

# Der Linearzeit MST Algorithmus

Der schnellste Algorithmus für das MST/ MSF Problem

---

Max Springenberg

Proseminar: Randomisierte Algorithmen, TU Dortmund

# Table of contents

1. MST in gewichteten Graphen
2. Bäume vs. Wälder
3. Borůvka Phasen
4. F-schwere/ -leichte Kanten
5. Randomisierte Stichproblem
6. Erkenntnis

# MST in gewichteten Graphen

---

# Definition MST

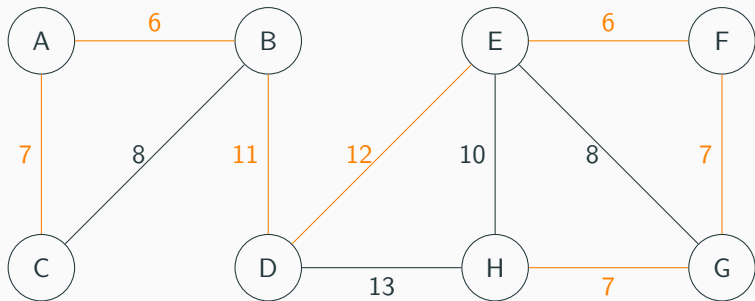
Ein Teilgraph  $T$  ist genau dann ein minimaler Spannbaum von  $G$ , wenn er ein Spannbaum in  $G$  ist und die Summe seiner Kantengewichte

$\sum_{e \in E_T} w(e)$  minimal ist.

# Definition MST

Ein Teilgraph  $T$  ist genau dann ein **minimaler Spannbaum** von  $G$ , wenn er ein Spannbaum in  $G$  ist und die Summe seiner Kantengewichte

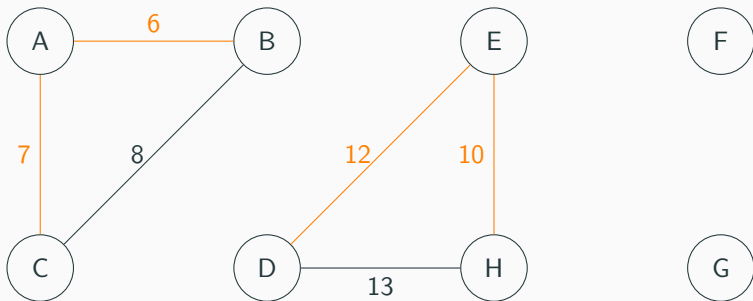
$\sum_{e \in E_T} w(e)$  **minimal** ist.



# Bäume vs. Wälder

---

Ist ein Graph nicht Zusammenhängend, so kann auch ein MSF den MST ersetzen. Der MSF ist ein Teilgraph aus disjunkten MSTs.



# Borůvka Phasen

---

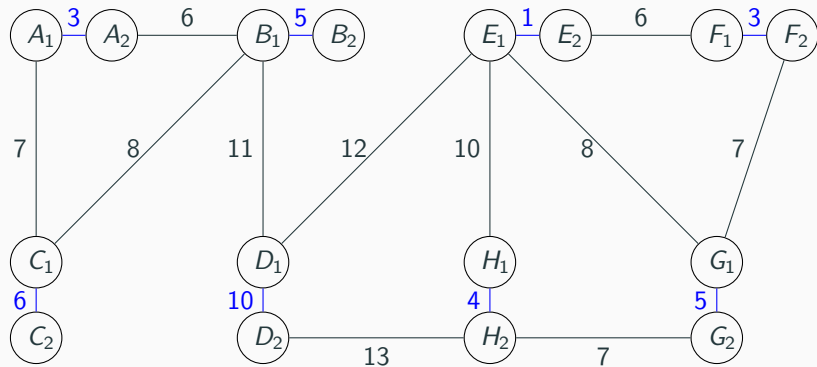


1. Kontraktierende Kanten markieren
2. Verbundene Komponente bestimmen
3. Verbundene Komponenten durch einzelnen Knoten ersetzen
4. Selbstschleifen entfernen

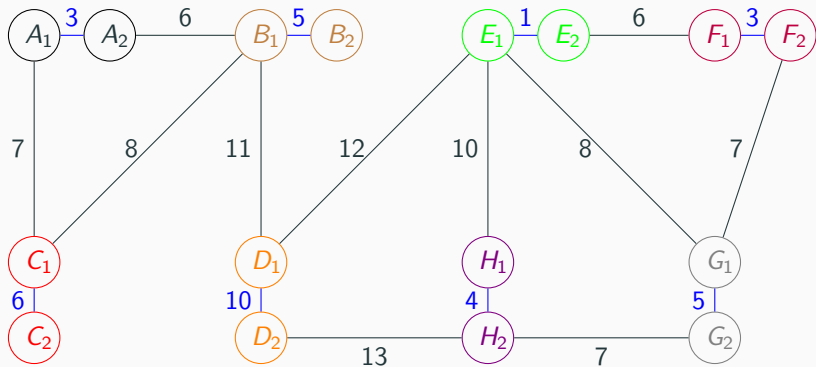
**Was bedeutet das für den reduzierten Graphen?**

⇒ Knoten werden auf maximal  $n/2$ ,  $n = |V|$  reduziert!

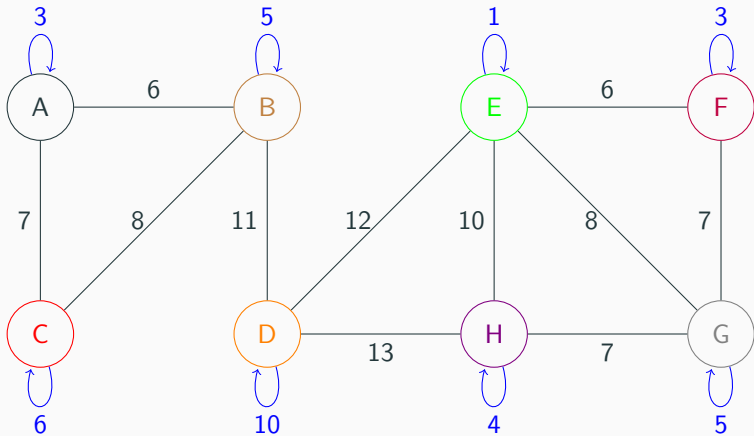
# 1. Kontraktierende Kanten markieren



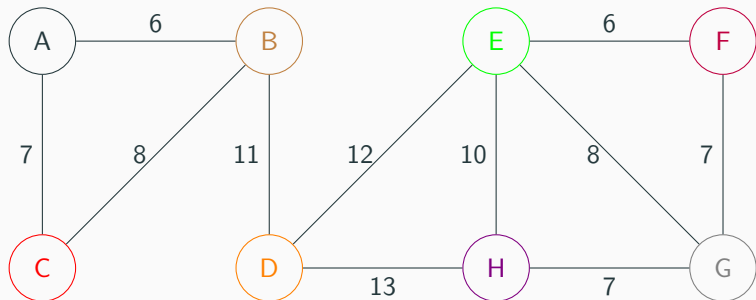
## 2. Verbundene Komponente bestimmen



### 3. Verbundene Komponenten durch einzelnen Knoten ersetzen



## 4. Selbstschleifen entfernen



**F-schwere/ -leichte Kanten**

---

# Definition

Sei  $P(e = \{u, v\})$  der Pfad, der die Knoten im MSF verbindet, in Kanten

Sei  $w : E \rightarrow \mathbb{R}$ , die Gewichtsfunktion von  $G$

Sei ferner definiert  $w(E) = \{w(e_1), \dots, w(e_m)\}$

Eine Kante ist F-schwer, wenn gilt:

$$w(e) > w_F(e)$$

, wobei:

$$w_F(e = (u, v)) = \begin{cases} \infty, & \text{u und v sind in verschiedenen Komponenten} \\ \max\{w(P(e))\}, & \text{sonst} \end{cases}$$

# Randomisierte Stichproblem

---



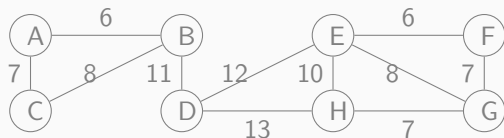
Wirf eine Münze!



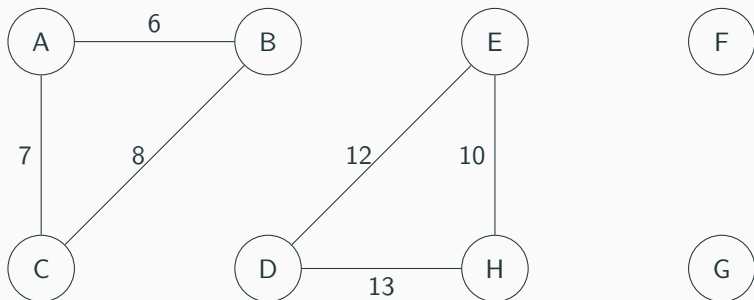
Quelle: <https://melbournechapter.net/explore/coin-flip-clipart/>

# Kanten 'würfeln'

G:

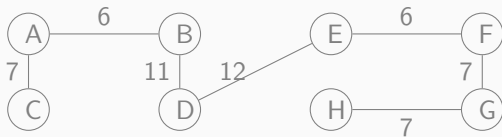


$G(p = 0,5)$  :

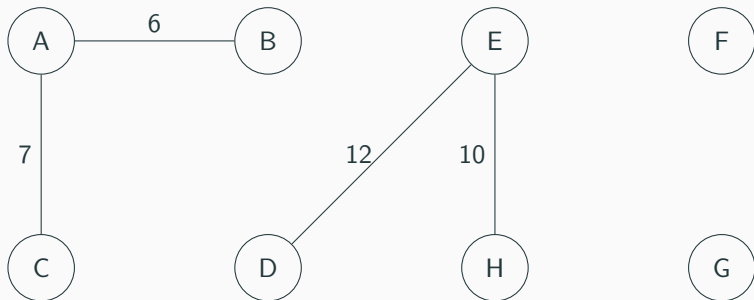


# MST vs. MSF

$MST_G$ :



$MSF_{G(0,5)}$ :

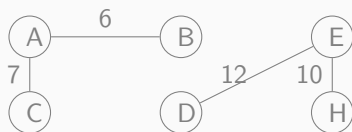


# Erkenntnis

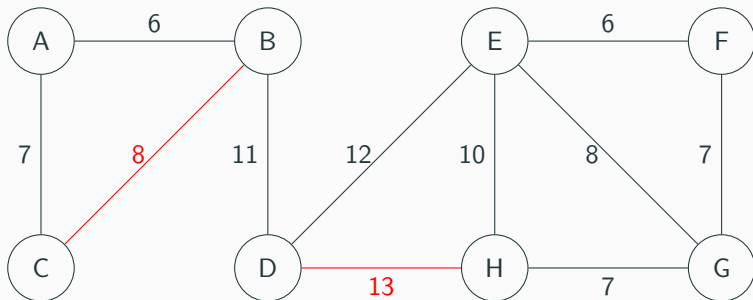
---

# Eleminierung von unnützen Kanten

$MSF_{G(0,5)}$ :



$G$ :



1. Nutze Borůvka-Phasen, um die Anzahl von Knoten zu reduzieren
2. Nutze Stickproben, um die Anzahl von Kanten zu reduzieren
3. Entferne alle F-schweren Kanten
4. **Rekursion**

- Wie Fassen wir die Erkenntnis geschickt in einem Algorithmus?
- Wie erhalten wir trotz rekursiven Aufrufen eine erwartete lineare Laufzeit?