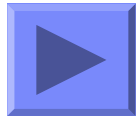


# Lokale Variante des Algorithmus von Dijkstra zur Berechnung kürzester Wege und das Problem der Aktualisierung von Wegen bei Leitungsausfällen

vorgetragen von Alex Prentki

# Agenda



Problemstellung



Grundlagen der Graphentheorie



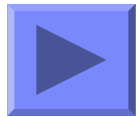
Der Dijkstra Algorithmus



Problem bei Leitungsausfällen



Lösungsansätze



Fazit

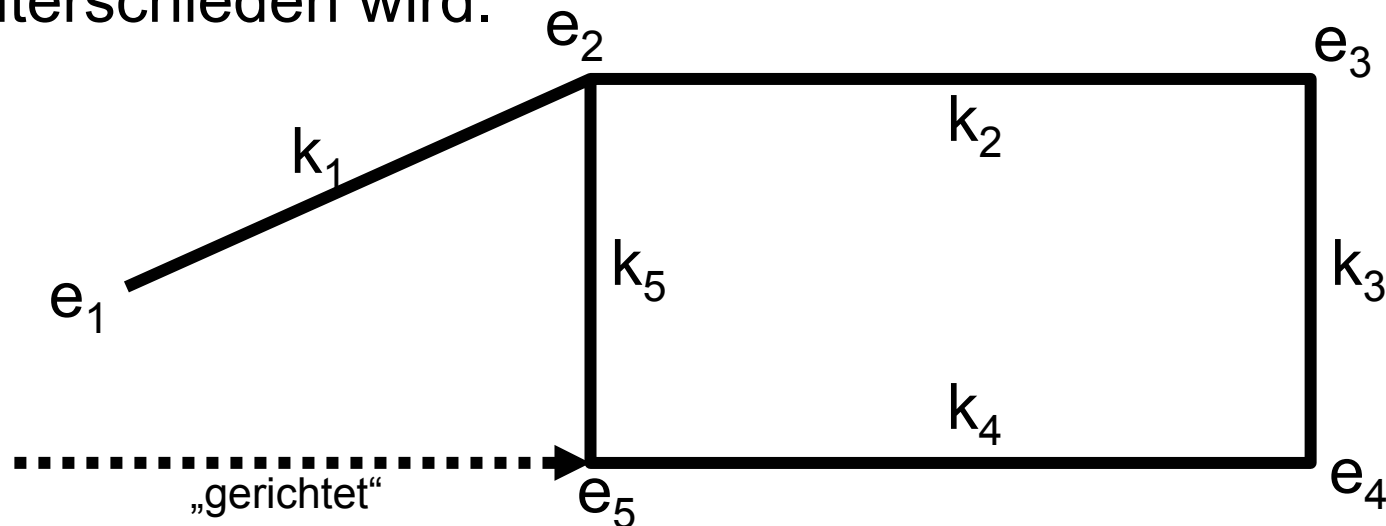
# Problemstellung

- **Finden kürzester Wege interessant in vielen Bereichen**
  - Routingverfahren (RIP & OSPF)
  - Logistik: Transportwege in einem Distributionssystem
  - KI: viele Probleme können als Suchprobleme dargestellt werden
  
- **Problem: Häufig sehr viele Knoten im Suchraum**
  - Beispiel Schach:  
25 Züge bei gegebener Stellung möglich  
5 eigene und 5 gegnerische Züge vorausplanen  
→ es müssen  $25^{13}$  Züge überschaut werden!

# Grundlagen der Graphentheorie

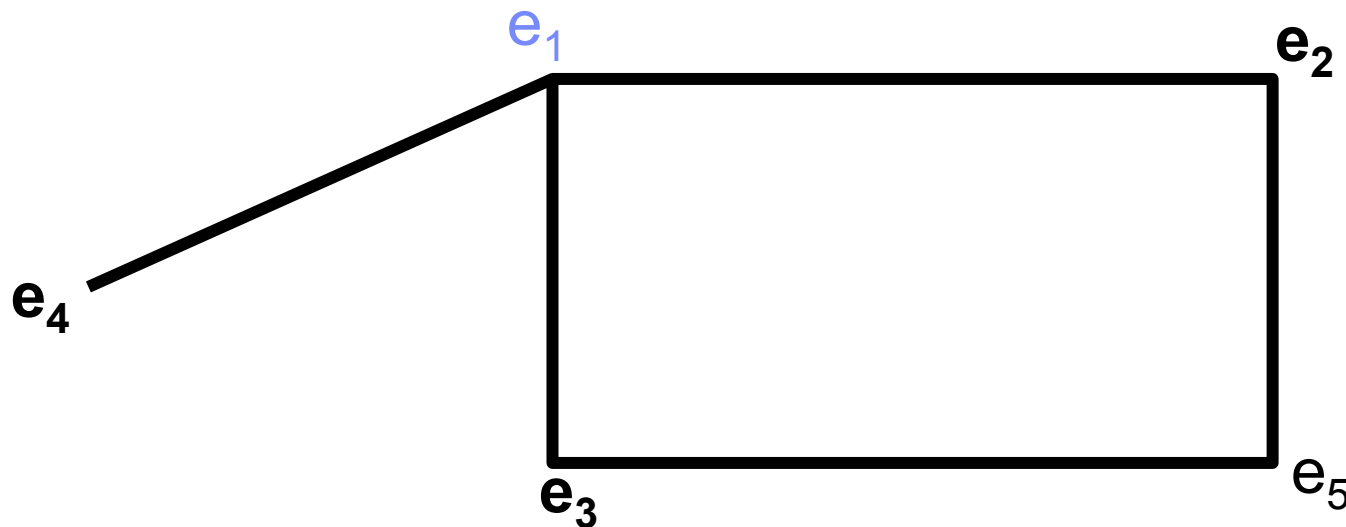
Ein **ungerichteter Graph** besteht aus einer Menge  $E$  von Ecken und einer Menge  $K$  von Kanten.

Die Ecken werden oft auch Knoten oder Punkte genannt. „Ungerichtet“ heißt es deswegen, weil zwischen Anfangs- und Endpunkt der Kante nicht unterschieden wird.



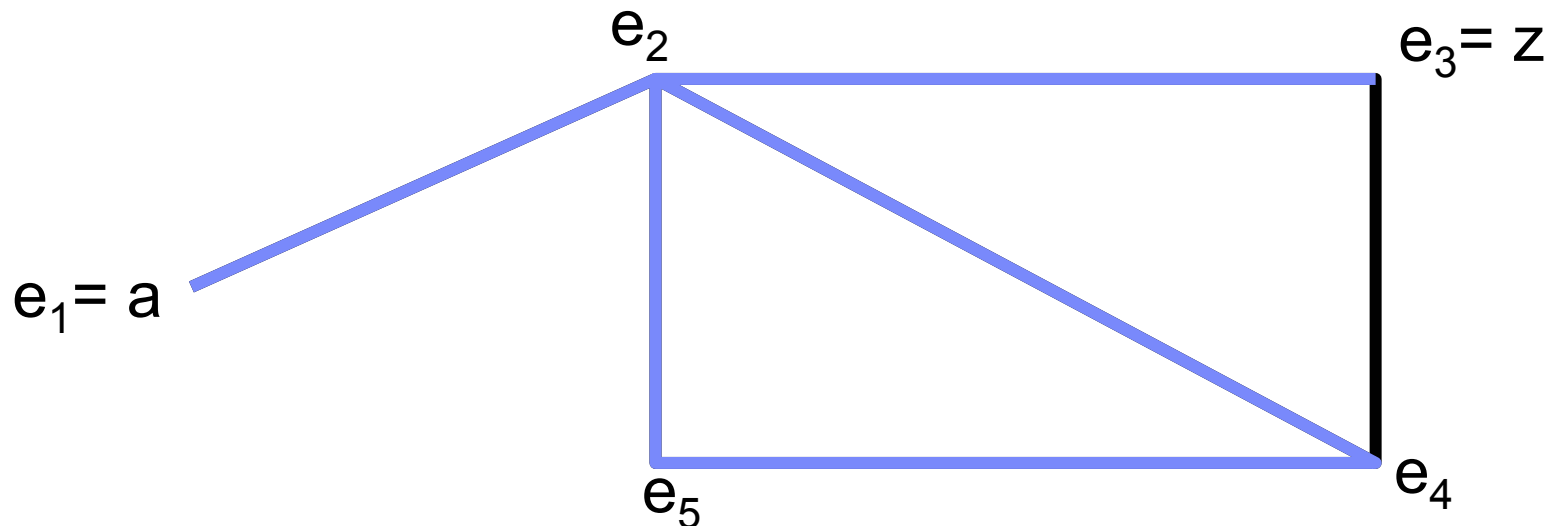
# Grundlagen der Graphentheorie

Von **Nachbarn** spricht man, wenn zwei Ecken  $e$  und  $f$  eines ungerichteten Graphen benachbart sind, d.h. wenn es eine Kante von  $e$  nach  $f$  gibt.



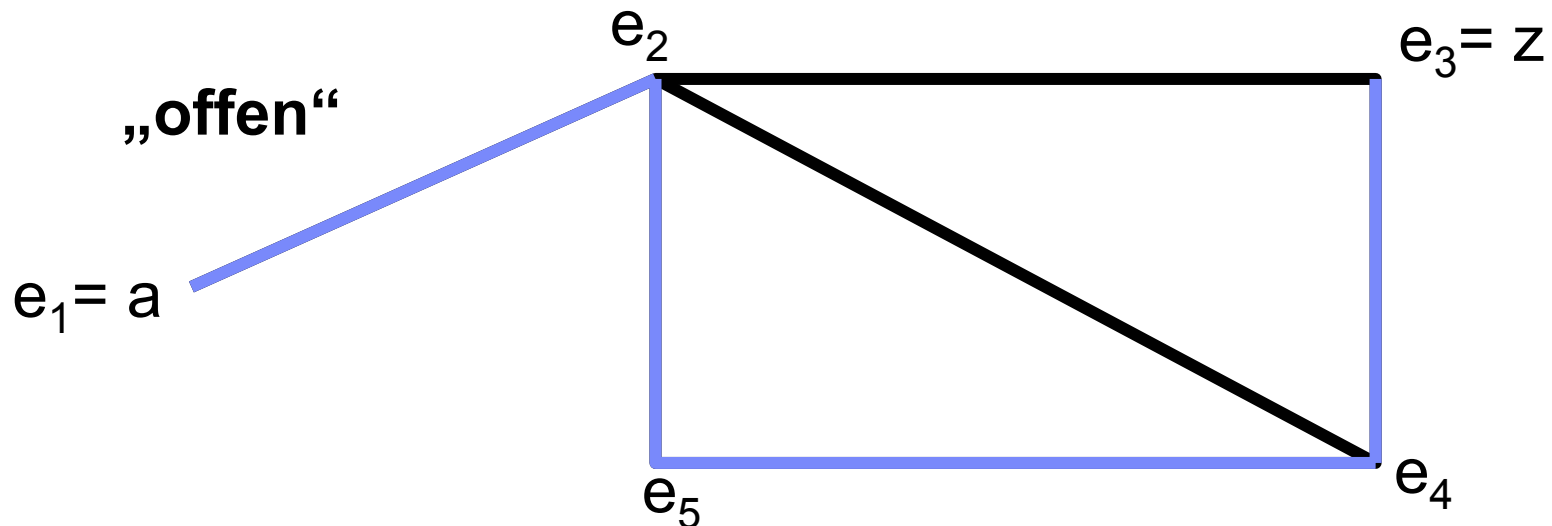
# Grundlagen der Graphentheorie

Ein **Pfad** von einem Anfangsknoten  $a$  zu einem Endknoten  $z$  ist eine Knotenfolge (oder auch Kantenfolge) von  $p$  Knoten  $e_1, e_2, \dots, e_p$  mit  $e_1 = a$ ,  $e_p = z$ . Die Knotenfolge kann einen Knoten mehrfach durchlaufen.  
(Wiederholungsmöglichkeit)



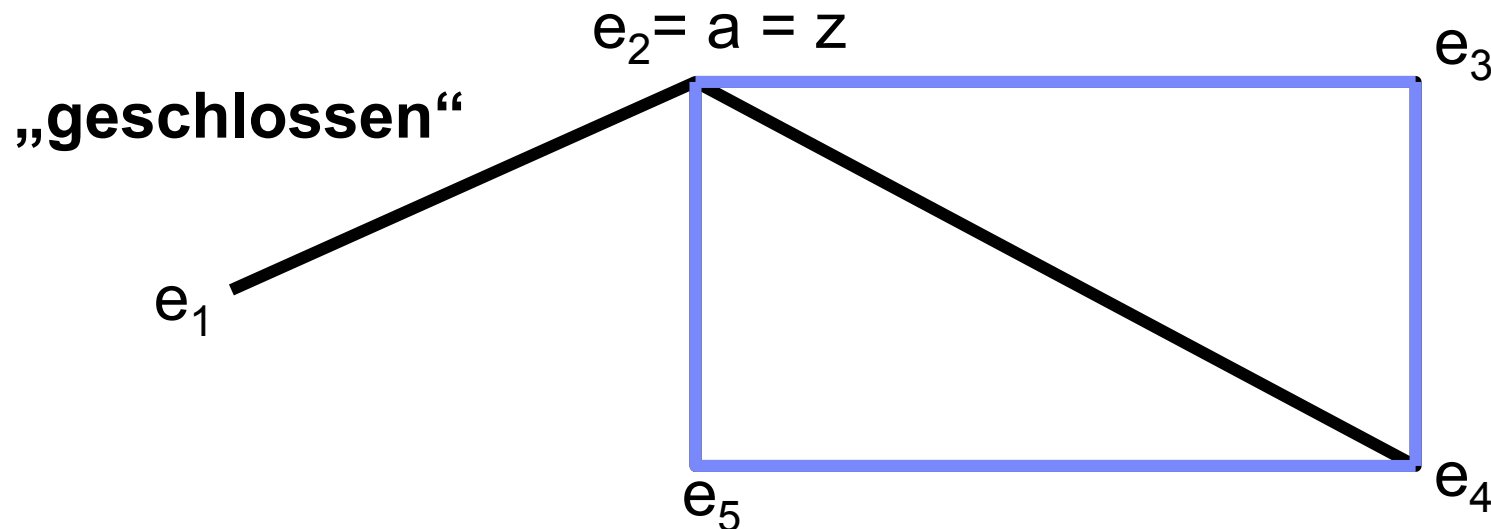
# Grundlagen der Graphentheorie

Ein **Weg** von einem Anfangsknoten  $a$  zu einem Endknoten  $z$  ist ein Pfad, bei dem alle verwendeten Kanten verschieden sind (ohne Wiederholungsmöglichkeit). Ist der Startknoten gleich dem Endknoten des Weges, so heißt der Weg **geschlossen**, andernfalls heißt er **offen**.



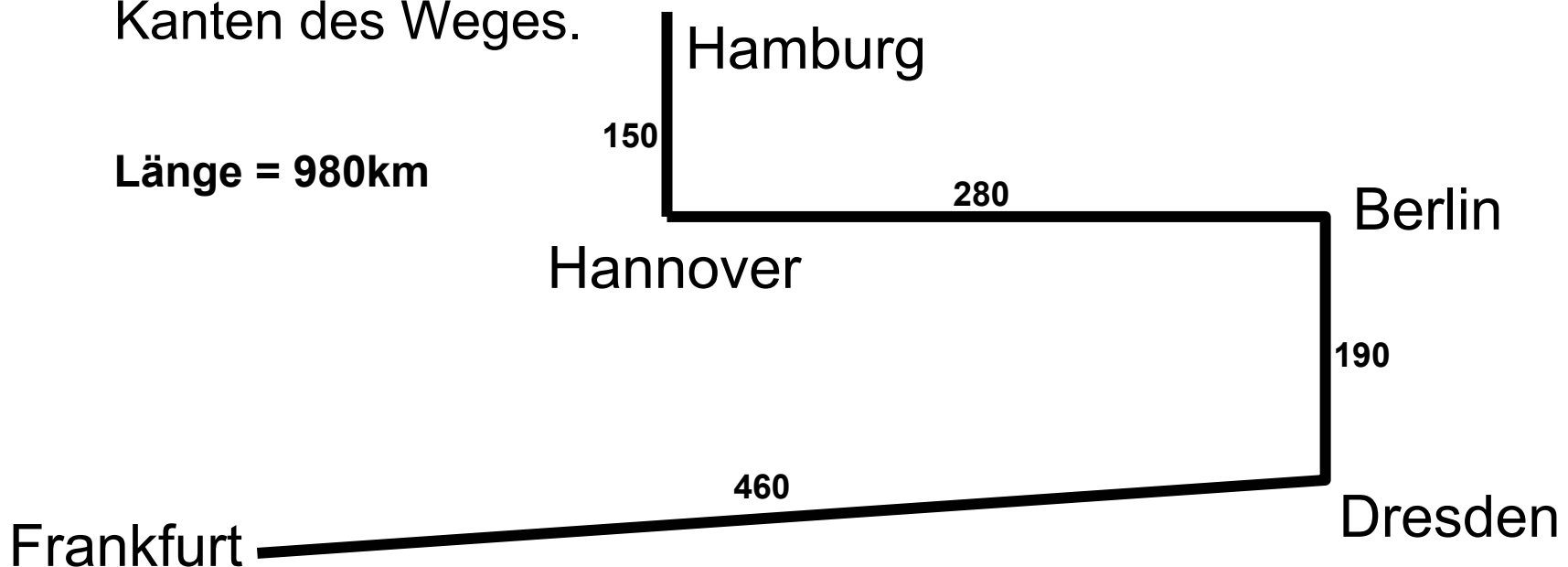
# Grundlagen der Graphentheorie

Ein **Weg** von einem Anfangsknoten  $a$  zu einem Endknoten  $z$  ist ein Pfad, bei dem alle verwendeten Kanten verschieden sind (ohne Wiederholungsmöglichkeit). Ist der Startknoten gleich dem Endknoten des Weges, so heißt der Weg **geschlossen**, andernfalls heißt er **offen**.



# Grundlagen der Graphentheorie

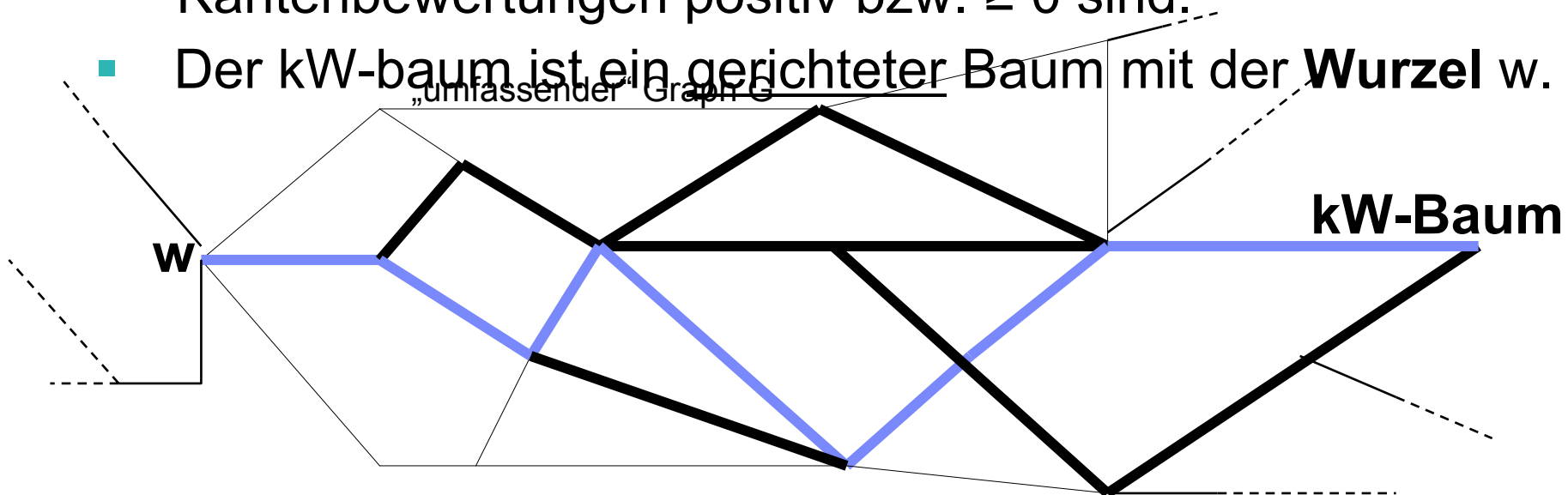
Bei **bewerteten Graphen** wird jeder Kante ein Wert aus der praktischen Anwendung zugewiesen. Die **Länge** eines Weges zwischen einem Startknoten  $s$  und einem Zielknoten  $z$  ist dann gegeben als die Summe der Bewertungen auf den Kanten des Weges.



# Grundlagen der Graphentheorie

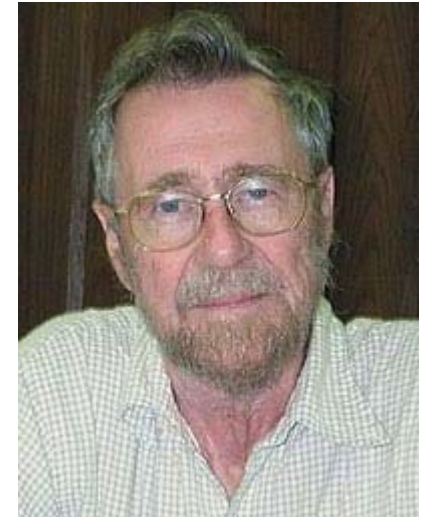
## kW-Baum

- Ein kürzeste-Wege-Baum ist immer ein Untergraph eines kantenbewerteten Graphen  $G$ , bei dem alle Kantenbewertungen positiv bzw.  $\geq 0$  sind.
- Der kW-baum ist ein gerichteter Baum mit der **Wurzel**  $w$ .



## Edsger Wybe Dijkstra (1930-2002)

- **Geboren in Rotterdam**
- **International angesehener Computer Wissenschaftler**
- **Lebte lange in Holland und arbeitete an der Eindhoven Universität für Technologie**
- **Anfang der 70er war er in der Forschungsabteilung der „Burroughs Corporation“ tätig**
- **1972 Turing Award**
- **1984 - 2000 Lehrstuhl an der University of Texas**



# Der Dijkstra Algorithmus

Basiert auf einer geschickten Auswahl des zu bearbeitenden Knotens.

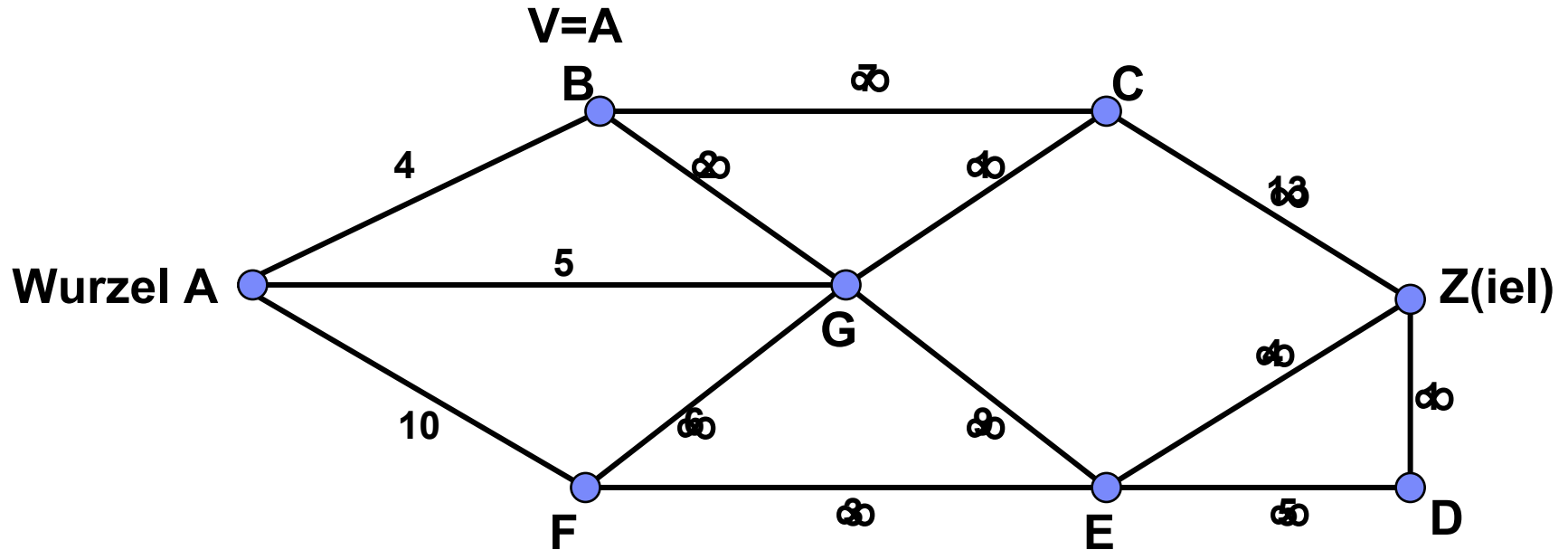
1. Definition und Initialisierung der **Distanz D**:  
Alle Knoten, die nicht unmittelbare Nachbarn der Wurzel sind, erhalten  $D = \infty$
2. Bildung von zwei Mengen  
**Menge B** enthält alle Knoten, für welche schon ein Weg von der Wurzel aus bekannt ist. Ein Knoten  $i$  ist in  $B$ , wenn  $D[i] \neq \infty$

**Menge E** enthält alle endgültig markierten Knoten.

## Der Dijkstra Algorithmus

3. Auswahl des Knotens mit der geringsten Distanz
  4. Vorgänger dieses Knotens merken
  5. Knoten aus B entfernen und in E einfügen
  6. Distanzen zu „neuen“ Nachbarn ermitteln
  7. Wiederholung ab Schritt 3 bis Zielknoten in E eingefügt oder alle Knoten bearbeitet wurden
- Jeder Knoten kennt nun den Vorgänger, über den er am schnellsten zu erreichen ist!

# Der Dijkstra Algorithmus - Anfangszustand

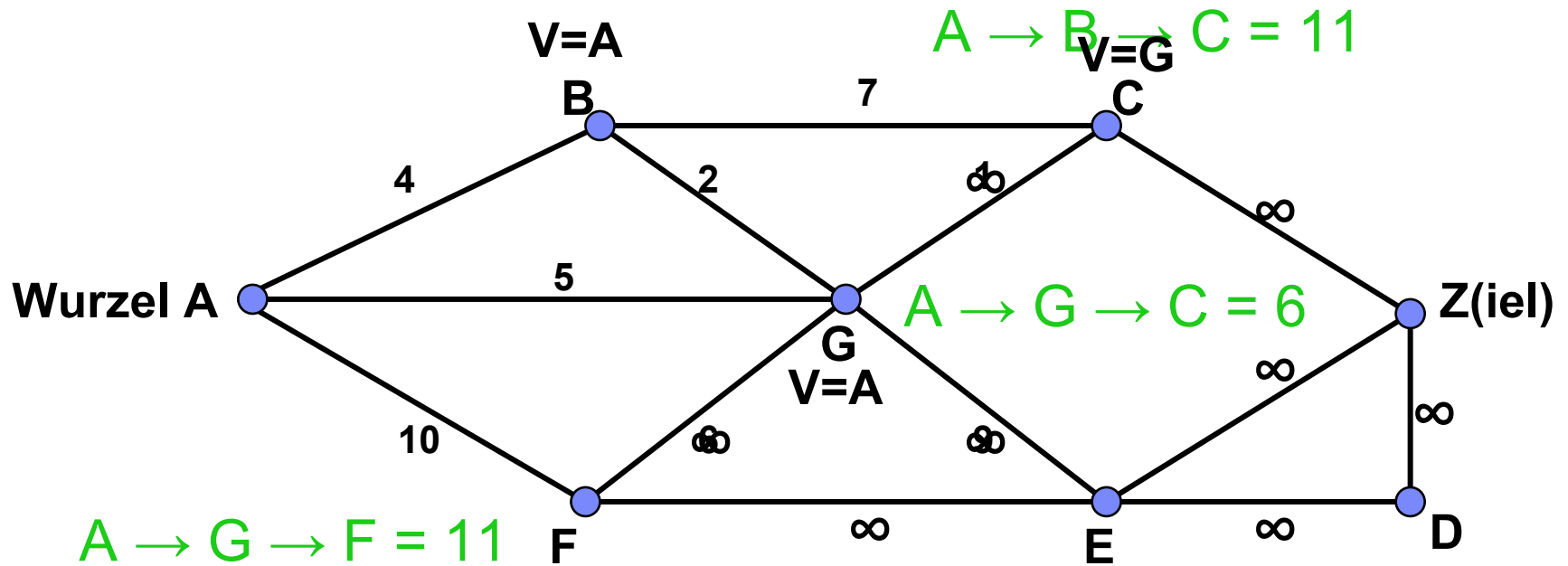


$$B = \{ \cancel{B}, G, F \}$$

$$E = \{ B \}$$



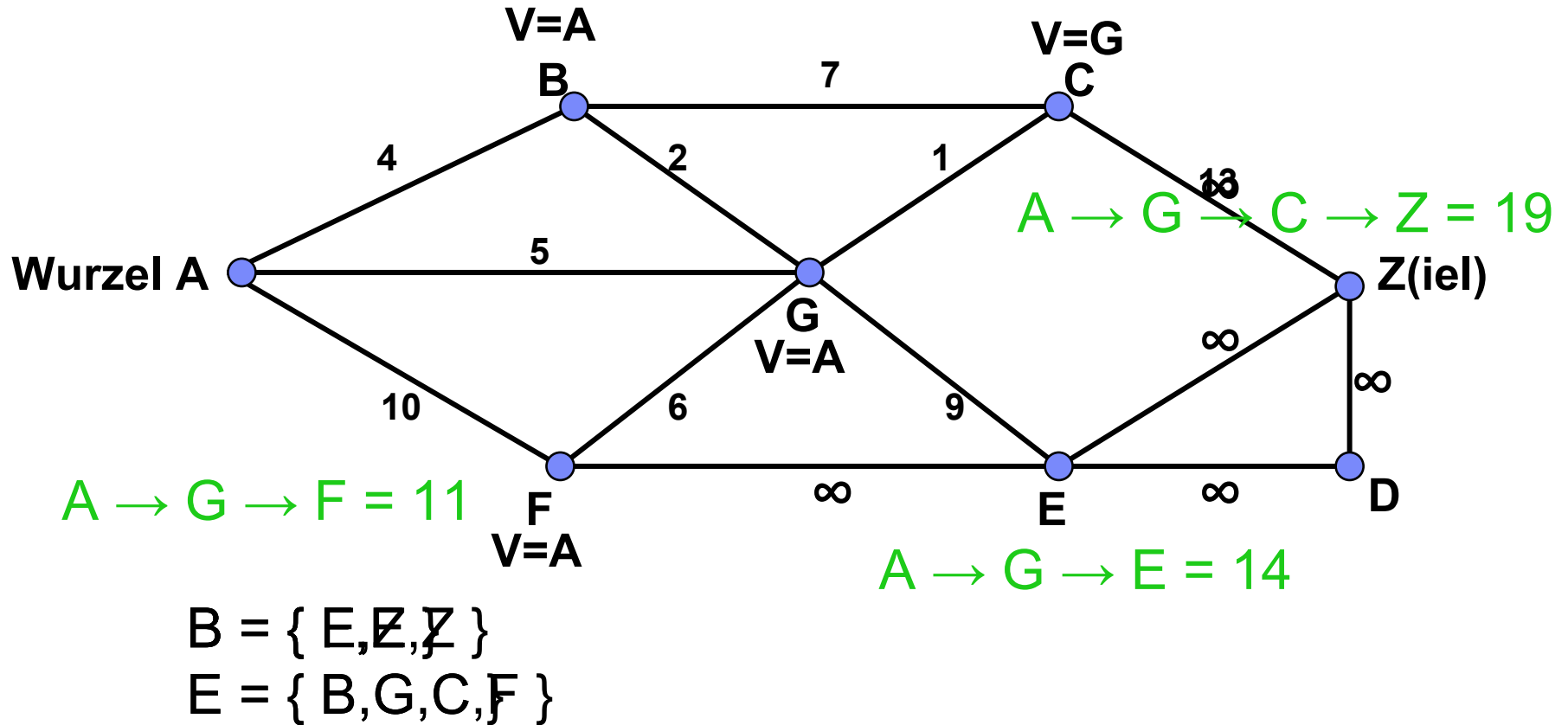
# Der Dijkstra Algorithmus – Schritt 3



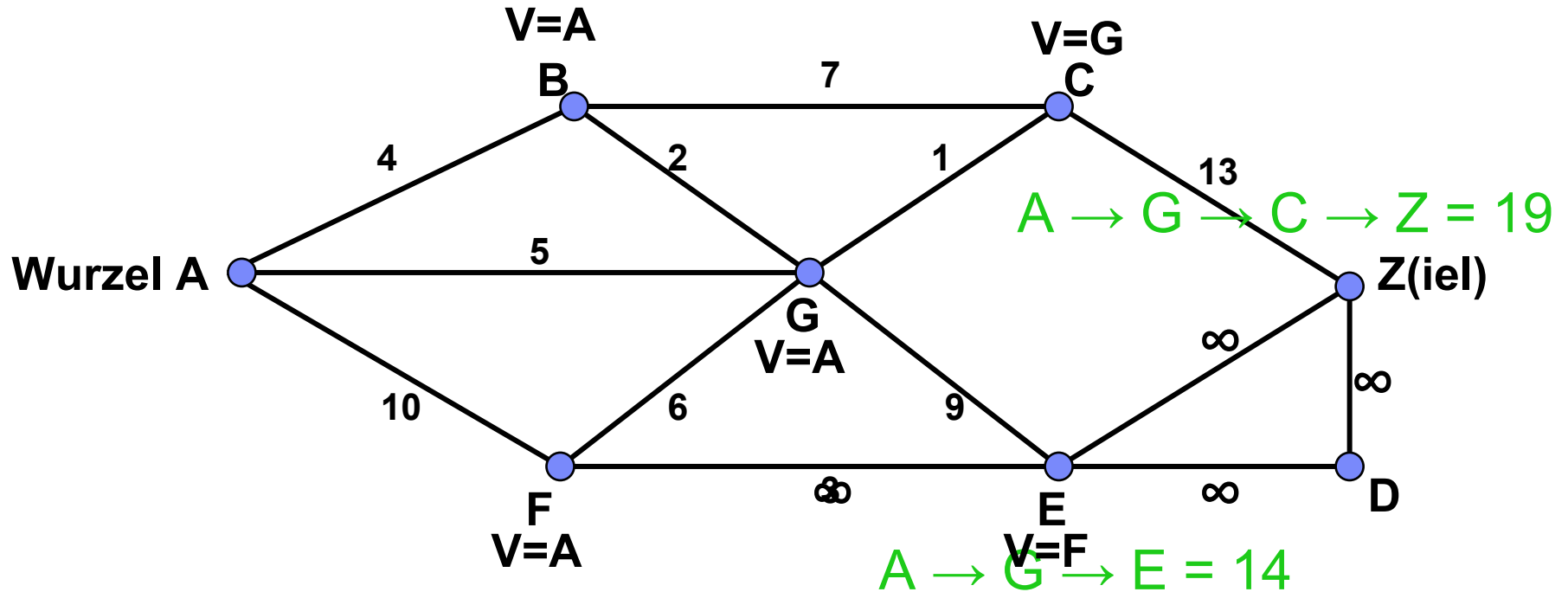
$$B = \{ \cancel{E}, \cancel{F} \}$$

$$E = \{ B, G, C \}$$

# Der Dijkstra Algorithmus – Schritt 4



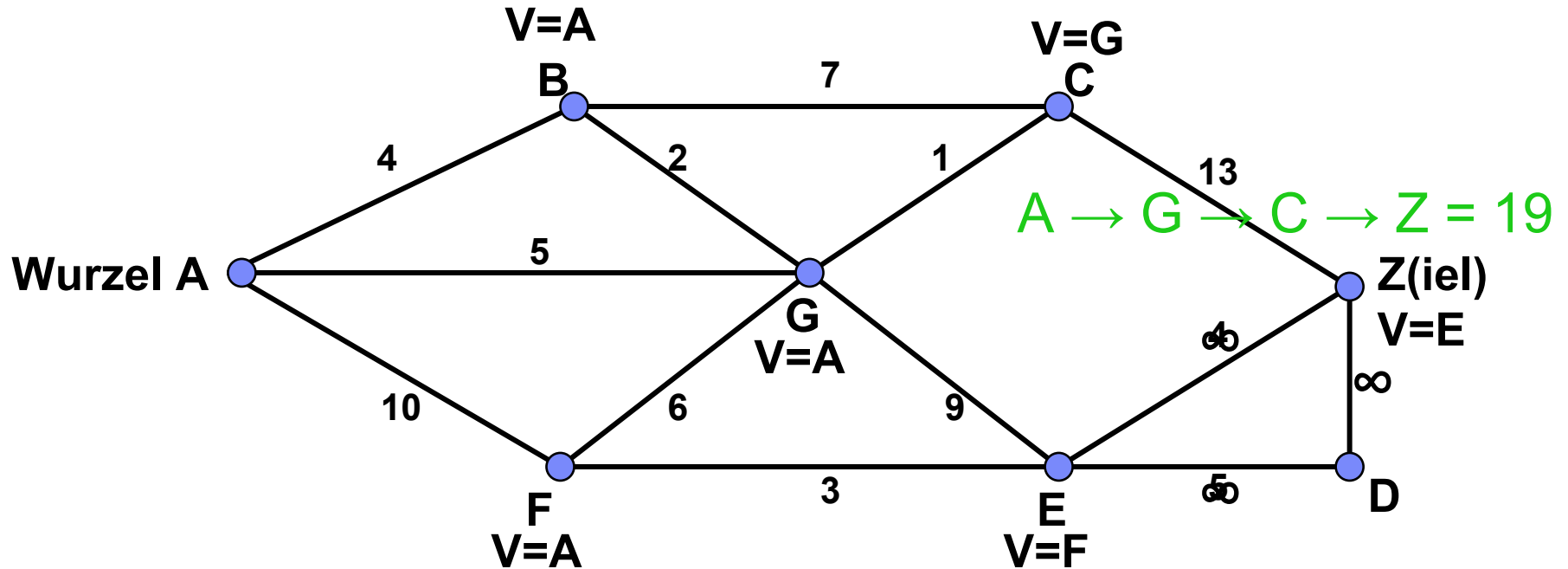
# Der Dijkstra Algorithmus – Schritt 5



$$B = \{ \cancel{E}, \cancel{Z} \}$$

$$E = \{ B, G, C, F, \cancel{E} \}$$

# Der Dijkstra Algorithmus – Schritt 6



$$B = \{ \emptyset, D \}$$

$$E = \{ B, G, C, F, E, Z \}$$

**Kürzester Weg von A nach Z = 17**

$$A \rightarrow F \rightarrow E \rightarrow D = 18$$

## Problem der Aktualisierung von Wegen bei Leitungsausfällen

### Szenario

- Leitung fällt auf Grund eines Unwetterschadens aus
- Router bemerken den Ausfall und starten eine komplette Neuberechnung ALLER Wege
- Router tauschen die Informationen untereinander aus

➔ sehr hoher Rechenaufwand

## Warum gibt es nichts effizienteres?

## Problem der Aktualisierung von Wegen bei Leitungsausfällen

- ... weil die **Wissenschaft** sich mit der Laufzeit  $O[n^2]$  des Dijkstra Algorithmus zufrieden gibt!
- Der scheinbar sehr hohe Rechenaufwand relativiert sich durch die steigende Rechenleistung der Hardware

**Wir** wollen uns damit jedoch jetzt nicht zufrieden stellen lassen!

# Lösungsansatz



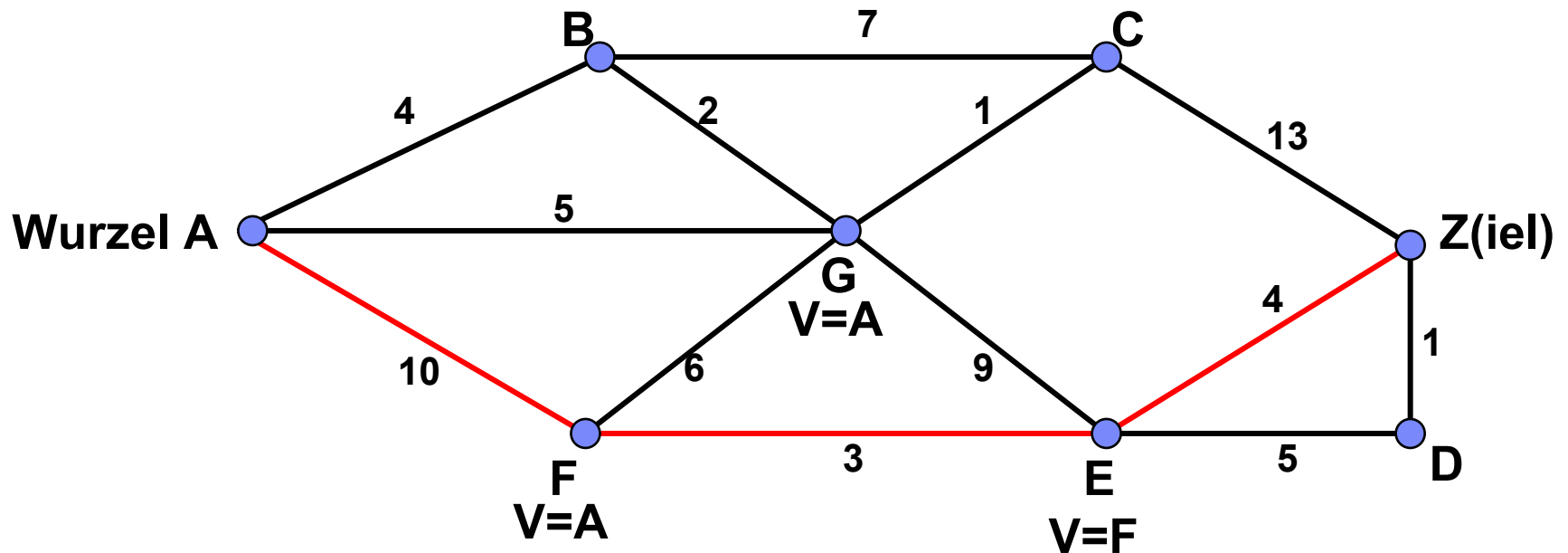
## Idee

- Router haben einmal alle kürzesten Wege für jedes Ziel berechnet **und diese abgespeichert**
- Der Router müsste bei einem Ausfall nun lediglich einen Weg kennen, über den er diesen umgeht, bis er auf einen anderen Knoten des ursprünglichen Pfades trifft. Ab dort würde der „alte“ Pfad dann fortgesetzt.
- Dieser Pfad soll im Folgenden Umgehungsweg heißen
- Umgehungspfade wurden vorher berechnet und Router soll bei Ausfall **sofort** auf diesen umschalten



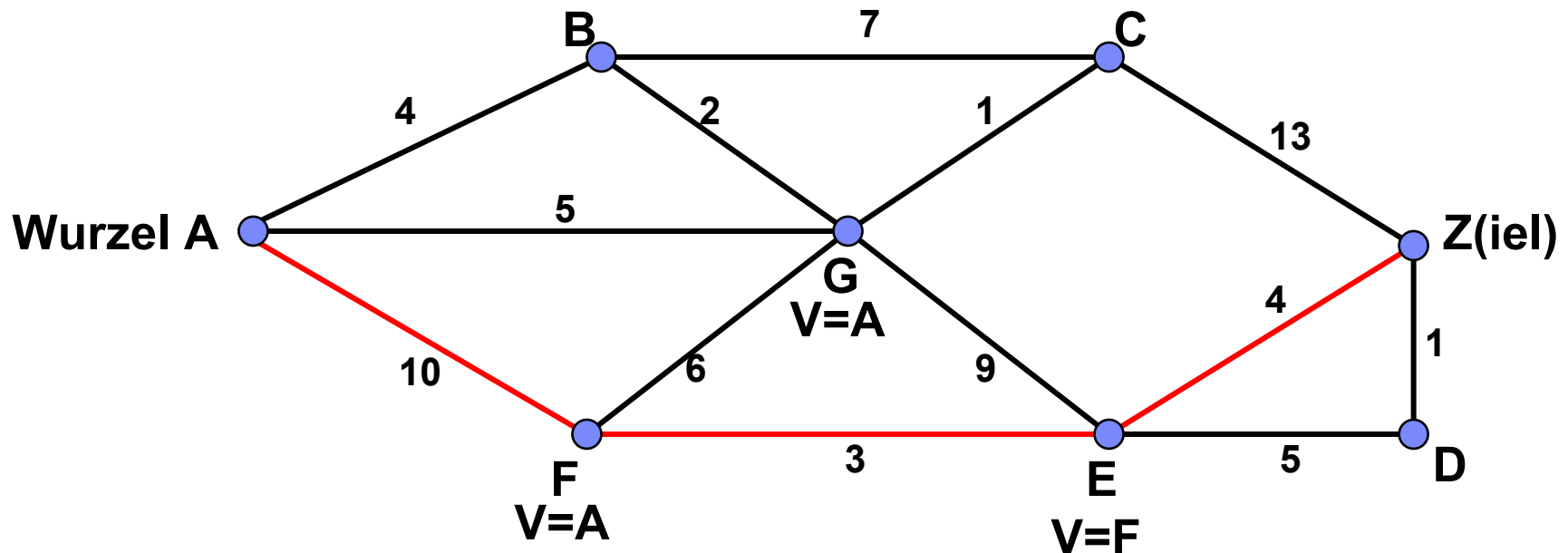
## Realisierung – Beispiel 1

**Router G** als Umgehungsweg kennen,  
**G** kennt dann bereits den Weg über **E** nach **Z**



## Realisierung – Beispiel 2

**F muss G als Umgehungsweg kennen,  
Router E ist defekt,  
G nutzt dann ebenfalls seinen  
Umgehungsweg über C nach Z**






## Bewertung des Lösungsansatzes

- + Leitungsausfall kann sofort umgangen werden
- + weniger Rechenaufwand
  
- Mehr Speicherbedarf
- Umgehungswege nur partiell optimal (Beispiel 2)

**→ Keine „echte“ Lösung des Problems, aber Einsatz für zeitkritische Leitungsabschnitte denkbar**

# Fazit

**Zielsetzung dieser Arbeit war es,**

- **eine Einführung in die graphen-  
theoretische Wegoptimierung zu geben** 
- **den Dijkstra Algorithmus zu veranschaulichen** 
- **Herrn Dijkstra mit einer besseren  
Lösung zu übertreffen!** 

Ende

**Vielen Dank für Ihre Aufmerksamkeit**