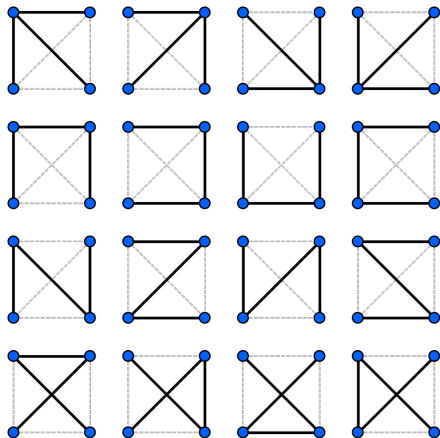


Kapitel 5

Bäume und Minimalgerüste



Inhalt

5 Bäume und Minimalgerüste

- Charakterisierung von Minimalgerüsten
- Berechnung von Minimalgerüsten
- Minimalgerüste und TSP
- Anwendung von Minimalgerüsten

Gerüst

Definition 5.1

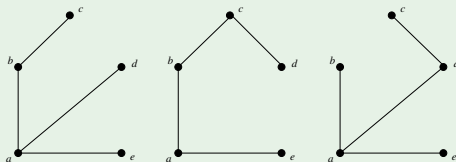
Es ein $G = (V, E)$ ein zusammenhängender Graph.

$H = (V, E')$ heißt **Gerüst** von G gdw. wenn H ein Baum ist und $E' \subseteq E$ gilt.

Ein Gerüst ist also ein **zusammenhängender, kreisfreier, aufspannender Untergraph** von G .

Beispiel 5.2

Einige Gerüste für das Haus vom Nikolaus:



Minimalgerüst

Definition 5.3

Es sei $G = (V, E)$ ein zusammenhängender Graph mit einer Kantengewichtsfunktion $w : E \rightarrow \mathbb{R}$.

- Für $F \subseteq E$ heißt

$$w(F) := \sum_{e \in F} w(e)$$

das **Gewicht der Kantenmenge** F .

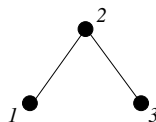
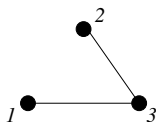
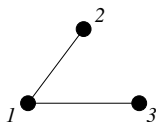
- Es sei $H = (V, F)$ ein Gerüst von G . Dann ist $w(H) := w(F)$ das **Gewicht des Gerüstes** H .
- Ein Gerüst H von G heißt **Minimalgerüst** von G gdw. $w(H) \leq w(H')$ für alle Gerüste H' von G .

Anzahl an Gerüsten

Satz 5.4 (Cayley)

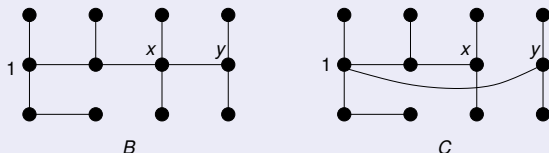
Es gibt n^{n-2} verschiedene Gerüste für den vollständigen Graphen K_n (mit n Knoten).

Bemerkung: Verschieden bedeutet hier wirklich verschieden und nicht "nichtisomorph".



Beweis.

- Idee: **Prinzip der doppelten Abzählung**: In jeder Matrix ist die Summe der Zeilensummen gleich der Summe der Spaltensummen.
- Sei $t(n, k)$ die Anzahl der Bäume auf $V = \{1, \dots, n\}$, in denen der Knoten 1 den Grad k hat.
- Es wird eine Formel für $t(n, k)$ bestimmt. Das Ergebnis erfolgt durch Aufsummieren über alle k .
- B sei ein Baum mit $\deg(1) = k - 1$, C sei ein Baum mit $\deg(1) = k$. (B, C) heißt **verwandtes Paar** gdw. C aus B entsteht, indem eine Kante $\{x, y\}$ entfernt und eine Kante $\{1, y\}$ eingefügt wird.



Fortsetzung Beweis.

- Es seien $B_1, B_2, \dots, B_{t(n, k-1)}$ die Bäume mit $\deg(1) = k - 1$ und $C_1, C_2, \dots, C_{t(n, k)}$ die Bäume mit $\deg(1) = k$.
- Die $t(n, k - 1) \times t(n, k)$ Matrix $A = (a_{ij})$ sei definiert durch:

$$a_{ij} := \begin{cases} 1 & \text{falls } (B_i, C_j) \text{ verwandtes Paar} \\ 0 & \text{sonst} \end{cases}$$

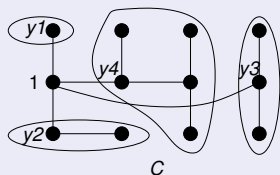
- Aus B_i kann jede der $n - 1$ Kanten entfernt werden, ausgenommen die $k - 1$ mit 1 inzidenten Kanten.

In der i -ten Zeile von A stehen so viele Einsen, wie es verwandte Paare (B_i, C) gibt, also genau $(n - 1) - (k - 1) = n - k$.

Summe der Zeilensummen von $A = t(n, k - 1)(n - k)$.

Fortsetzung Beweis.

- In der j -ten Spalte von A stehen so viele Einsen, wie es verwandte Paare (B, C_j) gibt.
- Die in C_j mit 1 verbundenen Knoten seien y_1, \dots, y_k .



- Entfernt man eine der Kanten $\{1, y_r\} (1 \leq r \leq k)$, so entsteht ein Graph mit zwei ZHKs. V_r sei die ZHK, die y_r enthält und $n_r := |V_r|$.
- Ein verwandtes Paar (B, C_j) entsteht genau dann, wenn $B = (C_j \setminus \{\{1, y_r\}\}) \cup \{\{x, y_r\}\}$ gilt, wobei x einer der $n - 1 - n_r$ Knoten in $V \setminus (\{1\} \cup V_r)$ ist.

Fortsetzung Beweis.

- Es gibt also genau $(n - 1 - n_1) + \dots + (n - 1 - n_k) = (k - 1)(n - 1)$ Einsen in der j -ten Spalte von A .

Summe der Spaltensummen von $A = t(n, k)(k - 1)(n - 1)$.

- Das Prinzip der doppelten Abzählung liefert

$$t(n, k - 1)(n - k) = t(n, k)(k - 1)(n - 1)$$

bzw.

$$t(n, k - 1) = (n - 1) \frac{k - 1}{n - k} t(n, k).$$

- Ausgehend von $t(n, n - 1) = 1$ ergibt sich durch Induktion

$$t(n, n - i) = (n - 1)^{i-1} \binom{n - 2}{i - 1}.$$

Fortsetzung Beweis.

- Für die Anzahl $t(n)$ aller Bäume ergibt sich somit

$$\begin{aligned}t(n) &= \sum_{i=1}^{n-1} t(n, n-i) \\ &= \sum_{i=1}^{n-1} (n-1)^{i-1} \binom{n-2}{i-1} \\ &= \sum_{i=0}^{n-2} (n-1)^i \binom{n-2}{i} \\ &= ((n-1) + 1)^{n-2} = n^{n-2}.\end{aligned}$$

Berechnung von Minimalgerüsten

Lemma 5.5

Es sei $G = (V, E)$ ein zusammenhängender Graph mit einer Kantengewichtsfunktion $w : E \rightarrow \mathbb{R}$. Weiterhin sei $U \subseteq V$ und e_0 eine Kante zwischen U und $V \setminus U$ mit minimalem Gewicht.

Dann existiert ein Minimalgerüst für G , das die Kante e_0 enthält.

Beweis.

Falls ein Minimalgerüst T_0 die Kante e_0 nicht enthält, so nehmen wir e_0 zu T_0 und entfernen eine Kante e_1 , die U und $V \setminus U$ verbindet.

Wegen der Minimalitätseigenschaft von e_0 erhöhen wir damit nicht das Gewicht des Gerüstes.

Der Algorithmus von Prim (1)

- Wir beginnen mit einem beliebigen Knoten v , d.h. $U := \{v\}$.
- In einem Iterationsschritt berechnen wir für alle $v \in V \setminus U$ den Knoten w , der am nächsten zu einem Knoten in U liegt.
- Dieser Knoten w wird selektiert, die entsprechende Kante wird in den Baum aufgenommen und w wird in U aufgenommen.
- Dies setzen wir fort, bis alle Knoten in U sind.

Der Algorithmus von Prim (2)

Algorithmus 5.6 (Prim)

Es sei $G = (V, E)$ ein zusammenhängender Graph mit $V = \{1, \dots, n\}$ und Kantengewichtsfunktion w . Es gelte $w(\{i, j\}) = \infty$, falls $\{i, j\} \notin E$. Der Algorithmus berechnet ein Minimalgerüst in B .

```

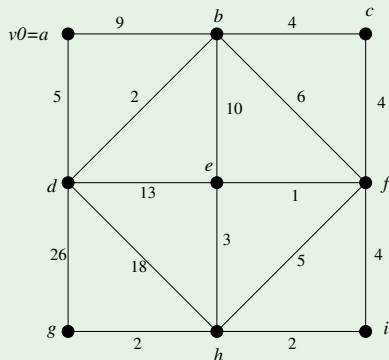
U := {1};
for i := 2 to n do  $d_U(i) := w(\{1, i\})$ ;
while  $U \neq V$  do
  bestimme Knoten  $u$  mit  $d_U(u) = \min\{d_U(v) : v \in V \setminus U\}$ ;      (*)
  bestimme Kante  $e := \{x, u\}$  mit  $w(e) = d_U(u)$  und  $x \in U$ ;      (*)
   $B := B \cup \{e\}$ ;
   $U := U \cup \{u\}$ ;
  for all  $v \in V \setminus U$  do  $d_U(v) := \min\{d_U(v), w(\{u, v\})\}$ ;    (**)
end

```

Der Algorithmus von Prim (3)

Beispiel 5.7

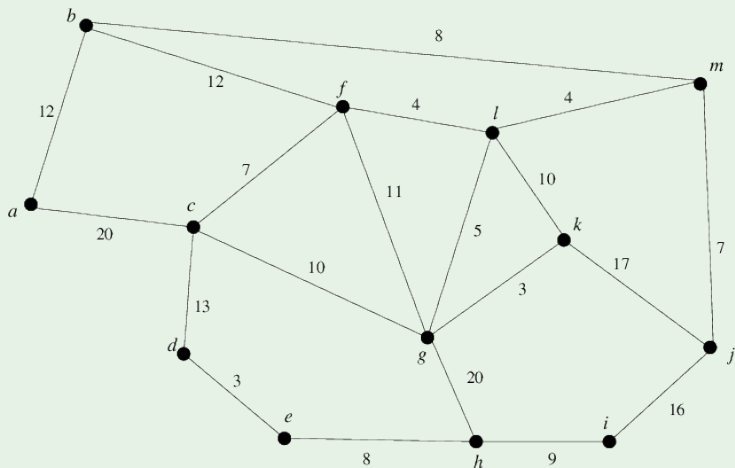
Wir betrachten den Graphen:



Es ergibt sich in dieser Reihenfolge:

$$B = \{\{a, d\}, \{b, d\}, \{b, c\}, \{c, f\}, \{e, f\}, \{e, h\}, \{g, h\}, \{h, i\}\}.$$

Beispiel 5.8



Satz 5.9

Algorithmus 5.6 berechnet ein Minimalgerüst in Zeit $O(|V|^2)$.

Beweis.

Es sind zwei Dinge zu beweisen:

- **Korrektheit des Algorithmus**

Wir zeigen induktiv: Nach jeder Ausführung von (*) (bzw. nach (**)) gibt $d_U(v)$ für alle $v \in V \setminus U$ den Abstand zu einem nächstgelegenen Knoten $u \in U$ an.

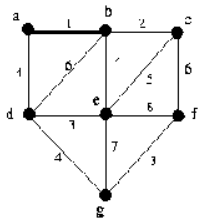
Korrektheit des Algorithmus folgt dann mit Lemma 4.2. Tafel .

- **Laufzeit**

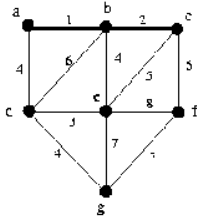
- ▶ Die dominieren Schritte innerhalb der While-Schleife: (*) und (**).
- ▶ Die While-Schleife wird $|V|$ -mal durchlaufen.
- ▶ Schritte (*) und (**) können in $O(|V|)$ ausgeführt werden.

Algorithmus von Kruskal (1)

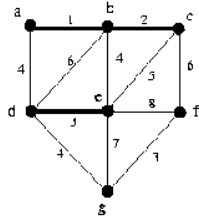
- Beim Algorithmus von Prim baut man ausgehend von einem Knoten den Baum sukzessive auf.
- Beim Algorithmus von Kruskal beginnt man stattdessen mit den einzelnen Knoten. Diese stellen einen Wald dar.
- Man versucht nun, diese Wälder optimal zu einem Baum zusammenzusetzen.
- Hierzu sortiert man zunächst alle Kanten aufsteigend nach ihrer Länge.
- Man beginnt, indem man die kürzeste Kante in den Wald aufnimmt. Der Wald hat jetzt eine ZHK weniger.
- In Iteration i : Falls die i -te Kante zu einem Kreis führen würde, verwirft man sie. Andernfalls nimmt man sie in den Wald auf, wodurch sich wiederum die Anzahl der ZHKs verringert.
- Dies macht man solange, bis ein aufspannender Baum entstanden ist.
- Dies ist genau dann der Fall, wenn man $n - 1$ Kanten aufgenommen hat (vgl. Satz 1.34 und Satz 1.42).



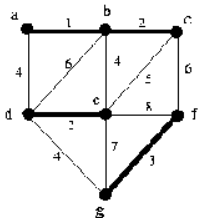
(a)



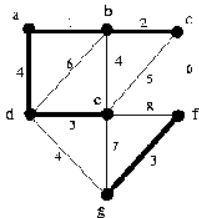
(b)



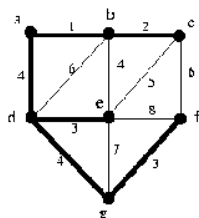
(c)



(d)



(e)



(f)

Algorithmus von Kruskal (2)

Algorithmus 5.10 (Kruskal)

Vorraussetzungen, Ein- und Ausgabe wie beim Algorithmus von Prim.

$B := \emptyset; ZHK := \emptyset; i := 0;$

$L :=$ Liste der Kanten aufsteigend sortiert nach ihrer Länge;

for all $v \in V$ *do* $ZHK := ZHK \cup \{\{v\}\};$

while $|ZHK| > 1$ *do*

$i := i + 1;$

Es sei $\{v, w\}$ *das* i -*te Element von* L ;

if v *und* w *gehören zu verschiedenen Komponenten* K_1 *und* K_2 *in* ZHK

$ZHK := (ZHK \setminus \{K_1, K_2\}) \cup \{\{K_1 \cup K_2\}\};$

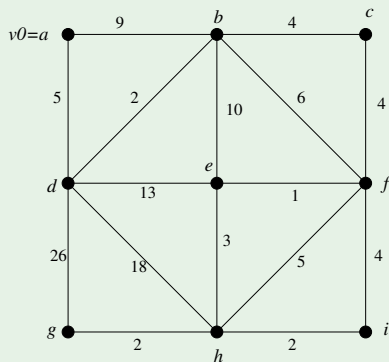
$B := B \cup \{v, w\};$

end

end

Beispiel 5.11

Wir betrachten den Graphen:



Es ergibt sich in dieser Reihenfolge:


$$B = \{\{e, f\}, \{b, d\}, \{g, h\}, \{h, i\}, \{e, h\}, \{b, c\}, \{c, f\}, \{a, d\}\}.$$

Algorithmus von Kruskal (3)

Satz 5.12

Algorithmus 5.10 berechnet ein Minimalgerüst in Zeit $O(|E| \log |V|)$.

Beweis.

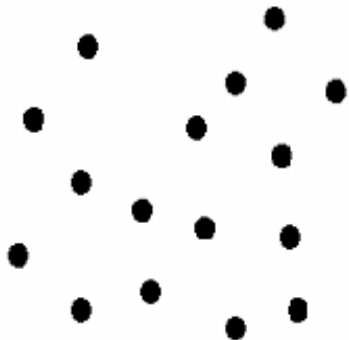
Die Korrektheit des Algorithmus von Kruskal kann induktiv durch wiederholte Anwendung von Lemma 5.5 gezeigt werden. Tafel .

Zeitaufwand: Der dominierende Schritt ist die Sortierung. Der Aufwand zur Sortierung der Kanten ist

$$O(|E| \log |E|) = O(|E| \log |V|^2) = O(|E| 2 \log |V|) = O(|E| \log |V|).$$

Für die effiziente Vereinigung der ZHKs und den vorangehenden Test benutzt man Datenstrukturen für das sogenannte **Union-Find-Problem**.

Minimalgerüste und TSP



Untere Schranke für TSP

Satz 5.13

Es sei $G = (V, E)$ ein vollständiger Graph mit Gewichtsfunktion $w : E \rightarrow \mathbb{N}$. Weiterhin sei TSP eine optimale TSP-Tour und MST sei ein Minimalgerüst.

Dann gilt:

$$w(MST) \leq w(TSP)$$

Beweis.

- Wenn man aus TSP eine beliebige Kante entfernt, erhält man einen hamiltonschen Weg HP .
- Jeder hamiltonsche Weg ist ein Gerüst.
- Also folgt $w(TSP) \geq w(HP) \geq w(MST)$.

Obere Schranke für TSP

Satz 5.14

Es sei $G = (V, E)$ ein vollständiger Graph mit Gewichtsfunktion $w : E \rightarrow \mathbb{N}$, für die die Dreiecksungleichung gilt, d.h.


$$w(\{i, k\}) \leq w(\{i, j\}) + w(\{j, k\}) \quad \forall i, j, k \in V$$

Weiterhin sei TSP eine optimale TSP-Tour und MST sei ein Minimalgerüst. Dann gilt:

$$w(MST) \leq w(TSP) \leq 2 w(MST)$$

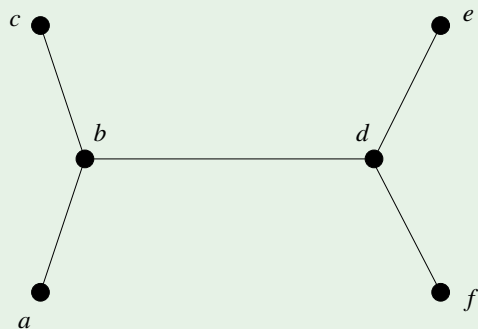
Beweis.

Die linke Ungleichung folgt aus Satz 5.13.

Für den Beweis der rechten Ungleichung konstruieren wir mit Hilfe der Tiefensuche eine Tour aus dem Minimalgerüst MST . Tafel .

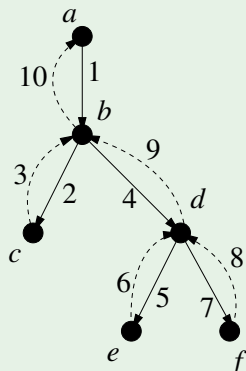
TSP-Tour mittels Tiefensuche auf MST (1)

Beispiel 5.15



v	a	b	c	d	e	f
$t(v)$	1	2	3	4	5	6

Tour: a, b, c, d, e, f, a

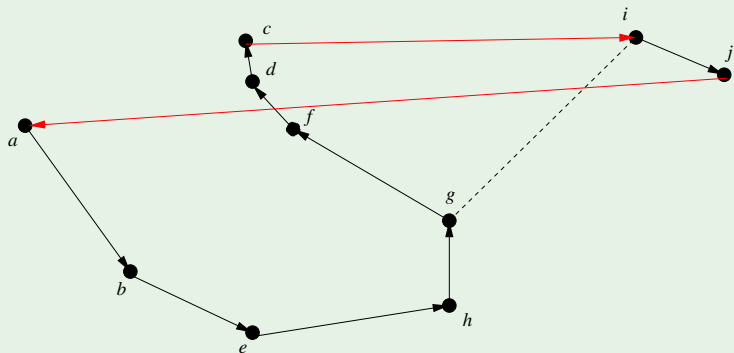


TSP-Tour mittels Tiefensuche auf MST (2)

- Das Verfahren aus Satz 5.14 liefert uns nicht nur eine **obere Schranke** für die Länge einer optimalen Tour,
- sondern auch ein effizientes Konstruktionsverfahren für eine Tour mit Länge $\leq 2 w(TSP)$.
- Die Tour ergibt sich dabei direkt aus der DFS-Numerierung.
- Alternative:
 - ▶ Jede Kante des *MST* wird durch zwei gerichtete Kanten ersetzt,
 - ▶ mit dem Algorithmus von Hierholzer wird ein Eulerkreis berechnet und
 - ▶ die Knoten werden in der Reihenfolge wie im Eulerkreis “angefahren”.

Verbesserung einer Tour (1)

Beispiel 5.16

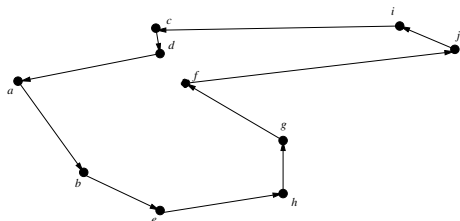
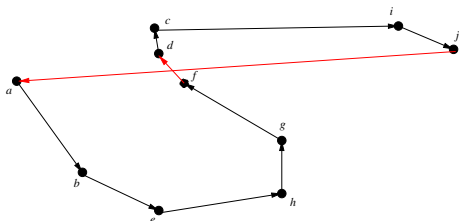


v	a	b	c	d	e	f	g	h	i	j
$t(v)$	1	2	8	7	3	6	5	4	9	10

Tour: $a, b, e, h, g, f, d, c, i, j, a$

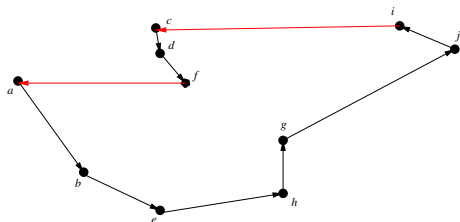
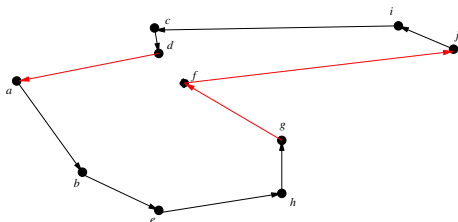
Verbesserung einer Tour (2)

- Anschließend können sogenannte **Verbesserungsverfahren** angewendet werden.
- Der Algorithmus **2-opt** sucht beispielsweise nach zwei Kanten, die gegen zwei andere Kanten ausgetauscht werden können, so dass eine kürzere Tour entsteht.
- Im euklidischen Fall werden damit alle **Kreuzungen** eliminiert.

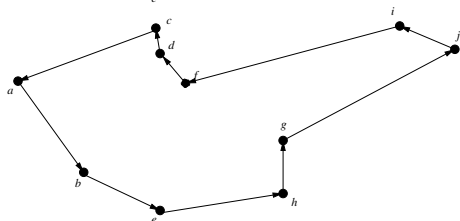


Verbesserung einer Tour (3)

- Analog sucht der Algorithmus **3-opt** nach drei Kanten für einen Austausch.



- Der nächste 2-opt-Tausch liefert eine optimale Tour.

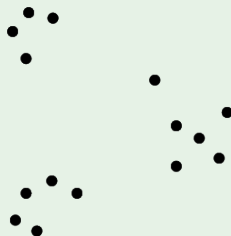


Clusteranalyse mit Minimalgerüsten (1)

Beispiel 5.17 (Clusteranalyse)

Gegeben sei eine Menge von Punkten, z. B. im \mathbb{R}^2 , die gewisse Häufungen aufweist.

Wie kann man diese **Häufungen** algorithmisch erkennen?



Clusteranalyse mit Minimalgerüsten (2)

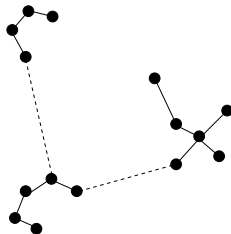
- In Häufungen liegen Knoten nahe zusammen.
- Ansatz: Wir legen zwischen zwei Knoten eine Kante, wenn sie nicht zu weit voneinander entfernt sind ($\leq \alpha$).

$$E := \{\{v, w\} \mid d(v, w) \leq \alpha\}$$

- Hierbei ist $d(v, w)$ die Entfernung auf Basis einer Metrik $d(\cdot, \cdot)$ und α ist die maximal erlaubte Entfernung.
- Die Zusammenhangskomponenten des entsprechenden Graphen sehen wir dann als Häufungen an.
- Zur Berechnung dieser können wir den Algorithmus von Kruskal verwenden.

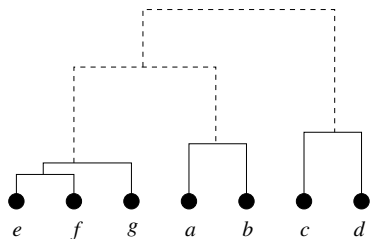
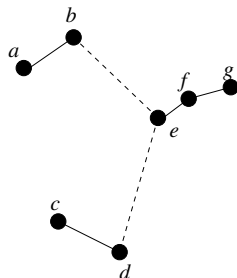
Clusteranalyse mit Minimalgerüsten (3)

- Berechnung eines Minimalgerüstes mit dem Algorithmus von Kruskal und
- Elimination der Kanten mit Länge $> \alpha$.



Clusteranalyse mit Minimalgerüsten (4)

- Die Benutzung des Algorithmus von Kruskal hat den Vorteil, dass die Kanten aufsteigend nach Ihrer Länge selektiert werden.
- Die Aufnahme einer Kante in den MST können wir als die Verschmelzung zweier Cluster ansehen.
- Der Verlauf der Cluster-Verschmelzung wird mit einem sogenannten **Dendrogramm** visualisiert.



Zusammenfassung des Kapitels

- Berechnung von Minimalgerüsten
- Algorithmen: Prim, Kruskal
- Anwendungen:
 - ▶ Gütegarantie beim Traveling Salesman Problem
 - ▶ Heuristik zur Konstruktion einer TSP-Tour
 - ▶ Clusteranalyse