

ALGORITHMEN UND DATENSTRUKTUREN

ÜBUNG 11: KÜRZESTE WEGE

Eric Kunze

`eric.kunze@tu-dresden.de`

TU Dresden, 20. Dezember 2021

letzte Änderung:
20.12.2021, 10:55

Dijkstra-Algorithmus

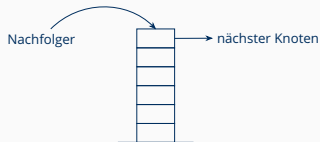
Tiefensuche:

- ▶ gehe in die Tiefe:
„entdecke erst Kinder, dann Geschwister“
- ▶ Nachfolger werden *oben* auf den Keller gelegt
- ▶ nächster Knoten wird *oben* vom Keller genommen

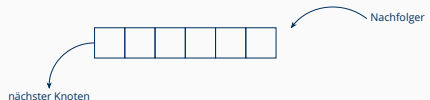
Breitensuche:

- ▶ gehe in die Breite:
„entdecke erst Geschwister, dann Kinder“
- ▶ Nachfolger stellen sich *hinten an*
- ▶ nächster Knoten wird von *vorn* genommen

Keller:



Warteschlange:



Beobachtung: Suche läuft ähnlich ab

- | | |
|--|--------|
| ‣ Operation 1: Lesen des nächsten Knotens | READ |
| ‣ Operation 2: Löschen des gewählten Knotens
(und seiner Duplikate) | REMOVE |
| ‣ Operation 3: Hinzufügen der Nachfolgerknoten | INSERT |
| ‣ Operation 4: Leerheit der Datenstruktur prüfen | EMPTY |

VERALLGEMEINERTE GRAPHENSUCHE

Beobachtung: Suche läuft ähnlich ab

- Operation 1: Lesen des nächsten Knotens READ
- Operation 2: Löschen des gewählten Knotens REMOVE
(und seiner Duplikate)
- Operation 3: Hinzufügen der Nachfolgerknoten INSERT
- Operation 4: Leerheit der Datenstruktur prüfen EMPTY

	STORAGE	READ	REMOVE	INSERT	EMPTY
Tiefensuche	Keller	top	pop	push	empty
Breitensuche	Queue	head	dequeue	enqueue	nil

weitere Möglichkeit für STORAGE: **Prioritätswarteschlange**

- ▶ READ – Auswahl des nächsten Knotens mit minimaler Priorität
- ▶ REMOVE – as usual
- ▶ INSERT – Nachfolger stellt sich entsprechend seiner Priorität in die Warteschlange
(oder Prioritätswert erhält ein Update, wenn er bereits in der Warteschlange steht)

Vorstellung: „geordnete“ Warteschlange

DIJKSTRA-ALGORITHMUS

Graphensuche mit Prioritätswarteschlange:

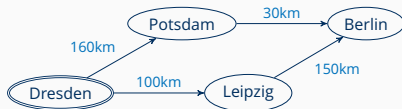
Priorität = Priorität des Vorgängers + Kantengewicht

DIJKSTRA-ALGORITHMUS

Graphensuche mit Prioritätswarteschlange:

Priorität = Priorität des Vorgängers + Kantengewicht

Beispiel:



Wir notieren Knoten in der Form (Knoten, Priorität, Vorgänger).

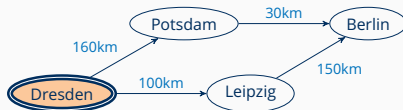
gewählter Knoten	Warteschlange

DIJKSTRA-ALGORITHMUS

Graphensuche mit Prioritätswarteschlange:

Priorität = Priorität des Vorgängers + Kantengewicht

Beispiel:



Wir notieren Knoten in der Form (Knoten, Priorität, Vorgänger).

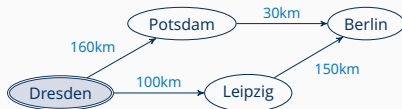
gewählter Knoten	Warteschlange
(Dresden, 0 km, -)	

DIJKSTRA-ALGORITHMUS

Graphensuche mit Prioritätswarteschlange:

Priorität = Priorität des Vorgängers + Kantengewicht

Beispiel:



Wir notieren Knoten in der Form (Knoten, Priorität, Vorgänger).

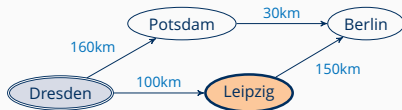
gewählter Knoten	Warteschlange
(Dresden, 0 km, -)	(Leipzig, 0 + 100 km, Dresden), (Potsdam, 0 + 160 km, Dresden)

DIJKSTRA-ALGORITHMUS

Graphensuche mit Prioritätswarteschlange:

Priorität = Priorität des Vorgängers + Kantengewicht

Beispiel:



Wir notieren Knoten in der Form (Knoten, Priorität, Vorgänger).

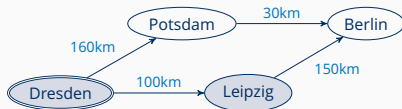
gewählter Knoten	Warteschlange
(Dresden, 0 km, -)	(Leipzig, 0 + 100 km, Dresden), (Potsdam, 0 + 160 km, Dresden)
(Leipzig, 100 km, Dresden)	

DIJKSTRA-ALGORITHMUS

Graphensuche mit Prioritätswarteschlange:

Priorität = Priorität des Vorgängers + Kantengewicht

Beispiel:



Wir notieren Knoten in der Form (Knoten, Priorität, Vorgänger).

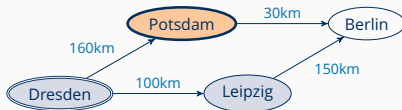
gewählter Knoten	Warteschlange
(Dresden, 0 km, -)	(Leipzig, 0 + 100 km, Dresden), (Potsdam, 0 + 160 km, Dresden)
(Leipzig, 100 km, Dresden)	(Potsdam, 160 km, Dresden), (Berlin, 100 + 150 km, Leipzig)

DIJKSTRA-ALGORITHMUS

Graphensuche mit Prioritätswarteschlange:

Priorität = Priorität des Vorgängers + Kantengewicht

Beispiel:



Wir notieren Knoten in der Form (Knoten, Priorität, Vorgänger).

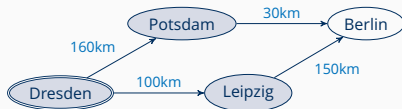
gewählter Knoten	Warteschlange
(Dresden, 0 km, -)	(Leipzig, 0 + 100 km, Dresden), (Potsdam, 0 + 160 km, Dresden)
(Leipzig, 100 km, Dresden)	(Potsdam, 160 km, Dresden), (Berlin, 100 + 150 km, Leipzig)
(Potsdam, 160 km, Dresden)	

DIJKSTRA-ALGORITHMUS

Graphensuche mit Prioritätswarteschlange:

Priorität = Priorität des Vorgängers + Kantengewicht

Beispiel:



Wir notieren Knoten in der Form (Knoten, Priorität, Vorgänger).

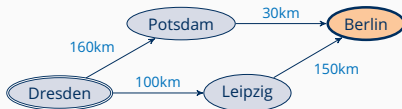
gewählter Knoten	Warteschlange
(Dresden, 0 km, -)	(Leipzig, 0 + 100 km, Dresden), (Potsdam, 0 + 160 km, Dresden)
(Leipzig, 100 km, Dresden)	(Potsdam, 160 km, Dresden), (Berlin, 100 + 150 km, Leipzig)
(Potsdam, 160 km, Dresden)	(Berlin, 160 + 30 km, Potsdam)

DIJKSTRA-ALGORITHMUS

Graphensuche mit Prioritätswarteschlange:

Priorität = Priorität des Vorgängers + Kantengewicht

Beispiel:



Wir notieren Knoten in der Form (Knoten, Priorität, Vorgänger).

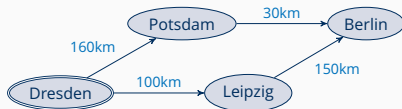
gewählter Knoten	Warteschlange
(Dresden, 0 km, -)	(Leipzig, 0 + 100 km, Dresden), (Potsdam, 0 + 160 km, Dresden)
(Leipzig, 100 km, Dresden)	(Potsdam, 160 km, Dresden), (Berlin, 100 + 150 km, Leipzig)
(Potsdam, 160 km, Dresden)	(Berlin, 160 + 30 km, Potsdam)
(Berlin, 190 km, Potsdam)	

DIJKSTRA-ALGORITHMUS

Graphensuche mit Prioritätswarteschlange:

Priorität = Priorität des Vorgängers + Kantengewicht

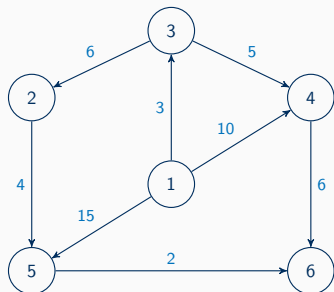
Beispiel:



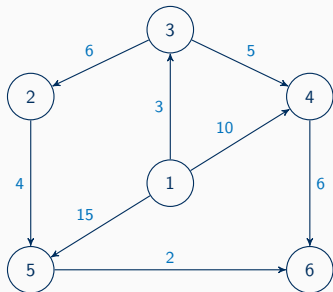
Wir notieren Knoten in der Form (Knoten, Priorität, Vorgänger).

gewählter Knoten	Warteschlange
(Dresden, 0 km, -)	(Leipzig, 0 + 100 km, Dresden), (Potsdam, 0 + 160 km, Dresden)
(Leipzig, 100 km, Dresden)	(Potsdam, 160 km, Dresden), (Berlin, 100 + 150 km, Leipzig)
(Potsdam, 160 km, Dresden)	(Berlin, 160 + 30 km, Potsdam)
(Berlin, 190 km, Potsdam)	—

AUFGABE 1

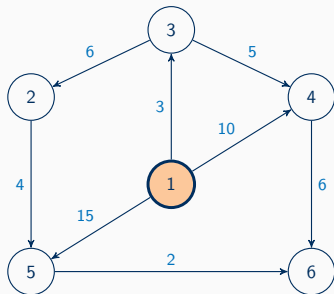


AUFGABE 1



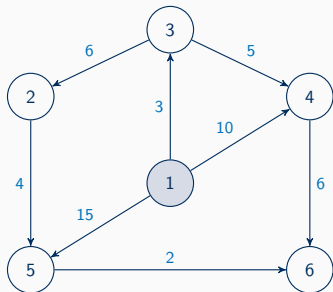
gewählter Knoten	Randknotenmenge

AUFGABE 1



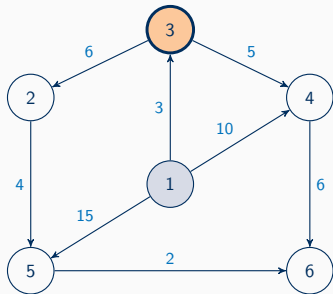
gewählter Knoten	Randknotenmenge
(1, 0, --)	

AUFGABE 1



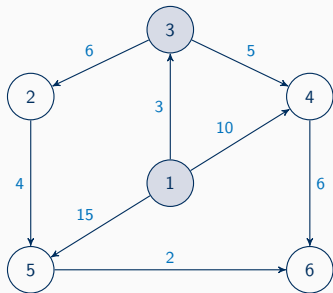
gewählter Knoten	Randknotenmenge
(1, 0, --)	(3, 3, 1), (4, 10, 1), (5, 15, 1)

AUFGABE 1



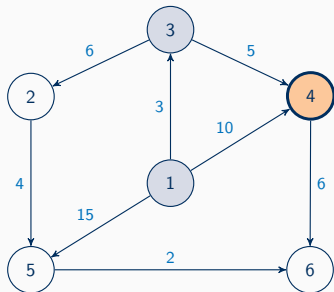
gewählter Knoten	Randknotenmenge
$(1, 0, --)$	$(3, 3, 1), (4, 10, 1), (5, 15, 1)$
$(3, 3, 1)$	

AUFGABE 1



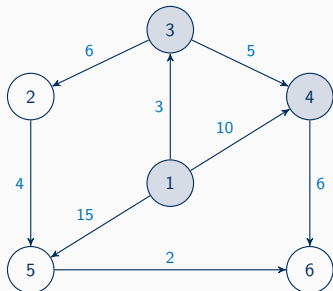
gewählter Knoten	Randknotenmenge
(1, 0, --)	(3, 3, 1), (4, 10, 1), (5, 15, 1)
(3, 3, 1)	(2, 9, 3), (4, 8, 3), (5, 15, 1)

AUFGABE 1



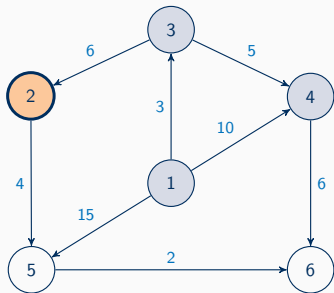
gewählter Knoten	Randknotenmenge
(1, 0, --)	(3, 3, 1), (4, 10, 1), (5, 15, 1)
(3, 3, 1)	(2, 9, 3), (4, 8, 3), (5, 15, 1)
(4, 8, 3)	

AUFGABE 1



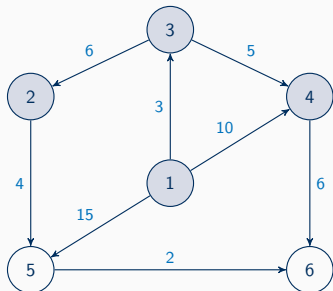
gewählter Knoten	Randknotenmenge
(1, 0, --)	(3, 3, 1), (4, 10, 1), (5, 15, 1)
(3, 3, 1)	(2, 9, 3), (4, 8, 3), (5, 15, 1)
(4, 8, 3)	(2, 9, 3), (5, 15, 1), (6, 14, 4)

AUFGABE 1



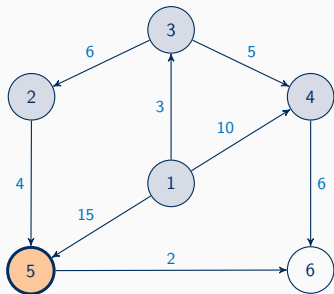
gewählter Knoten	Randknotenmenge
(1, 0, --)	(3, 3, 1), (4, 10, 1), (5, 15, 1)
(3, 3, 1)	(2, 9, 3), (4, 8, 3), (5, 15, 1)
(4, 8, 3)	(2, 9, 3), (5, 15, 1), (6, 14, 4)
(2, 9, 3)	

AUFGABE 1



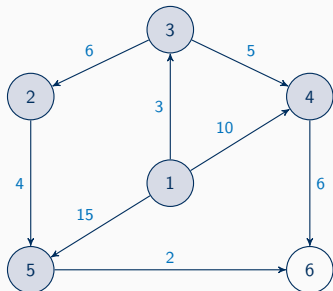
gewählter Knoten	Randknotenmenge
(1, 0, --)	(3, 3, 1), (4, 10, 1), (5, 15, 1)
(3, 3, 1)	(2, 9, 3), (4, 8, 3), (5, 15, 1)
(4, 8, 3)	(2, 9, 3), (5, 15, 1), (6, 14, 4)
(2, 9, 3)	(5, 13, 2), (6, 14, 4)

AUFGABE 1



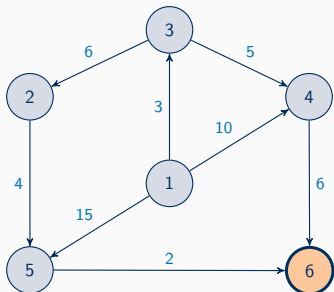
gewählter Knoten	Randknotenmenge
(1, 0, --)	(3, 3, 1), (4, 10, 1), (5, 15, 1)
(3, 3, 1)	(2, 9, 3), (4, 8, 3), (5, 15, 1)
(4, 8, 3)	(2, 9, 3), (5, 15, 1), (6, 14, 4)
(2, 9, 3)	(5, 13, 2), (6, 14, 4)
(5, 13, 2)	

AUFGABE 1



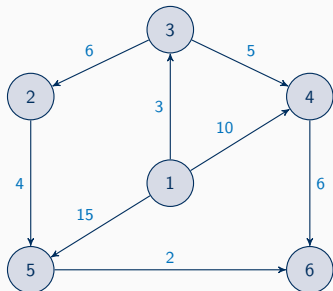
gewählter Knoten	Randknotenmenge
(1, 0, --)	(3, 3, 1), (4, 10, 1), (5, 15, 1)
(3, 3, 1)	(2, 9, 3), (4, 8, 3), (5, 15, 1)
(4, 8, 3)	(2, 9, 3), (5, 15, 1), (6, 14, 4)
(2, 9, 3)	(5, 13, 2), (6, 14, 4)
(5, 13, 2)	(6, 14, 4)

AUFGABE 1



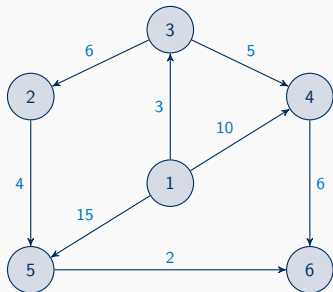
gewählter Knoten	Randknotenmenge
$(1, 0, --)$	$(3, 3, 1), (4, 10, 1), (5, 15, 1)$
$(3, 3, 1)$	$(2, 9, 3), (4, 8, 3), (5, 15, 1)$
$(4, 8, 3)$	$(2, 9, 3), (5, 15, 1), (6, 14, 4)$
$(2, 9, 3)$	$(5, 13, 2), (6, 14, 4)$
$(5, 13, 2)$	$(6, 14, 4)$
$(6, 14, 4)$	

AUFGABE 1



gewählter Knoten	Randknotenmenge
$(1, 0, --)$	$(3, 3, 1), (4, 10, 1), (5, 15, 1)$
$(3, 3, 1)$	$(2, 9, 3), (4, 8, 3), (5, 15, 1)$
$(4, 8, 3)$	$(2, 9, 3), (5, 15, 1), (6, 14, 4)$
$(2, 9, 3)$	$(5, 13, 2), (6, 14, 4)$
$(5, 13, 2)$	$(6, 14, 4)$
$(6, 14, 4)$	—

AUFGABE 1



gewählter Knoten	Randknotenmenge
(1, 0, --)	(3, 3, 1), (4, 10, 1), (5, 15, 1)
(3, 3, 1)	(2, 9, 3), (4, 8, 3), (5, 15, 1)
(4, 8, 3)	(2, 9, 3), (5, 15, 1), (6, 14, 4)
(2, 9, 3)	(5, 13, 2), (6, 14, 4)
(5, 13, 2)	(6, 14, 4)
(6, 14, 4)	—

Pfadtable:

Zielknoten	Pfadlänge	kürzester Pfad
1	0	1
2	9	1, 3, 2
3	3	1, 3
4	8	1, 3, 4
5	13	1, 3, 2, 5
6	13	1, 3, 4, 6

Floyd-Warshall-Algorithmus

FLOYD-WARSHALL-ALGORITHMUS

- gewichteter Graph $G = (V, E, c)$ mit Weglängen c und ohne Schlingen
- **Ziel:** kürzeste Wege von beliebigem Startknoten zu beliebigem Zielknoten
- oBdA: $V = \{1, \dots, n\}$

FLOYD-WARSHALL-ALGORITHMUS

- ▶ gewichteter Graph $G = (V, E, c)$ mit Weglängen c und ohne Schlingen
- ▶ **Ziel:** kürzeste Wege von beliebigem Startknoten zu beliebigem Zielknoten
- ▶ oBdA: $V = \{1, \dots, n\}$
- ▶ $P_{u,v}$ = Menge aller Wege von u nach v
- ▶
$$D_G(u, v) = \begin{cases} \min \{c_p : p \in P_{u,v}\} & \text{wenn } P_{u,v} \neq \emptyset \\ \infty & \text{sonst} \end{cases}$$

FLOYD-WARSHALL-ALGORITHMUS

- ▶ gewichteter Graph $G = (V, E, c)$ mit Weglängen c und ohne Schlingen
- ▶ **Ziel:** kürzeste Wege von beliebigem Startknoten zu beliebigem Zielknoten
- ▶ oBdA: $V = \{1, \dots, n\}$
- ▶ $P_{u,v}$ = Menge aller Wege von u nach v
- ▶
$$D_G(u, v) = \begin{cases} \min \{c_p : p \in P_{u,v}\} & \text{wenn } P_{u,v} \neq \emptyset \\ \infty & \text{sonst} \end{cases}$$
- ▶ $P_{u,v}^{(k)}$ = Menge aller Wege von u nach v , deren innere Knoten in $\{1, \dots, k\}$ liegen
- ▶
$$D_G^{(k)}(u, v) = \begin{cases} \min \{c_p : p \in P_{u,v}^{(k)}\} & \text{wenn } P_{u,v}^{(k)} \neq \emptyset \\ \infty & \text{sonst} \end{cases}$$

FLOYD-WARSHALL-ALGORITHMUS

- ▶ gewichteter Graph $G = (V, E, c)$ mit Weglängen c und ohne Schlingen
- ▶ **Ziel:** kürzeste Wege von beliebigem Startknoten zu beliebigem Zielknoten
- ▶ oBdA: $V = \{1, \dots, n\}$
- ▶ $P_{u,v}$ = Menge aller Wege von u nach v
- ▶
$$D_G(u, v) = \begin{cases} \min \{c_p : p \in P_{u,v}\} & \text{wenn } P_{u,v} \neq \emptyset \\ \infty & \text{sonst} \end{cases}$$
- ▶ $P_{u,v}^{(k)}$ = Menge aller Wege von u nach v , deren innere Knoten in $\{1, \dots, k\}$ liegen
- ▶
$$D_G^{(k)}(u, v) = \begin{cases} \min \{c_p : p \in P_{u,v}^{(k)}\} & \text{wenn } P_{u,v}^{(k)} \neq \emptyset \\ \infty & \text{sonst} \end{cases}$$
- ▶ Es gilt $P_{u,v}^{(n)} = P_{u,v}$ und somit $D_G^{(n)} = D_G$.

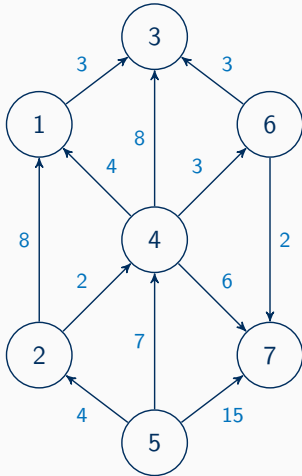
- ▶ modifizierte Adjazenzmatrix

$$mA_G = \min \{A_G, \mathbf{0}_n\} = \begin{cases} c(u, v) & \text{wenn } u \neq v, (u, v) \in E \\ 0 & \text{wenn } u = v \\ \infty & \text{sonst} \end{cases}$$

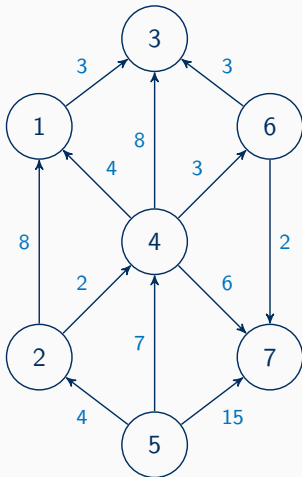
- ▶ **Initialisierung:** $D_G^{(0)} = mA_G$
- ▶ **Rekursion:**

$$D_G^{(k+1)}(u, v) = \min \left\{ D_G^{(k)}(u, v), D_G^{(k)}(u, k+1) + D_G^{(k)}(k+1, v) \right\}$$

AUFGABE 2



AUFGABE 2 — TEIL (A)



$$mA_G = \begin{pmatrix} 0 & \infty & 3 & \infty & \infty & \infty & \infty \\ 8 & 0 & \infty & 2 & \infty & \infty & \infty \\ \infty & \infty & 0 & \infty & \infty & \infty & \infty \\ 4 & \infty & 8 & 0 & \infty & 3 & 6 \\ \infty & 4 & \infty & 7 & 0 & \infty & 15 \\ \infty & \infty & 3 & \infty & \infty & 0 & 2 \\ \infty & \infty & \infty & \infty & \infty & \infty & 0 \end{pmatrix}$$

AUFGABE 2 — TEIL (B)

$$D_G^{(2)} = \begin{pmatrix} 0 & \infty & 3 & \infty & \infty & \infty & \infty \\ 8 & 0 & 11 & 2 & \infty & \infty & \infty \\ \infty & \infty & 0 & \infty & \infty & \infty & \infty \\ 4 & \infty & 7 & 0 & \infty & 3 & 6 \\ 12 & 4 & 15 & 6 & 0 & \infty & 15 \\ \infty & \infty & 3 & \infty & \infty & 0 & 2 \\ \infty & \infty & \infty & \infty & \infty & \infty & 0 \end{pmatrix}$$

d.h. es ändern sich folgende Einträge:

$$\underbrace{(4, 3, 7), (2, 3, 11)}_{\text{aus } D_G^{(1)}}, \underbrace{(5, 3, 15), (5, 1, 12), (5, 4, 6)}_{\text{aus } D_G^{(2)}}$$

AUFGABE 2 — TEIL (C)

Für $k \in \{4, 6\}$, d.h. durch Zulassen der Knoten 4 und 6 als innere Knoten eines Weges, erreichen wir eine Verbesserung. Dagegen sind die Knoten 3 und 7 *Senken*, d.h. es bringt nichts, diese zu besuchen, weil wir nicht wieder weg kommen. Ebenso bringt uns der Knoten 5 als *Quelle* keine Verbesserung, weil wir diesen gar nicht erreichen können. Somit gilt also

$$D_G^{(2)} = D_G^{(3)} \quad D_G^{(4)} = D_G^{(5)} \quad D_G^{(6)} = D_G^{(7)}$$

und wir müssen lediglich $D_G^{(4)}$ sowie $D_G^{(6)}$ explizit berechnen.

AUFGABE 2 — TEIL (D)

$$D_G^{(4)} = \begin{pmatrix} 0 & \infty & 3 & \infty & \infty & \infty & \infty \\ 6 & 0 & 9 & 2 & \infty & 5 & 8 \\ \infty & \infty & 0 & \infty & \infty & \infty & \infty \\ 4 & \infty & 11 & 0 & \infty & 3 & 6 \\ 10 & 4 & 13 & 6 & 0 & 9 & 12 \\ \infty & \infty & 3 & \infty & \infty & 0 & 2 \\ \infty & \infty & \infty & \infty & \infty & \infty & 0 \end{pmatrix} = D_G^{(5)}$$

$$D_G^{(6)} = \begin{pmatrix} 0 & \infty & 3 & \infty & \infty & \infty & \infty \\ 6 & 0 & 8 & 2 & \infty & 5 & 7 \\ \infty & \infty & 0 & \infty & \infty & \infty & \infty \\ 4 & \infty & 6 & 0 & \infty & 3 & 5 \\ 10 & 4 & 12 & 6 & 0 & 9 & 11 \\ \infty & \infty & 3 & \infty & \infty & 0 & 2 \\ \infty & \infty & \infty & \infty & \infty & \infty & 0 \end{pmatrix} = D_G^{(7)} = D_G$$