Technische Universität Braunschweig

Decision Support

Institut für Wirtschaftsinformatik

Operations Research

Vorlesung 8

Graphen und Netzwerke: Spannende Bäume & kürzeste Wege

Lernziele und Literatur

- Neue Art der Modellierung: Graphentheorie
- Zwei Verfahren für graphentheoretische Probleme
- Äquivalenz: Alternatives Modell und Verfahren für ein bereits bekanntes Problem
- Übersichtsliteratur: Peter Tittmann - Graphentheorie: Eine anwendungsorientierte Einführung
- Vertiefend: Reinhard Diestel - Graphentheorie

Überblick

1. Einführung und Grundlagen
2. Minimale spannende Bäume in Graphen
3. Kürzeste Wege in Graphen
4. Dynamisches Losgrößenproblem

Überblick

1. **Einführung und Grundlagen**
2. Minimale spannende Bäume in Graphen
3. Kürzeste Wege in Graphen
4. Dynamisches Losgrößenproblem

Anwendungen der Netzwerkmodellierung

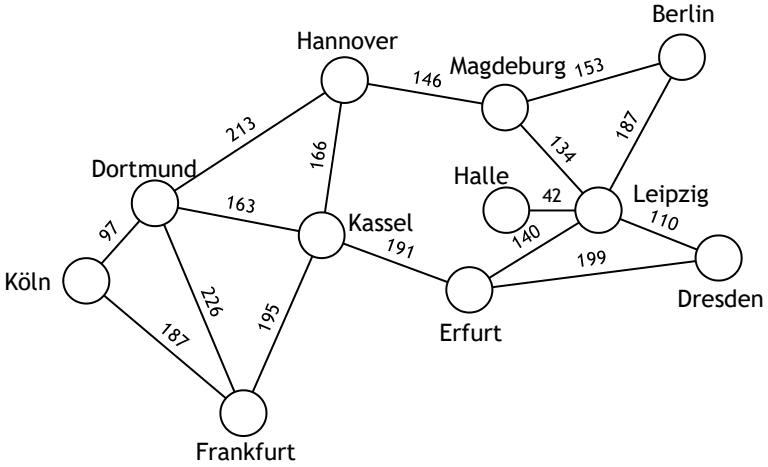
Mit Graphen lassen sich Systeme modellieren.

- Verkehrssysteme mit Flughäfen, Bahnhöfen und Haltestellen
- Distributionssysteme mit Zentrallagern und Regionallagern
- Finanzpläne mit Ein- und Auszahlungen
- Elektrische Schaltungen, Baupläne
- Produktkonfigurationen,
- das Internet, Elektrizitätsnetze, Telefonnetze, etc.

Relation	Knoten	Kanten	Gewichte
Physisch	Orte	Verkehrswege	Distanzen
Logisch	Partner	Relationen	Flüsse
Hierarchisch	Objekte	Ordnung	Mengen
Zeitlich	Zustände	Übergänge	Beliebig

Beispiele zur Netzwerkmodellierung

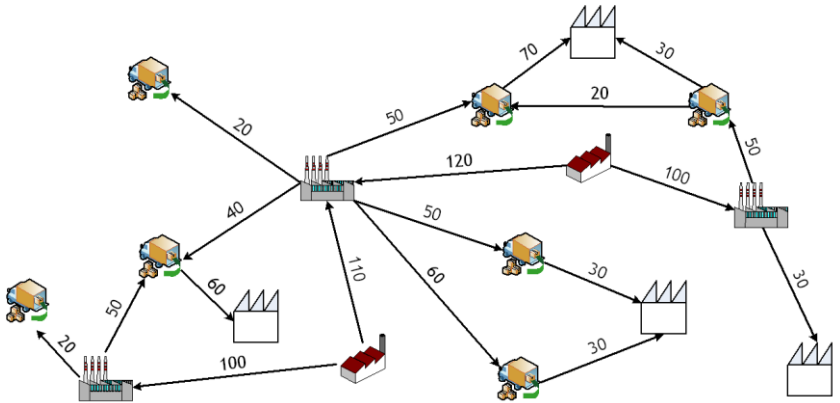
Autobahnnetz: Welcher Weg ist der kürzeste?



Relation	Knoten	Kanten	Gewichte
Physisch	Orte	Verkehrswege	Distanzen
Logisch	Partner	Relationen	Flüsse
Hierarchisch	Objekte	Ordnung	Mengen
Zeitlich	Zustände	Übergänge	Beliebig

Beispiele zur Netzwerkmodellierung

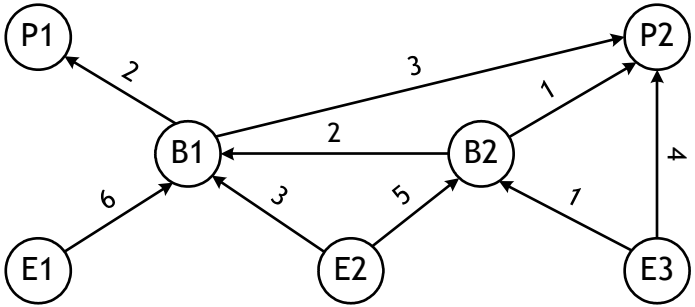
Beschaffungs- und Liefernetzwerk: Wo wird was produziert?



Relation	Knoten	Kanten	Gewichte
Physisch	Orte	Verkehrswege	Distanzen
Logisch	**Partner**	**Relationen**	**Flüsse**
Hierarchisch	Objekte	Ordnung	Mengen
Zeitlich	Zustände	Übergänge	Beliebig

Beispiele zur Netzwerkmodellierung

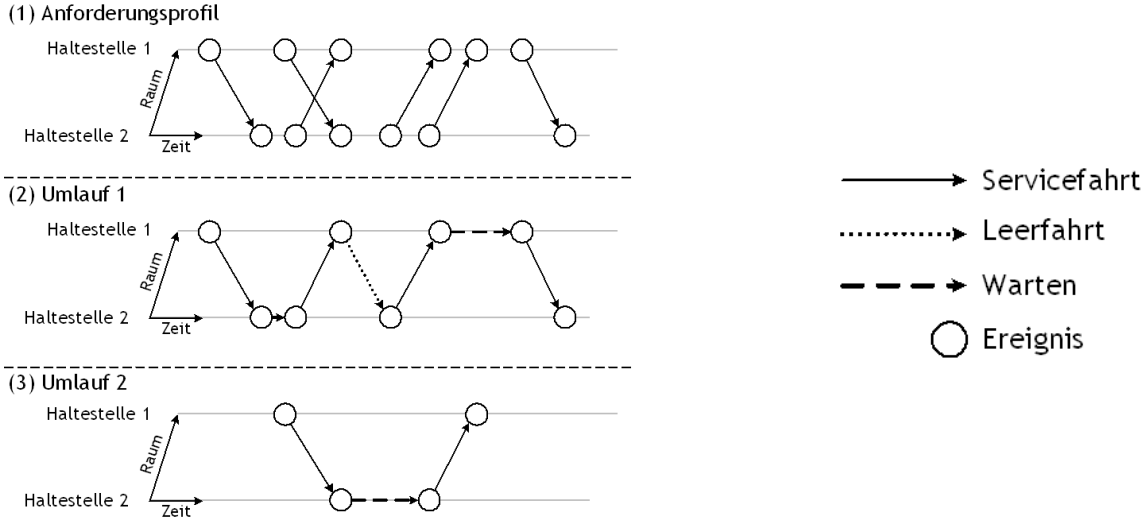
- Erzeugnisstruktur als Gozintograph
  - Teilenachweis: In welchen Produkten wird ein Teil verwendet?
  - Stückliste: Wie viel Teile gegen in ein Produkt ein?



Relation	Knoten	Kanten	Gewichte
Physisch	Orte	Verkehrswege	Distanzen
Logisch	Partner	Relationen	Flüsse
Hierarchisch	Objekte	Ordnung	Mengen
Zeitlich	Zustände	Übergänge	Beliebig

Beispiele zur Netzwerkmodellierung

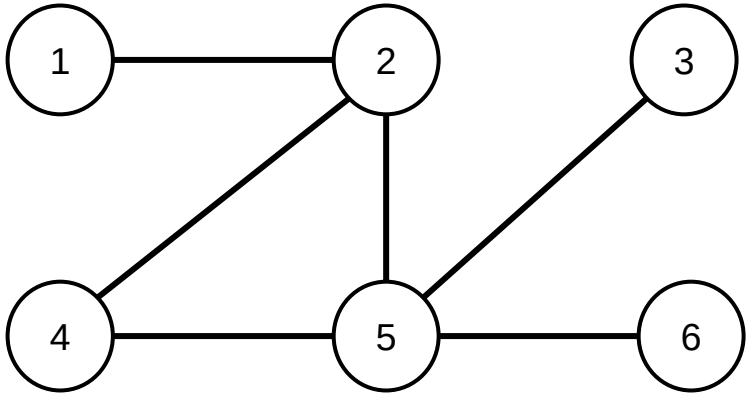
Umlaufplanung im ÖPNV



Relation	Knoten	Kanten	Gewichte
Physisch	Orte	Verkehrswege	Distanzen
Logisch	Partner	Relationen	Flüsse
Hierarchisch	Objekte	Ordnung	Mengen
Zeitlich	Zustände	Übergänge	Beliebig

Ungerichtete Graphen

- Ein **ungerichteter Graph** besteht aus einer
  - Menge V von n **Knoten** und einer
  - Menge von Verbindungen unter den Knoten,
  - die als **Kantenmenge** E bezeichnet wird.
  - Eine Kante zwischen zwei Knoten i und j wird dargestellt durch (i, j) bzw. (j, i).



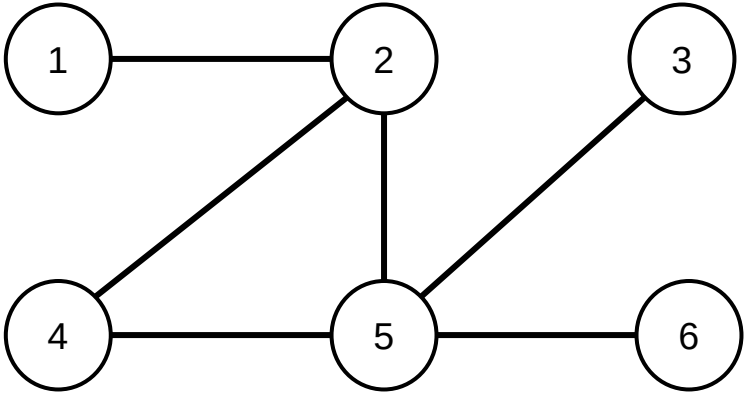
- Beispielnetzwerk mit
  - Knotenmenge $V = \{1,2,3,4,5,6\}$; $n = 6$ Knoten und
  - Kantenmenge $E = \{(1,2), (3,5), (2,4), (2,5), (4,5), (5,6)\}$; $m = 6$ Kanten

Strukturen in ungerichteten Graphen

- **Grad** von i : Anzahl von benachbarten Knoten von i
- **Isoliert:** Knoten i mit $grad(i) = 0$
- **Vollständiger** Graph: Knoten sind durch Kanten mit allen anderen Knoten verbunden.
- **Pfad** von i nach j: endliche Knotenfolge mit dem Startknoten i und Endknoten j, in der Knoten mehrfach vorkommen können.
- **Zyklus:** Pfad von i nach j mit $i = j$
- **Weg:** Pfad ohne Teilzyklen in der Knotenfolge
- **Erreichbarer** Knoten j: von i, wenn Weg von Knoten i zu Knoten j führt
- **Zusammenhängender** Graph: alle $i, j \in V$ sind durch Weg verbunden

Beispiel

Graph G sei gegeben als:



Grad(1) = 1

Grad(2) = 3

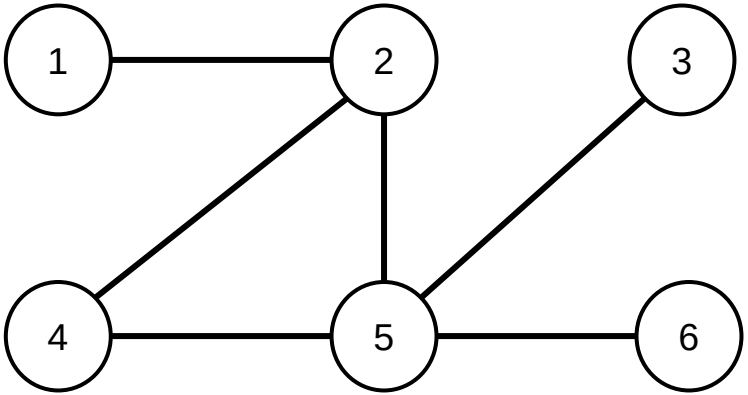
Grad(4) = 2

Kein Knoten ist isoliert

G ist nicht vollständig

Beispiel

Graph G sei gegeben als:



Beispiele für Wege von 1 nach 5:

(1,2,5)

(1,2,4,5)

Beispiele für Pfade von 1 nach 5:

(1,2,5)

(1,2,5,6,5)

Von 1 sind alle Knoten erreichbar

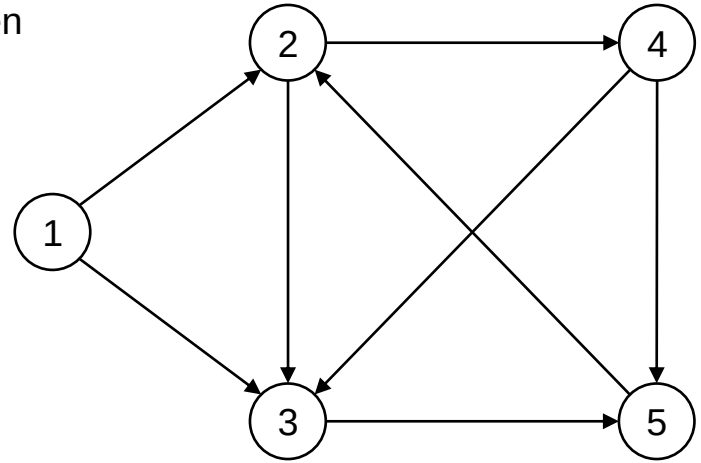
→ G ist zusammenhängend

Gerichtete Graphen

- Ein gerichteter Graph $G = (V, E)$ besteht aus
  - einer Menge V von n Knoten und
  - einer Menge E von m gerichteten Verbindungen zwischen den Knoten, die Pfeile heißen.

- Pfeil $(i, j) \in E$ mit $i, j \in V$ ist
  - positiv inzident mit Knoten i (Anfangsknoten) und
  - negativ inzident mit Knoten j (Endknoten).

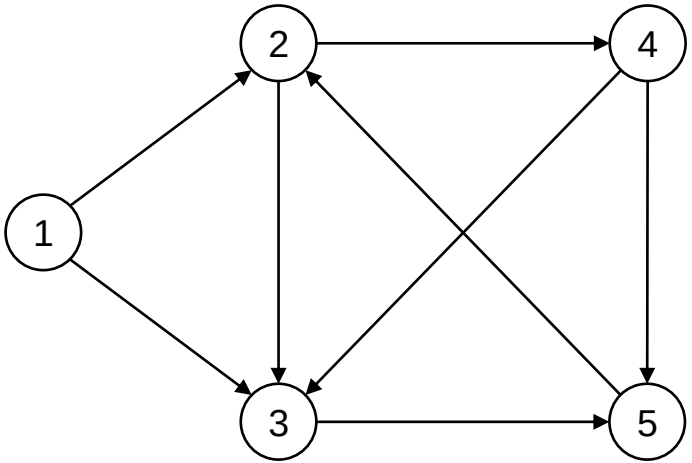
- Ungerichtete Verbindungen zwischen benachbarten Knoten i, j in gerichteten Graphen werden durch ein Pfeilpaar $(i, j), (j, i) \in E$ dargestellt.



Strukturen in gerichteten Graphen

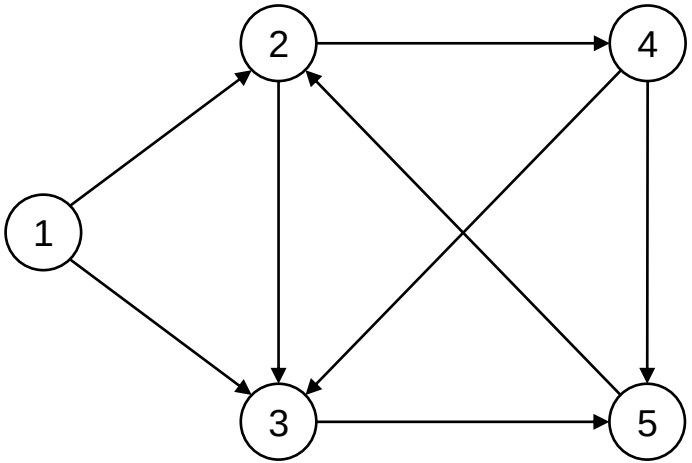
- Für Pfeil (i,j) heißt i **Vorgänger** von j und j **Nachfolger** von i.
- $P(i)$ - **Menge der Vorgänger** von i
- $N(i)$ - **Menge der Nachfolger** von i
- $grad^{+}(i)$ - **(positiver) Ausgangsgrad** ist Mächtigkeit von $N(i)$
- $grad^{-}(i)$ - **(negativer) Eingangsgrad** ist Mächtigkeit von $P(i)$
- **Gerichtete Pfade** und **gerichtete Wege** analog zu Pfaden und Wegen in ungerichteten Netzwerken definiert (unter Berücksichtigung der Pfeilorientierung)
- **Semiwege** bezeichnen Wege in gerichteten Graphen ohne Berücksichtigung der Pfeilorientierung

Strukturen in gerichteten Graphen



Knoten	*Vorgänger*	*Eingangsgrad*	*Nachfolger*	*Ausgangsgrad*
1	-	0	{2; 3}	2
2	{1; 5}	2	{3; 4}	2
3	{1; 2; 4}	3	{5}	1

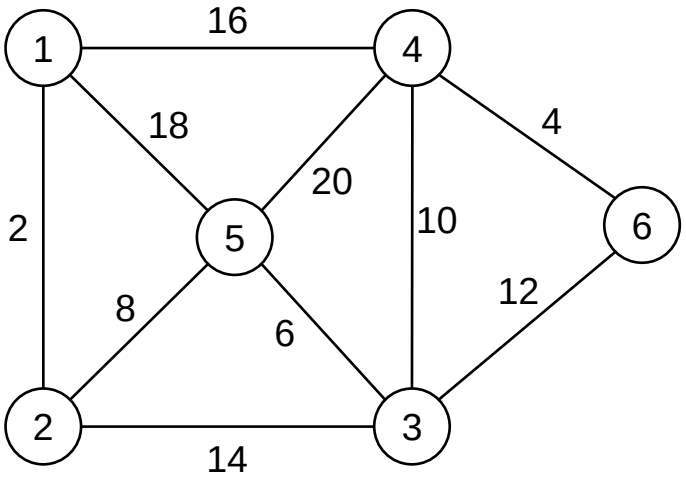
Strukturen in gerichteten Graphen



(1, 2, 4, 5, 2, 3)	Pfad
(1, 2, 4, 3)	Weg
(2, 3, 4, 5)	Semiweg
(2, 4, 5, 2)	Zyklus (geschlossener Weg)

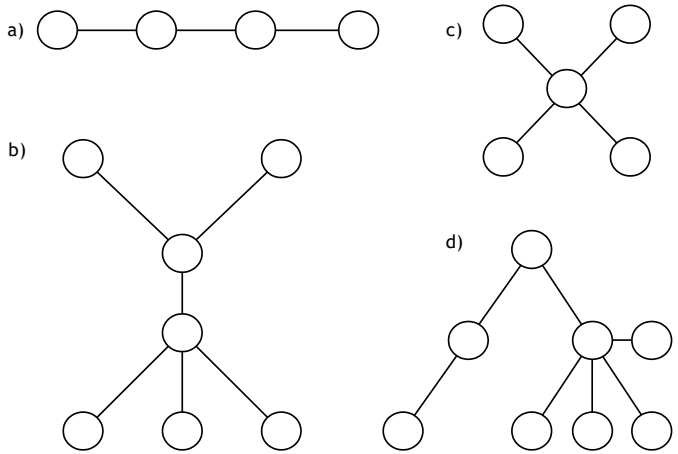
Gewichtete Graphen

- $G = (V, E)$, dessen Kanten / Pfeile eine *Gewichtung* c_{ij} besitzen ($c_{ij} \in \mathbb{R}$), bezeichnet man als *gewichteten Graphen*.
- z.B. Transportkosten pro ME von i nach j
- Es ergibt sich damit $G = (V, E, c)$



Bäume

- Definitionen zu Bäumen:
  - n Knoten bei $n-1$ Kanten
  - Ein Baum hat keinen Zyklus
  - Knoten mit dem Grad 1 heißen Blätter



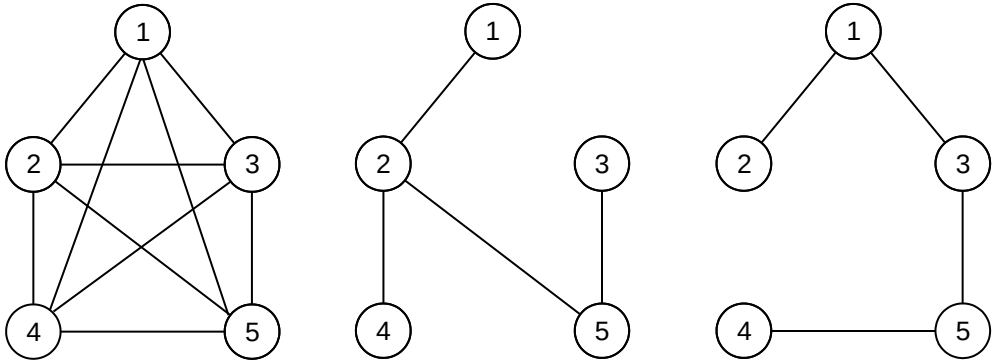
- Aufspannender Baum
  - Ein aufspannender Baum $T = (V, E_T)$ ist ein Teilgraph eines ungerichteten zusammenhängenden Graphen $G = (V, E)$,
  - der mit einer minimalen Anzahl von Kanten $E_T \subset E$
  - gerade noch den Zusammenhang zwischen den Knoten in *V* erhält.
  - Einen Graphen aus mehreren Bäumen nennen wir *Wald*

Überblick

1. Einführung und Grundlagen
2. **Minimale spannende Bäume in Graphen**
3. Kürzeste Wege in Graphen
4. Dynamisches Losgrößenproblem

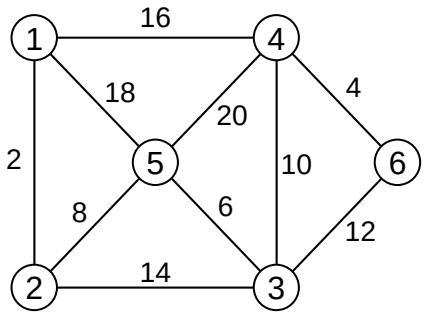
Spannende Bäume

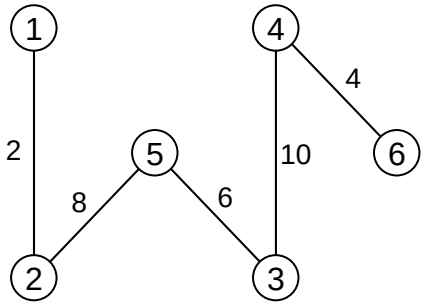
- Ein Baum T, der alle Knoten eines zusammenhängenden Graphen $G = (V, E)$ beinhaltet, wird spannender Baum genannt.
  - In einem vollständigen Graphen (jeweils zwei Knoten sind direkt durch eine Kante verbunden) existieren n^{n-2} spannende Bäume.
  - Für den Beispielgraphen unten links existieren 125 verschiedene spannende Bäume, zwei davon sind rechts und in der Mitte gezeigt.



Minimaler spannender Baum

- *Ziel*: In einem gewichteten, ungerichteten Graphen soll unter der Vielzahl von möglichen spannenden Bäumen derjenige aufspannende Baum bestimmt werden, dessen Gewichtssumme minimal ist
- Der minimale spannende Baum ist eindeutig, wenn die Kanten unterschiedliche Gewichte tragen.
- Praktische Bedeutung von Bäumen: Auslegung einer Infrastruktur bei Straßensystemen, Kommunikationsnetzwerken oder Versorgungsleitungen





Verfahren zur Bestimmung des minimalen spannenden Baumes

- Konstruktion:
  - Verfahren von Prim oder von Kruskal
  - Arbeiten mit sortierter Distanzliste der Kanten

- Kruskal (folgendes Beispiel):
  - Kanten in aufsteigender Sortierung einfügen, solange hierdurch kein Zyklus entsteht
    → jeweils Knoten durch eingefügte Kante anbinden
  - Aufbau eines Waldes, bis Wald *n*-1 Kanten hat und somit ein Baum ist

- Prim (hier nicht weiter betrachtet):
  - Ausgangspunkt: Endknoten der Kante mit der kleinsten Distanz
  - Schrittweise die inzidente Kante mit der jeweils kürzesten Distanz dem Baum zuschlagen

Verfahren von Kruskal

Initialisierung

1 Alle Kanten sind frei, d.h. nicht belegt

2 Wähle in der Liste Kante (i_1, j_1) mit den kleinsten Kosten

3 Setze Baum $T_1 = (i_1, j_1)$

4 Markiere die Kante (i_1, j_1) als belegt

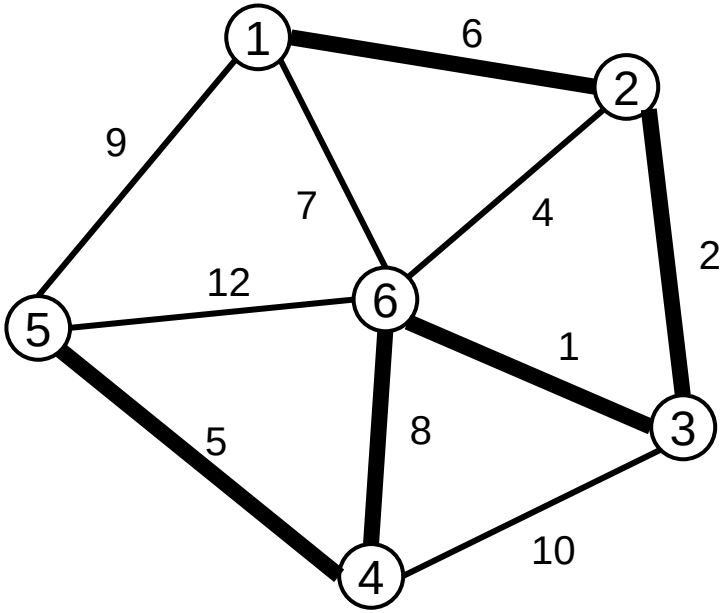
Verarbeitung

5 Gehe obige Liste von Kanten von der Spitze durch und nehme nacheinander alle freien Kanten (i, j) in den Wald auf, ohne einen Zyklus zu bilden. Markiere Kante (i, j) als belegt

Terminierung

6 Abbruch, wenn T_p $n - 1$ Kanten enthält, sonst gehe zu 5

Beispiel-Aufgabe: Verfahren von Kruskal



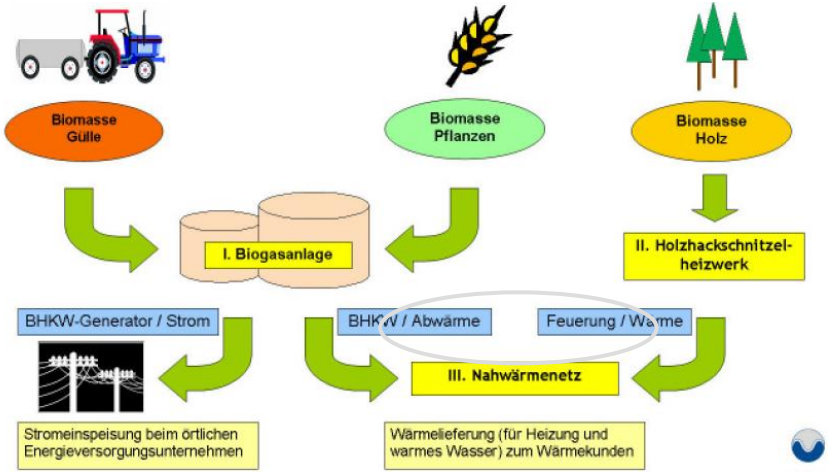
Kante	Kosten	drin?
(3,6)	1	✓
(2,3)	2	✓
(2,6)	4	✗
(4,5)	5	✓
(1,2)	6	✓
(1,6)	7	✗
(4,6)	8	✓
(1,5)	9	
(1,6)	10	
(5,6)	12	

Minimales Gesamtgewicht: 22

Beispiel: Bioenergiedorf Jühnde (1)

- Das Bioenergiedorf Jühnde
  - Interdisziplinäres Aktionsforschungsprojekt
  - Umstellung der Energieversorgung eines Dorfes
  - Eigenständige und kosteneffiziente Versorgung mit Strom und Wärme auf Basis von Biomasse

- Problemstellung
  - Stromnetz vorhanden
  - Aufbau eines Nahwärmenetzes

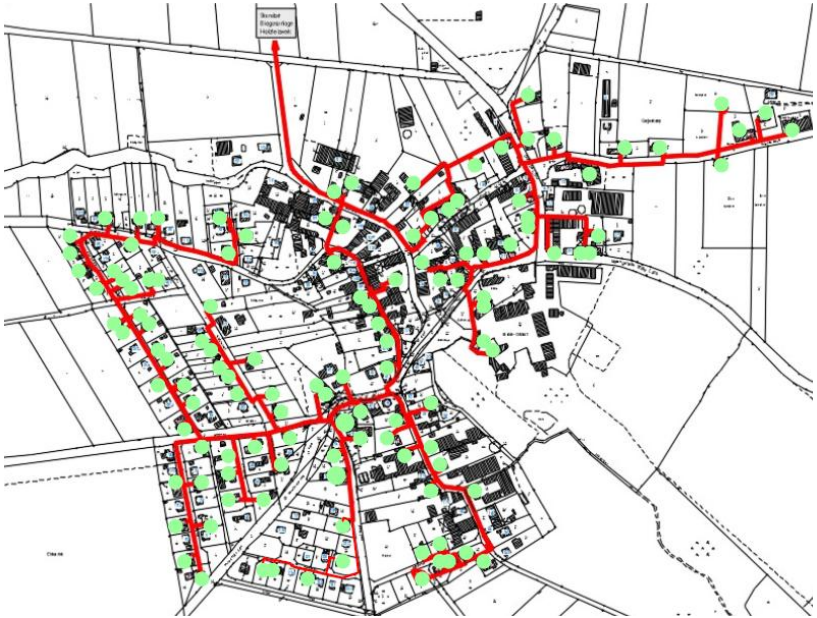


Beispiel: Bioenergiedorf Jühnde (2)

- Ausgangslage
  - Standort des Heizwerks vorgegeben
  - Anschluss von 140 Haushalten

- Ziel: Minimierung der Investitionskosten des Nahwärmenetzes

➔ KAB



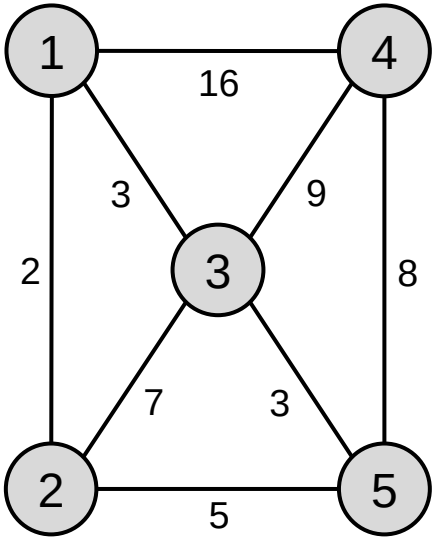
Überblick

1. Einführung und Grundlagen
2. Minimale spannende Bäume in Graphen
3. **Kürzeste Wege in Graphen**
4. Dynamisches Losgrößenproblem

Kürzeste Wege in Graphen

Anwendungsgebiete:

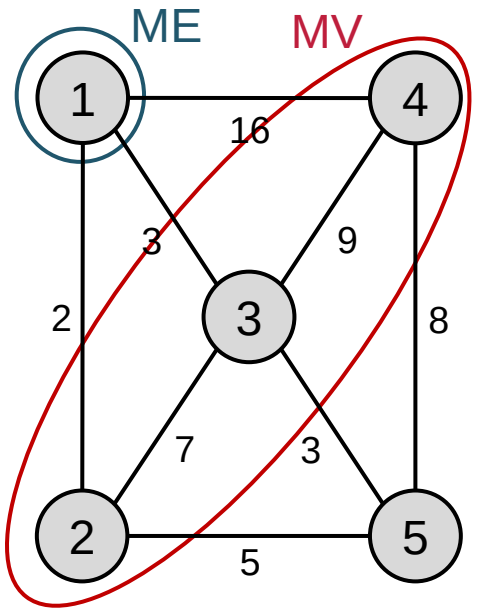
- Transport- und Tourenplanung
- Kostenminimale Flüsse in Netzwerken
- Optimales Packen von Rucksäcken
- Investitionsprogrammplanung
- Mehrperiodige Losgrößenplanung



Der Dijkstra-Algorithmus

- Der Dijkstra-Algorithmus bestimmt kürzeste Wege von einem Knoten zu allen anderen Knoten.

- Idee:
  - Eine Knotenmenge mit bereits bekannten Wegen (ME)
  - Eine benachbarte Kandidatenmenge (MV)
  - Schrittweise Kandidaten von MV nach ME verschieben und Nachbarschaft aktualisieren



Dijkstra-Algorithmus - Labeling

- Label-Setting: Sukzessive Besuch und Markierung von Knoten
- Labels werden durch Mengen *MV* und *ME* repräsentiert
  - *MV* – vorläufig markierte Knoten (zur Auswahl vorhalten)
  - *ME* – endgültig markierte Knoten (bereits abgearbeitet)
- Algorithmus:
  1. Nehme Knoten i in die Menge *MV* auf
  2. Knoten $i \in MV$ mit kleinster aktueller Distanz $d(i)$ auswählen, aus *MV* löschen, in *ME* aufnehmen
  3. Die Nachfolger $j \in N(i)$ zu *MV* hinzufügen falls $j \notin ME$
  4. Falls $MV \neq \emptyset$, gehe zu 2.
  5. Terminiere

Fortschreiben der Knoteninformation

- Variablen:
  - s – Startknoten
  - $d(i)$ – aktuell kürzeste bekannte Distanz von s zu i
  - $p(i)$ – Vorgängerknoten von i (von wo bin ich auf dem kürzesten Weg nach i gekommen; dient der Dokumentation)
  - *MV* – vorläufig markierte Knoten
  - *ME* – endgültig markierte Knoten

- Auswahlregel:
  - Wähle $i \in MV$, mit kleinster aktueller Distanz $d(i)$
- Fortschreiberegel:
  - Aktualisieren Information zu j in *MV* nur, wenn der aktuelle Besuch des Knotens die bisher bekannte Distanz $d(j)$ verringert

Dijkstra-Algorithmus - Pseudocode

Initialisierung

1 $d(s) \coloneqq 0,\ p(s) \coloneqq 0;\ MV\ \coloneqq \{s\};\ ME \coloneqq \emptyset$

2 $d(s) \coloneqq \infty$ für $j \coloneqq 1, \ldots, n;\ j \neq s$

Verarbeitung

3 WHILE $MV \neq\ \emptyset$ DO

4 Wähle ein $i \in MV$ mit $d(i) =\ Min\,\{d(p): p \in MV\}$ und setze $\mathrm{MV} \coloneqq \mathrm{MV} \backslash \{i\}$ und $\mathrm{ME} \coloneqq \mathrm{ME} \cup \{i\}$

5 Für alle Nachfolger $k \in N(i)$ mit $k \in N \backslash \mathrm{ME}$ prüfe, ob $d(k) > d(i) + d_{ik}$

6 IF $d(k) > d(i) + d_{ik}$, THEN

7 setze: $d(k) \coloneqq d(i) + d_{ik}; p(k) \coloneqq i; MV \coloneqq MV \cup \{k\}$

8 ENDIF

9 ENDWHILE

Terminierung

10 MV ist leer

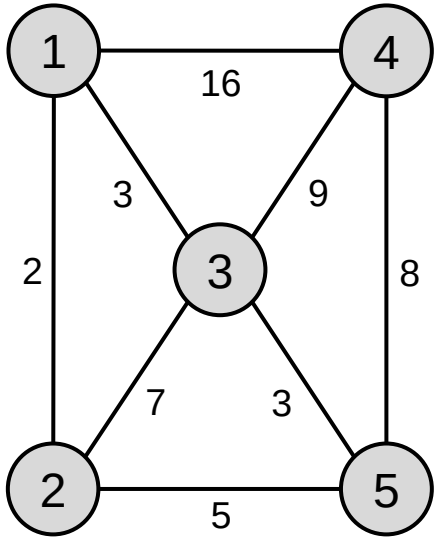
Beispiel zum Dijkstra-Algorithmus (1)

Initialisierung:

- $d(1) := 0$
- $d(2), \dots, d(5) := \infty$
- $p(1) := 0$
- MV $= \{1\}$
- *ME* $= \emptyset$

Verarbeitung:

- 5 Durchläufe

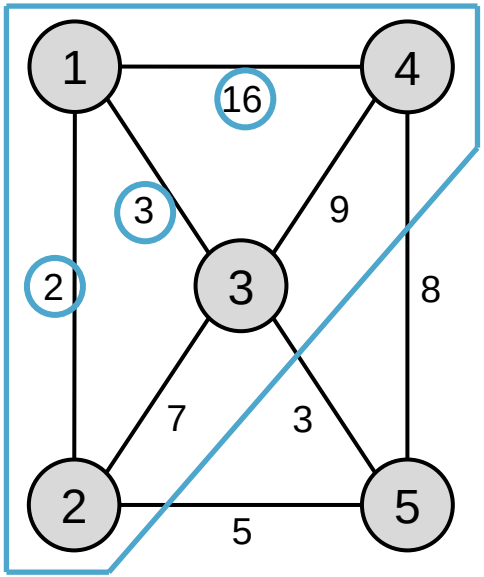


	$d(1)$	$d(2)$	$d(3)$	$d(4)$	$d(5)$	$P(1)$	$P(2)$	$P(3)$	$P(4)$	$P(5)$	MV	ME
Initialisierung	0	∞	∞	∞	∞	0					1	∅

Beispiel zum Dijkstra-Algorithmus (2)

<u>1. Durchlauf:</u>

- MV $= \{1\}$ ist nicht leer: Wähle Knoten $i = 1$
- MV := MV\$\{1\} = \emptyset$ und ME := $\emptyset$ AND $\{1\} = \{1\}$
- Nachfolger von 1: $j = 2,3,4$
- $j = 2$: $d(2) > d(1) + d_{12}$? Ja, weil $\infty > 0 + 2 = 2$
  - $d(2) = 2, p(2) = 1$
  - MV := $\emptyset$ AND $\{2\} = \{2\}$
- $j = 3$: $d(3) > d(1) + d_{13}$? Ja, weil $\infty > 0 + 3 = 3$
  - $d(3) = 3, p(3) = 1$
  - MV := $\{2\}$ AND $\{3\} = \{2,3\}$
- $j = 4$: $d(4) > d(1) + d_{14}$? Ja, weil $\infty > 0 + 16 = 16$
  - $d(4) = 16, p(4) = 1$
  - MV := $\{2,3\}$ AND $\{4\} = \{2,3,4\}$

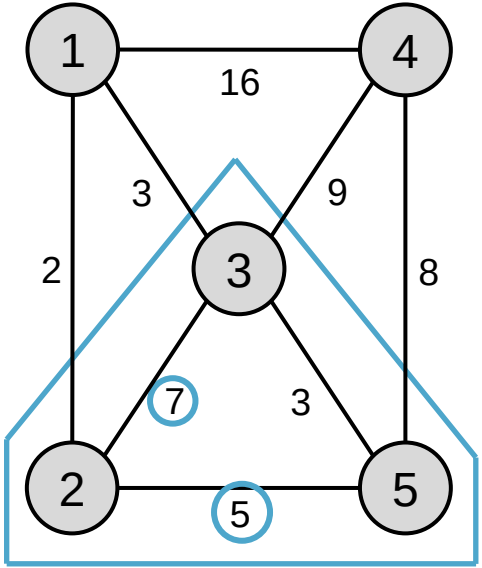


		$d(1)$	$d(2)$	$d(3)$	$d(4)$	$d(5)$	$P(1)$	$P(2)$	$P(3)$	$P(4)$	$P(5)$	MV	ME
1. Durchlauf	$i = 1, j = 2$	0	2	∞	∞	∞	0	1				2	1
	$i = 1, j = 3$	0	2	3	∞	∞	0	1	1			2,3	1
	$i = 1, j = 4$	0	2	3	16	∞	0	1	1	1		2,3,4	1

Beispiel zum Dijkstra-Algorithmus (3)

2. Durchlauf:

- MV $= \{2,3,4\}$ ist nicht leer: Wähle Knoten $i = 2$, da $d(2) = 2, d(3) = 3, d(4) = 16$
- MV := MV\$\{2\} = \{3,4\}$ und ME $:= \{1\}$ AND $\{2\} = \{1,2\}$
- Nachfolger von 2: $j = 3,5$
- $j = 3$: $d(3) > d(2) + d_{23}$? Nein, weil $3 < 2 + 7 = 9$
  - $d(3) = 3, p(3) = 1$
  - MV := $\{3,4\}$ AND $\{0\} = \{3,4\}$
- $j = 5$: $d(5) > d(2) + d_{25}$? Ja, weil $\infty > 2 + 5 = 7$
  - d(5) = 7, p(5) = 2
  - MV := $\{3,4\}$ AND $\{5\} = \{3,4,5\}$

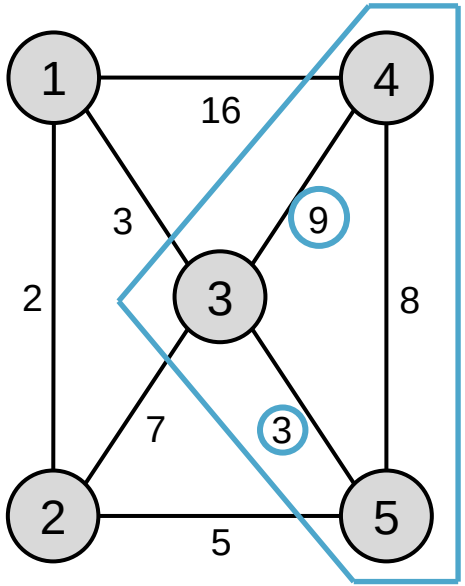


		$d(1)$	$d(2)$	$d(3)$	$d(4)$	$d(5)$	$P(1)$	$P(2)$	$P(3)$	$P(4)$	$P(5)$	MV	ME
2. Durchlauf	$i = 2, j = 3$	0	2	3	16	∞	0	1	1	1		3,4	1,2
	$i = 2, j = 5$	0	2	3	16	7	0	1	1	1	2	3,4,5	1,2

Beispiel zum Dijkstra-Algorithmus (4)

3. Durchlauf:

- MV $= \{3,4,5\}$ ist nicht leer: Wähle Knoten $i = 3$, da $d(3) = 3, d(4) = 16, d(5) = 7$
- MV := MV$\backslash\{3\} = \{4,5\}$ und ME := $\{1,2\}$ AND $\{3\} = \{1,2,3\}$
- Nachfolger von 3: $j = 4,5$
- $j = 4$: $d(4) > d(3) + d_{34}$? Ja, weil $16 > 3 + 9 = 12$
  - $d(4) = 12, p(4) = 3$
  - MV := $\{4,5\}$ AND $\{4\} = \{4,5\}$
- $j = 5$: $d(5) > d(3) + d_{35}$? Ja, weil $7 > 3 + 3 = 6$
  - $d(5) = 6, p(5) = 2$
  - MV := $\{4,5\}$ AND $\{5\} = \{4,5\}$



		$d(1)$	$d(2)$	$d(3)$	$d(4)$	$d(5)$	$P(1)$	$P(2)$	$P(3)$	$P(4)$	$P(5)$	MV	ME
3. Durchlauf	$i = 3, j = 4$	0	2	3	12	7	0	1	1	3	2	4,5	1,2,3
	$i = 3, j = 5$	0	2	3	12	6	0	1	1	3	3	4,5	1,2,3

Beispiel zum Dijkstra-Algorithmus (5)

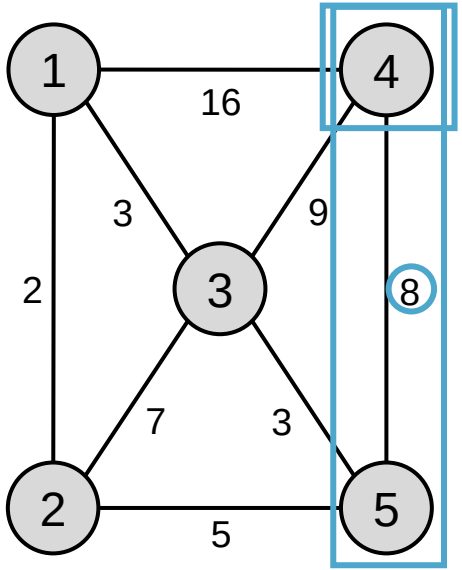
4. Durchlauf:

- MV $= \{4,5\}$ ist nicht leer: Wähle Knoten $i = 5$, da $d(5) = 6, d(4) = 12$
- MV :=MV$\backslash\{5\} = \{4\}$, ME $:= \{1,2,3\}$ AND $\{5\} = \{1,2,3,5\}$
- Nachfolger von 5: $j = 4$
- $j = 4$: $d(4) > d(5) + d_{54}$? Nein, weil $12 > 6 + 8 = 14$
  - $d(4) = 12, p(4) = 3$
  - MV $:= \{4\}$ AND $\{4\} = \{4\}$

5. Durchlauf:

- MV $= \{4\}$ ist nicht leer: Wähle Knoten $i = 4$
- MV=MV$\backslash\{4\} = \emptyset$ und ME $:= \{1,2,3,5\}$ AND $\{4\} = \{1,2,3,5,4\}$:
- 4 hat keine Nachfolger!

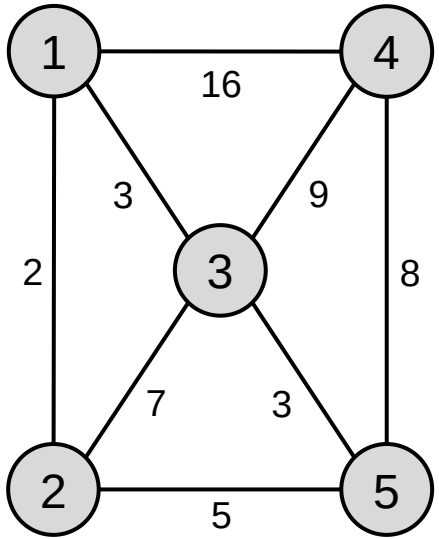
Abbruch: MV ist leer.



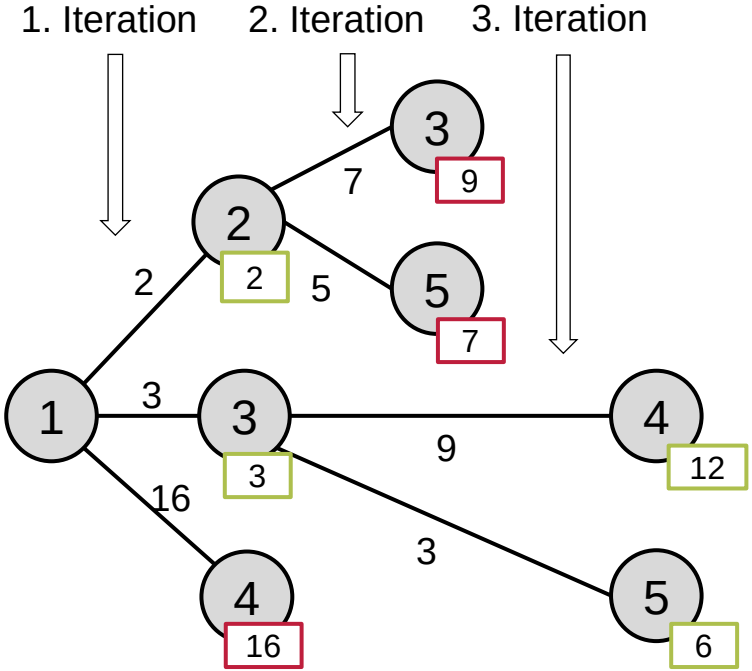
		$d(1)$	$d(2)$	$d(3)$	$d(4)$	$d(5)$	$P(1)$	$P(2)$	$P(3)$	$P(4)$	$P(5)$	MV	ME
4. Durchlauf	$i = 5, j = 4$	0	2	3	12	6	0	1	1	3	3	4	1,2,3,5
5. Durchlauf	$i = 4$	0	2	3	12	6	0	1	1	3	3	∅	1,2,3,5,4

Beispiel zum Dijkstra-Algorithmus (6)

Ausgangsgraph:



Kürzeste Wege durch Dijkstra:



Beispiel zum Dijkstra-Algorithmus (7)

Verarbeitung:

		$d(1)$	$d(2)$	$d(3)$	$d(4)$	$d(5)$	$P(1)$	$P(2)$	$P(3)$	$P(4)$	$P(5)$	MV	ME
Initialisierung		0	∞	∞	∞	∞	0					1	$\varnothing$
1. Durchlauf	$i = 1, j = 2$	0	2	∞	∞	∞	0	1				2	1
	$i = 1, j = 3$	0	2	3	∞	∞	0	1	1			2,3	1
	$i = 1, j = 4$	0	2	3	16	∞	0	1	1	1		2,3,4	1
2. Durchlauf	$i = 2, j = 3$	0	2	3	16	∞	0	1	1	1		3,4	1,2
	$i = 2, j = 5$	0	2	3	16	7	0	1	1	1	2	3,4,5	1,2
3. Durchlauf	$i = 3, j = 4$	0	2	3	12	7	0	1	1	3	2	4,5	1,2,3
	$i = 3, j = 5$	0	2	3	12	6	0	1	1	3	3	4,5	1,2,3
4. Durchlauf	$i = 5, j = 4$	0	2	3	12	6	0	1	1	3	3	4	1,2,3,5
5. Durchlauf	$i = 4$	0	2	3	12	6	0	1	1	3	3	$\varnothing$	1,2,3,5,4

Überblick

1. Einführung und Grundlagen
2. Minimale spannende Bäume in Graphen
3. Kürzeste Wege in Graphen
4. **Dynamisches Losgrößenproblem**

Dynamisches Losgrößenproblem

Problemstellung:

- Der Bedarf d_t an Produkten ist $40, 20, 10, 30, 50$ und 20 Einheiten in den nächsten $t = 1, \ldots, 6$ Wochen.
- Rüstkosten werden mit $c_B = 50$ € und Lagerhaltungskosten mit $c_L = 1$ € / Einheit und Woche angenommen.
- Welche Menge q_t soll in welcher Woche produziert werden, so dass die Gesamtkosten minimal sind?

α \ β	1	2	3	4	5	6
1	50	70	90	180	380	480
2		50	60	120	270	350
3			50	80	180	240
4				50	100	140
5					50	70
6						50

Kosten $K_{\alpha,\beta} = c_B + c_L \cdot \sum_{t=\alpha}^{\beta} d_t \cdot (t - \alpha)$

mit

α: Produktionswoche

β: Reichweite des Loses

Dynamisches Losgrößenproblem: Berechnung der Kosten

Variablen

- α: Produktionswoche
- β: Reichweite des Loses
- $K_{\alpha,\beta}$: Kosten bei Produktion in Woche α mit Reichweite von β Wochen

β

α	1	2	3	4	5	6
1	50	70	90	180	380	480
2		50	60	120	270	350
3			50	80	180	240
4				50	100	140
5					50	70
6						50

Beispiele für die Berechnung

$K_{1,1} = 50$

$K_{1,2} = 50 + 20 \cdot 1 = 70$

$K_{1,3} = 50 + 20 \cdot 1 + 10 \cdot 2 = 90$

$K_{1,4} = 50 + 20 \cdot 1 + 10 \cdot 2 + 30 \cdot 3 = 180$

$\ldots.$

$K_{4,5} = 50 + 50 \cdot 1 = 100$

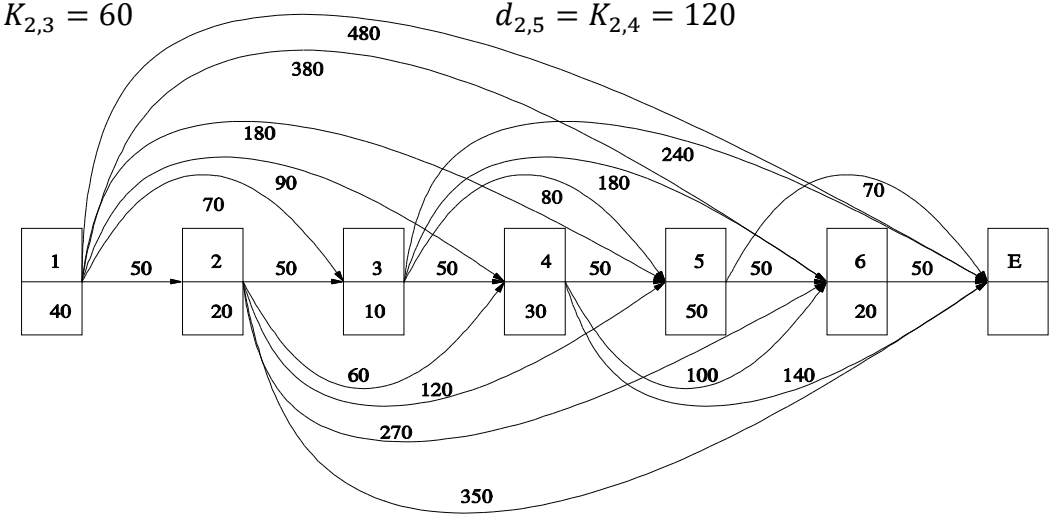
$K_{4,6} = 50 + 50 \cdot 1 + 20 \cdot 2 = 140$

Dynamisches Losgrößenproblem: Kürzeste-Wege-Problem

- Nutzung der Werte $K_{\alpha,\beta}$ als Distanzmatrix für einen gerichteten Graphen:
  - Ein Knoten je Woche t
  - Eine Kante von jedem Knoten zu jedem Knoten für eine spätere Woche
  - $K_{\alpha,\beta}$ wird als Distanz d_{ij} von Knoten $i = \alpha$ zu Knoten $j = (\alpha + \beta)$ verwendet d.h.:
    - $d_{1,2} = K_{1,1} = 50$ $\quad d_{1,3} = K_{1,2} = 70$ $\quad d_{1,4} = K_{1,3} = 90$
    - $d_{2,3} = K_{2,1} = 50$ $\quad d_{2,4} = K_{2,3} = 60$ $\quad d_{2,5} = K_{2,4} = 120$

α \ β

	1	2	3	4	5	6
1	50	70	90	180	380	480
2		50	60	120	270	350
3			50	80	180	240
4				50	100	140
5					50	70
6						50

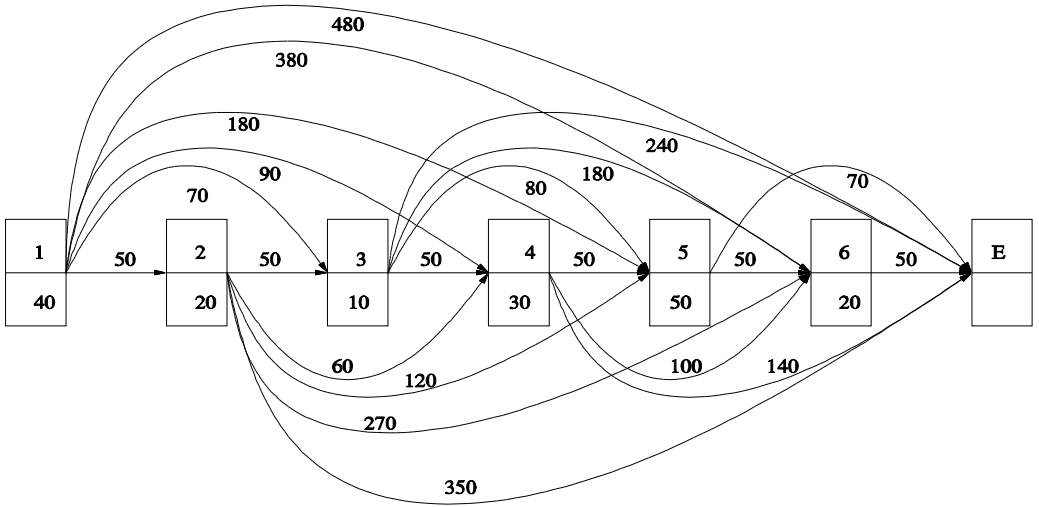


Dynamisches Losgrößenproblem: Lösung

Ergebnis

- Kürzester Weg: (1,4,5,E)
- Kosten: $d_{1,4} + d_{4,5} + d_{5,E} = 90 + 50 + 70 = 210$
- Produzierte Menge: $q_1 = 70, q_2 = 0, q_3 = 0, q_4 = 30, q_5 = 70, q_6 = 0$

α \ β	1	2	3	4	5	6
1	50	70	90	180	380	480
2		50	60	120	270	350
3			50	80	180	240
4				50	100	140
5					50	70
6						50



Zusammenfassung

- Grundlagen von Graphentheorie
  - Gerichtete und ungerichtete Graphen
  - Gewichtete Graphen
  - Bäume

- Minimal spannende Bäume in Graphen
  - Verfahren von Kruskal

- Kürzeste Wege in Graphen
  - Dijkstra-Algorithmus

- Dynamisches Losgrößenproblem