Technische Universität Braunschweig

Institut für Betriebssysteme und Rechnerverbund
Connected and Mobile Systems

Computernetze 1

Übung 7

Network Layer – Routing-Verfahren

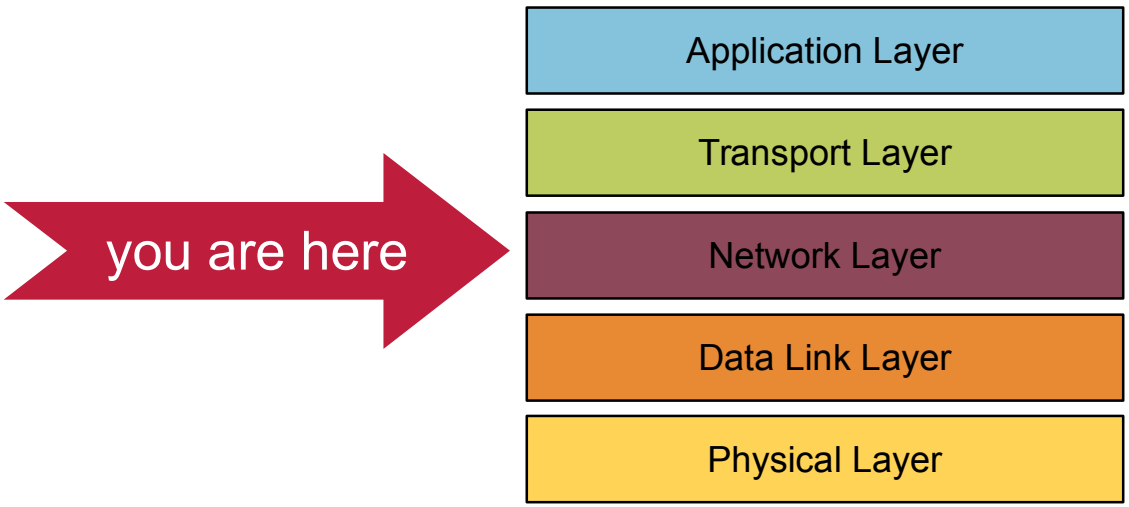
Fynn Schulze, 03. Juli 2025

Technische Universität Braunschweig, IBR

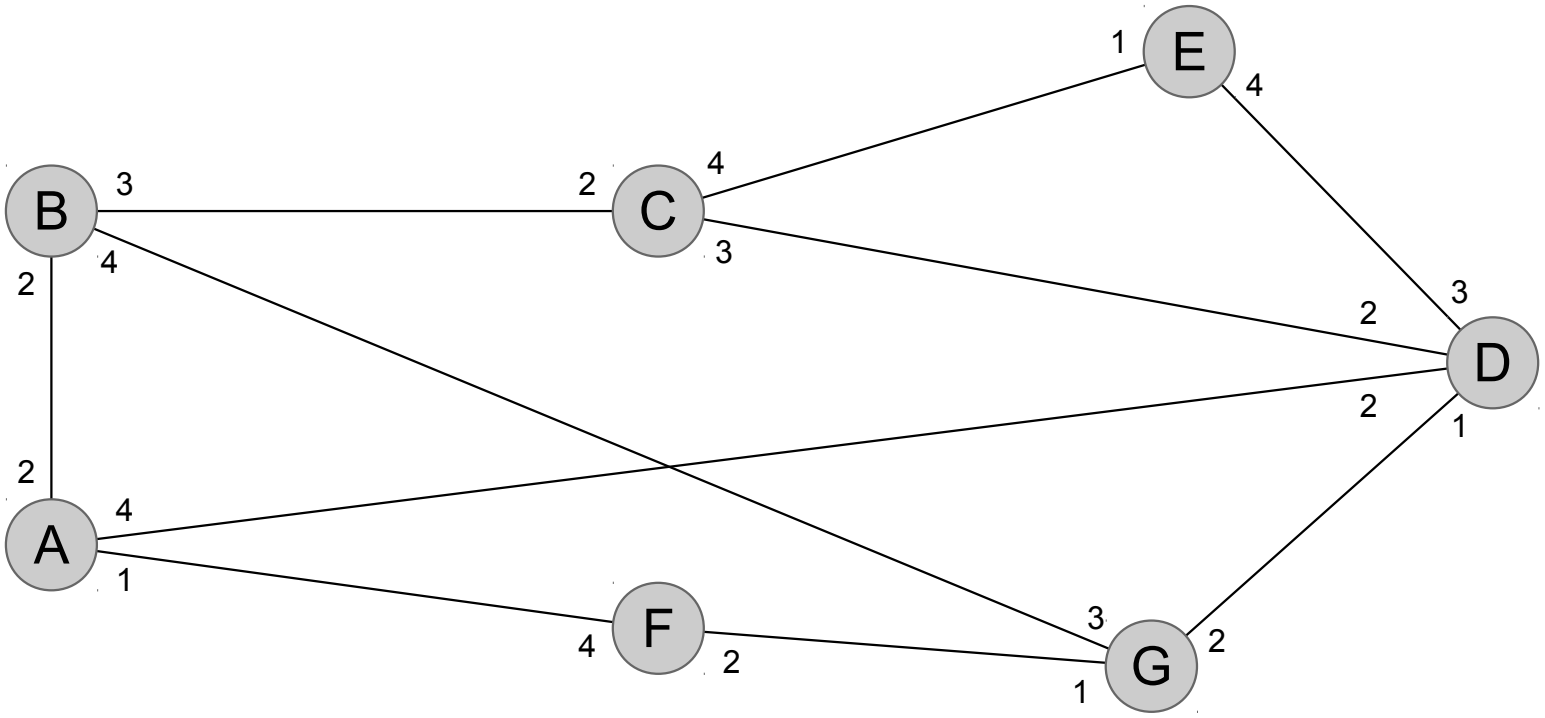
Überblick

- 1) Shortest Path Routing
- 2) Distance Vector Routing
- 3) Flooding und Shortest Path
- 4) Link State Routing

Überblick

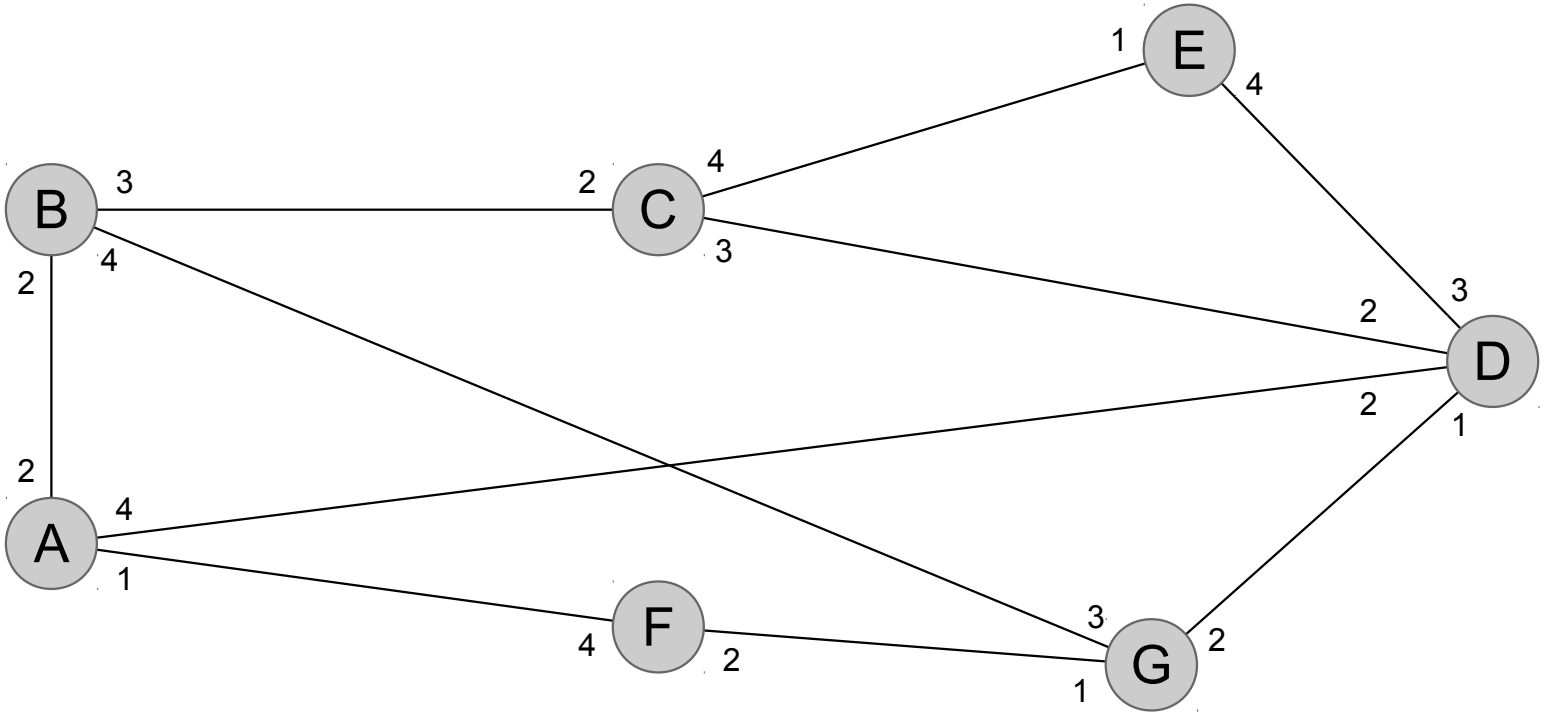

you are here
Application Layer
Transport Layer
Network Layer
Data Link Layer
Physical Layer

Aufgabe 1: Shortest Path Routing



Ermitteln Sie den Shortest-Path-Tree nach Dijkstras Algorithmus für den Router A im folgenden Netz. Geben Sie dabei alle Schritte der Berechnung an.

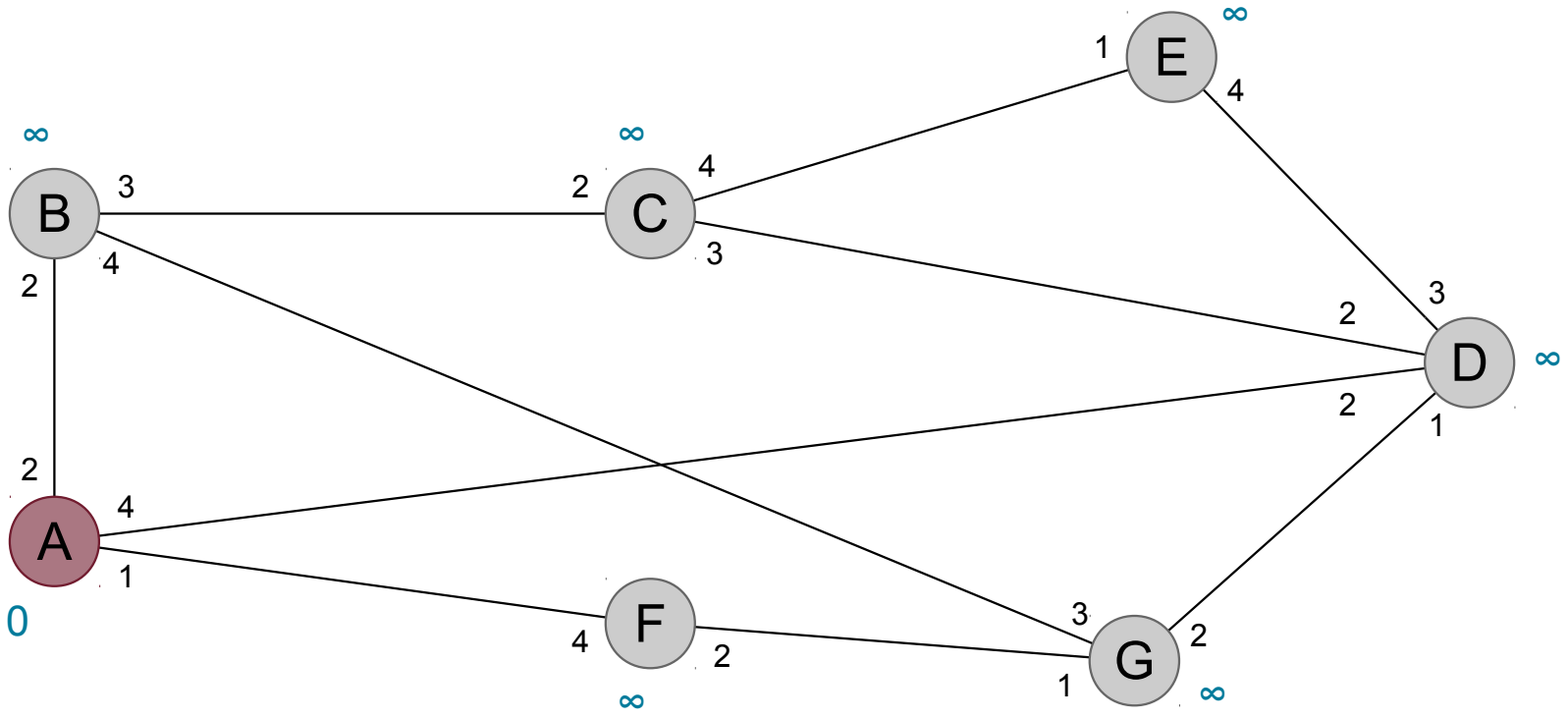
Aufgabe 1: Shortest Path Routing



Ziel

Finde die kürzesten Wege zu allen anderen Knoten und entferne „teure“ Kanten aus dem Graphen.

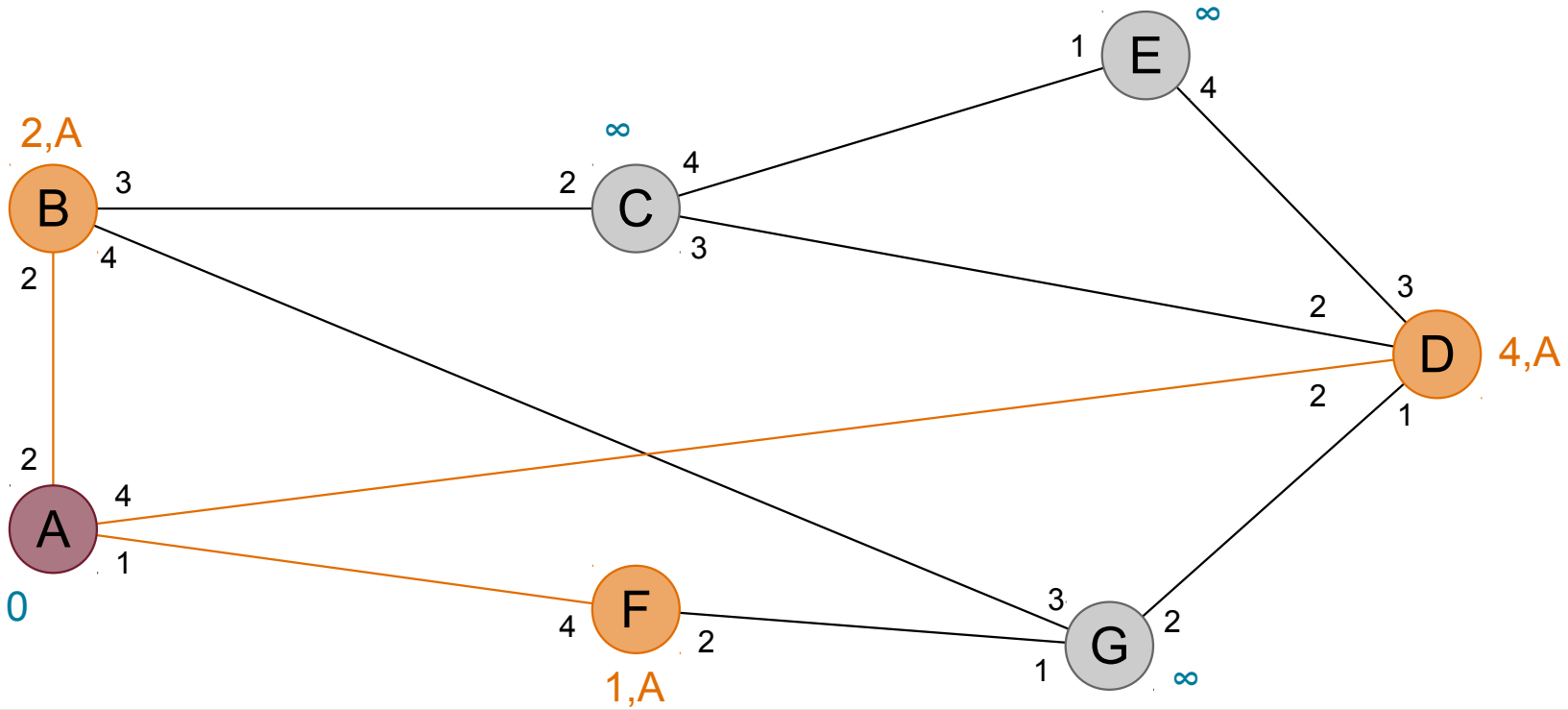
Aufgabe 1: Shortest Path Routing



Initial

Alle Knoten mit unendlicher Entfernung zu A markieren.
A hat die Entfernung 0.

Aufgabe 1: Shortest Path Routing

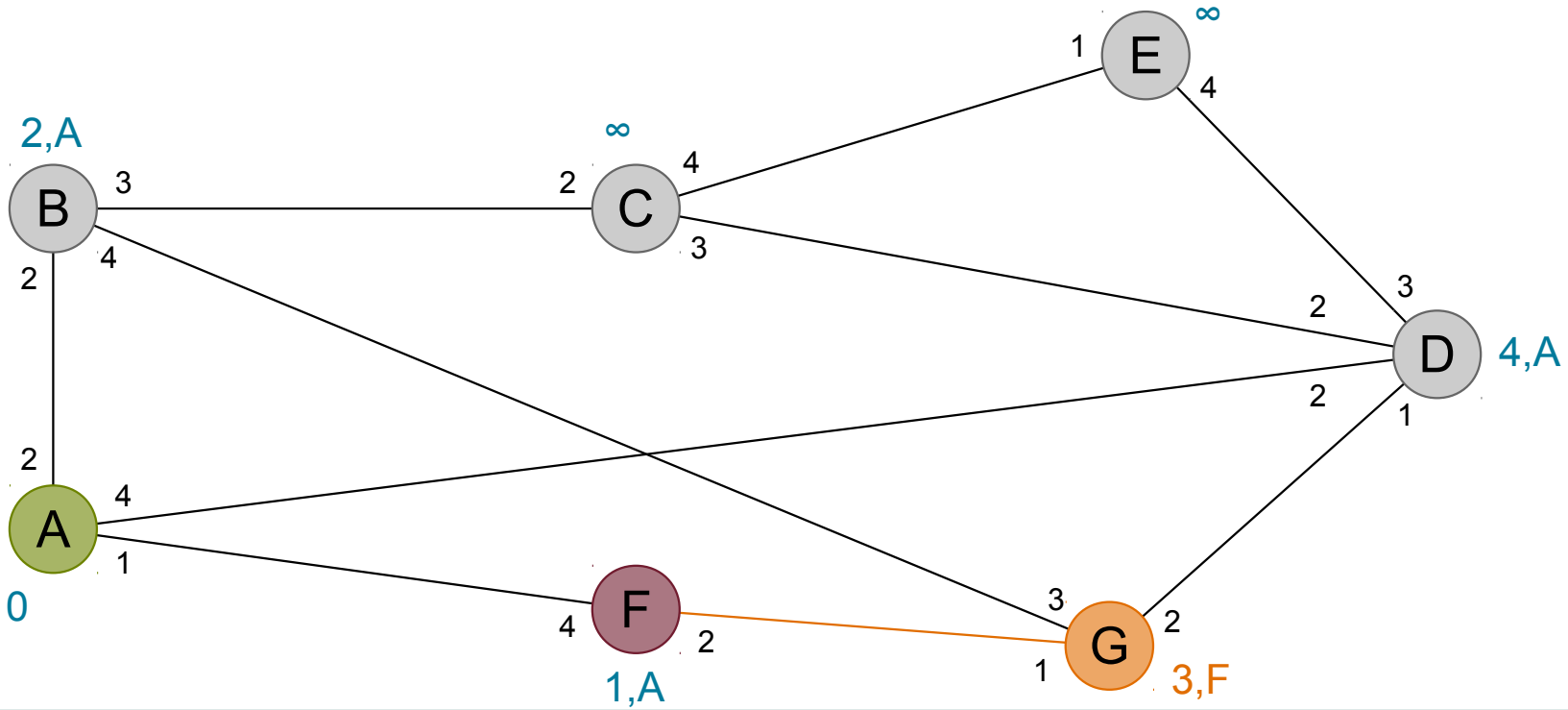


Wähle Knoten mit der geringsten Entfernung: A

Markiere **A** als permanent und betrachte alle Nachbarn

$\Rightarrow$ B(2,A), F(1,A), D(4,A)

Aufgabe 1: Shortest Path Routing

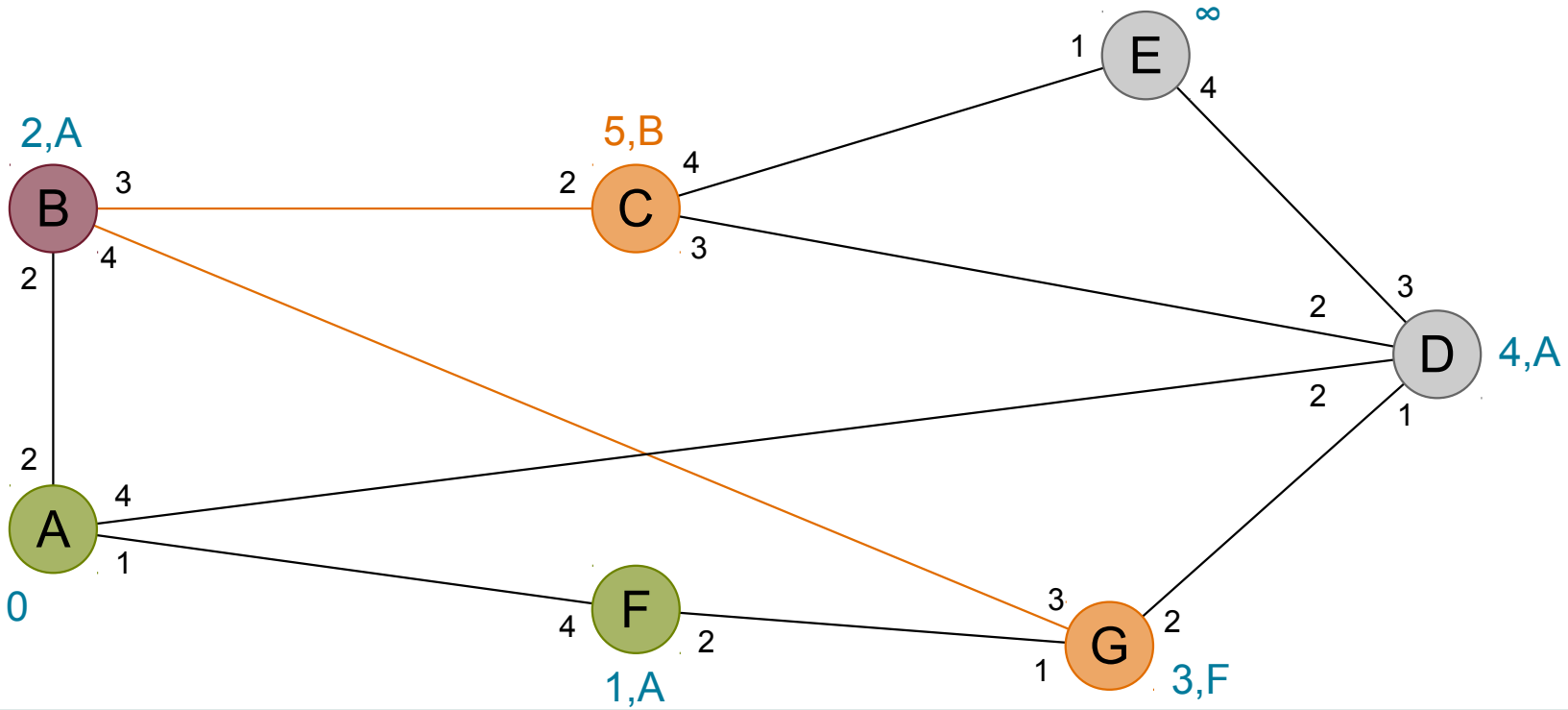


Wähle Knoten mit der geringsten Entfernung: F

Markiere **F** als permanent und betrachte alle Nachbarn

$\Rightarrow$ G(3,F)

Aufgabe 1: Shortest Path Routing

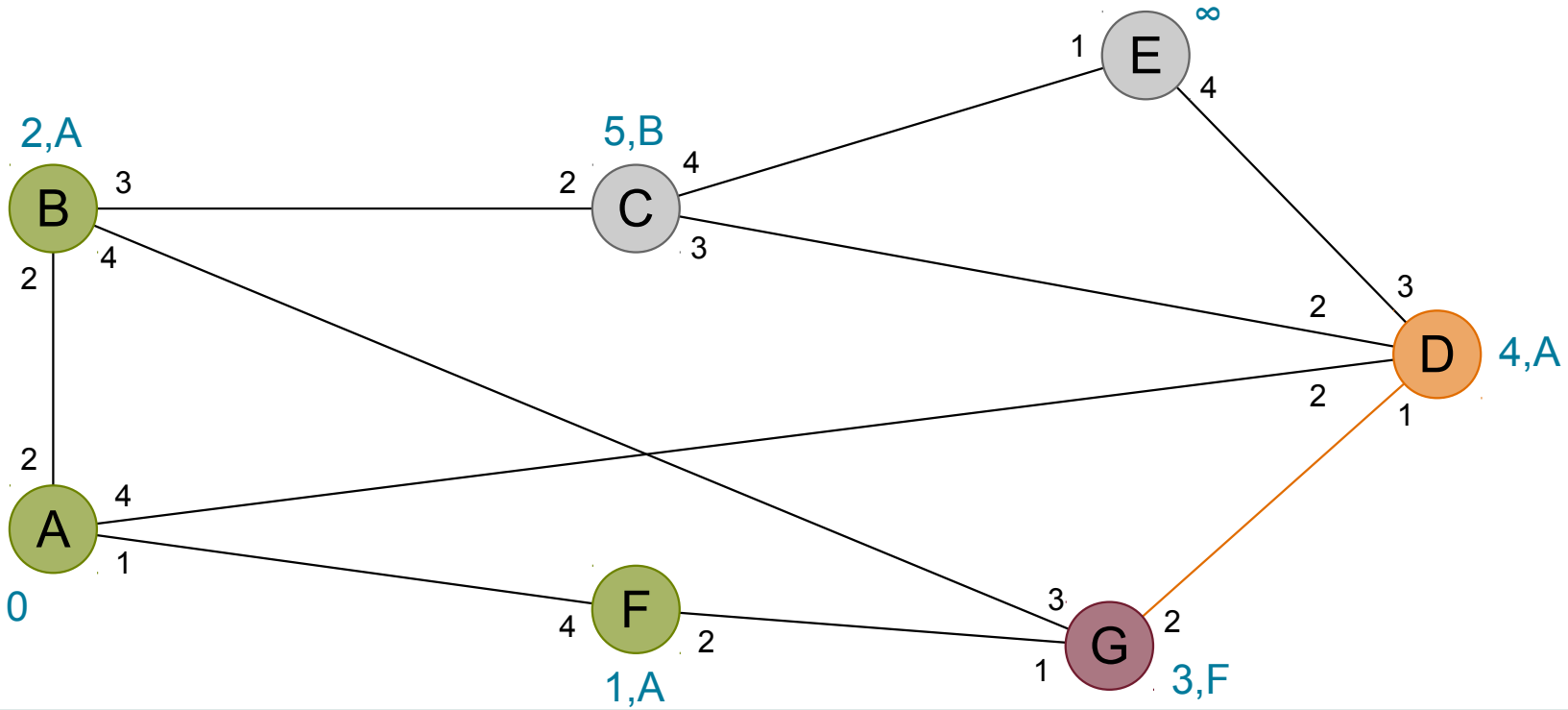


Wähle Knoten mit der geringsten Entfernung: B

Markiere **B** als permanent und betrachte alle Nachbarn

$\Rightarrow$ C(5,B); G bleibt, da nicht besser (6 vs. 3)

Aufgabe 1: Shortest Path Routing

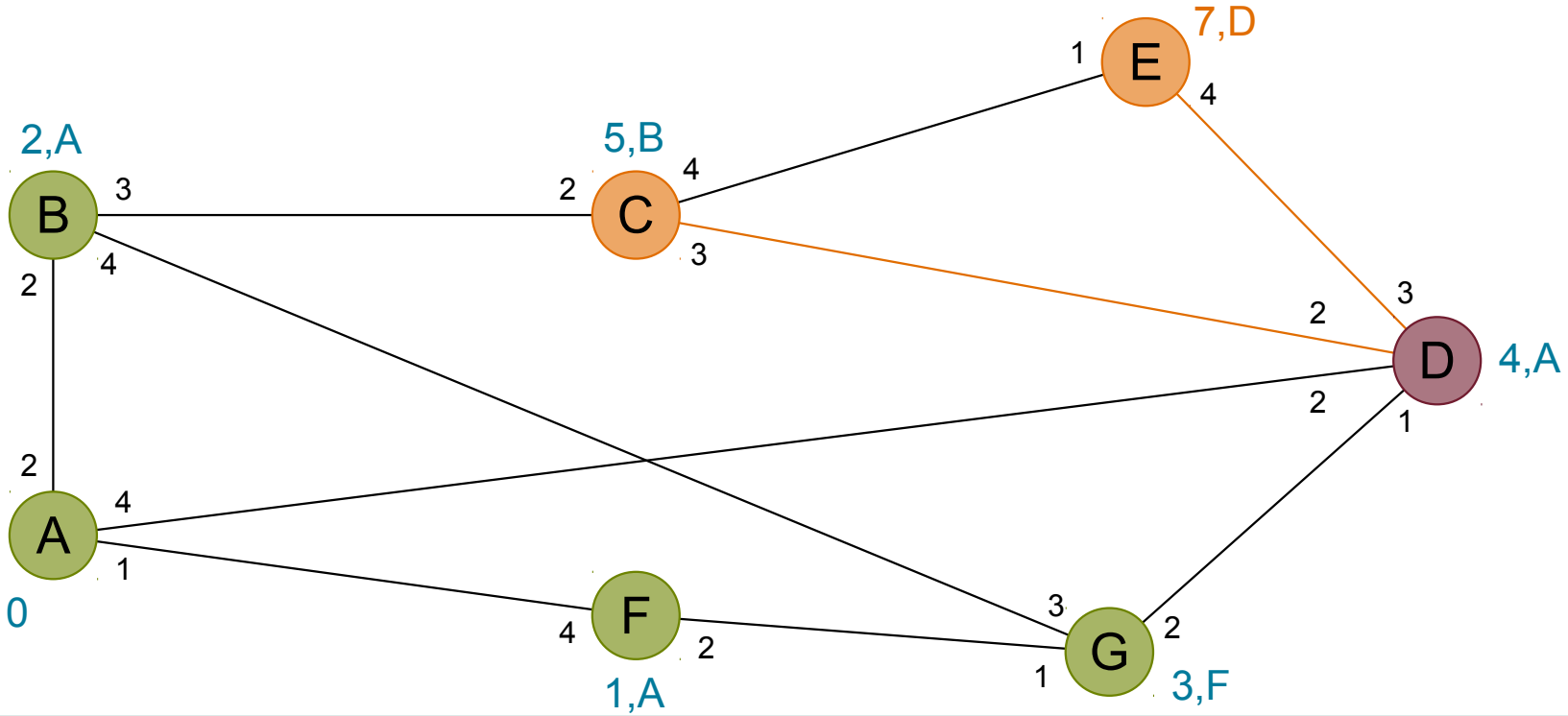


Wähle Knoten mit der geringsten Entfernung: G

Markiere **G** als permanent und betrachte alle Nachbarn

$\Rightarrow$ D bleibt, da nicht besser (5 vs. 4)

Aufgabe 1: Shortest Path Routing

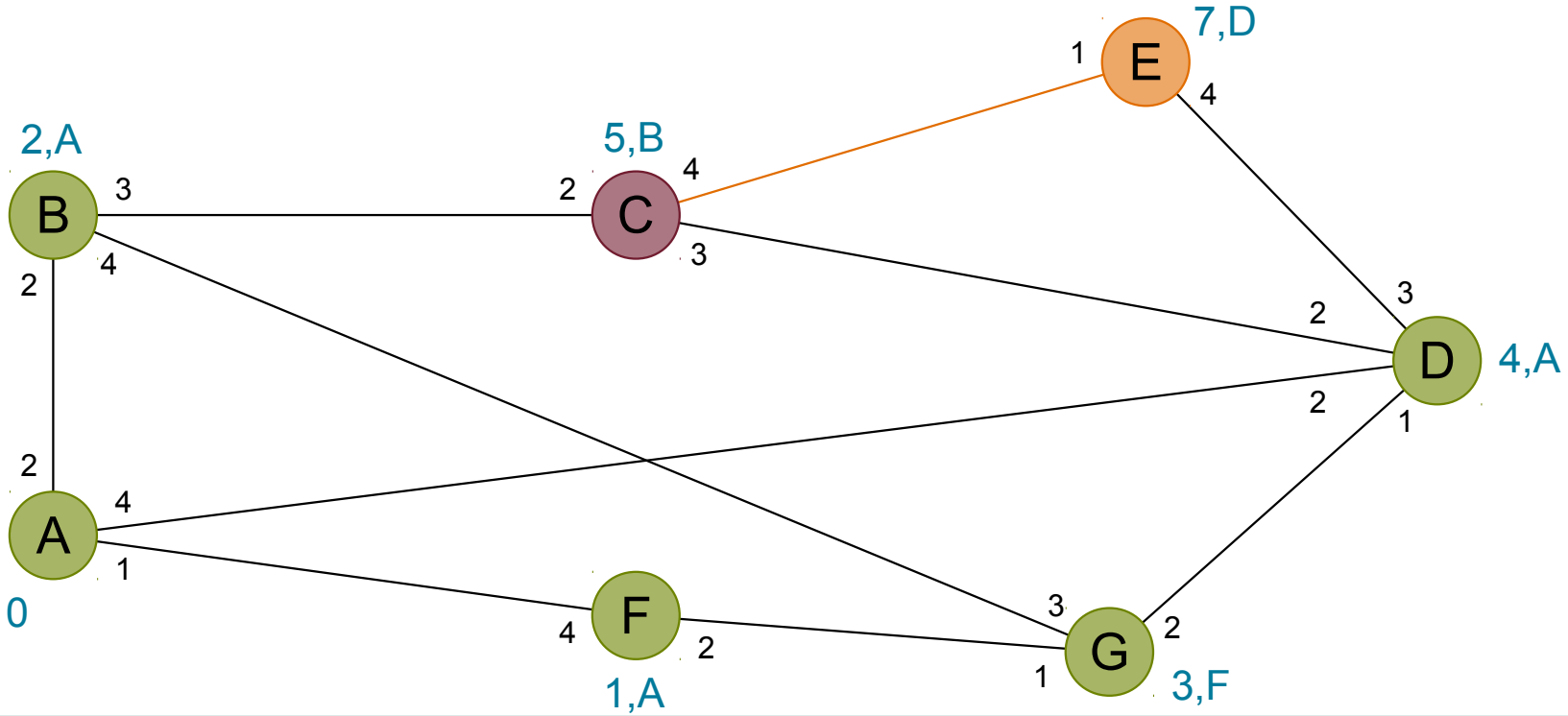


Wähle Knoten mit der geringsten Entfernung: D

Markiere **D** als permanent und betrachte alle Nachbarn

$\Rightarrow$ C bleibt, da nicht besser (6 vs. 5); E(7,D)

Aufgabe 1: Shortest Path Routing

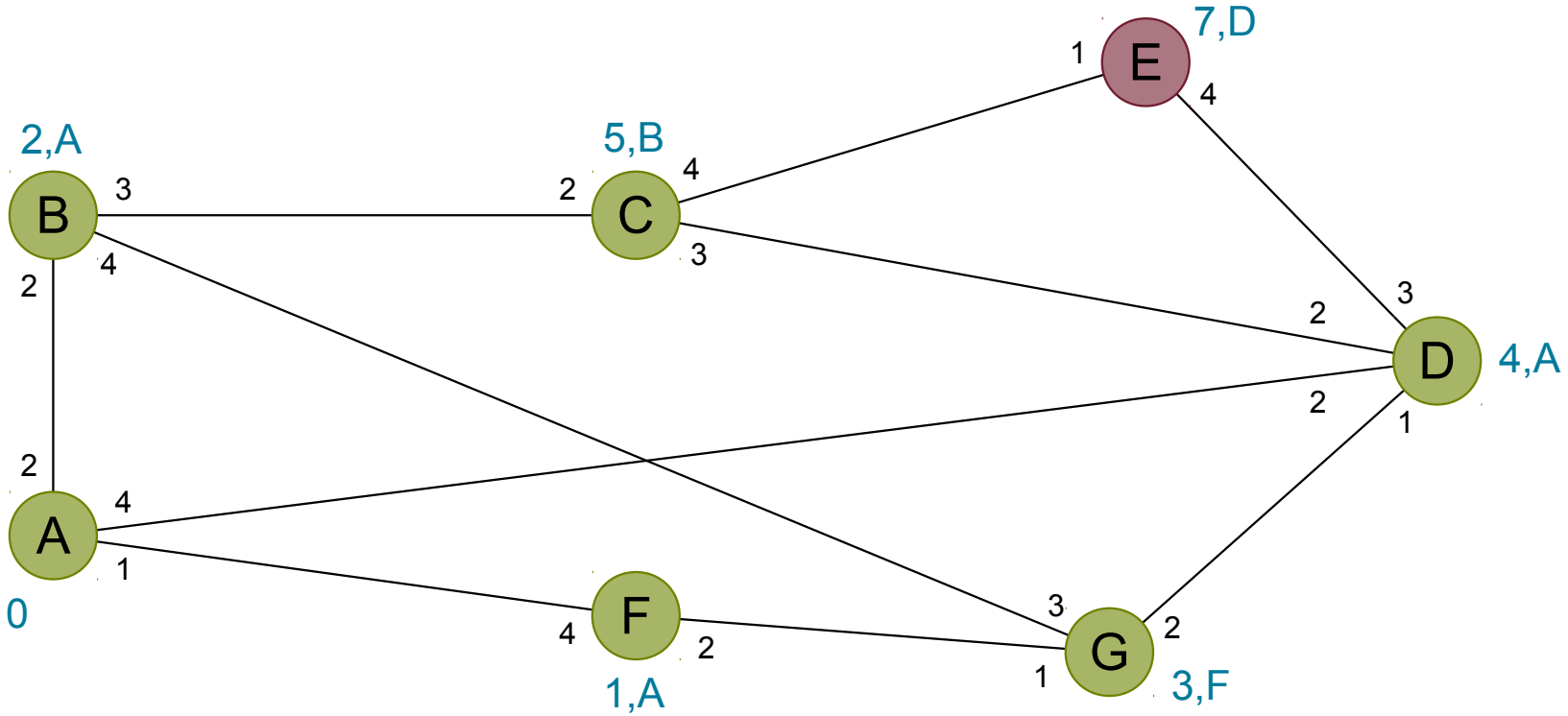


Wähle Knoten mit der geringsten Entfernung: C

Markiere **C** als permanent und betrachte alle Nachbarn

$\Rightarrow$ E bleibt, da nicht besser (9 vs. 7)

Aufgabe 1: Shortest Path Routing



Wähle Knoten mit der geringsten Entfernung: E

Markiere **E** als permanent

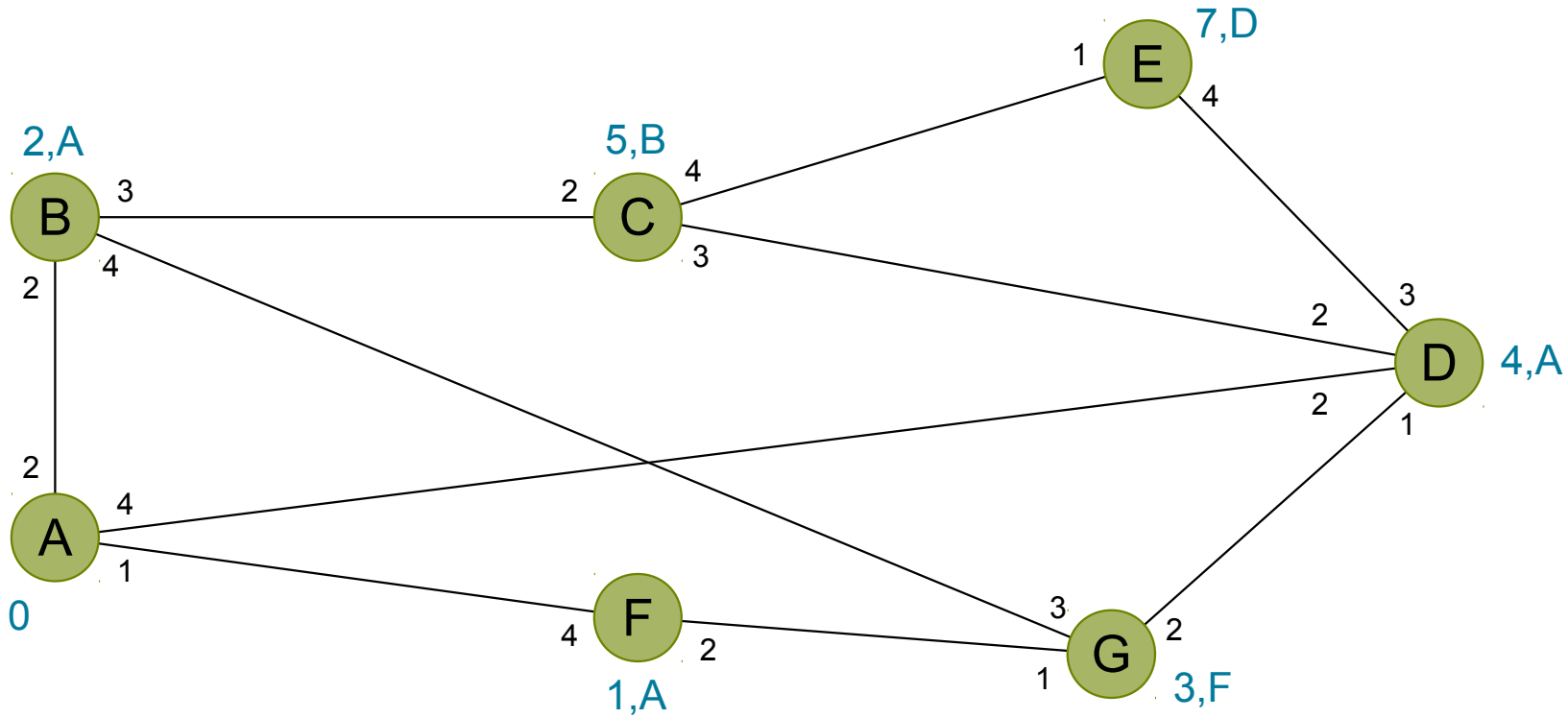
$\Rightarrow$ keine nicht permanenten Nachbarn

Aufgabe 1: Shortest Path Routing

Lösung

1. A permanent $\Rightarrow$ B(2,A), F(1,A), D(4,A)
2. F permanent $\Rightarrow$ G(3,F)
3. B permanent $\Rightarrow$ C(5,B); G bleibt, da nicht besser (6 vs. 3)
4. G permanent $\Rightarrow$ D bleibt, da nicht besser (5 vs. 4)
5. D permanent $\Rightarrow$ C bleibt, da nicht besser (6 vs. 5); E(7,D)
6. C permanent $\Rightarrow$ E bleibt, da nicht besser (9 vs. 7)
7. E permanent

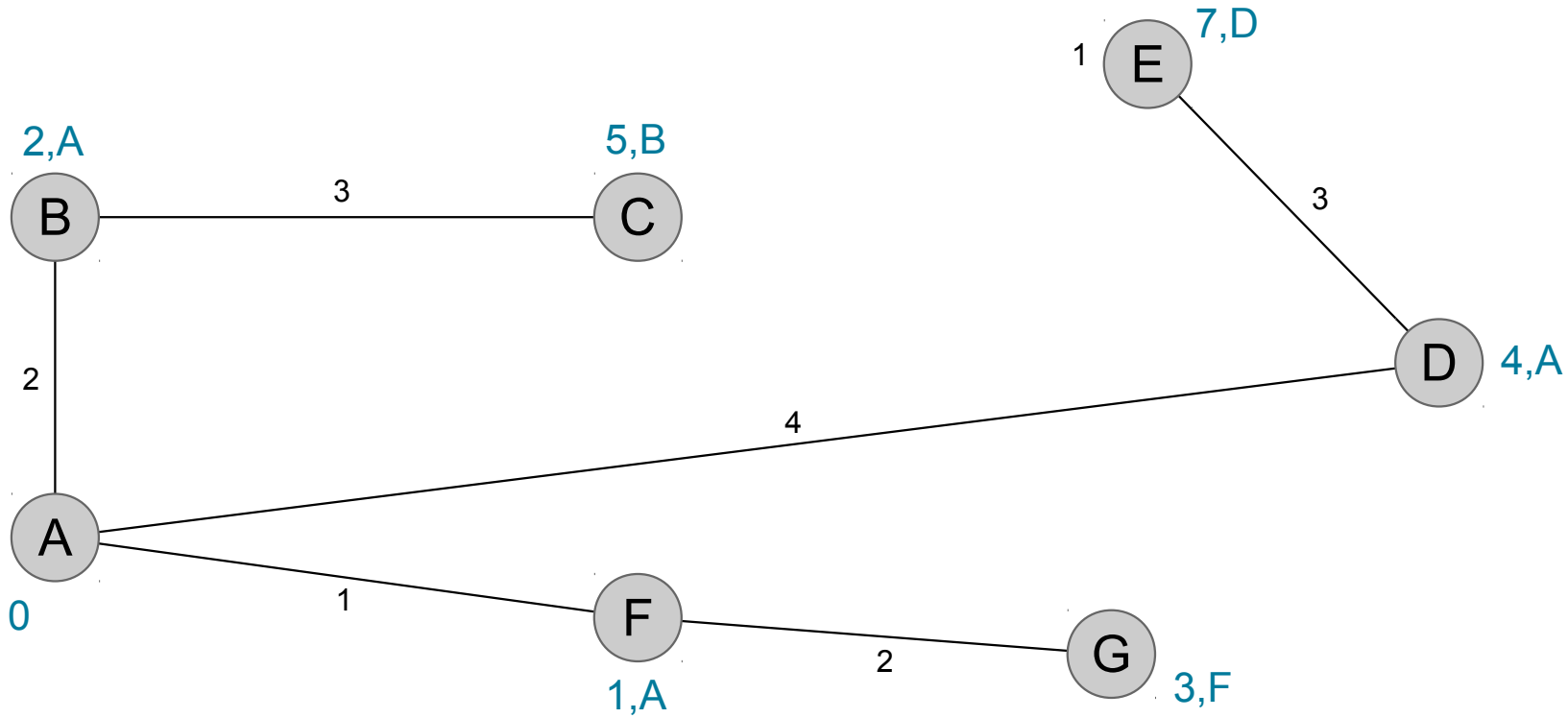
Aufgabe 1: Shortest Path Routing



Lösung

Alle Knoten sind permanent

Aufgabe 1: Shortest Path Routing

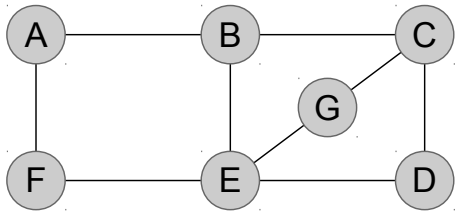


Lösung

Shortest-Path-Tree für Router A

Aufgabe 2: Distance Vector Routing

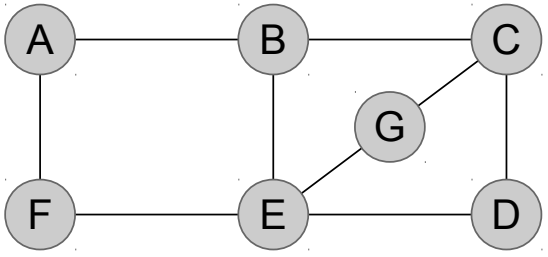
Gegeben sei das folgende Netz von Routern:



a) Berechnen Sie die Routing-Tabellen für alle Router nach dem in RIP verwendeten Bellmann/Ford-Algorithmus. Die zugrunde liegende Metrik für die Distanz zwischen zwei Routern sei die Anzahl der Hops (= 1 zwischen zwei direkt benachbarten Systemen). Nehmen Sie an, dass die Router ihre Vektoren in alphabetischer Reihenfolge senden, beginnend bei Router A. Brechen Sie den Algorithmus ab, wenn jeder Router seinen Vektor einmal gesendet hat.

b) Beschreiben Sie das beim Distance Vector Routing mögliche Problem anhand eines Beispiels.

2 a) Routingtabelle

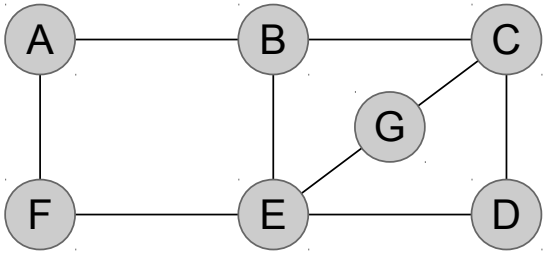


Ziel	A	B	C	D	E	F	G
A	0						
B		0					
C			0				
D				0			
E					0		
F						0	
G							0

Hinweis

In der Realität würden die Router in zufälliger Reihenfolge und möglicherweise auch gleichzeitig senden! Dies ist in der Übungsaufgabe jedoch schwer umsetzbar...

2 a) Routingtabelle

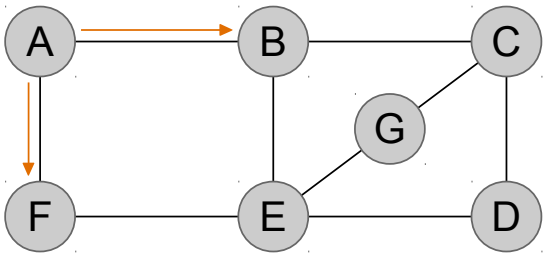


Ziel	A	B	C	D	E	F	G
A	0	1				1	
B	1	0	1		1		
C		1	0	1			1
D			1	0	1		
E		1		1	0	1	1
F	1				1	0	
G			1		1		0

Ausgangszustand

- Jeder Knoten kennt seine direkten Nachbarn
  - z.B. durch „Hello“-Nachrichten
- „1“ in Routingtabelle zeigt, dass Systeme direkt benachbart sind
- Hier in Übung: Reihenfolge des Sendens alphabetisch

2 a) Routingtabelle



Ziel	A	B	C	D	E	F	G
A	0	1				1	
B	1	0	1		1	(2,A)	
C		1	0	1			1
D			1	0	1		
E		1		1	0	1	1
F	1	(2,A)			1	0	
G			1		1		0

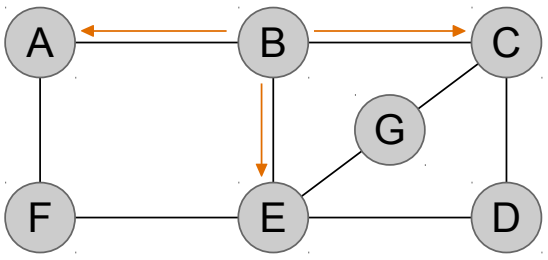
Schritt 1

- A sendet seinen Vektor an B und F: $[0, 1, \infty, \infty, \infty, 1, \infty]$

$\Rightarrow$ B lernt, dass er F über A in 2 Hops erreichen kann

$\Rightarrow$ F lernt, dass er B über A in 2 Hops erreichen kann

2 a) Routingtabelle



Ziel	A	B	C	D	E	F	G
A	0	1	(2,B)		(2,B)	1	
B	1	0	1		1	(2,A)	
C	(2,B)	1	0	1	(2,B)		1
D			1	0	1		
E	(2,B)	1	(2,B)	1	0	1	1
F	1	(2,A)	(3,B)		1	0	
G			1		1		0

Schritt 2

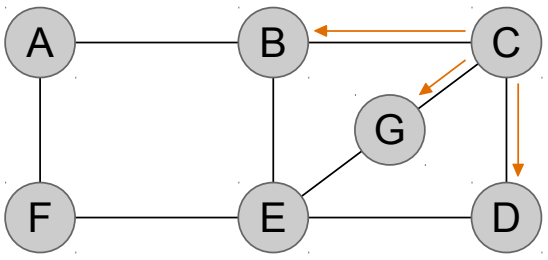
- B sendet seinen Vektor an A, C und E: $[1, 0, 1, \infty, 1, 2, \infty]$

$\Rightarrow$ A lernt, dass er C und E über B in jeweils 2 Hops erreichen kann

$\Rightarrow$ C lernt, dass er A und E in 2 Hops und F in 3 Hops über B erreichen kann

$\Rightarrow$ E lernt, dass er A und C über B in jeweils 2 Hops erreichen kann

2 a) Routingtabelle



Ziel	A	B	C	D	E	F	G
A	0	1	(2,B)	(3,C)	(2,B)	1	(3,C)
B	1	0	1	(2,C)	1	(2,A)	(2,C)
C	(2,B)	1	0	1	(2,B)		1
D		(2,C)	1	0	1		(2,C)
E	(2,B)	1	(2,B)	1	0	1	1
F	1	(2,A)	(3,B)	(4,C)	1	0	(4,C)
G		(2,C)	1	(2,C)	1		0

Schritt 3

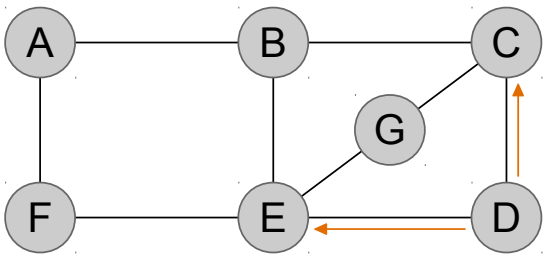
- C sendet seinen Vektor an B, D, und G: $[2, 1, 0, 1, 2, 3, 1]$

$\Rightarrow$ B lernt, dass er D und G über C in jeweils 2 Hops erreichen kann

$\Rightarrow$ D lernt unter Anderem, dass er F über C in 4 Hops erreichen kann

$\Rightarrow$ G lernt unter Anderem, dass er A über C in 3 Hops erreichen kann

2 a) Routingtabelle



Ziel	A	B	C	D	E	F	G
A	0	1	(2,B)	(3,C)	(2,B)	1	(3,C)
B	1	0	1	(2,C)	1	(2,A)	(2,C)
C	(2,B)	1	0	1	(2,B)		1
D		(2,C)	1	0	1		(2,C)
E	(2,B)	1	(2,B)	1	0	1	1
F	1	(2,A)	(3,B)	(4,C)	1	0	(4,C)
G		(2,C)	1	(2,C)	1		0

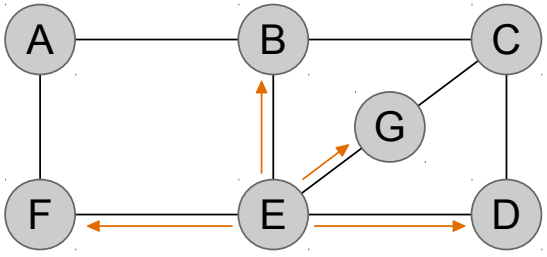
Schritt 4

- D sendet seinen Vektor an C und E: $[3, 2, 1, 0, 1, 4, 2]$

$\Rightarrow$ C lernt keine besseren Routen und ändert seine Tabelle nicht

$\Rightarrow$ E lernt keine besseren Routen und ändert seine Tabelle nicht

2 a) Routingtabelle



Ziel	A	B	C	D	E	F	G
A	0	1	(2,B)	(3,C)	(2,B)	1	(3,C)
B	1	0	1	(2,C)	1	(2,A)	(2,C)
C	(2,B)	1	0	1	(2,B)	(3,E)	1
D		(2,C)	1	0	1	(2,E)	(2,C)
E	(2,B)	1	(2,B)	1	0	1	1
F	1	(2,A)	(3,B)	(2,E)	1	0	(2,E)
G		(2,C)	1	(2,C)	1	(2,E)	0

Schritt 5

- E sendet seinen Vektor an B, D, F und G: $[2, 1, 2, 1, 0, 1, 1]$

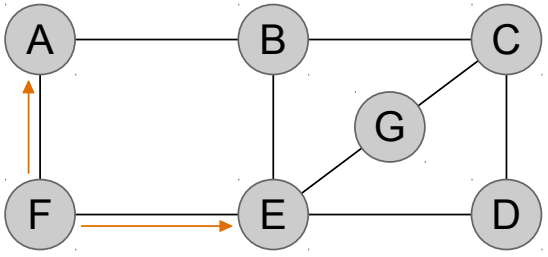
$\Rightarrow$ B lernt keine besseren Routen und ändert seine Tabelle nicht

$\Rightarrow$ D lernt eine bessere Route zu F und aktualisiert den Eintrag

$\Rightarrow$ F lernt, dass er D und G in 2 Hops und C in 3 Hops erreichen kann

$\Rightarrow$ G lernt eine bessere Route zu F und aktualisiert den Eintrag

2 a) Routingtabelle



Ziel	A	B	C	D	E	F	G
A	0	1	(2,B)	(3,C)	(2,B)	1	(3,C)
B	1	0	1	(2,C)	1	(2,A)	(2,C)
C	(2,B)	1	0	1	(2,B)	(3,E)	1
D	(3,F)	(2,C)	1	0	1	(2,E)	(2,C)
E	(2,B)	1	(2,B)	1	0	1	1
F	1	(2,A)	(3,B)	(2,E)	1	0	(2,E)
G	(3,F)	(2,C)	1	(2,C)	1	(2,E)	0

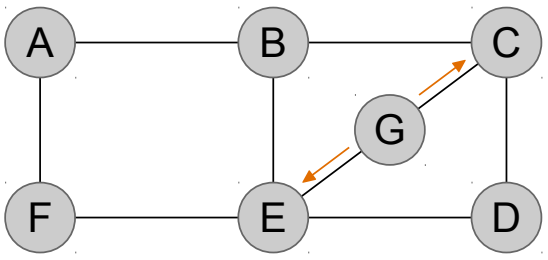
Schritt 6

- F sendet seinen Vektor an A und E: $[1, 2, 3, 2, 1, 0, 2]$

$\Rightarrow$ A lernt, dass er D und G über F in 3 Hops erreichen kann

$\Rightarrow$ E lernt keine besseren Routen und ändert seine Tabelle nicht

2 a) Routingtabelle



Ziel	A	B	C	D	E	F	G
A	0	1	(2,B)	(3,C)	(2,B)	1	(3,C)
B	1	0	1	(2,C)	1	(2,A)	(2,C)
C	(2,B)	1	0	1	(2,B)	(3,E)	1
D	(3,F)	(2,C)	1	0	1	(2,E)	(2,C)
E	(2,B)	1	(2,B)	1	0	1	1
F	1	(2,A)	(3,B)	(2,E)	1	0	(2,E)
G	(3,F)	(2,C)	1	(2,C)	1	(2,E)	0

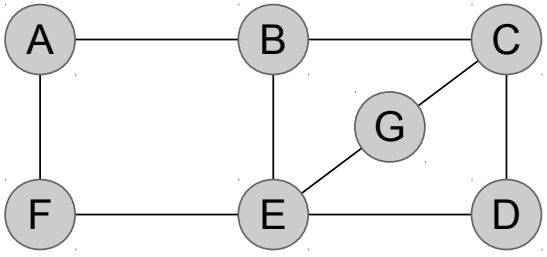
Schritt 7

- G sendet seinen Vektor an C und E: $[3, 2, 1, 2, 1, 2, 0]$

$\Rightarrow$ C lernt keine besseren Routen und ändert seine Tabelle nicht

$\Rightarrow$ E lernt keine besseren Routen und ändert seine Tabelle nicht

2 a) Routingtabelle



Ziel	A	B	C	D	E	F	G
A	0	1	(2,B)	(3,C)	(2,B)	1	(3,C)
B	1	0	1	(2,C)	1	(2,A)	(2,C)
C	(2,B)	1	0	1	(2,B)	(3,E)	1
D	(3,F)	(2,C)	1	0	1	(2,E)	(2,C)
E	(2,B)	1	(2,B)	1	0	1	1
F	1	(2,A)	(3,B)	(2,E)	1	0	(2,E)
G	(3,F)	(2,C)	1	(2,C)	1	(2,E)	0

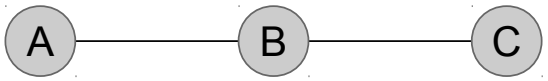
Hinweis

Der Algorithmus wird nach Aufgabenstellung hier abgebrochen. Zufällig hat sich dabei bereits eine optimale Routingtabelle ergeben. Dies kann je nach Netztopologie jedoch länger dauern!
Weiterhin kann sich ein reales Netz über die Zeit verändern.

2 b) Count-to-Infinity-Problem

b) Beschreiben Sie das beim Distance Vector Routing mögliche Problem anhand eines Beispiels.

2 b) Count-to-Infinity-Problem

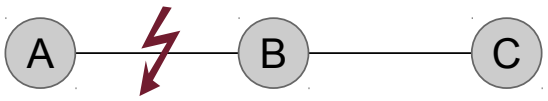


Ziel	A	B	C
A	0	1	(2,B)
B	1	0	1
C	(2,B)	1	0

Ausgangszustand

- Routingtabelle ist gefüllt

2 b) Count-to-Infinity-Problem

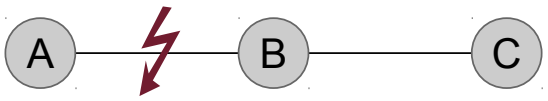


Ziel	A	B	C
A	0		(2,B)
B		0	1
C		1	0

Schritt 1

- Link zwischen A und B bricht zusammen
- A löscht alle Routen über B nach Timeout
- B löscht Route zu A nach Timeout

2 b) Count-to-Infinity-Problem

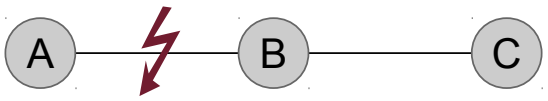


Ziel	A	B	C
A	0	(3,C)	(2,B)
B		0	1
C		1	0

Schritt 2

- B erhält von C den Vektor:
  - $[2, 1, 0]$
- C zeigt an, dass er A in 2 Hops erreichen kann
- B glaubt, dass er A in 3 Hops über C erreichen kann

2 b) Count-to-Infinity-Problem



Ziel	A	B	C
A	0	(3,C)	(4,B)
B		0	1
C		1	0

Schritt 3

- C erhält von B den Vektor:
  - $[3, 0, 1]$
- B zeigt an, dass er A in 3 Hops erreichen kann
- C muss seinen Eintrag für A aktualisieren

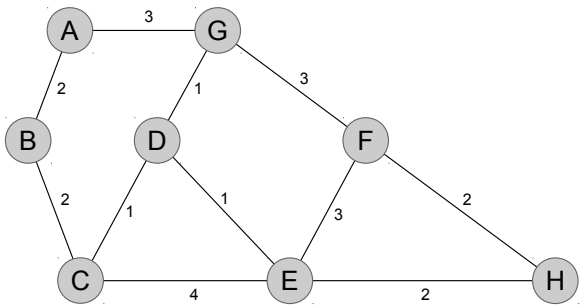
Achtung

Distanzen erhöhen sich unendlich weiter!

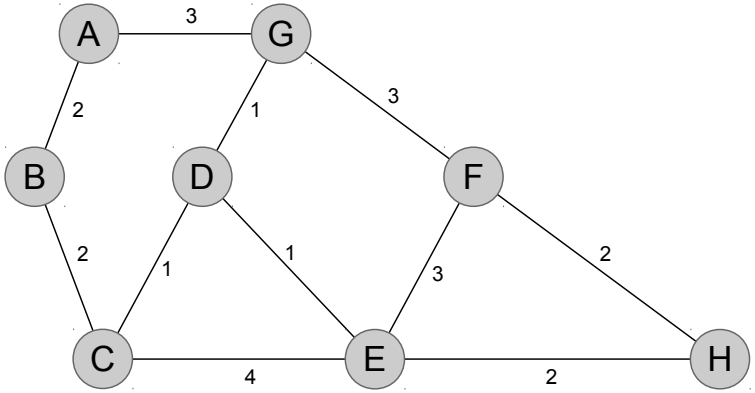
Aufgabe 3: Flooding und Shortest Path

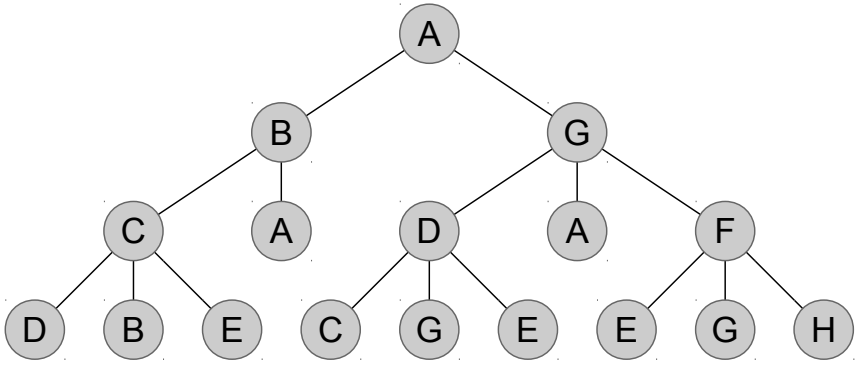
In diesem Netzwerk soll ein Paket von Knoten A zum Knoten H geschickt werden.

a) Listen Sie alle Routen auf, die das Paket nimmt, wenn das Verfahren *Fluten/Flooding* zum Einsatz kommt. Die maximale Anzahl der Hops beträgt 3 (der Zielknoten des dritten Hops leitet das Paket also nicht weiter sondern er wirft es weg, falls er nicht der ursprüngliche Zielknoten ist). Geben Sie auch an, wie viele Hops insgesamt benötigt werden. Annahme: Der Knoten verwirft ein Paket, das zu ihm zurückkehrt.



3 a) Routen des Pakets





Lösung

A sendet an B und G

B sendet an A und C
⇒ A verwirft das Paket

C sendet an B, D und E
⇒ max. Hopzahl erreicht

G sendet an A, D und F
⇒ A verwirft das Paket

D sendet an C, E und G
⇒ max. Hopzahl erreicht

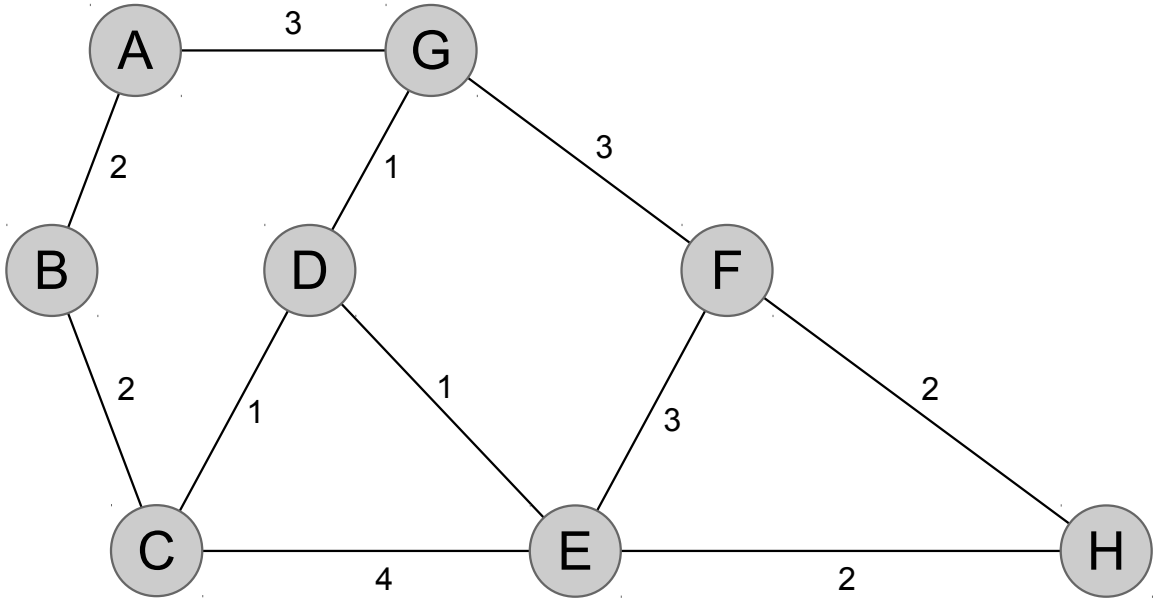
F sendet an E, G und H
⇒ max. Hopzahl erreicht

Gesamte Hopzahl: **16**

3 b) Dijkstras Algorithmus

b) Berechnen Sie die beste Route von A nach H, indem Sie mit Dijkstras Algorithmus den kürzesten Weg vom A nach H bestimmen.

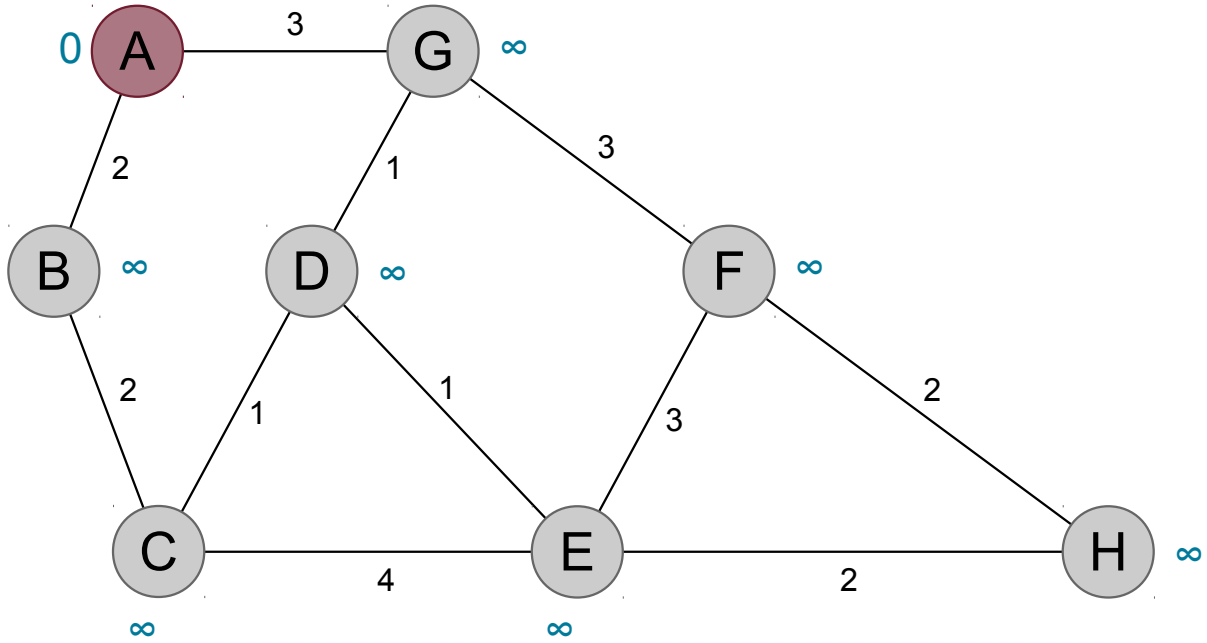
3 b) Dijkstras Algorithmus



Ziel

Finde den kürzesten Weg zu Knoten H und entferne „teure“ Kanten aus dem Graphen.

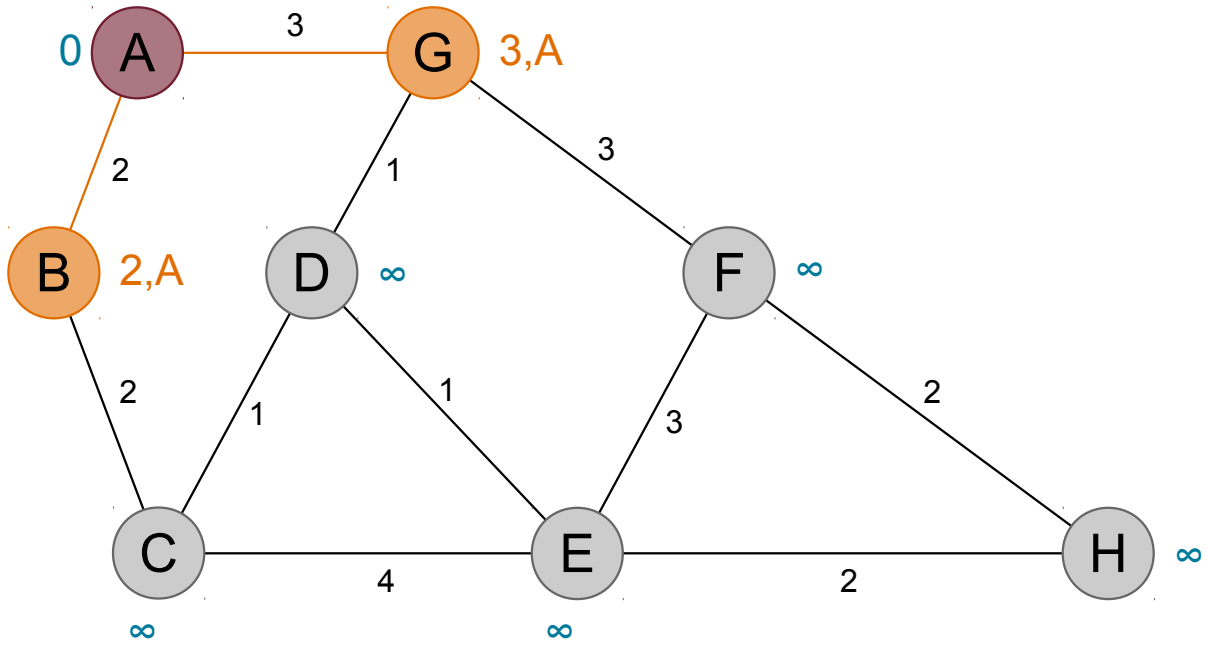
3 b) Dijkstras Algorithmus



Initial

Alle Knoten mit unendlicher Entfernung zu A markieren.
A hat die Entfernung 0.

3 b) Dijkstras Algorithmus

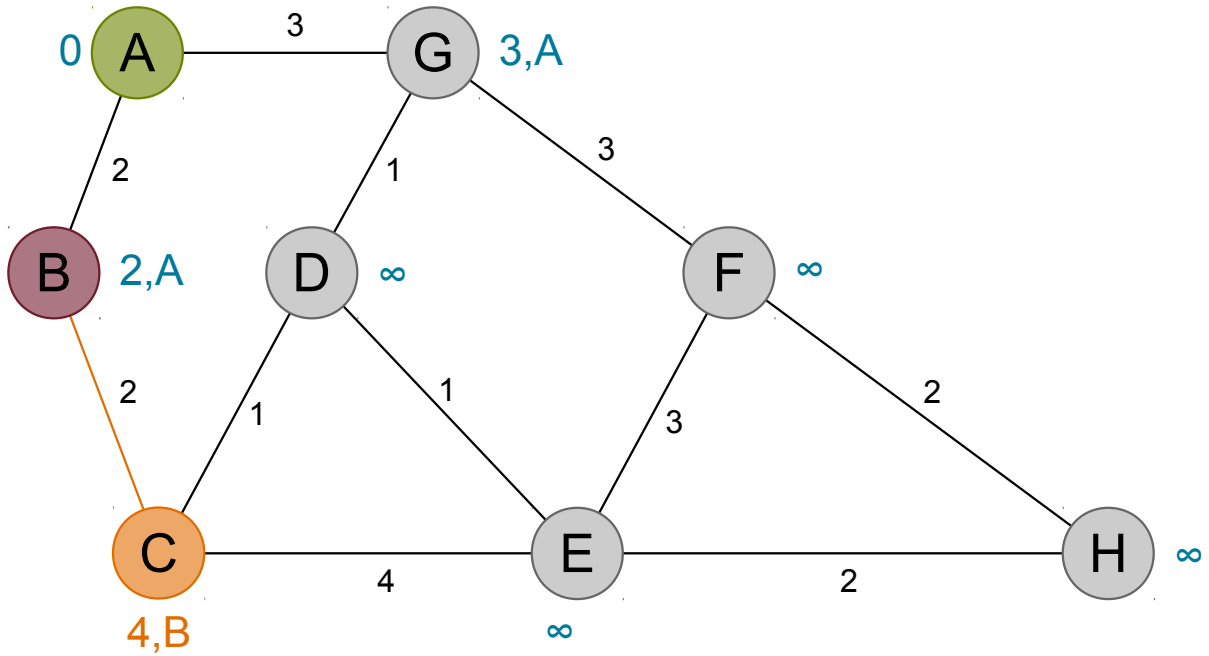


Wähle Knoten mit der geringsten Entfernung: A

Markiere **A** als permanent und betrachte alle Nachbarn

$\Rightarrow$ B(2,A), G(3,A)

3 b) Dijkstras Algorithmus

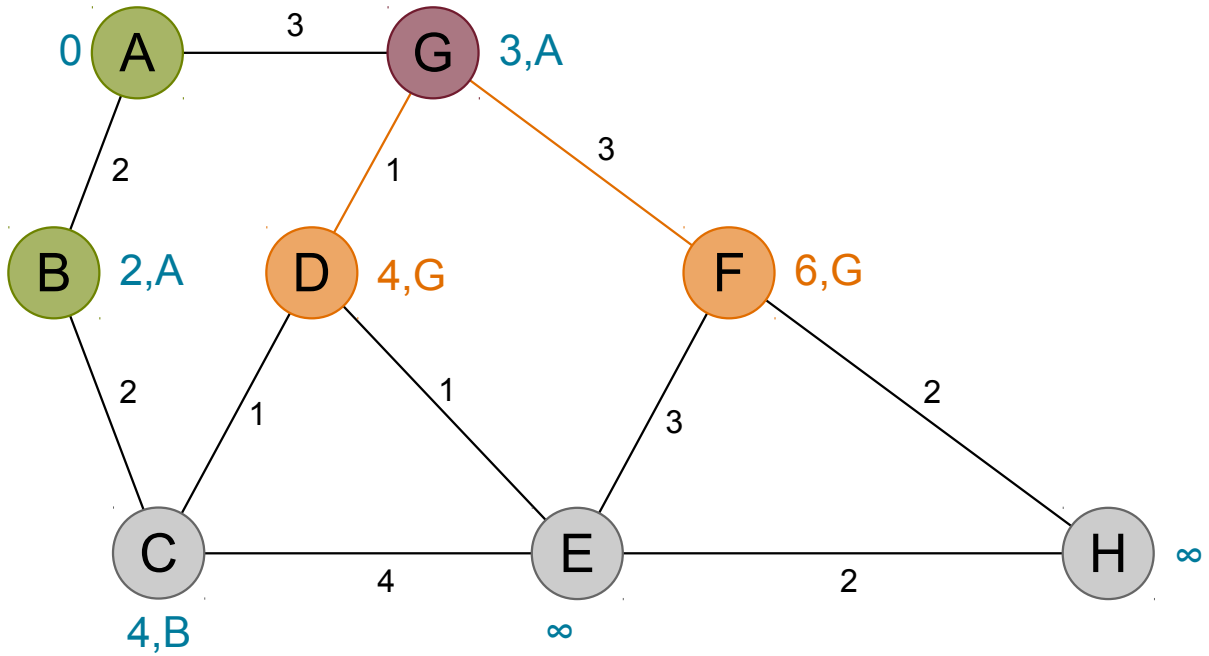


Wähle Knoten mit der geringsten Entfernung: B

Markiere **B** als permanent und betrachte alle Nachbarn

$\Rightarrow$ C(4,B)

3 b) Dijkstras Algorithmus

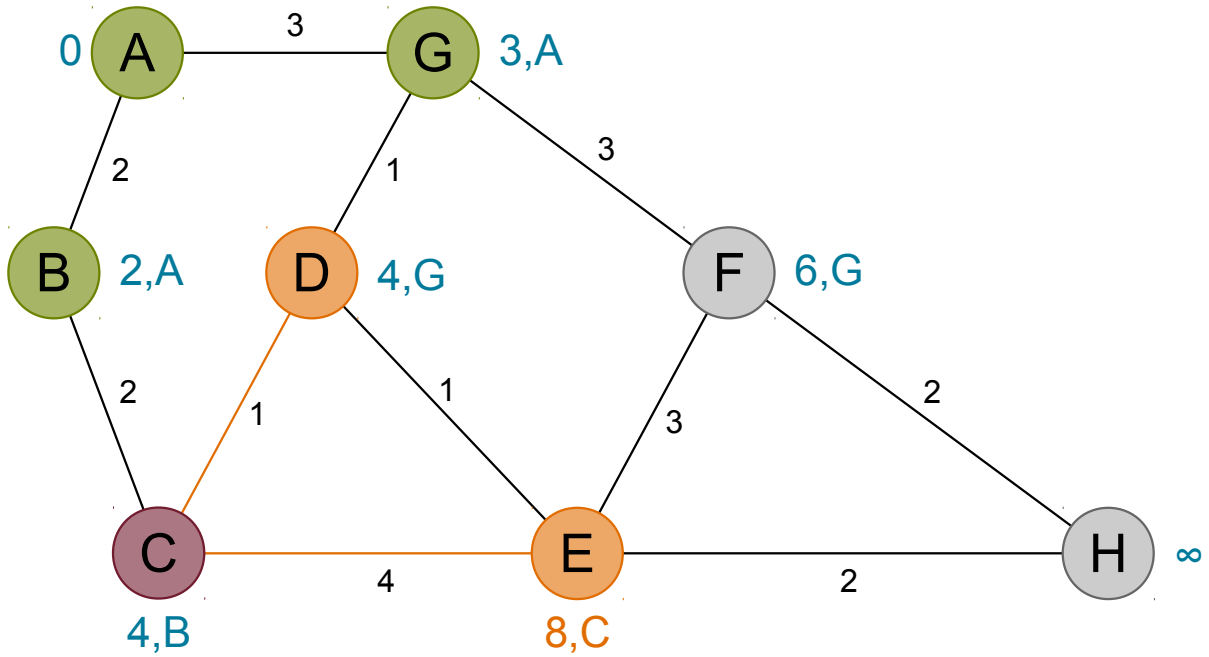


Wähle Knoten mit der geringsten Entfernung: G

Markiere **G** als permanent und betrachte alle Nachbarn

$\Rightarrow$ D(4,G), F(6,G)

3 b) Dijkstras Algorithmus

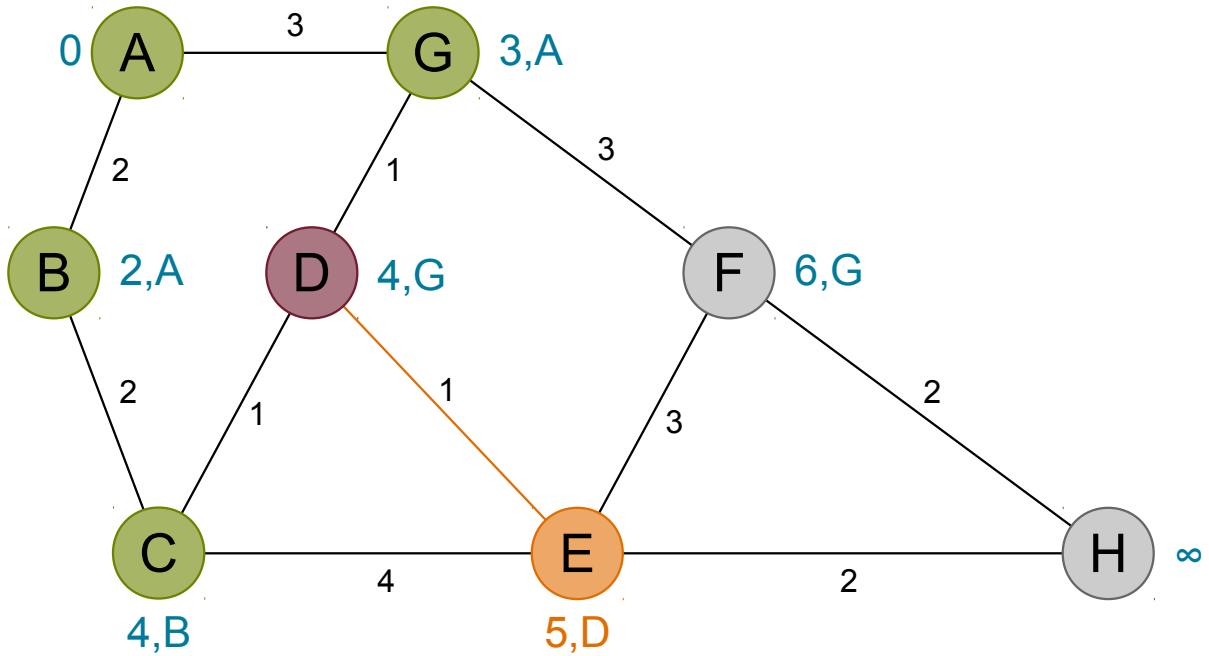


Wähle Knoten mit der geringsten Entfernung: C

Markiere **C** als permanent und betrachte alle Nachbarn

$\Rightarrow$ D bleibt, da nicht besser (5 vs. 4); E(8,C)

3 b) Dijkstras Algorithmus

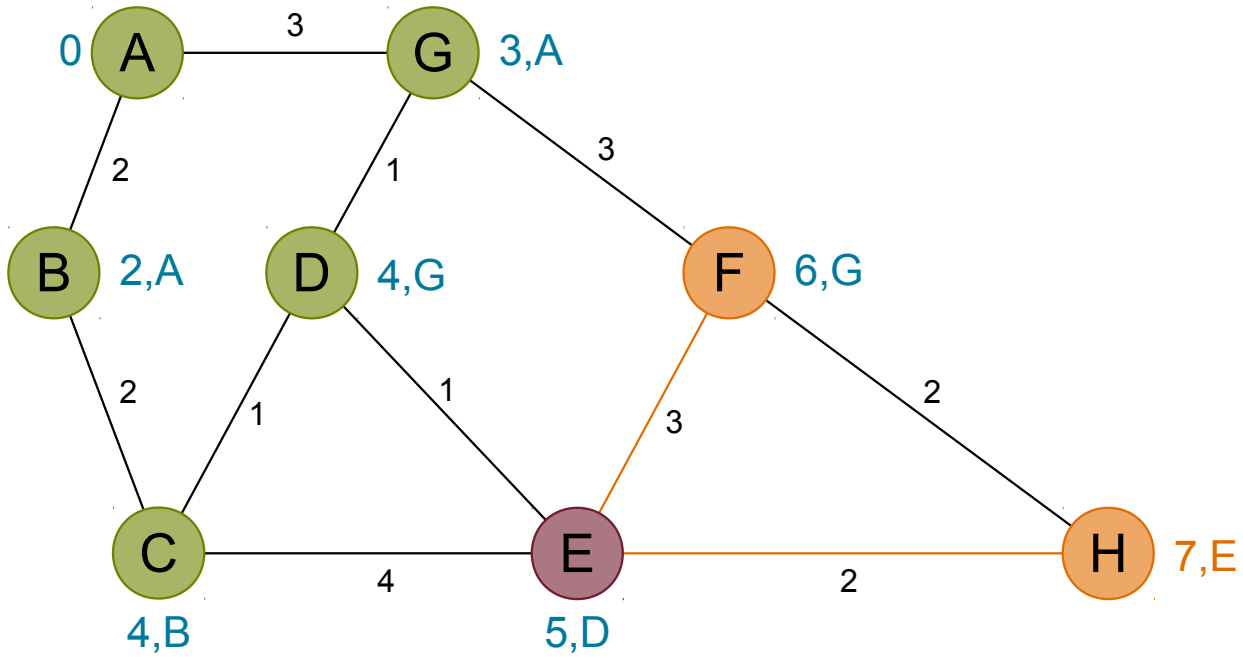


Wähle Knoten mit der geringsten Entfernung: D

Markiere **D** als permanent und betrachte alle Nachbarn

$\Rightarrow$ E geändert, da besser (5 vs. 8)

3 b) Dijkstras Algorithmus

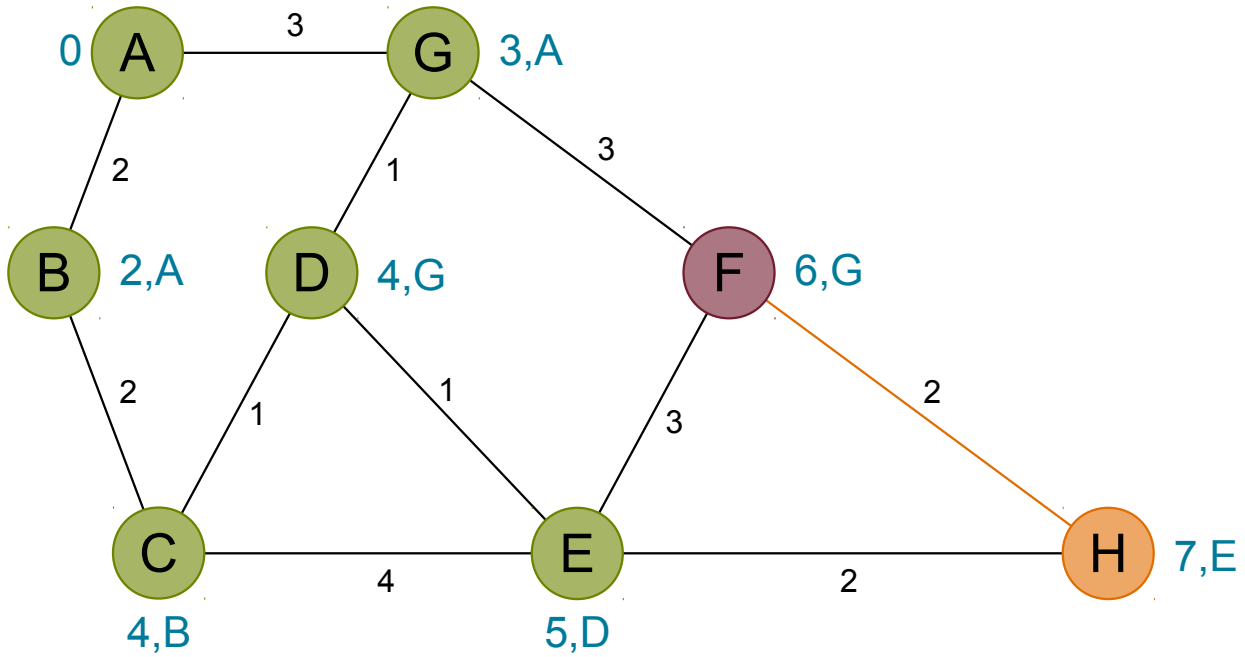


Wähle Knoten mit der geringsten Entfernung: E

Markiere **E** als permanent und betrachte alle Nachbarn

$\Rightarrow$ F bleibt, da nicht besser (8 vs. 6); H(7,E)

3 b) Dijkstras Algorithmus

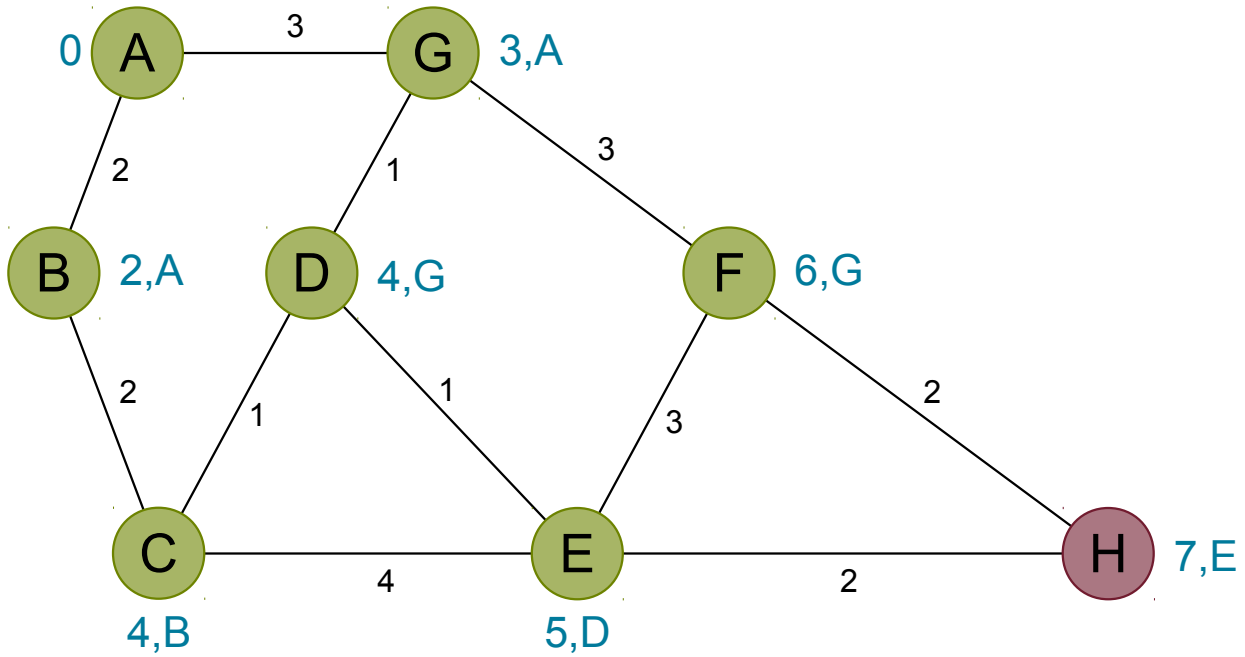


Wähle Knoten mit der geringsten Entfernung: F

Markiere **F** als permanent und betrachte alle Nachbarn

$\Rightarrow$ H bleibt, da nicht besser (8 vs. 7)

3 b) Dijkstras Algorithmus



Wähle Knoten mit der geringsten Entfernung: H

Markiere **H** als permanent

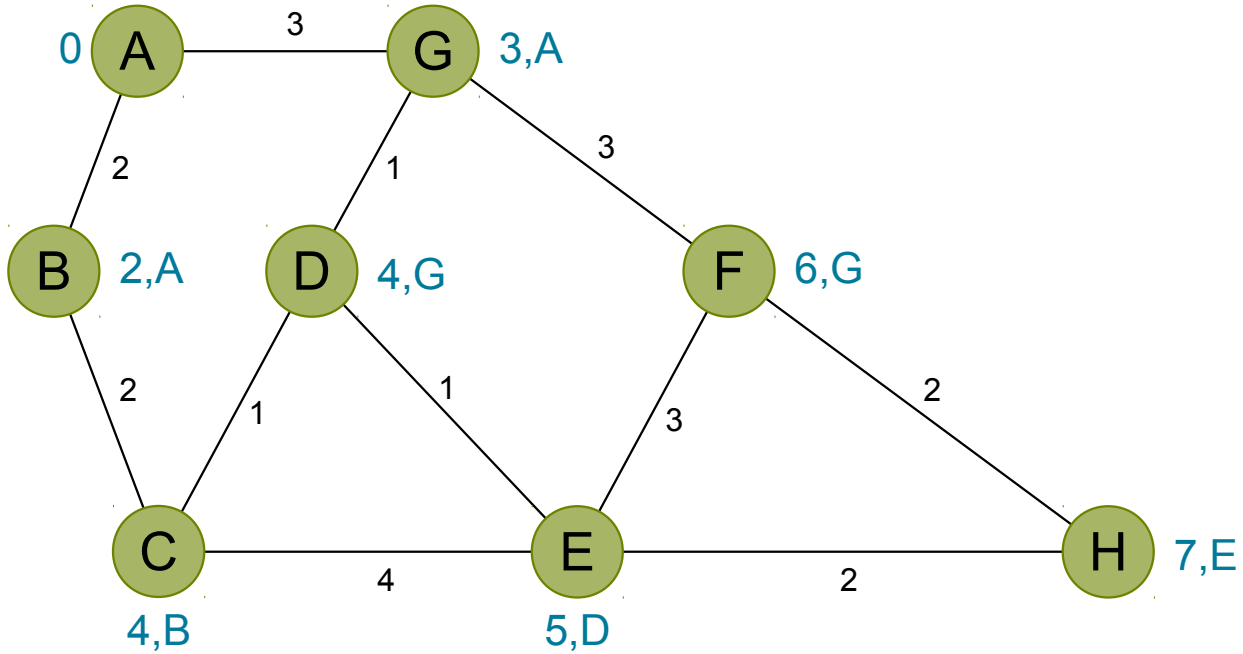
$\Rightarrow$ keine nicht permanenten Nachbarn

3 b) Dijkstras Algorithmus

Lösung

1. A permanent $\Rightarrow$ B(2,A), G(3,A)
2. B permanent $\Rightarrow$ C(4,B)
3. G permanent $\Rightarrow$ D(4,G), F(6,G)
4. C permanent $\Rightarrow$ D bleibt, da nicht besser (5 vs. 4); E(8,C)
5. D permanent $\Rightarrow$ E geändert, da besser (5 vs. 8)
6. E permanent $\Rightarrow$ F bleibt, da nicht besser (8 vs. 6); H(7,E)
7. F permanent $\Rightarrow$ H bleibt, da nicht besser (8 vs. 7)
8. H permanent

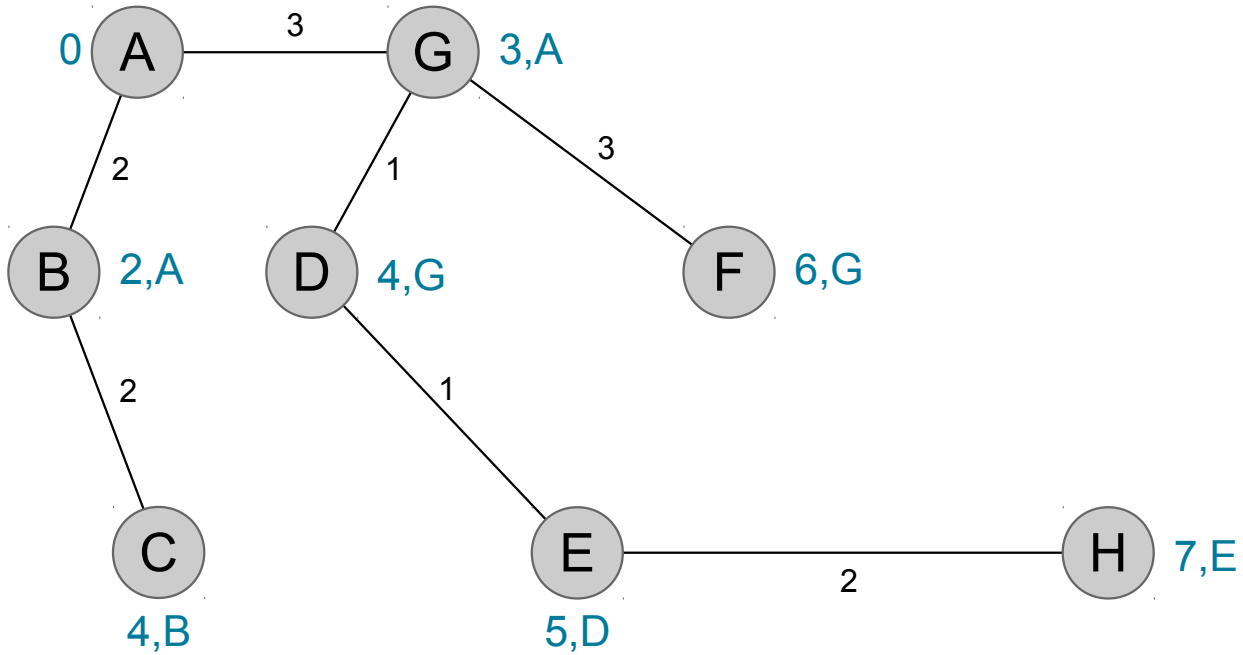
3 b) Dijkstras Algorithmus



Lösung

Alle Knoten sind permanent

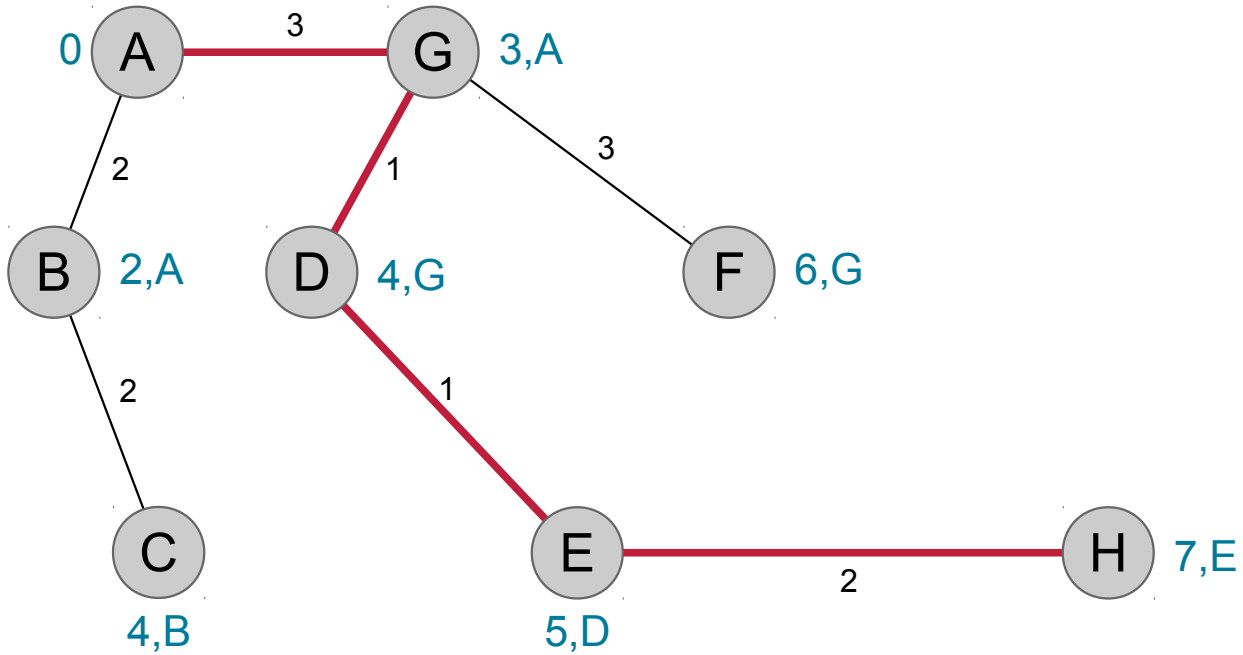
3 b) Dijkstras Algorithmus



Lösung

Shortest-Path-Tree für Router A

3 b) Dijkstras Algorithmus

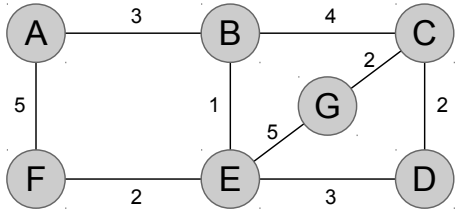


Lösung

Kürzeste Route von Router A zu Router H

Aufgabe 4: Link State Routing

Für das folgende Netz soll nun Link State Routing eingesetzt werden.

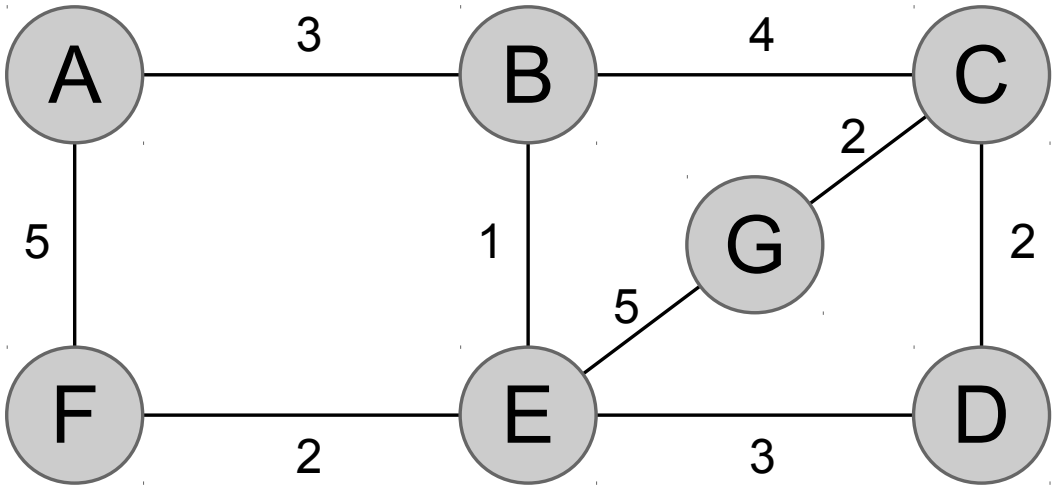


a) Beschreiben Sie die fünf Schritte des Verfahrens im Detail. Stellen Sie die Informationen, die zwischen den Routern ausgetauscht werden für die Router A, B und C dar.

b) Erläutern Sie, warum Link State Routing nicht für sehr große Netze eingesetzt werden sollte.

c) Berechnen Sie die im vierten Schritt insgesamt versendete Datenmenge, wenn das Netz aus 1000 Knoten besteht, jeder Knoten 10 Nachbarn hat und pro Nachbar 10 Byte Informationen verschickt werden müssen.

4 a) Schritte des Verfahrens

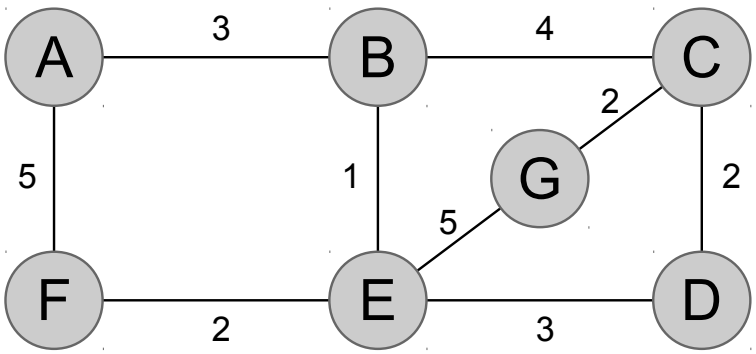


a) Beschreiben Sie die fünf Schritte des Verfahrens im Detail. Stellen Sie die Informationen, die zwischen den Routern ausgetauscht werden für die Router A, B und C dar.

4 a) Schritt 1

Schritt 1

Hello-Nachrichten versenden um benachbarte Systeme zu finden



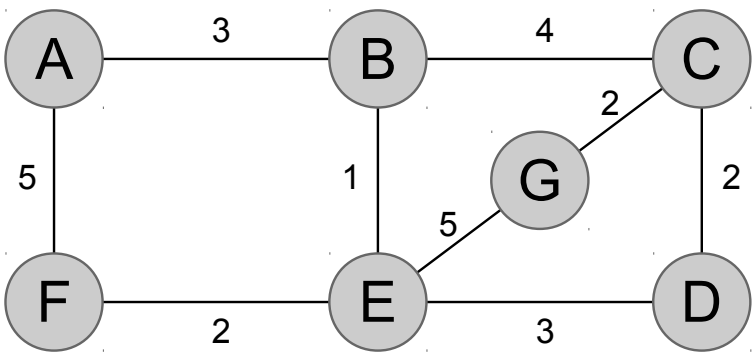
Hello-Nachrichten

- A sendet an B und F
- B sendet an A, C und E
- C sendet an B, D und G
- D sendet an C und E
- E sendet an B, D, F und G
- F sendet an A und E
- G sendet an C und E

4 a) Schritt 2

Schritt 2

Distanz bestimmen (Realität: meist Delay-Messungen (ICMP-Pakete), Round-Trip-Time (RTT) halbieren, symmetrischer Link angenommen)



Distanzen (vorgegeben)

A (B,3), (F,5)

B (A,3), (C,4), (E,1)

C (B,4), (D,2), (G,2)

D (C,2), (E,3)

E (B,1), (D,3), (F,2), (G,5)

F (A,5), (E,2)

G (C,2), (E,5)

4 a) Schritt 3

Schritt 3

Link-State-Pakete erstellen (periodisch, bei Änderungen werden Extra-Pakete versendet)

Pakete der Router A, B und C

A	
1	
13:00	
B	3
F	5

B	
1	
13:01	
A	3
C	4
E	1

C	
1	
13:02	
B	4
D	2
G	2

- Paket enthält dabei Absender, Sequenznummer, Alter und Distanzen

4 a) Schritt 4

Schritt 4

Pakete per globalem Broadcast (Flooding) versenden

4 a) Schritt 5

Schritt 5

Neues Routing berechnen (z.B. Shortest Path)

4 b) Skalierbarkeit

b) Erläutern Sie, warum Link State Routing nicht für sehr große Netze eingesetzt werden sollte.

Schritt 4

Pakete per globalem Broadcast (Flooding) versenden

Lösung

Durch das verwendete Flooding werden sehr große Datenmengen im Netz übertragen (siehe Berechnung in Aufgabenteil c).
Außerdem muss jeder Knoten alle Daten speichern und den Shortest Path Tree berechnen.

4 c) Versendete Datenmenge

c) Berechnen Sie die im vierten Schritt insgesamt versendete Datenmenge, wenn das Netz aus 1000 Knoten besteht, jeder Knoten 10 Nachbarn hat und pro Nachbar 10 Byte Informationen verschickt werden müssen.

Lösung

- $K = 1000$, $N = 10$, $L = 10$ Byte (Adressen + Distanzen)

$\Rightarrow$ 1 Knoten sendet an 10 Nachbarn $\Rightarrow$ 100 Byte $(N \times L)$

$\Rightarrow$ 1000 Knoten senden an 10 Nachbarn $\Rightarrow$ 100 KByte $(K \times N \times L)$

$\Rightarrow$ jeder leitet jede Nachricht 1x weiter $\Rightarrow$ **100 MByte** $(K^2 \times N \times L)$
(bei K = 10.000 $\Rightarrow$ 10 GByte!)

Hinweis

Die Datenmenge hängt vom verwendeten Broadcast-Algorithmus ab (mehr dazu in Computernetze 2).

Zusammenfassung 1/2

Shortest Path Routing

- Berechnung des Shortest Path Trees per Dijkstra-Algorithmus

Distance Vector Routing

- Identifikation von Nachbarn per Hello-Nachricht
- Berechnung des jeweils nächsten Hops per Bellmann/Ford-Algorithmus
- Count-to-Infinity-Problem möglich!

Zusammenfassung 2/2

Link State Routing

- Verteilung von Link-State-Paketen im Netz
- nur für kleine Netze geeignet
- Berechnung des Shortest Path Trees per Dijkstra-Algorithmus

Flooding

- Verteilung von Paketen im Netz ohne Routing
- Abbruchbedingung muss vorhanden sein!

Nächste Übung
10. Juli 2025