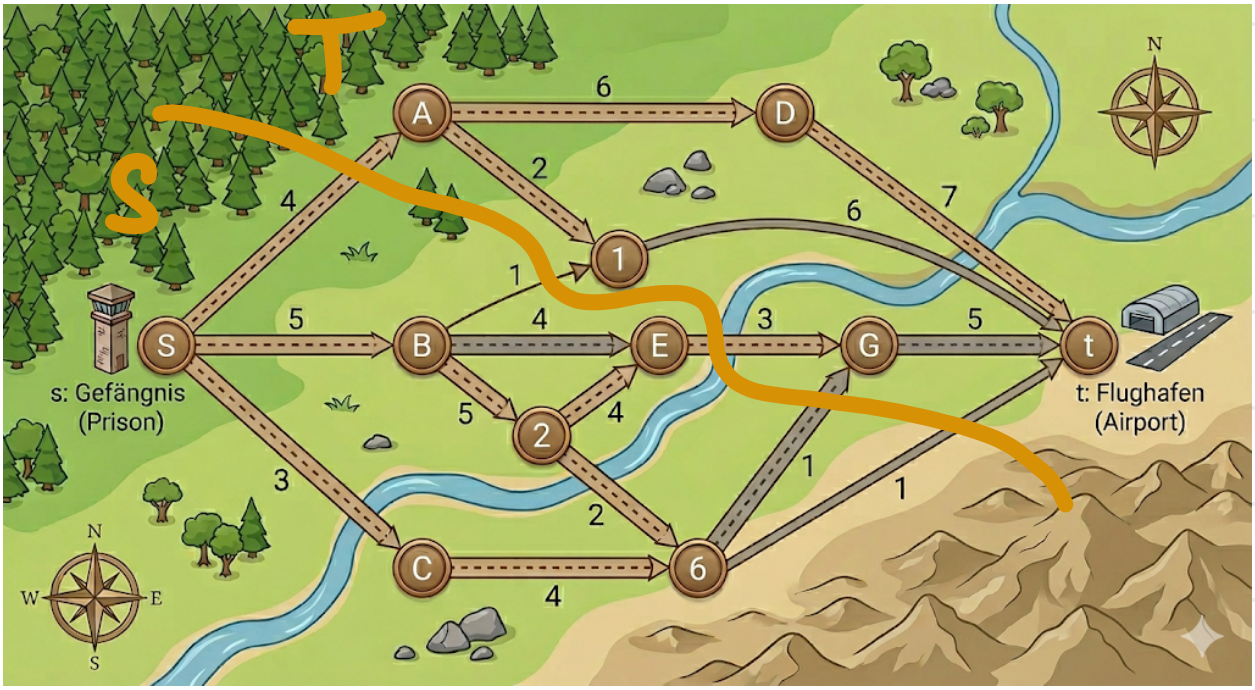
  (A_i, T_j) mit Kapazität 1 $\quad \forall 1 \leq i \leq n \; \forall 1 \leq j \leq F$ mit $s_i \leq j \leq e_i$

  (T_j, t) mit Kapazität m $\quad \forall 1 \leq j \leq F$



Einhaltung der Frist ist möglich $\Leftrightarrow \max_{\text{Fluss } f} \text{val}(f) = \underbrace{n}_{\text{alle Aufträge}}$

Ein Dieb bricht aus dem Gefängnis s aus, und möchte zum Flughafen t flüchten. Dabei benutzt er das Strassennetz. Die Polizei verfügt über 10 Polizisten, die sie auf Strassen platzieren können. Für jede Strasse ist angegeben, wie viele Polizisten benötigt werden, um diese zu blockieren.
Ist es möglich, mit 10 Polizisten sicherzustellen, dass der Dieb nicht zum Flughafen gelangt?



Ja es ist möglich, weil $\max_{\text{Fluss } f} \text{val}(f) = 10$ $\Rightarrow$ minimaler Schnitt = 10