Grundlagen der Informationstechnik

Übung 04 - Routing

Technische Universität Carolo-Wilhelmina zu Braunschweig
Institut für Datentechnik und Kommunikationsnetze (IDA)
Abteilung Kommunikationsnetze

Ankündigung

Die Vorlesung am **22.12.2023** fällt aus.

Anstatt der Vorlesung laden wir Ihnen Zusatzaufgaben hoch, die Ihnen bei der Wiederholung der bisherigen Übungen helfen sollen. Die Bearbeitung der Aufgaben ist freiwillig.
Als kleinen Wettbewerb erhalten alle einen kleinen Preis, die uns die richtigen Lösungen bis zum 10.01.2024 an a.jukan@tu-bs.de oder an k.thang@tu-bs.de zuschicken.

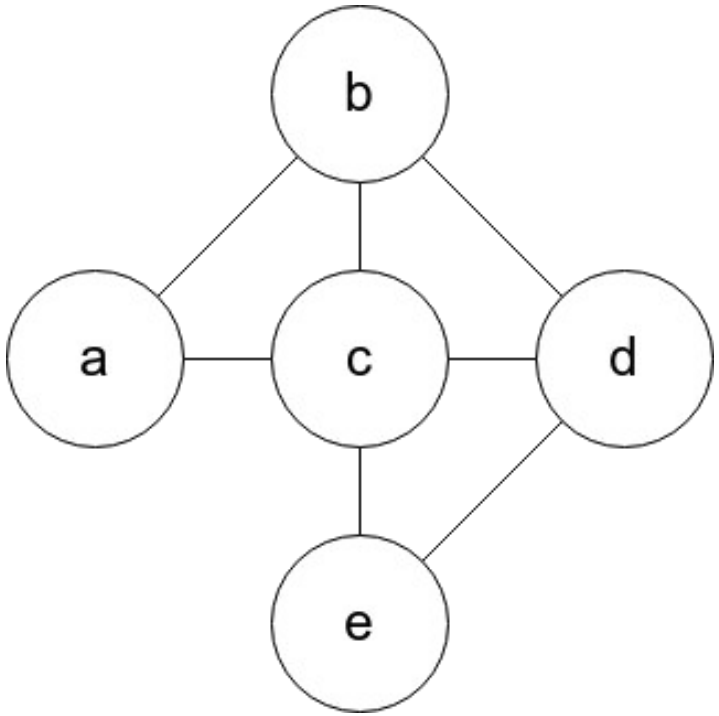
Wir wünschen schöne Feiertage und viel Spaß beim Lösen!

A1) Graphenstruktur

Zeichnen Sie den Graphen G = (V,E), mit V = {a, b, c, d, e} und E = {{a,b},{a,c},{b,c},{b,d},{c,d},{c,e},{d,e}}. Stellen Sie anschließend die zugehörige Adjazenzmatrix und Adjazenzliste auf.

V = {a, b, c, d, e}

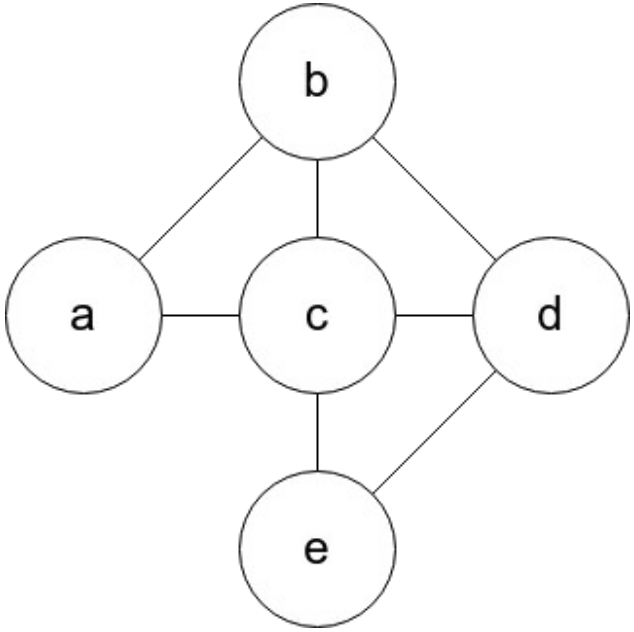
E = {{a,b},{a,c},{b,c},{b,d},{c,d},{c,e},{d,e}}.



V = {a, b, c, d, e}
E = {{a,b},{a,c},{b,c},{b,d},{c,d},{c,e},{d,e}}.

Adjazenzmatrix

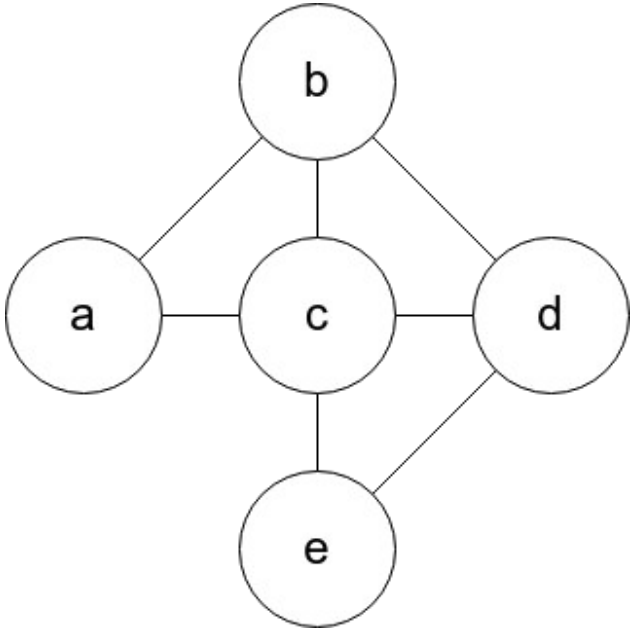
	a	**b**	**c**	**d**	**e**
a	0	1	1	0	0
b	1	0	1	1	0
c	1	1	0	1	1
d	0	1	1	0	1
e	0	0	1	1	0



V = {a, b, c, d, e}
E = {{a,b},{a,c},{b,c},{b,d},{c,d},{c,e},{d,e}}.

Adjazenzliste

a	b, c
b	a, c, d
c	a, b, d, e
d	b, c, e
e	c, d

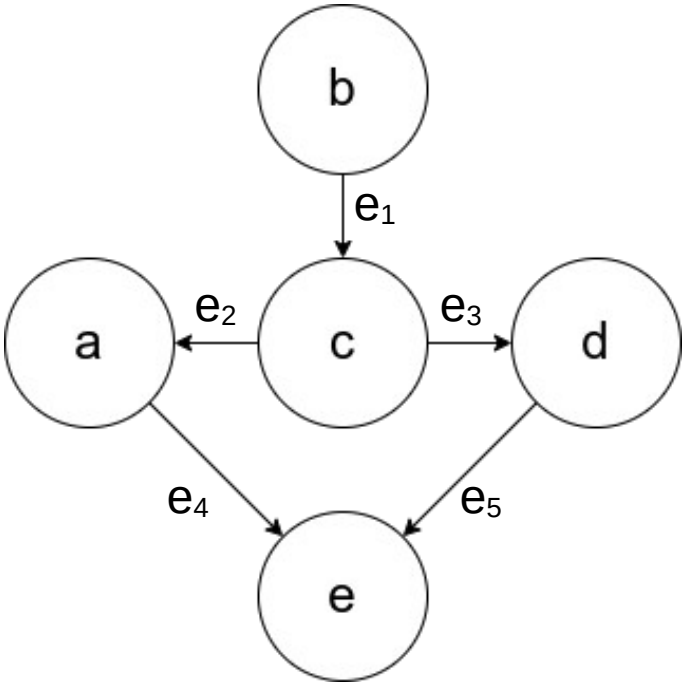


A2) Graphenstruktur

Zeichnen Sie den Graphen G = (V,E), mit V = {a, b, c, d, e} und E = {(b,c),(c,a),(c,d),(a,e),(d,e)}. Stellen Sie anschließend die zugehörige Inzidenzmatrix auf.

V = {a, b, c, d, e}

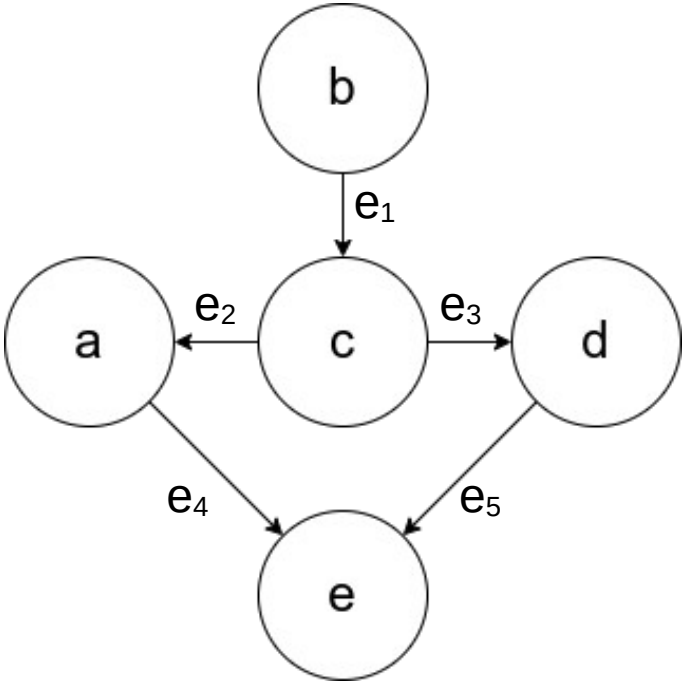
E = {(b,c),(c,a),(c,d),(a,e),(d,e)}



V = {a, b, c, d, e}
E = {(b,c),(c,a),(c,d),(a,e),(d,e)}

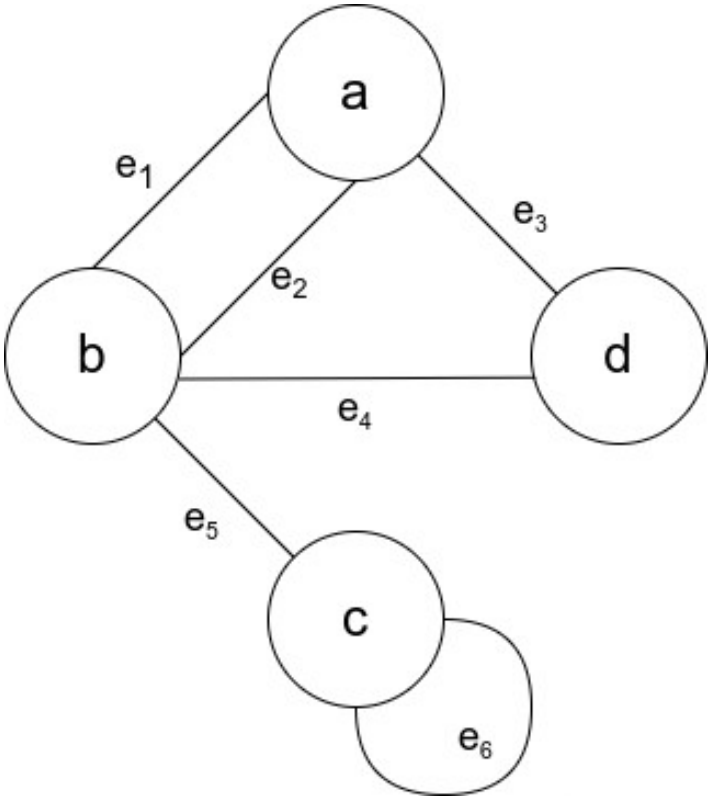
Inzidenzmatrix

	e_1	e_2	e_3	e_4	e_5
a	0	1	0	-1	0
b	-1	0	0	0	0
c	1	-1	-1	0	0
d	0	0	1	0	-1
e	0	0	0	1	1



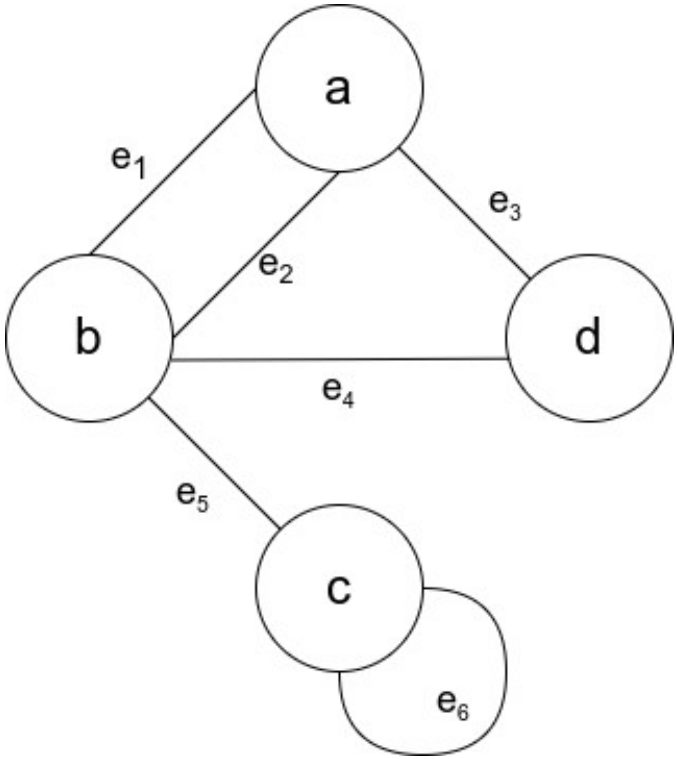
A3) Graphenstruktur

Stellen Sie für den unten gezeigten Graphen die Inzidenzmatrix auf.



Stellen Sie für den unten gezeigten Graphen die Inzidenzmatrix auf.

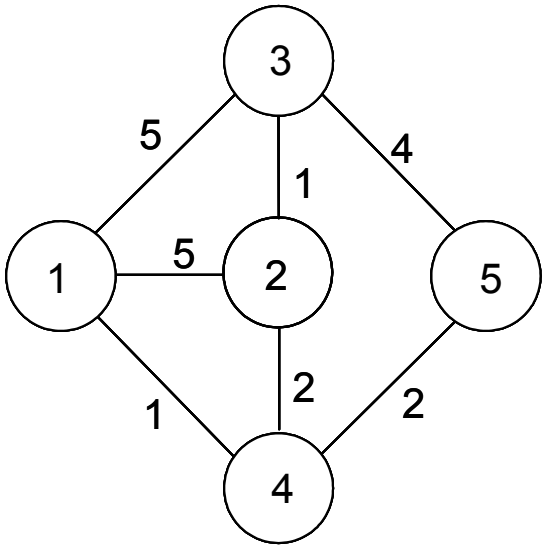
	e_1	e_2	e_3	e_4	e_5	e_6
a	1	1	1	0	0	0
b	1	1	0	1	1	0
c	0	0	0	0	1	1
d	0	0	1	1	0	0



A4) Distanz-Vektor Routing Algorithmus

a) Stellen Sie für jeden Quellknoten des Netzbeispiels die Kostentabelle auf, wobei die Konvergenz des Verfahren vorausgesetzt wird. Verwenden Sie dafür die Topologie des im Bild gezeigten Netzbeispiels und zeigen Sie die potentielle Gefahr von Routing Schleifen.

b) Verdeutlichen Sie die Funktionsweise des Distanz-Vektor Algorithmus anhand des Netzbeispiels, falls die Metrik von c(2,3) = 1 auf c(2,3) = 3 ändert.



Routing-Algorithmus: Distanzvektor

Dezentrale (verteilte) Zustandsinformation

- Knoten X
  - kennt **nur** die Linkkosten zu allen Nachbarknoten V: c(x,v)
- ermittelt die Kosten des Least-Cost-Path zum Zielknoten Y: $D_X(y) \approx k_X(y)$
- bildet seinen Distanzvektor zu allen Zielen: $\mathbf{D}_X = (D_X(y):| \; y \in \mathbf{N})$
- kennt die Distanzvektoren seiner Nachbarknoten V: $\mathbf{D}_V = (D_V(y):| \; y \in \mathbf{N})$

Update und Kostenberechnung (verteiltes Routing)

- Von Zeit zu Zeit sendet jeder Knoten seinen Distanzvektor an die Nachbarn
- Falls Knoten X ein Update ($\mathbf{D}_V$) von seinem Nachbarknoten V erhält aktualisiert er seinen eigenen Distanzvektor:

  für jeden Zielknoten Y: $D_X(y) = \min_V \{c(x,v) + D_V(y)\}$
- unter Bedingungen konvertiert $D_X(y)$ gegen die optimalen Kosten $k_X(y)$

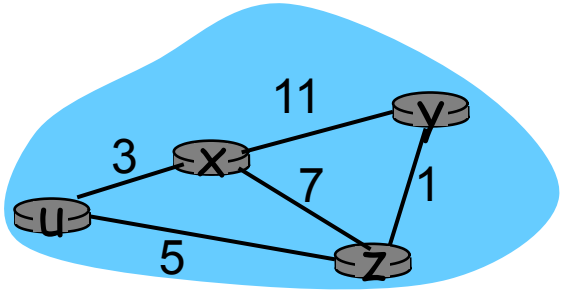
Routing-Algorithmus: Distanzvektor

Verteilte Zustandsinformation und Routing Tabelle

- c(x,y): Kosten des Links von Knoten X zu Nachbarknoten Y
- $D_X(y,v)$: Kosten des Weges von Knoten X zu Zielknoten Y über Nachbar V
- $D_X(y)$: Kosten des Least-Cost-Path von Knoten X zu Knoten Y

$v \in \{u, y, z\}$ $D_X(y,v) = c(x,v) + D_V(y)$

$D_X(y) = \min_v D_X(y,v)$ $v \in \{u, y, z\}$



über Knoten v

zum Ziel

$D_X(.,.)$	u	y	z
u	3	17	12
y	9	11	8
z	8	12	7

Routing Database in Knoten X

Ziel	via	$D_X(.)$
u	u	3
y	z	8
z	z	7

Forwarding Table in Knoten X

Bellman-Ford Algorithmus

[Initialisierung in Knoten x]

$D_X(v,v) = c(x,v)$ for all neighbours v, $D_X(y,w) = \infty$ for w is not neighbour node

For all destination nodes y: sent $D_X(y) = \min_V D_X(y,v)$ to all neighbour nodes v

[loop (in each node x)]

wait (until a link cost changes or until an update receives from neighbour v)

if (c(x,v) changes by d)

for all destinations y: $D_X(y,v) = D_X(y,v) + d$

← *(Kosten zu allen Zielen über Nachbar v um d ändern. Anmerkung: d kann positiv oder negativ sein)*

else if (update $D_V(y)$ received from v for destination y)

for the single destination y: $D_X(y,v) = c(x,v) + D_V(y)$

[Neuberechnung aller Vektoren]

for all destinations y: $D_X(y) = \min_V D_X(y,v)$

← *(der kürzeste Weg von v zu y hat sich geändert, v hat einen neuen Wert $D_V(y)$ gesendet)*

if (we have a new $D_X(y)$ for any destination y)

send new value $D_X(y)$ of to all neighbour v

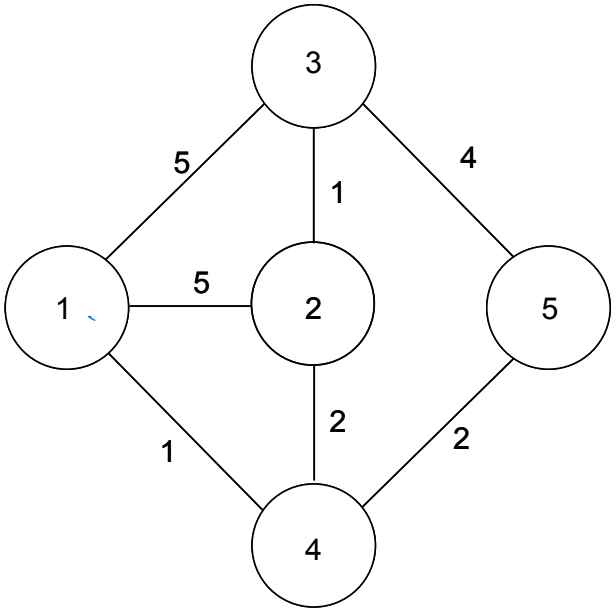
← *(send Update)*

4a)

a) Stellen Sie für jeden Quellknoten des Netzbeispiels die Kostentabelle auf, wobei die Konvergenz des Verfahren vorausgesetzt wird. Verwenden Sie dafür die Topologie des im Bild gezeigten Netzbeispiels und zeigen Sie die potentielle Gefahr von Routing Schleifen.

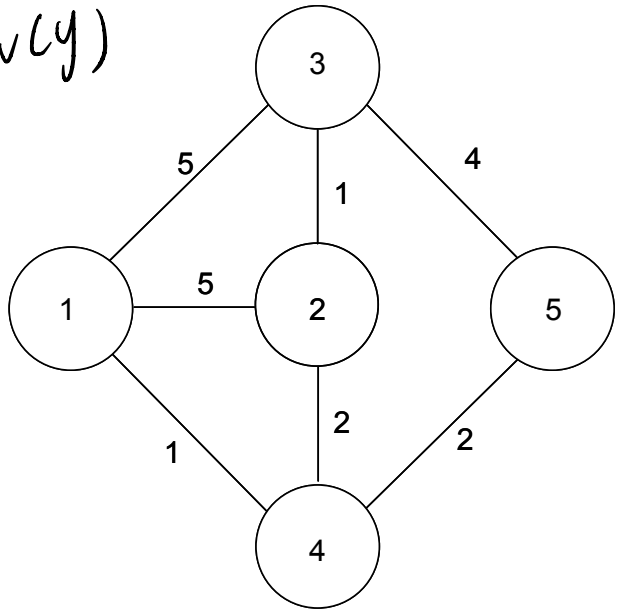
	Kosten zum Ziel über Knoten j			
	$D_1(v,j)$	2	3	4
Z	2	5	6	3
i	3	6	5	4
e	4	7	8	1
l	5	9	9	3

	Kosten zum Ziel über Knoten j			
	$D_2(v,j)$	1	3	4
Z	1	5	5	3
i	3	9	1	5
e	4	6	4	2
l	5	8	5	4



4a)

$D_x(y,v) = c(x,v) + D_v(y)$



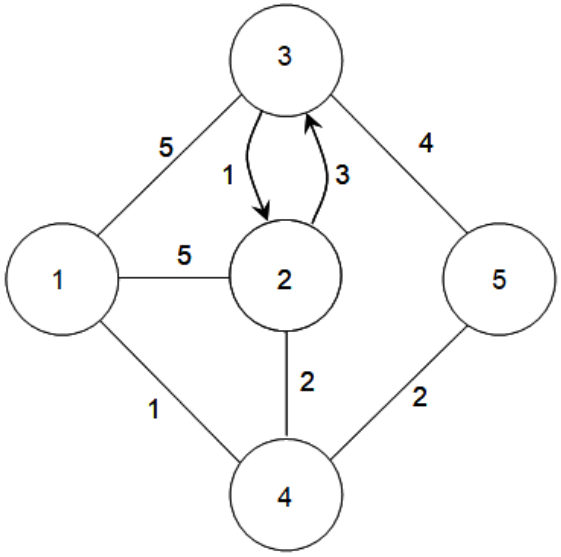
		Kosten zum Ziel über Knoten $D_3(v,j)$		
		1	2	5
Z	1	5	4	7
i	2	8	1	8
e	4	6	3	6
l	5	8	5	4

		Kosten zum Ziel über Knoten $D_4(v,j)$		
		1	2	5
Z	1	1	5	5
i	2	4	2	6
e	3	5	3	6
l	5	4	6	2

		Kosten zum Ziel über Knoten $D_5(v,j)$	
		3	4
Z	1	8	3
i	2	5	4
e	3	4	5
l	4	7	2

b) Verdeutlichen Sie die Funktionsweise des Distanz-Vektor Algorithmus anhand des Netzbeispiels, falls die Metrik von c(2,3) = 1 auf c(2,3) = 3 ändert.

1. Iteration: Update Knoten 2



	Kosten zum Ziel *v* über Knoten j			
	$D_2(v,j)$	1	3	4
Z	1	5	7	3
i	3	9	3	5
e	4	6	6	2
l	5	8	7	4

III) neuen Wert $D_2(3) = min_w D_2(3,w) = 3$ an Nachbarknoten 1, 3 und 4 senden

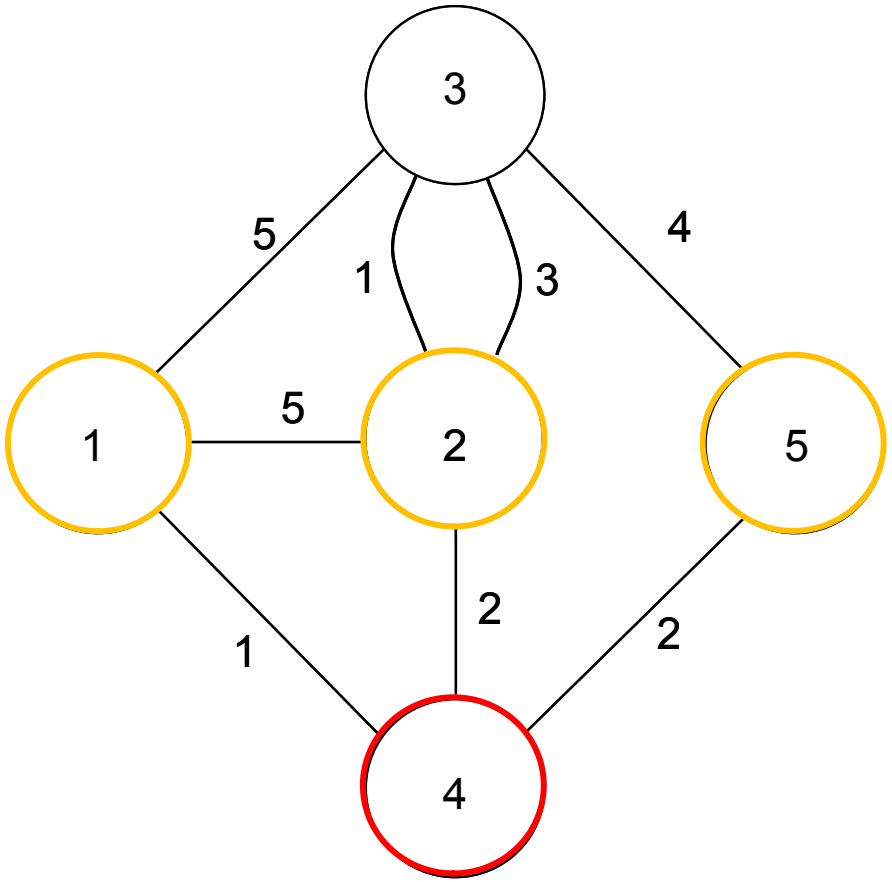
2. Iteration: Update der Knoten 1 und 4, Knoten 3 keine Änderung da Zielknoten

	Kosten zum Ziel v über Knoten j			
	$D_1(v,j)$	2	3	4
Z	2	5	6	3
i	3	8	5	4
e	4	7	8	1
l	5	9	9	3

	Kosten zum Ziel v über Knoten j			
	$D_4(v,j)$	1	2	5
Z	1	1	5	5
i	2	4	2	6
e	3	5	5	6
l	5	4	6	2

II)

III) neuen Wert $D_4(3) = min_w D_4(3,w) = 5$ an Nachbarknoten 1, 2, 5 senden


3
5
1
3
4
1
5
2
5
2
1
2
4

3. Iteration: Update der Knoten 1,2 und 5

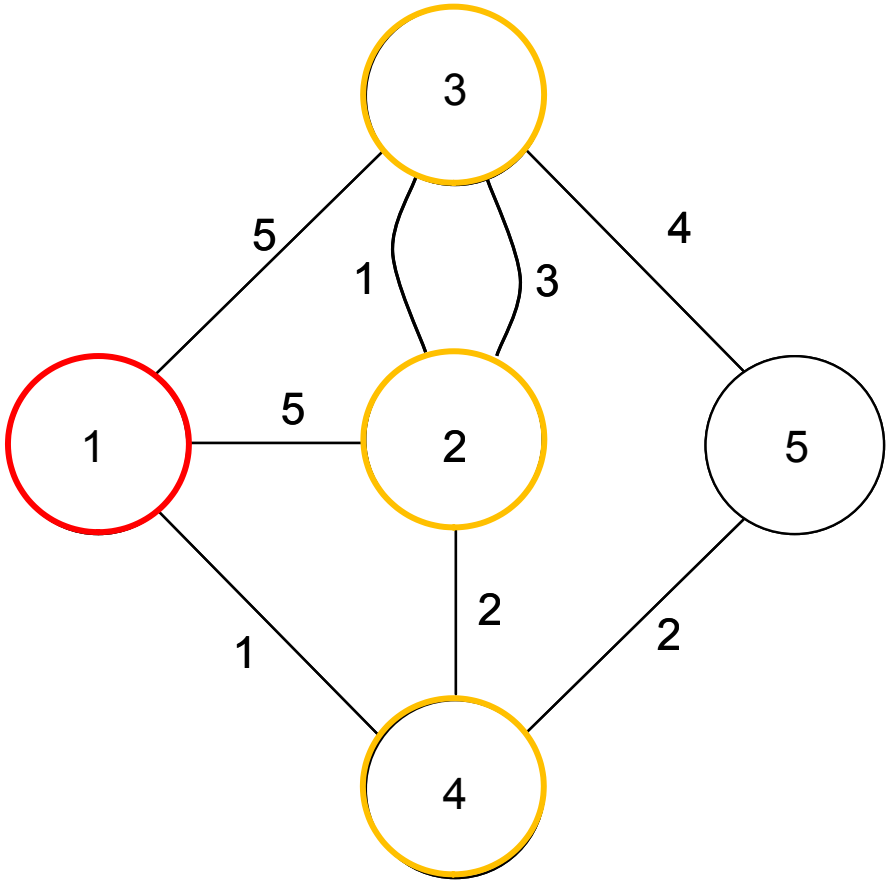
	Kosten zum Ziel v über Knoten j			
	$D_1(v,j)$	2	3	4
Z	2	5	6	3
i	3	8	5	6
e	4	7	8	1
l	5	9	9	3

	Kosten zum Ziel v über Knoten j			
	$D_2(v,j)$	1	3	4
Z	1	5	7	3
i	3	9	3	7
e	4	6	6	2
l	5	8	7	4

II)

III) neuen Wert $D_1(3) = min_w D_1(3,w) = 5$
an Nachbarknoten 2, 3, 4 senden

	Kosten zum Ziel v über Knoten j		
	$D_5(v,j)$	3	4
Z	1	8	3
i	2	5	4
e	3	4	7
l	4	7	2


3
5
1
3
4
1
5
2
5
2
1
2
4

4. Iteration: Update der Knoten 2 und 4, Knoten 3 keine Änderung da Zielknoten

	Kosten zum Ziel v über Knoten j			
	$D_2(v,j)$	1	3	4
Z	1	5	7	3
i	3	10	3	7
e	4	6	6	2
l	5	8	7	4

	Kosten zum Ziel v über Knoten j			
	$D_4(v,j)$	1	2	5
Z	1	1	5	5
i	2	4	2	6
e	3	6	5	6
l	5	4	6	2

Keine weiteren Updates mehr; Routing-Tabellen sind stationär

4b)

Router 1		
Ziel	Nächster Hop	Hops zum Ziel
1		
2	4	3
3	3	5
4	4	1
5	4	3

Router 2		
Ziel	Nächster Hop	Hops zum Ziel
1	4	3
2		
3	3	3
4	4	2
5	4	4

Router 3		
Ziel	Nächster Hop	Hops zum Ziel
1	2	4
2	2	1
3		
4	2	3
5	5	4

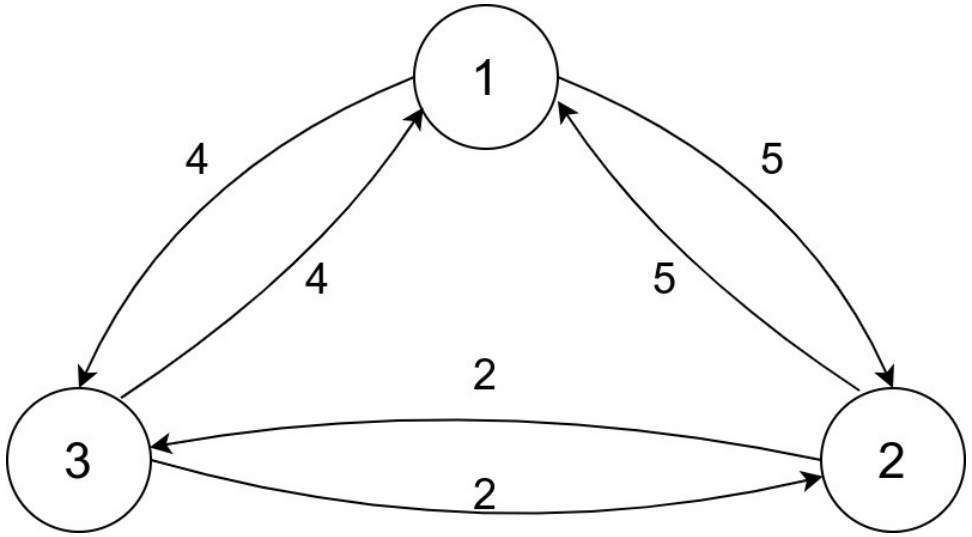
Router 4		
Ziel	Nächster Hop	Hops zum Ziel
1	1	1
2	2	2
3	2	5
4		
5	5	2

Router 5		
Ziel	Nächster Hop	Hops zum Ziel
1	4	3
2	4	4
3	3	4
4	4	2
5		

A5) Distanz-Vektor Routing Algorithmus

a) Nennen Sie ein mit dem Distanz-Vektor Routing Algorithmus arbeitendes IP-Routing Protokoll. Für welche Hierarchieebene des Internet-Routings wird dieses Protokoll verwendet und welche Eigenschaften besitzt dieses?

b) Im Bild ist die Topologie eines Netzbeispiels mit den Knoten 1, 2 und 3, sowie den Kosten c(i, j) der gerichteten Verbindungslinks von Knoten i nach Knoten j gezeigt.



A5) Distanz-Vektor Routing Algorithmus

b) Die Kostentabelle für den Distanz-Vektor Routing Algorithmus mit Kosten $D_k(v,j)$ des Weges von Knoten k zu Zielknoten v über den Nachbarknoten j sind für die Knoten 1 und 2 in den folgenden Tabellen dargestellt.

	$D_1(v,j)$ Kosten zum Ziel v über Knoten j	
Ziel	2	3
2	5	6
3	7	4

	$D_2(v,j)$ Kosten zum Ziel v über Knoten j	
Ziel	1	3
1	5	6
3	9	2

	$D_3(v,j)$ Kosten zum Ziel v über Knoten j	
Ziel	1	2
1		
2		

Geben Sie die Kostentabelle für den Knoten 3 an und kennzeichnen Sie die für die Forwarding-Tabelle verwendeten Einträge.

A5) Distanz-Vektor Routing Algorithmus

c) Nehmen Sie an, dass sich die Link-Metriken c(1,2) = c(2,1) = 5 auf c(1,2) = c(2,1) = 1 verändern. Stellen Sie die sich hieraus ergebenden neuen Kostentabellen der durch diese Änderung der Link-Metriken betroffenen Knoten dar. Erläutern Sie, ob sich diese Änderungen der Forwarding-Tabellen (bzw. Routing-Tabellen) ergeben und welche Updates ggf. gesendet werden müssen.

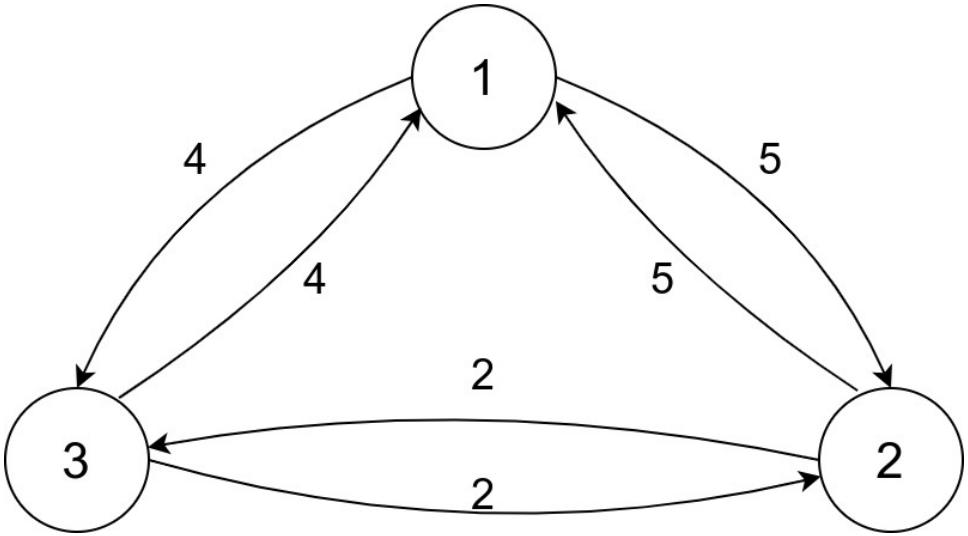
a) Nennen Sie ein mit dem Distanz-Vektor Routing Algorithmus arbeitendes IP-Routing Protokoll. Für welche Hierarchieebene des Internet-Routings wird dieses Protokoll verwendet und welche Eigenschaften besitzt dieses?

Distanz-Vektor Algorithmus: RIP-Protokoll

- Wird innerhalb eines Autonomen Systems verwendet (bzw. Intra-Domain Routing)
- Erhält von den Nachbarknoten die Informationen über deren minimale Kosten zu allen Zielen. (Alternativ: Es gibt nur eine verteilte Netzzustands- und Routinginformation)
- Berechnete Wege können bei Änderung der Netzkosten transiente Schleifen enthalten

5b)

b) Geben Sie die Kostentabelle für den Knoten 3 an und kennzeichnen Sie die für die Forwarding-Tabelle verwendeten Einträge.



Ziel	$D_1(v,j)$ Kosten zum Ziel v über Knoten j	
	2	3
2	5	6
3	7	4

Ziel	$D_2(v,j)$ Kosten zum Ziel v über Knoten j	
	1	3
1	5	6
3	9	2

Ziel	$D_3(v,j)$ Kosten zum Ziel v über Knoten j	
	1	2
1	4	7
2	9	2

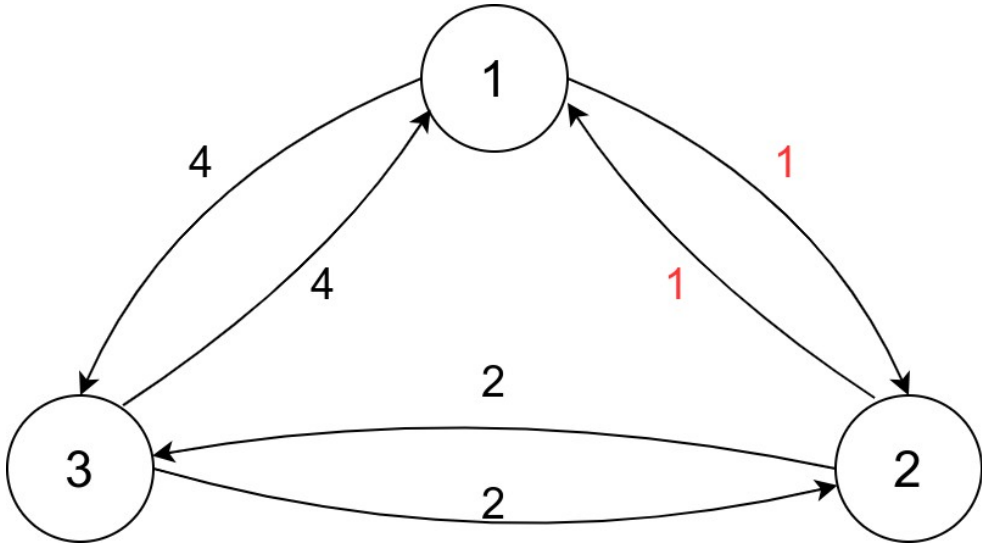
5c)

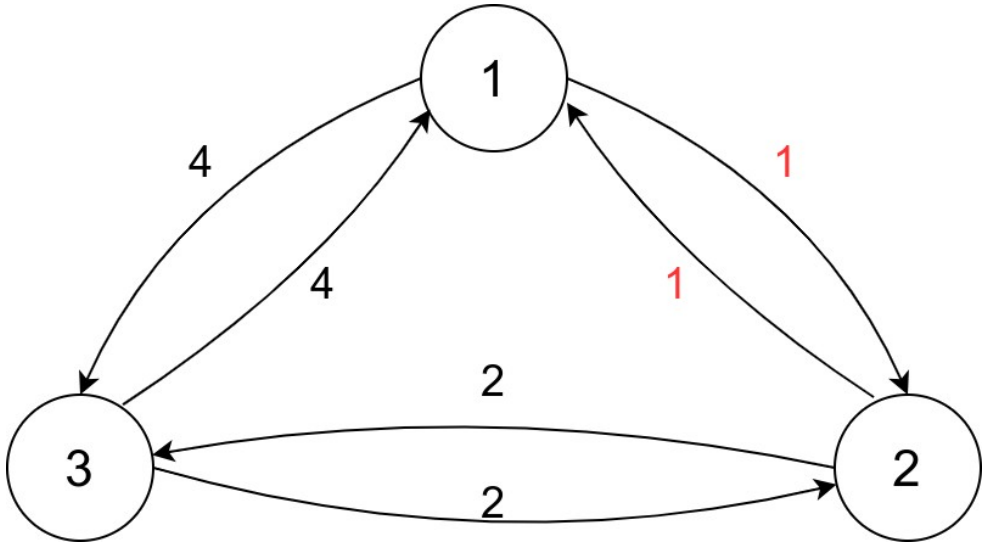
c) Nehmen Sie an, dass sich die Link-Metriken c(1,2) = c(2,1) = 5 auf c(1,2) = c(2,1) = 1 verändern. Stellen Sie die sich hieraus ergebenden neuen Kostentabellen der durch diese Änderung der Link-Metriken betroffenen Knoten dar. Erläutern Sie, ob sich diese Änderungen der Forwarding-Tabellen (bzw. Routing-Tabellen) ergeben und welche Updates ggf. gesendet werden müssen.

Veränderung der Metrik von c(1,2) = c(2,1) = 5

→ c(1,2) = c(2,1) = 1

→ *d= -4*





Neue Kostentabellen mit Kennung kürzester Wege

Ziel	$D_1(v,j)$ Kosten zum Ziel v über Knoten j	
	2	3
2	**1**	6
3	**3**	4

Ziel	$D_2(v,j)$ Kosten zum Ziel v über Knoten j	
	1	3
1	**1**	6
3	**5**	2

Ziel	$D_3(v,j)$ Kosten zum Ziel v über Knoten j	
	1	2
1	4	7
2	9	2

Updates für die 1. Iteration

Knoten 1:

- $D_1(2) = \min_w D_1(2,w) = 1$ liefert neuen Kostenwert und
  - es wird das Update $D_1(2) = 1$ an die Nachbarknoten 2 und 3 gesendet.
- $D_1(3) = \min_w D_1(3,w) = 3$ liefert neuen Kostenwert und neuen Nachbarknoten 2 für den kürzesten Wert zum Zielknoten 3, (der in der Forwarding Tabelle verwendet werden muss (siehe Schattierung).
  - Es wird das Update D1(3) = 3 an die Nachbarknoten 2 und 3 gesendet.

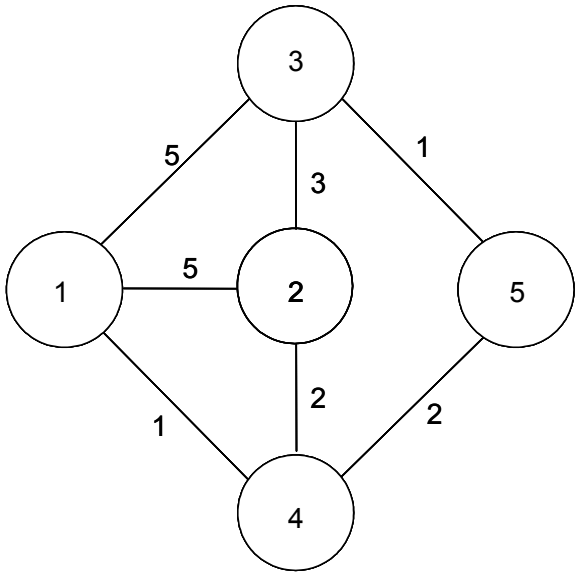
Knoten 2:

- $D_2(1) = \min_w D_2(1,w) = 1$ liefert neuen Kostenwert,
  - Es wird ein Update $D_2(1) = 1$ an die Nachbarknoten 1 und 3 gesendet.

A6) Link-State Routing Algorithmus

Führen Sie den Algorithmus für das Netzbeispiel bezüglich des Ursprungsknotens 1 aus. Wie viele Schritte sind erforderlich? Wie wird die Wegeinformation vom Knoten 1 zu Knoten 5 ermittelt?

Topologie



Link-Metrik

c(i,j)	1	2	3	4	5
1	-	5	5	1	∞
2	5	-	3	2	∞
3	5	3	-	∞	1
4	1	2	∞	-	2
5	∞	∞	1	2	-

Routing-Algorithmus: Link-State

Globale Zustandsinformation

- Netzwerktopologie und Linkkosten sind in jedem Knoten (Router) bekannt
  - Die Zustandsinformation ist im statischen Fall in jedem Router gleich
  - Wird durch ein Link-State Broadcast Protokoll realisiert
- Nur positive, meist zusatndsabhängige Linkkosten (Metrik)

Dezentral ausgeführter Routing Algorithmus

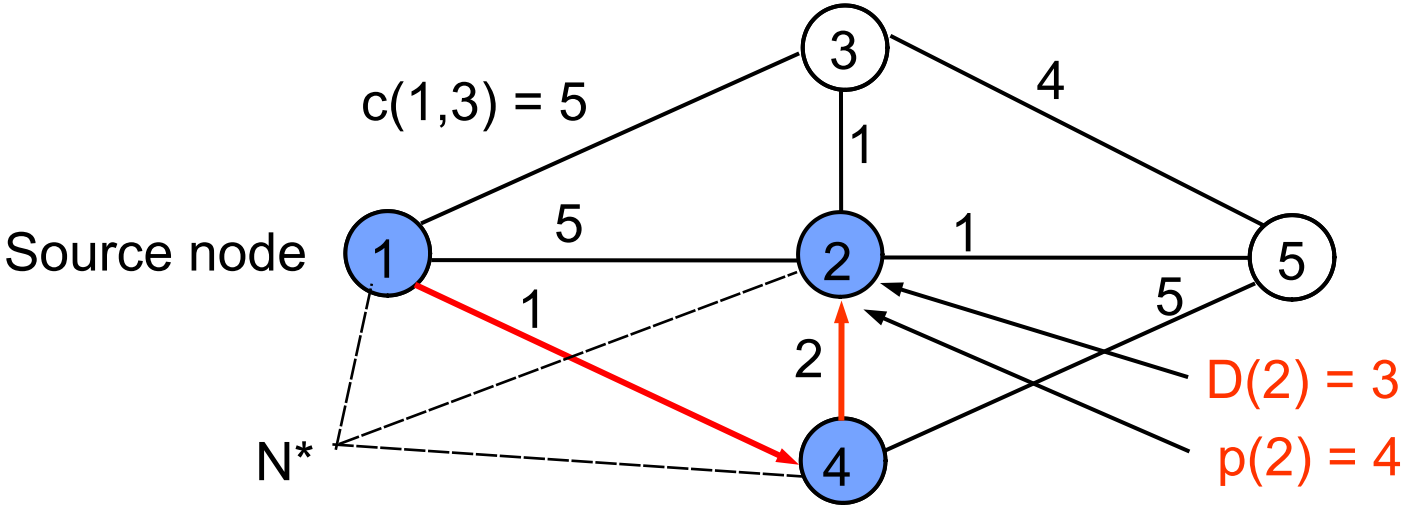
- Shortest-Path Algorithmus von Dijkstra
- In jedem Knoten:
  - Finde den Weg geringster Kosten zu allen anderen Knoten
  - Quellknoten ist Wurzel eines Baums kürzester Wege
  - Wege sind schleifenfrei

Shortest-Path Algorithmus von Dijkstra

Notation

Für einen Quellknoten (source node):

- c(i,j): Linkkosten von Knoten i zu j, c(i,j) = ∞ falls kein Nachbarknoten.
  - Linkkosten aller Links im Quellknoten bekannt
- D(v): Kosten des Weges vom Quellknoten zum Knoten v mit den momentan geringsten Kosten (Label)
- p(v): Vorgänger von v auf dem momentan kürzesten Weg zu Knoten v.
- N*: Menge der Knoten, zu denen ein Weg geringster Kosten besteht.



Shortest-Path Algorithmus von Dijkstra

Initialisierung

N* = {A}; D(v) = ∞; p(v) = 0 ← *(A ist der Quellknoten)*

I. for all nodes v adjacent to A:

D(v) = c(A,v); p(v) = A ← *(initiale Kosten zu allen Nachbarknoten)*

Iteration

loop

II. find w not in N* such that D(w) is a minimum; add w to N* ← *(Wähle den Knoten w aus der Menge N\N* der vom Quellknoten mit minimalen Pfadkosten erreicht wird)*

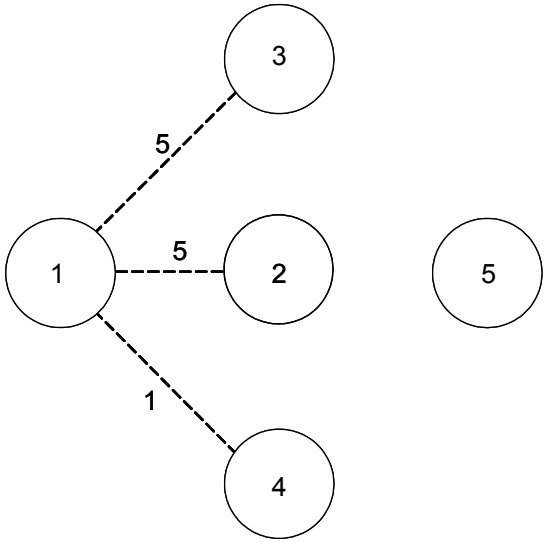
III. Update D(v) for all v adjacent to w and not in N*:
D(v) = min(D(v), D(w) + c(w,v))
if new shortest path: p(v) = w ← *(Die neuen Kosten zu v sind entweder die alten Kosten zu v oder die bekannten kosten des kürzesten Weges zu w, zuzüglich der Kosten von w zu v)*

until all nodes n ∈ N*;

0. Iteration

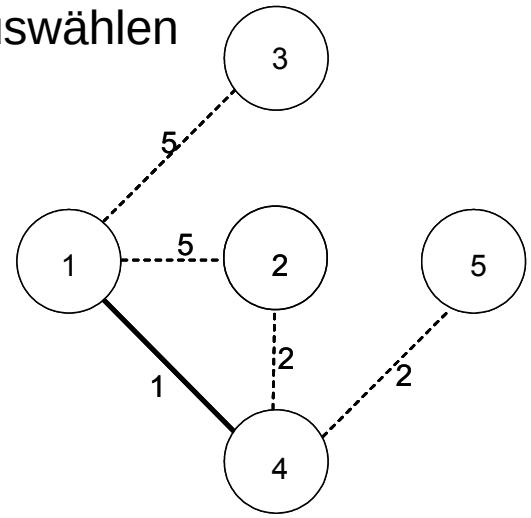
- Momentan bekannte Pfad mit geringsten Kosten zu 1 zu seinen direkten Nachbarn

		Knotenmarkierungen					
Iteration	Schritt	N	D(1), p(1)	D(2), p(2)	D(3), p(3)	D(4), p(4)	D(5), p(5)
0.	I	{1}		5, 1	5, 1	1, 1	∞, 0



1. Iteration

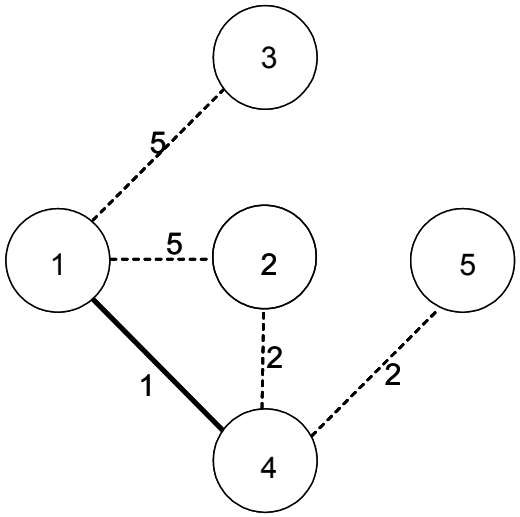
- Schritt II:
  - Nach Knoten suchen, die noch nicht in Menge N aufgenommen sind
  - Knoten mit geringsten Kosten zur vorherigen Iteration auswählen
    → Knoten 4 wird zur Menge N hinzugefügt



		Knotenmarkierungen					
Iteration	Schritt	N	D(1), p(1)	D(2), p(2)	D(3), p(3)	D(4), p(4)	D(5), p(5)
0.	I	{1}		5, 1	5, 1	1, 1	∞, 0
1.	II III	{1, 4}		3, 4	5, 1	**min, *w*=4** (1, 1)	3, 4

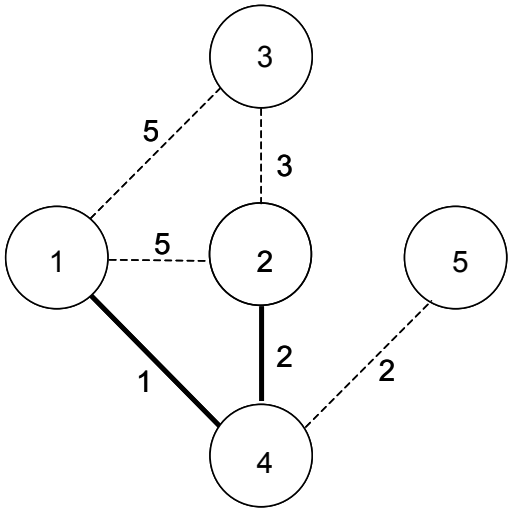
1. Iteration

- Schritt III:
  - D(v) für alle Knoten aktualisieren
    - → Kosten zu Knoten 2 durch Knoten 4 aktualisieren
    - → Kosten des Pfades zu Knoten 5 aktualisieren



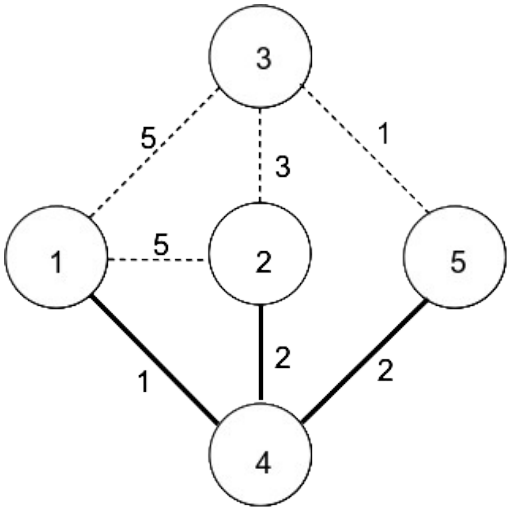
		Knotenmarkierungen					
Iteration	Schritt	N	D(1), p(1)	D(2), p(2)	D(3), p(3)	D(4), p(4)	D(5), p(5)
0.	I	{1}		5, 1	5, 1	1, 1	∞, 0
1.	II III	{1, 4}		3, 4	5, 1	**min, *w*=4** (1, 1)	3, 4

2. Iteration



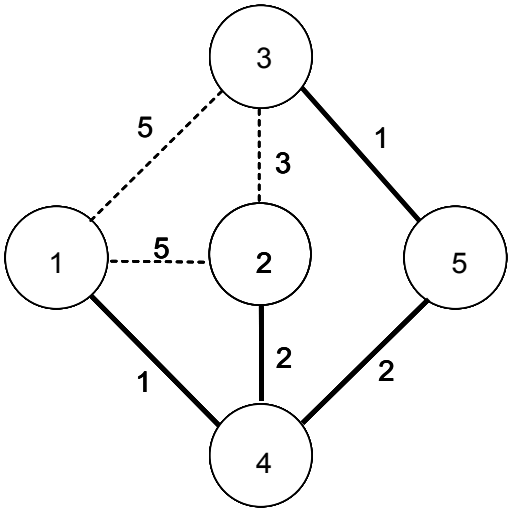
		Knotenmarkierungen					
Iteration	Schritt	N	D(1), p(1)	D(2), p(2)	D(3), p(3)	D(4), p(4)	D(5), p(5)
0.	I	{1}		5, 1	5, 1	1, 1	∞, 0
1.	II III	{1, 4}		3, 4	5, 1	**min, *w*=4** (1, 1)	3, 4
2.	II III	{1, 4, 2}		**min, *w*=2** (3, 4)	5, 1		3, 4

3. Iteration



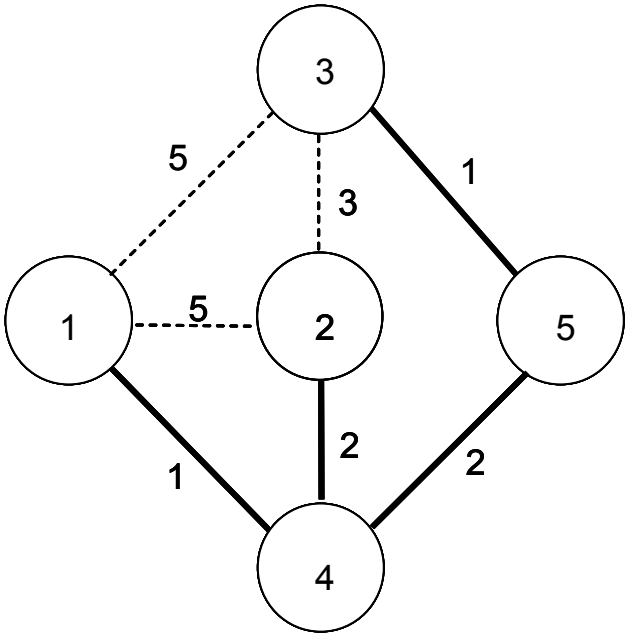
		Knotenmarkierungen					
Iteration	Schritt	N	D(1), p(1)	D(2), p(2)	D(3), p(3)	D(4), p(4)	D(5), p(5)
0.	I	{1}		5, 1	5, 1	1, 1	∞, 0
1.	II III	{1, 4}		3, 4	5, 1	**min, *w*=4** (1, 1)	3, 4
2.	II III	{1, 4, 2}		**min, *w*=2** (3, 4)	5, 1		3, 4
3.	II III	{1, 4, 2 ,5}			4, 5		**min, *w*=5** (3, 4)

4. Iteration



		Knotenmarkierungen					
Iteration	Schritt	N	D(1), p(1)	D(2), p(2)	D(3), p(3)	D(4), p(4)	D(5), p(5)
0.	I	{1}		5, 1	5, 1	1, 1	∞, 0
1.	II III	{1, 4}		3, 4	5, 1	**min, *w*=4** (1, 1)	3, 4
2.	II III	{1, 4, 2}		**min, *w*=2** (3, 4)	5, 1		3, 4
3.	II III	{1, 4, 2 ,5}			4, 5		**min, *w*=5** (3, 4)
4.	II	{1, 4, 2, 5, 3}			**min, *w*=3**		

4. Iteration



Weginformation von 1 nach 5

p(5) = 4; p(4) = 1 → Weg: 1 – 4 – 5