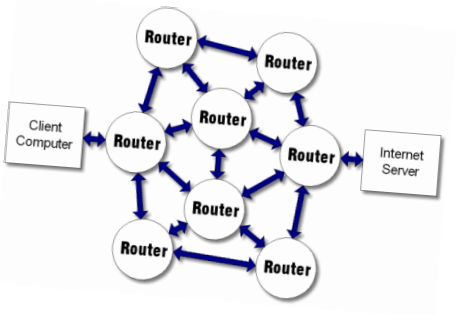
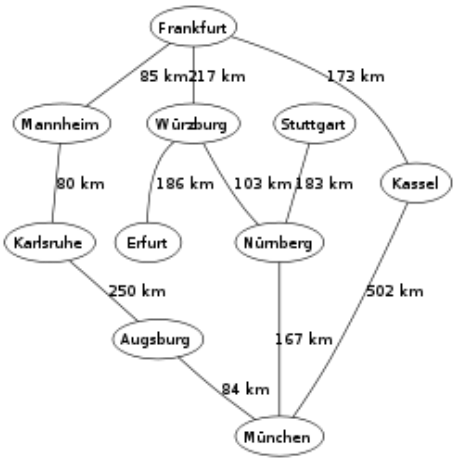
Routing

in Rechnernetzen

- wo geht's lang?





Reist eine Person im Alltag von A nach B, so wird sie sich automatisch fragen, mit welchen Transportmitteln (Auto, Bus, Bahn, …) sie diese Strecke zurücklegen will. Hinzu kommt stets die Überlegung nach der Optimalität bzgl. Kosten, Zeit, Gefahr, etc.

Dieses Verhalten ist in der Computerwelt nicht anders. In der heutigen vernetzten digitalen Welt müssen Datenpakete in einem komplizierten Geflecht von verschiedenen Datenkanälen (Telefon-, Koaxial- und Glasfaserleitungen sowie Funk- und Richtstrahlstrecken) ihren Weg finden.

Diese Wegbereitung und die Optimierung (bzgl. der Netzauslastung/ der Netzstabilität/ der Laufzeit/ der Kosten/ der Verfügbarkeit von Routern/ etc.) für die Auswahl des Weges ist zumeist die Aufgabe der Netzwerkbetreiber. Sie garantieren für eine zuverlässige, kostengünstige und zeitlich vertretbare Übermittlung der Datenpakete. Wie diese Wege ausgewählt werden (können), lässt sich anhand so genannter **Routing-Algorithmen für dynamisches Routing** veranschaulichen (statisches Routing bedeutet im Gegensatz dazu einfach manuelles Konfigurieren). Im Allgemeinen generieren Routing-Algorithmen geeignete Werte – so genannte Metriken – die auf den Eigenschaften basieren, die das zugehörige Routing-Protokoll als optimal (vgl. oben) ansieht und die die Router in ihre Routing-Tabellen übernehmen. Letztere könnte z.B. wie folgt aussehen.

Netzwerkziel	Netzwerkmaske	Gateway	Metrik
0.0.0.0	0.0.0.0	192.168.0.1	100
127.0.0.0	255.0.0.0	127.0.0.1	1
192.168.0.0	255.255.255.0	192.168.0.10	100
192.168.0.10	255.255.255.255	127.0.0.1	100

Das **Netzwerkziel** steht dabei für das Zielnetzwerk, die **Netzwerkmaske** definiert die Größe dieses Zielnetzwerks und das **Gateway** steht für die Adresse des Rechners, über die das Zielnetzwerk zu erreichen ist. Im Einzelnen liest sich die Routing-Tabelle wie folgt:

- Das Netzwerk 0.0.0.0 mit der Subnetzmaske 0.0.0.0 steht bezeichnend für den gesamten IP-Adressraum, also quasi für alle IP-Adressen im Internet. Das Gateway, das für das Netzwerkziel 0.0.0.0/0.0.0.0 zuständig ist, ist für gewöhnlich das Standard-Gateway.
- Das Netzwerk 127.0.0.0/8 ist ein im Internet reservierter IP-Adressbereich, der immer für den lokalen Rechner reserviert ist. Erkennbar ist das auch daran, dass als Gateway für dieses Netzwerkziel die 127.0.0.1 angegeben ist, also unser Rechner selbst.
- Das Netzwerk 192.168.0.0/255.255.255.0 ist das Netzwerk, aus dem die IP-Adresse unseres Rechners stammt. Da wir zum Erreichen von Zieladressen deshalb kein Standard-Gateway benötigen, ist für dieses Netzwerkziel unser Rechner selbst das Gateway – 192.168.0.10.
- Mit dem Netzwerkziel 192.168.0.10/255.255.255.255 (mit dieser Subnetzmaske ist also nur die einzelne IP-Adresse 192.168.0.10 als Ziel gemeint) ist schließlich unser Rechner selbst gemeint, es wird hier deshalb als Gateway auf die 127.0.0.1 geroutet.

Nun aber zu den angesprochenen (, heiß erwarteten) Algorithmen.

Routingalgorithmen

(aus der Graphentheorie)

Dijkstra & Bellman-Ford

Im Folgenden werden die zwei wichtigsten Routingalgorithmen (Dijkstra-/ Bellman-Ford-Algorithmus) – anlehnend an die Internetpräsenz „studyfix“ – vorgestellt und anhand einfacher Beispiele verdeutlicht.
Dabei ist zu beachten, dass beim Dijkstra-Algorithmus die so genannten Kantengewichte des Graphen (d.h. die Kosten, um von einem Punkt zum nächsten zu kommen) nicht negativ sein dürfen. Der Bellman-Ford-Algorithmus hingegen funktioniert allgemein.

Dijkstra-Algorithmus
(„folge dem aktuell kürzesten Weg“)

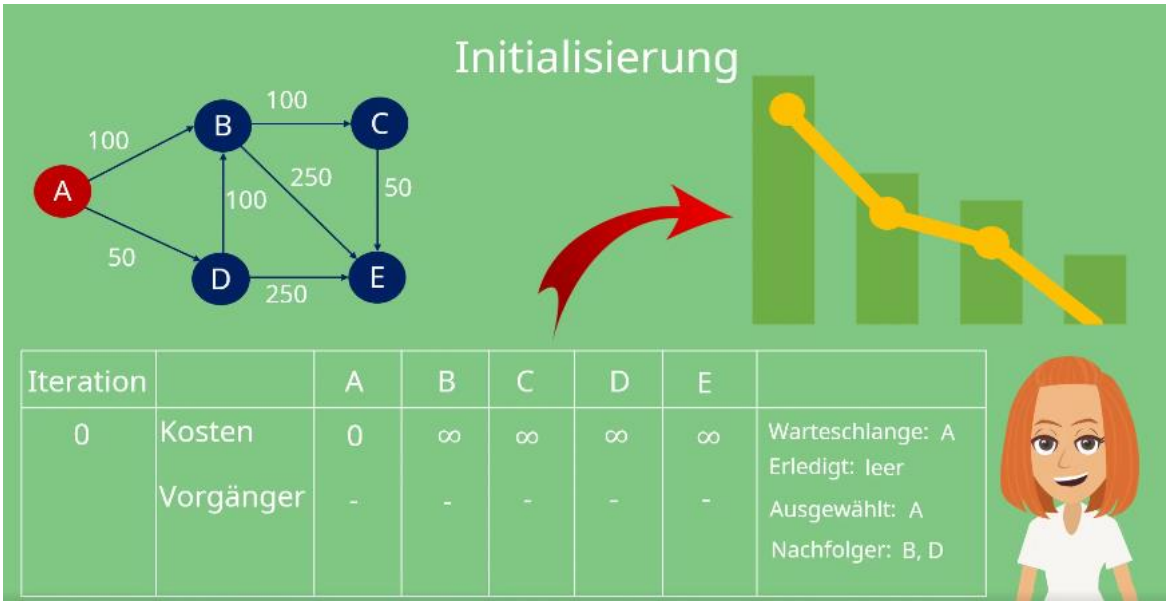
Pseudocode (Implementations-nah, Definitionen implizit enthalten):

Initialisierung
Warteschlange W = Startknoten
Erledigte Knoten E = Ø
Kosten festsetzen - Startknoten = 0, andere Knoten = ∞

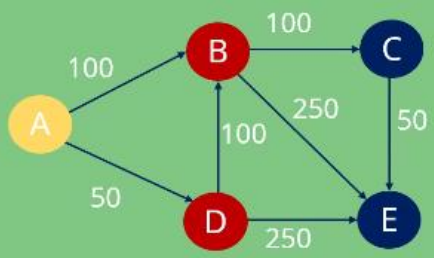
Iterationen
solange W ≠ Ø
wähle Knoten k aus W mit den geringsten Kosten
wiederhole für alle Nachfolger j von k, die nicht in E sind
berechne neue Kosten zu j über k
falls neue Kosten geringer sind als aktuelle
setze Kosten zu j = neue Kosten
setze Vorgänger von j = k
füge j zu W hinzu
entferne k aus W
füge k zu E hinzu

[Algorithmus endet, wenn W = Ø]

Beispiel (Anwendung des Pseudocodes, tabellarisches Erfassen wichtiger Zwischenschritte)

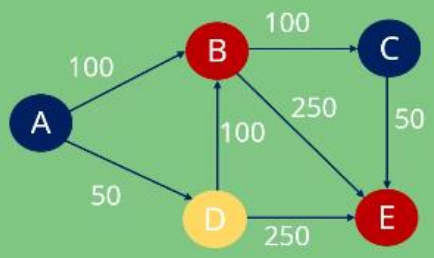


Iteration 1



Iteration		A	B	C	D	E	
1	Kosten	0	100	∞	50	∞	Warteschlange: B, D Erledigt: A
	Vorgänger	-	A	-	A	-	Ausgewählt: D Nachfolger: B, E

Iteration 2



Iteration		A	B	C	D	E	
2	Kosten	0	100	∞	50	300	Warteschlange: B, E Erledigt: A, D
	Vorgänger	-	A	-	A	D	Ausgewählt: B Nachfolger: C, E

⋮

Iteration 5

Iteration		A	B	C	D	E	
0	Kosten	0	∞	∞	∞	∞	
	Vorgänger	-	-	-	-	-	
1	Kosten	0	100	∞	50	∞	
	Vorgänger	-	A	-	A	-	
2	Kosten	0	100	∞	50	300	
	Vorgänger	-	A	-	A	D	
3	Kosten	0	100	200	50	300	
	Vorgänger	-	A	B	A	D	
4	Kosten	0	100	200	50	250	
	Vorgänger	-	A	B	A	C	
5	Kosten	0	100	200	50	250	Warteschlange: leer
	Vorgänger	-	A	B	A	C	Erledigt: A, B, C, D, E

In der letzten Zeile der Tabelle sind schließlich schön übersichtlich die Kosten aller Routen von Startknoten A zum jeweiligen Endknoten aufgeführt – die exakten kürzesten Wegstrecken ergeben sich dann stets rekursiv. Der kürzeste Weg von A zu E ist z.B. mit Kosten von 250 verbunden und ergibt sich durch Rückwärts-Gehen von E aus zu den jeweiligen Vorgängern (E-C-B-A) – der entsprechende Weg ist also A-B-C-E.

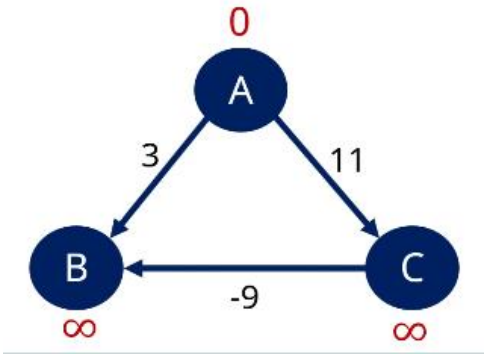
<u>Bellman-Ford-Algorithmus</u> („überprüfe alle Kanten so oft wie nötig“)

Pseudocode (Implementations-nah):

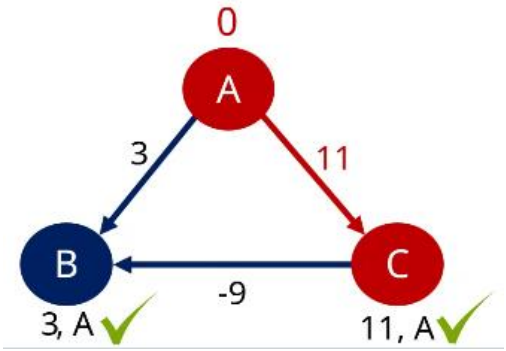
```
n = Anzahl der Knoten
Initialisierung
    für jeden Knoten:
        setze Distanz auf unendlich
    Distanz von Startknoten = 0

Iterationen
    führe n–1 Mal aus:
        für jede Kante im Graph:
            wenn (Kantengewicht + Distanz zum Ausgangsknoten) < Distanz zum Zielknoten
                Distanz zum Zielknoten = Kantengewicht + Distanz zum Ausgangsknoten
                Vorgänger von Zielknoten = Ausgangsknoten

Abschluss-Prüfung
    für jede Kante im Graph:
        wenn (Kantengewicht + Distanz zum Ausgangsknoten) < Distanz zum Zielknoten
            Fehler-Ausgabe „Negativer Zyklus“
```

Beispiel (Anwendung des Pseudocodes mit n=3, Anpassen des Graphen)

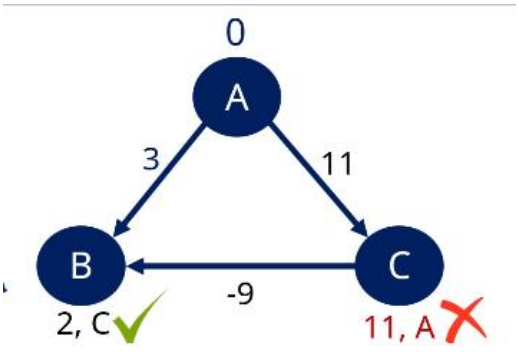
Initialisierung



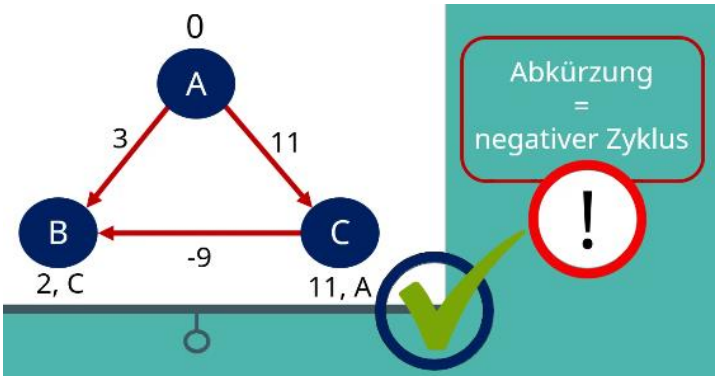
1. Durchlauf



2. Durchlauf



Abschluss-Prüfung



Die Kosten einer Route vom Startknoten A zum jeweiligen Endknoten stehen nun direkt ersichtlich neben dem Endknoten. Der gesamte, kürzeste Weg ergibt sich wiederum rekursiv.