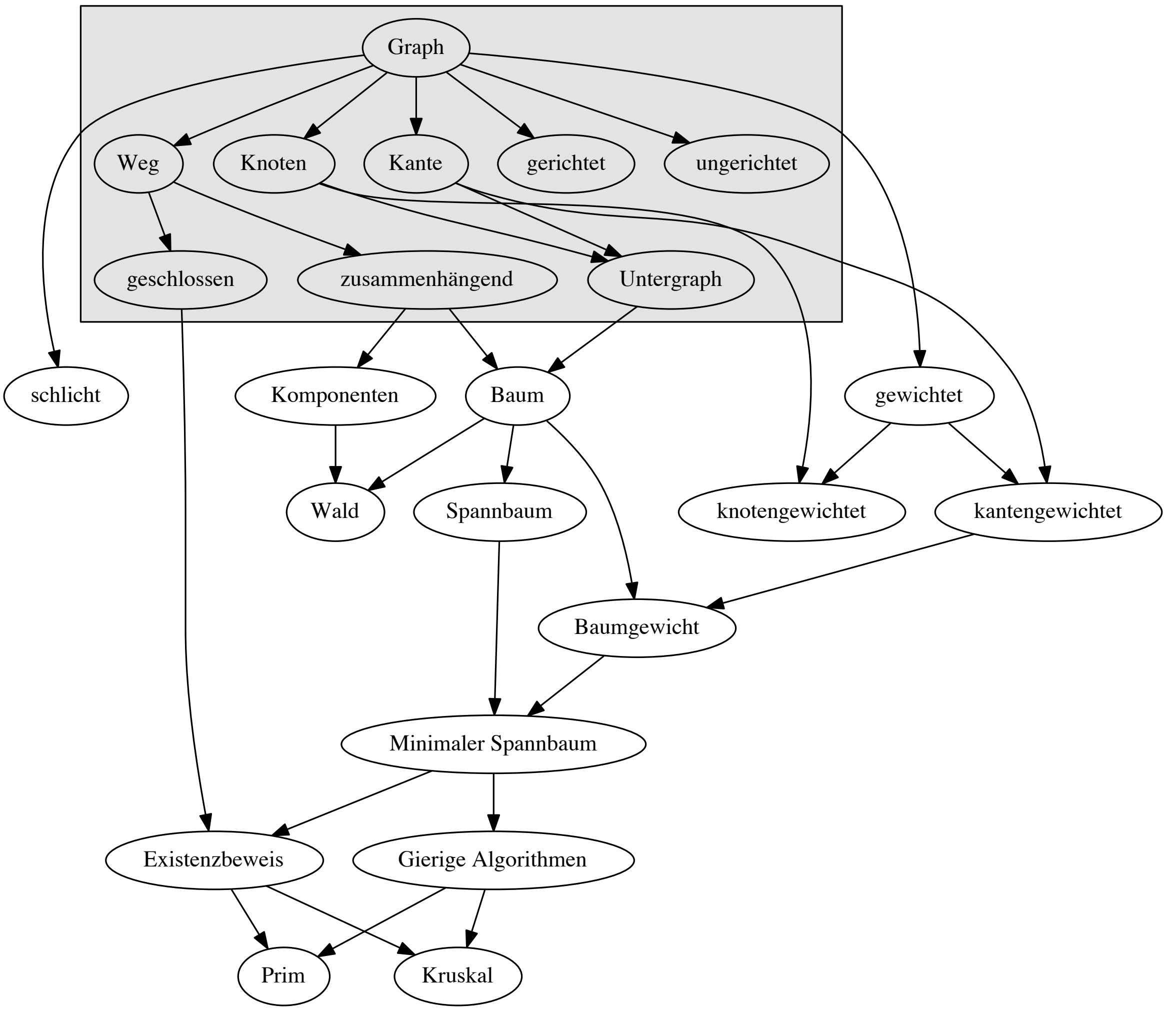
Aufgabe 1: Konzeption der Unterrichtseinheit

Matthias Neeracher, Gabriel Katz

a) Wir haben die zentralen Konzepte der Unterrichtseinheit in einem Lernzielgraphen dargestellt. Dabei betrachten wir den grau unterlegten Bereich als die erwarteten Vorkenntnisse:



Wir nehmen an, dass die Klasse bereits eine Einführung in Graphentheorie hatte. Zumindest den ungerichteten Graphen sollte die Klasse kennen, und wissen, was ein Knoten, und was eine Kante ist. Die zwei Darstellungsformen des Graphen (Adjazenzmatrix und Menge der Kanten) sollten den SchülerInnen auch bekannt sein. Wir gehen aber auch davon aus, dass gewisse Konzepte noch nicht, oder nicht mehr sehr detailliert präsent sein werden, und beginnen deshalb mit einer Repetition aller Begriffe.

Unsere Unterrichtseinheit verlangt von den SchülerInnen kein eigentliches Programmieren, aber sie erwartet, dass sie eine algorithmische Beschreibung in Pseudocode nachvollziehen können.

Ein heikles Problem in unserem Text war, ob wir Beweise für die vorgestellten Algorithmen präsentieren sollten. Einerseits waren wir der Ansicht, dass die Beweise für viele Schüler zu kompliziert sind, für die Anwendung der Algorithmen nicht unbedingt erforderlich, und den Präsentationsablauf eigentlich stören. Andererseits fanden wir es nicht akzeptabel für einen mathematischen Text, nicht-triviale Algorithmen ohne Beweis für deren Korrektheit zu präsentieren. Die Lösung, zu der wir fanden, war, die Beweise als fakultatives Material in einen separaten Anhang zu platzieren.

Bei der Präsentation des Stoffes gehen wir wie folgt vor:
- Zunächst diskutieren wir **Graphen**, beginnend mit einer Repetition der Vorkenntnisse, und dann einer Einführung von weiterführenden Konzepten, vor allem das der **Gewichtung** eines Graphen.
- Dann fahren wir mit **Bäumen** fort, was uns dann zu **Spannbäumen** und schliesslich **minimalen Spannbäumen** führt. Die SchülerInnen sollten nun verstehen, was ein minimaler Spannbaum ist, aber haben bei nicht-trivialen Graphen noch keine Methode, einen zu konstruieren.
- Schliesslich diskutieren wir zwei konkrete **Algorithmen** und zeigen, wie sie das gleiche Ziel mit unterschiedlichem Vorgehen erreichen.

Literatur:

- Volker Turau, *Algorithmische Graphentheorie (3. Auflage)*, Oldenbourg 2009
- Donald E. Knuth, *The Stanford GraphBase*, ACM Press 1993
- J. Nesetril, E. Milková, H. Nesetrilová, *Otakar Boruvka on minimum spanning tree problem*, Discrete Mathematics 233 (2001)

b) In der theoretischen Informatik werden algorithmische Probleme gelöst. Eine wichtige Kategorie dieser algorithmischen Probleme sind Optimierungsprobleme. Optimierungsprobleme sind Probleme, in welchen ein Wert unter Einhalten von Bedingungen möglichst gross oder klein werden sollte. Viele solche Optimierungsprobleme lassen sich am besten lösen, wenn man sie in die Graphentheorie überträgt und dort löst. Das minimale Spannbaum-Problem ist ein klassisches graphentheoretisches Optimierungsproblem und bietet durch die relative Einfachheit ein guter Einstieg in die Optimierungsprobleme.

Daher sollte man schon früh im Informatikunterricht das minimale Spannbaum-Problem und deren Lösungsalgorithmen studieren. Um die Unterscheidung von Problem und Lösungsalgorithmus zu demonstrieren, ist es von Nutzen, verschiedene Algorithmen für das gleiche Problem zu behandeln.

c) • Nach dieser Lerneinheit sind die Schüler in der Lage, im Alltag Probleme, welche in die Graphentheorie gelöst werden können, zu erkennen, und gleich graphentheoretisch zu modellieren.
• Die Schüler sollen motiviert sein, weitere graphentheoretische Optimierungsprobleme kennenzulernen.

d) Die folgenden Ziele sollten problemlos mit Graphen mit ca. 8 Knoten erreicht werden können.
Untergraph: Die Schüler können bestimmen, ob ein Graph ein Untergraph eines anderen ist, egal, ob der Graph als Abbildung, in der Adjazenzmatrix, oder als Kantenmenge dargestellt ist.
Baum / Wald: Die Schüler können anhand der Abbildung eines Graphen bestimmen, ob der Graph ein Baum, bzw. ein Wald ist.

Gewichtete Graphen: Die Schüler sollte in der Lage sein, eine Tabelle mit Distanzen/ Reisezeiten/Kosten in einem gewichteten Graphen darzustellen.
Spannbaum: Die Schüler erkennen, ob ein Graph ein Spannbaum eines anderen Graphen ist.
Minimaler Spannbaum: Die Schüler können auf mindestens eine Art den minimalen Spannbaum eines Graphen konstruieren. Sie können verifizieren, ob ein Spannbaum minimal ist.

Minimale Spannbäume

Gabriel Katz
Matthias Neeracher

27. Oktober 2013

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Wie kann hier gearbeitet werden?

Dieser Text wird Sie mit einem wichtigen algorithmischen Problem bekannt machen: dem Finden eines minimalen Spannbaums. Im nächsten Unterkapitel finden Sie eine Einführung in das Thema, doch bevor Sie loslegen, soll Ihnen hier noch kurz der Aufbau des Textes und die Arbeitsweise damit erläutert werden.

Die folgenden Kapitel sind zur selbstständigen Bearbeitung gedacht. Sie werden zuerst kurz Ihre Kenntnisse von ungerichteten Graphen nochmals auffrischen, lernen dann die Theorie von Bäumen und gewichteten Graphen kennen, und lernen dann zwei verschiedene Algorithmen, um einen *minimalen Spannbaum* zu finden.

Wenn Sie einige dieser Begriffe jetzt noch nicht verstehen, macht das nichts. Mitbringen sollten Sie aber die folgenden Kenntnisse:

- Sie wissen, was ein Algorithmus ist.
- Sie kennen die Grundbegriffe ungerichteter Graphen (wir werden diese allerdings in Kapitel 2 kurz repetieren).

Zur behandelten Theorie finden Sie immer auch Aufgaben, anhand derer Sie das Gelernte prüfen können. Die Lösungen dieser Aufgaben stehen jeweils im zweitletzten Unterkapitel für jedes Thema. Das letzte Unterkapitel ist dann der Kapiteltest, den Sie bearbeiten und mit Ihrer Lehrperson besprechen sollten.

1.2 Worum geht es hier?

In den frühen 20er Jahren beschäftigte sich der Mathematiker Otakar Borůvka mit dem Problem, ein Gebiet (40 Städte in Süd-Mähren) möglichst effizient mit Elektrizität zu erschliessen.

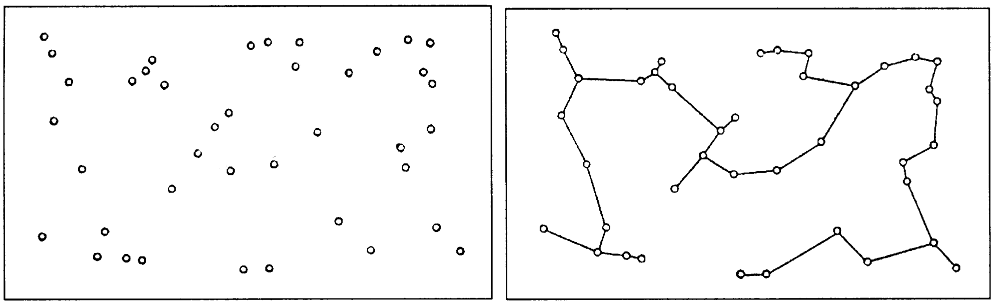


Abbildung 1.1: Elektrische Erschliessung von Süd-Mähren.[1]

Sein Ansatz war, das Problem als Graphen abzubilden, in dem die anzuschliessenden Städte als Knoten, und die Distanzen zwischen ihnen als Kanten repräsentiert wurden. Das Elektrifizierungsproblem besteht somit darin, alle Knoten so zu verbinden, dass die Gesamtlänge der Kanten so kurz wie möglich bleibt. Eine solche Verbindung wird **minimal aufspannender Baum** oder kurz **minimaler Spannbaum** genannt.

Borůvka's Lösung, die er 1926 publizierte, gilt als der erste Algorithmus zur Konstruktion eines minimalen Spannbaums, und als einer der ersten Optimierungs-Algorithmen. Weil sowohl die Problemstellung als auch die verwendeten Algorithmen interessant und einigermassen leicht verständlich sind, werden wir uns jetzt diesem Problem widmen.

[1]Otakar Borůvka, *O jistém problému minimálním* (Über ein gewisses Minimalisierungsproblem).
Zitiert aus: J. Nešetřil, E. Milková, H. Nešetřilová, *Otakar Borůvka on minimum spanning tree problem: translation of both the 1926 papers, comments, history.*, Discrete Mathematics 233 (2001)

Kapitel 2

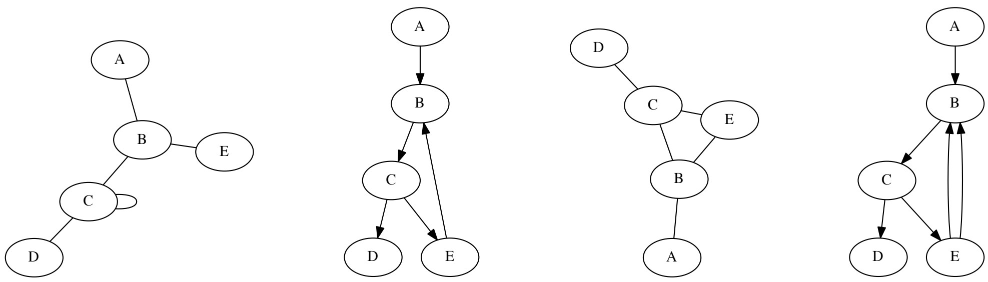
Graphen

2.1 Rekapitulation: Was ist ein Graph?

Ein **Graph** $G(V, E)$ besteht aus einer Menge von **Knoten** V und einer Menge von Kante E. Eine **Kante** besteht aus einem Paar von Knoten, den **Endknoten** der Kante. Wenn diese Paare geordnet sind, ist der Graph **gerichtet**; die Kanten verbinden die Knoten nur in eine Richtung.

In diesem Text befassen wir uns aber nur mit **ungerichteten** Graphen, in welchen die Knotenpaare ungeordnet und somit die Knoten symmetrisch verbunden sind. Des weiteren beschränken wir uns auf **schlichte** Graphen, in denen keine Kanten einen Knoten mit sich selbst verbinden, und zwischen zwei Knoten nicht mehr als eine Kante existiert.

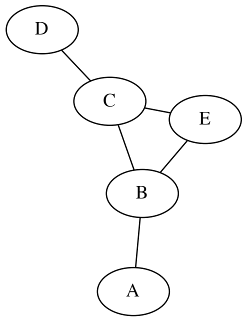
Aufgabe 2.1.1
Welche dieser vier Graphen sind gerichtet, welche ungerichtet? Welche sind schlicht?



2.2 Beschreibung von Graphen

Ein Graph $G(V, E)$ kann auf verschiedene Arten beschrieben werden:

Grafisch indem die Knoten und Kanten aufgezeichnet werden:



gerade bei kleineren Graphen ist diese Darstellung für Menschen am Einfachsten zu verstehen, aber sie ist z.B. für Computerverarbeitung nicht besonders gut geeignet.

Mengentheoretisch indem die Knoten- und die Kantenmenge beschrieben werden:

$V = \{\mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{D}, \mathsf{E}\}$ $\qquad E = \{\{\mathsf{A}, \mathsf{B}\}, \{\mathsf{B}, \mathsf{C}\}, \{\mathsf{C}, \mathsf{D}\}, \{\mathsf{C}, \mathsf{E}\}, \{\mathsf{B}, \mathsf{E}\}\}$

als **Adjazenzmatrix** indem die Kanten als $n \times n$ Matrix $A(G)$ dargestellt werden, wo der Eintrag a_{ij} gleich 1 ist, wenn eine Kante von Knoten i nach Knoten j existiert, und sonst gleich 0.

$$A(G) = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{pmatrix}$$

Wie zu beobachten ist, ist die Adjazenzmatrix bei einem ungerichteten Graphen immer symmetrisch.

Aufgabe 2.2.1

- Zeichnen Sie einen ungerichteten Graphen $G(V, E)$ mit Knoten $V = \{\mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{D}, \mathsf{E}\}$ und Kanten $E = \{\{\mathsf{A}, \mathsf{B}\}, \{\mathsf{A}, \mathsf{D}\}, \{\mathsf{C}, \mathsf{D}\}, \{\mathsf{A}, \mathsf{E}\}\}$.
- Konstruieren sie die Adjazenzmatrix dieses Graphen.

2.3 Zusammenhängende Graphen

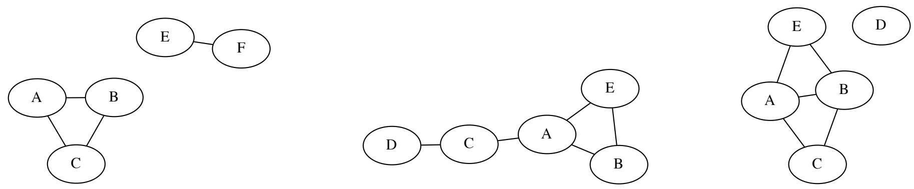
Wenn in einem Graphen $G(V, E)$ eine Folge von Knoten $v_0, v_1, \ldots, v_n \in V$ existiert, die von Kanten $e_i = \{v_i, v_{i+1}\} \in E$ verbunden werden, dann existiert ein **Weg** von v_0 nach v_n.

Falls ein Weg für *jedes* Knotenpaar $v_0, v_n \in V$ existiert, nennen wir den Graphen **zusammenhängend**.

Ob ein Graph zusammenhängend ist, lässt sich intuitiv recht einfach in seiner grafischen Darstellung verstehen: Ein Graph ist zusammenhängend, wenn alle Knoten miteinander verbunden sind. Sind die Knoten nicht alle miteinander verbunden, so heissen die verbundenen Teile des Graphen **Komponenten**.

Ein Weg $v_0 \cdots v_1 \cdots \ldots \cdots v_0$, der von einem Knoten v_0 wieder auf diesen selbst zurückführt, heisst **geschlossen**.

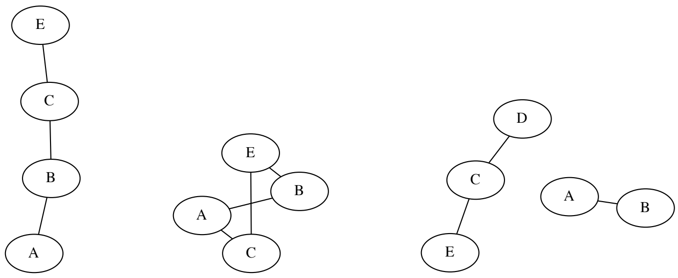
Aufgabe 2.3.1
Welcher der folgenden Graphen ist zusammenhängend? Markiere bei den nicht zusammenhängenden Graphen die grösste Komponente.



2.4 Untergraphen

Ein Graph $U(W, F)$ ist ein **Untergraph** eines Graphen $G(V, E)$ wenn seine Knotenmenge W eine Untermenge der Knotenmenge V ist und seine Kantenmenge F eine Untermenge der Kantenmenge E. Ein Untergraph $U(W, F)$ eines Graphen G ist **spannend**, wenn $W = V$. Ein spannender Untergraph muss weder zusammenhängend sein, noch muss jeder Knoten mit einer Kante verbunden sein.

Aufgabe 2.4.1
Welche der folgenden drei Graphen sind Untergraphen des Graphen in Abschnitt 2.2 ?



Aufgabe 2.4.2
FAKULTATIV Ein *vollständiger* Graph K_n ist ein Graph mit n Knoten, in welchem jedes Knotenpaar mit einer Kante verbunden ist.

$$K_n(V = \{v_1, v_2, \ldots, v_n\}, E = \{\{v_a, v_b\} \mid v_a, v_b \in V, v_a \neq v_b\})?$$

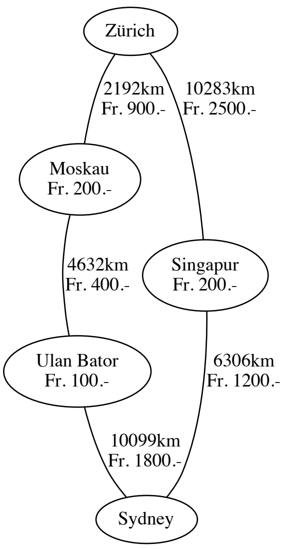
Wie viele spannende Untergraphen hat K_2? Wieviele K_3? Finde eine Formel, um die Anzahl Kanten von K_n zu berechnen.

2.5 Gewichtete Graphen

In vielen praktischen Anwendungen sind die Knoten und/oder Kanten eines Graphen mit **Gewichten** versehen. Die Kantengewichte können z.B. Distanz oder Flugkosten darstellen, die Knotengewichte z.B. Hotelkosten.

In der weiteren Diskussion werden wir hier nur noch an **Kantengewichteten** Graphen interessiert sein.

Aufgabe 2.5.1
Der folgende Graph zeigt Flug- und Hotelkosten, sowie Reisedistanzen für eine Reise.



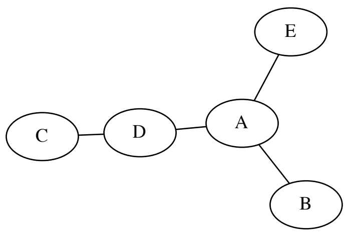
1. Was ist die *kürzeste* Route zwischen Zürich und Sydney?
2. Was ist die *billigste* Route?

2.6 Lösungen

Aufgabe 2.1.1

1. Ungerichtet, nicht schlicht weil C mit sich selbst verbunden ist.
2. Gerichtet und schlicht.
3. Ungerichtet und schlicht.
4. Gerichtet, nicht schlicht weil zwischen B und E zwei Kanten existieren.

Aufgabe 2.2.1



$$\begin{pmatrix} 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Aufgabe 2.3.1 Graph 1 ist nicht verbunden. Die grsste Komponente besteht aus den Knoten A, B und C. Graph 2 ist verbunden. Graph 3 ist nicht verbunden. Die grsste Komponente besteht aus den Knoten A, B, C und E.

Aufgabe 2.4.1 Graphen 1 und 3 (Ein Untergraph muss *nicht* notwendig zusammenhängend sein). Graph 2 enthält eine Kante $\{A, C\}$ die im ursprünglichen Graphen nicht existiert.

Aufgabe 2.4.2 Ein kompletter Graph K_n hat $n(n-1)/2$ Kanten. Für jede Kante wird die Anzahl der mglichen Subgraphen verdoppelt, da die Kante entweder im Subgraphen vorkommen kann oder nicht. Somit hat K_2 2 verschiedene Subgraphen, K_3 hat 8 Subgraphen, und K_n hat $2^{n(n-1)/2}$ Subgraphen.

Aufgabe 2.5.1

1. Zürich—Singapur—Sydney (16589 km)
2. Zürich—Moskau—Ulan Bator—Sydney (Fr. 3400.-)

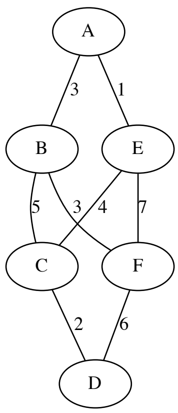
2.7 Kapiteltest

Aufgabe 2.7.1
Gegeben sei die Adjazenzmatrix

$$A(G) = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

1. Zeichnen Sie den Graphen auf (verwenden Sie $A..Z$ als Knotennamen).
2. Stellen Sie den Graphen in Mengennotation dar.
3. Aus welchen Komponenten besteht der Graph?

Aufgabe 2.7.2
Gegeben sei der folgende Graph:



1. Welcher Weg von A nach D weist die kleinste Summe von Kantengewichten auf?
2. Welcher geschlossene Weg im Graphen weist die kleinste Summe von Kantengewichten auf?

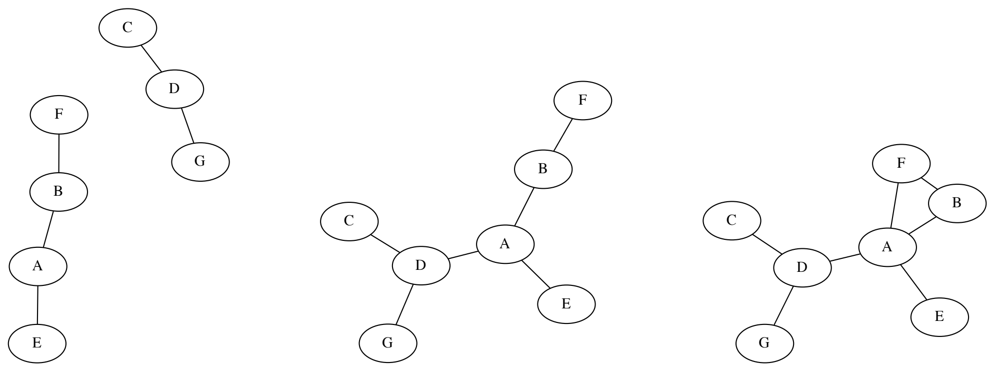
Kapitel 3

Bäume

3.1 Was ist ein Baum?

Als **Baum** bezeichnet man einen Graphen, der *zusammenhängend* ist und *keinen geschlossenen Weg* enthält. Eine Menge von Bäumen, die keine gemeinsamen Knoten haben, bezeichnet man (anschaulicherweise) als **Wald**.

Aufgabe 3.1.1
Welche der folgenden drei Graphen sind Bäume? “Reparieren” Sie die anderen Graphen durch Hinzufügen bzw. Weglassen von Kanten, damit sie ebenfalls zu Bäumen werden.



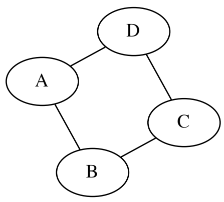
Satz 3.1.1 Ein Baum mit n Knoten hat exakt $n - 1$ Kanten.

BEWEIS: Ein Baum mit 1 Knoten kann keine Kanten haben. In einem Baum mit n Knoten sucht man sich einen beliebigen Knoten aus, folgt einer beliebigen Kante, und beim nächsten Knoten sucht man sich eine andere Kante aus. Weil der Baum ja keinen geschlossenen Weg enthält, landet man irgendwann in einem Knoten, der von nur 1 Kante erreicht wird. Wenn man diesen Knoten und seine Kante aus dem Baum entfernt, erhält man einen Baum mit $n - 1$ Knoten, und durch Induktion lässt sich sehen, dass der Satz gilt.

3.2 Spannbäume

Für jeden zusammenhängenden Graphen $G(V, E)$ kann man einen oder mehrere **Spannbäume** konstruieren, die aus *allen Knoten* V des Graphen und einer *Untermenge der Kanten* E bestehen.

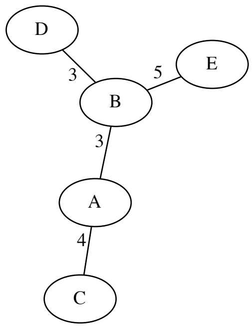
Aufgabe 3.2.1
Zählen Sie alle möglichen Spannbäume dieses Graphen auf:



3.3 Minimale Spannbäume

Wenn wir jetzt wieder an das in Kapitel 2.5 eingeführten Konzept des *kantengewichteten Graphen* zurückdenken, können wir dieses auch auf Bäume anwenden. Das **Gewicht** eines Baumes lässt sich dann als Summe der Kantengewichte des Baumes definieren.

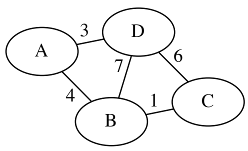
Aufgabe 3.3.1
Bestimmen Sie das Gewicht dieses Baumes:



Somit können wir nun den zentralen Begriff dieses Textes definieren: Ein Spannbaum B eines Graphen G ist ein **minimaler Spannbaum** von G wenn kein Spannbaum von G existiert, dessen Gewicht kleiner als das Gewicht von B ist.

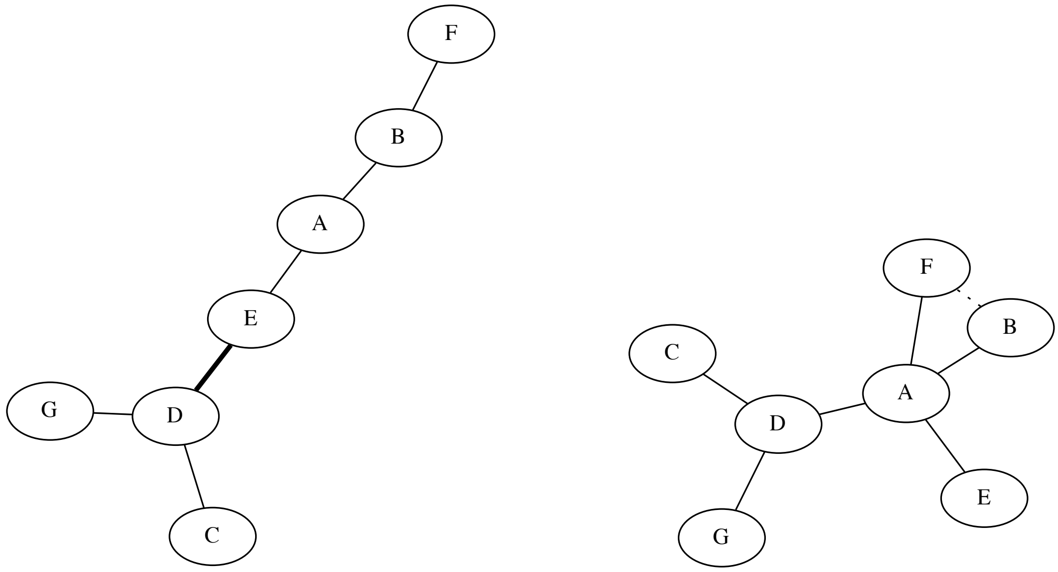
Ein Graph kann ohne weiteres mehrere minimale Spannbäume haben: Wenn z.B. alle Kantengewichte identisch sind, sind *alle* Spannbäume minimal!

Aufgabe 3.3.2
Finden Sie den minimalen Spannbaum dieses Graphen:



3.4 Lösungen

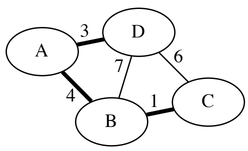
Aufgabe 3.1.1 Der mittlere Graph war bereits ein Baum.



Aufgabe 3.2.1 $\{\{A,B\},\{A,D\},\{B,C\}\}, \{\{A,B\},\{A,D\},\{C,D\}\}, \{\{A,B\},\{B,C\},\{C,D\}\}, \{\{A,D\},\{B,C\},\{C,D\}\}$

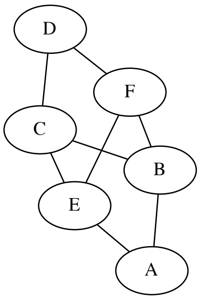
Aufgabe 3.3.1 15

Aufgabe 3.3.2

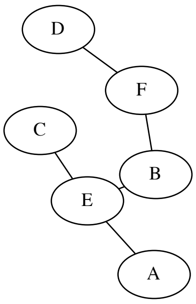


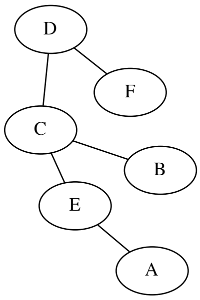
3.5 Kapiteltest

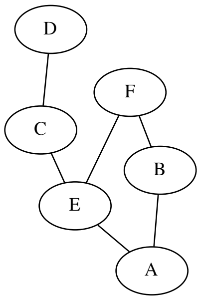
Aufgabe 3.5.1
Gegeben sei folgender Graph G:



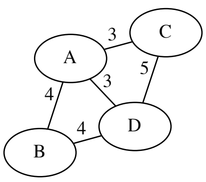
Welche der folgenden Graphen sind Bäume, welche sind Spannbäume von G?







Aufgabe 3.5.2
Zeichne alle minimalen Spannbäume dieses Graphen auf:



Kapitel 4

Algorithmen

Wie findet man denn nun einen minimalen Spannbaum in einem nicht-trivialen Graphen? In diesem Kapitel werden wir zwei verschiedene Algorithmen kennenlernen, die dies bewerkstelligen.

4.1 Gierige Algorithmen

Ein Algorithmus wird **gierig** (bzw. das Englische Aequivalent **greedy**) genannt, wenn er in jedem Schritt einen *lokal optimalen* Folgezustand wählt. In Reiseproblem, das in Aufgabe 2.5.1 besprochen wurde, würde ein gieriger Algorithmus z.B. in jedem Schritt den kürzesten oder den billigsten Flug aus der gegenwärtigen Stadt in eine noch nicht besuchte Stadt wählen.

Gierige Algorithmen sind oft einfach zu verstehen und schnell, aber sie lösen viele Probleme nicht optimal: In unserem Beispiel würde der Algorithmus zwar den billigsten Weg finden, aber im allgemeinen Fall ist das nicht garantiert, und der Algorithmus würde hier die falsche Lösung für den kürzesten Weg finden.

Glücklicherweise ist das Finden eines minimalen Spannbaums eines der Probleme, bei denen gierige Algorithmen ein optimales Resultat garantieren können:

Satz 4.1.1 Es sei $G(V, E)$ ein kantenbewerteter zusammenhängender Graph. U sei eine Teilmenge der Knoten V und e_{min} die Kante mit dem kleinsten Gewicht in der Kantenmenge $\{e = \{u, v\} \mid e \in E, u \in U, v \in (V \setminus U)\}$. Dann existiert ein minimaler Spannbaum von G, der e_{min} enthält.

BEWEIS: Nehmen wir das Gegenteil an: e_{min} liegt in *keinem* minimalen Spannbaum von G. Wenn wir dann B, einen beliebigen minimalen Spannbaum von G, nehmen und e_{min} in B einfügen, erhält man einen geschlossenen Weg W in B, der e_{min} enthält. Da W Knoten sowohl aus U als auch aus $V \setminus U$ enthält, muss

W eine andere Kante $\{u', v'\}$ enthalten, in der $u' \in U, v' \in (V \setminus U)$ sind. Wenn man dann $\{u', v'\}$ aus B entfernt, bleibt der entstehende Baum B' ein Spannbaum, und da nach Definition das Gewicht von e_{min} nicht grösser sein kann als das von $\{u', v'\}$, kann auch das Gewicht von B' nicht grösser sein als das von B: B' muss ebenfalls ein minimaler Spannbaum sein, und da B' e_{min} enthält, ist die Gegenannahme widerlegt.

4.2 Der Algorithmus von Kruskal

Der erste Algorithmus, den wir behandeln, wurde von dem amerikanischen Mathematiker Joseph Kruskal entwickelt und erstmals 1956 publiziert. Der minimale Spannbaum für den Graphen $G(V, E)$ wird konstruiert, indem ein Wald $W(V, E')$ von minimalen Bäumen sukzessive verbunden wird, bis ein einziger minimaler Spannbaum übrigbleibt:

1. Der Anfangszustand des Waldes besteht aus einem Baum pro Knoten: $W \leftarrow (V, \{\})$ (Beachten Sie, dass ein einzelner Knoten bereits einen Baum darstellt)

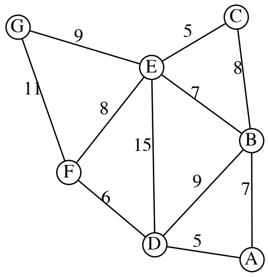
2. So lange es Kanten K, $K \subset (E \setminus E')$ gibt, die in W noch nicht verwendet wurden, und die mit den verwendeten Kanten E' keinen geschlossenen Weg bilden:

   2.1. Wählen Sie die Kante $e_i \in K$ mit dem kleinsten Kantengewicht aus.

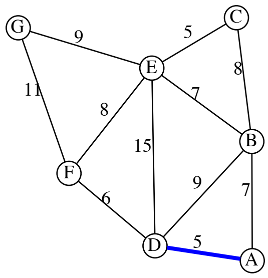
   2.2. Fügen Sie diese Kante zu W hinzu: $E' \leftarrow E' + \{e_i\}$.

FAKULTATIV: Der Beweis des Algorithmus von Kruskal ist etwas schwieriger als die anderen Beweise, die wir in diesem Text diskutiert haben. Sie können ihn im Abschnitt A.1.1 finden.

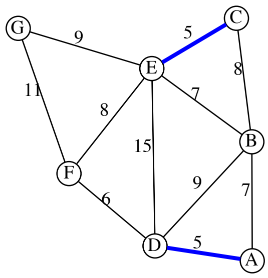
Hier soll ein einfaches Beispiel graphisch illustriert werden:



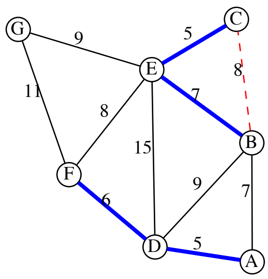
Der Anfangszustand, $E' = \{\}$



Schritt 1: Eine der beiden Kanten mit Gewicht 5 (egal welche) wurde hinzugefügt: $E' = \{\{\mathbf{A}, \mathbf{D}\}\}$



Schritt 2: Die nächstkürzere Kante wurde hinzugefügt: $E' = \{\{A, D\}, \{\mathbf{C}, \mathbf{E}\}\}$



Schritt 4: Nachdem zwei weitere Kanten hinzugefügt wurden, wird jetzt klar, dass die Kante $\{B, C\}$ nie mehr hinzugefügt werden wird, weil sie mit $\{B, E\}$ und $\{C, E\}$ einen geschlossenen Weg bilden würde. $E' = \{\{A, D\}, \{\mathbf{B}, \mathbf{E}\}, \{C, E\}, \{\mathbf{D}, \mathbf{F}\}\}$

Aufgabe 4.2.1
Führen Sie den Algorithmus von Kruskal im vorstehenden Beispiel zu Ende, bis der minimale Spannbaum konstruiert ist.

4.3 Der Algorithmus von Prim

Viele Optimierungsprobleme können auf viele verschiedene Arten gelöst werden. Auch für das minimale Spannbaumproblem gibt es viele Algorithmen, und wir wollen deshalb noch einen zweiten Algorithmus vorstellen, um einen anderen Ansatz zu zeigen.

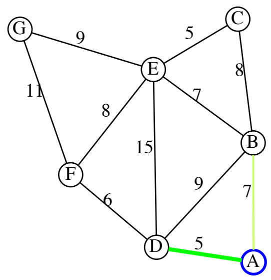
Dieser Algorithmus wurde bereits 1930 von dem Mathematiker Vojtěch Jarník entwickelt, geriet dann aber in Vergessenheit, bis er 1957 von Robert C. Prim und 1959 von Edsger W. Dijkstra unabhängig wiederentdeckt wurde (Aus diesem Grund ist der Algorithmus auch unter anderen Namen, z.B. *Prim-Dijkstra Algorithmus* oder *Algorithmus von Jarnik, Prim, und Dijkstra* bekannt).

Der Algorithmus von Prim konstruiert einen minimalen Spannbaum für den zusammenhängenden Graphen $G(V, E)$ indem er einen bestehenden Baum $B(V_B, E_B)$ sukzessive durch Hinzufügen minimaler Kanten erweitert:

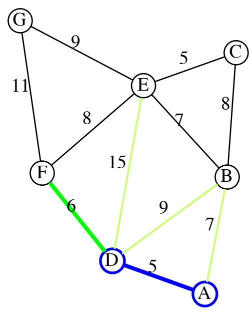
1. Wählen Sie einen beliebigen Knoten $v_0 \in V$ als Anfangszustand für B: $V_B \leftarrow \{v_0\}$, $E_B \leftarrow \{\})$.
2. Solange $V_B \neq V$:
   2.1. Wählen Sie unter den Kanten $\{e = \{u, v\} \mid u \in V_B, v \in (V \backslash V_B)\}$ die Kante $e_i = \{u_i, v_i\}$ mit dem kleinsten Kantengewicht aus.
   2.2. Fügen Sie die Kante und ihren Endpunkt zu B hinzu: $V_B \leftarrow V_B + \{v_i\}$, $E_B \leftarrow E_B + \{e_i\}$.

FAKULTATIV: Auch beim Algorithmus von Prim ist der Beweis etwas kompliziert. Sie können ihn im Abschnitt A.2.1 finden.

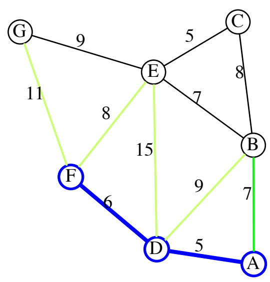
Auch hier werden wir das gleiche Beispiel illustrieren:



Der Anfangszustand, $V_B = \{A\}$ (beliebige Wahl), $E_B = \{\}$. Mögliche Erweiterungskanten sind $\{A, B\}$ und $\{A, D\}$; letztere hat das kleinere Kantengewicht.



Schritt 1: $\{A, D\}$ bzw. D wurden hinzugefügt. $V_B = \{A, D\}, E_B = \{\{A, D\}\}$. Nun ist $\{D, F\}$ die beste Erweiterungskante.



Schritt 2: $\{D, F\}$ bzw. F wurden hinzugefügt. $V_B = \{A, D, F\}, E_B = \{\{A, D\}, \{D, F\}\}$. Nun kommt endlich $\{A, B\}$ als Erweiterungskante zum Zug.

Aufgabe 4.3.1
Führen Sie den Algorithmus von Prim im vorstehenden Beispiel zu Ende, bis der minimale Spannbaum konstruiert ist.

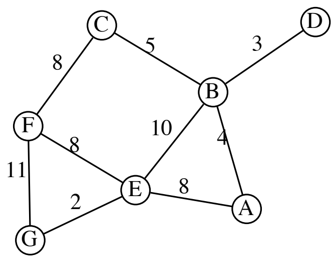
4.4 Schlussbetrachtung: Wer Gewinnt?

Wenn mehrere verschiedene Algorithmen zur Lösung eines Problems zur Wahl stehen, ist es eine naheliegende Frage, welcher davon denn der effizienteste ist. Leider würde eine umfassende Erforschung dieser Frage den Rahmen dieses Textes bei weitem sprengen, deshalb soll nur kurz erwähnt sein:

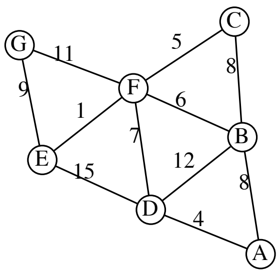
- Da alle Algorithmen hier optimale Lösungen finden, unterscheidet sich die Qualität der Resultate nicht.
- Wenn man diese Algorithmen auf einem Computer programmiert, hängt deren Laufzeit- und Speichereffizienz enorm von den verwendeten Datenstrukturen ab, weil die Auswahl- und Testschritte (z.B. auf geschlossene Wege in Kruskal's Algorithmus, auf Mitgliedschaft von Knoten in Untergraphen in Prim's Algorithmus) entscheidend auf eine effiziente Datenrepräsentation angewiesen sind.
- In einer Studie von mehreren Implementierungen von Kruskal's und Prim's Algorithmus ist der Informatiker Donald E. Knuth[1] zum Schluss gelangt, dass sich die Effizienz der beiden Algorithmen nicht wesentlich voneinander unterscheidet.

[1] Donald E. Knuth, *The Stanford GraphBase*, Seiten 460–497

Aufgabe 4.4.1
Berechnen Sie die minimalen Spannbäume für den folgenden Graphen.

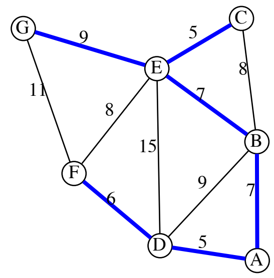


Aufgabe 4.4.2
Berechnen Sie den minimalen Spannbaum für den folgenden Graphen.

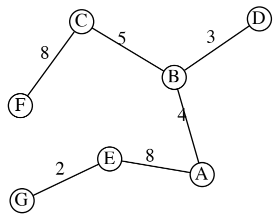


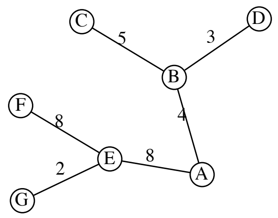
4.5 Lösungen

Aufgaben 4.2.1 und 4.3.1 konstruieren letztendlich beide den gleichen minimalen Spannbaum:

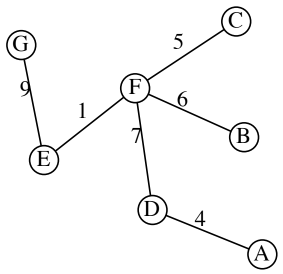


Aufgabe 4.4.1





Aufgabe 4.4.2



4.6 Kapiteltest

Aufgabe 4.6.1
Sie haben den Auftrag erhalten, ein Glasfaser-Netzwerk zu entwerfen, das die grössten Schweizer Städte verbindet (Elektrizität haben wir ja inzwischen alle). Gehen Sie dabei von der Annahme aus, dass Sie die Glasfaser zwischen zwei beliebigen Städten in Luftlinie verlegen können.

Stadt	Abkürzung	Koordinaten
Zürich	ZRH	683248 / 248161
Genf	GVA	500532 / 117325
Basel	BSL	611587 / 267423
Lausanne	LAU	538200 / 152026
Bern	BRN	600000 / 200000
Winterthur	WIN	698805 / 261852
Luzern	LZN	665450 / 211356
St. Gallen	QGL	746265 / 254310
Lugano	LUG	718030 / 96560
Biel	BIE	585481 / 220742
Thun	THU	614620 / 178664
Köniz	CHT	598221 / 197101
La Chaux-de-Fonds	LCF	553419 / 216894
Schaffhausen	SCH	689677 / 283948
Freiburg	FRB	578943 / 183921
Chur	CHR	759742 / 190895
Neuenburg	QNC	561352 / 204483
Vernier	VNR	496673 / 117390
Uster	USR	696755 / 245077
Sitten	SIR	594446 / 120213

Auf der nächsten Seite finden Sie eine Distanzentabelle, die Ihnen sicher gute Dienste leisten wird.

1. Bestimmen Sie mit Hilfe eines der beiden Algorithmen, die Sie nun kennen, den minimalen Spannbaum für entweder die ersten 10 der Städte, oder (fakultativ) für alle 20 Städte.

2. Wieviel km Glasfaserkabel müssen Sie bestellen?

	ZRH	GVA	BSL	LAU	BRN	WIN	LZN	QGL	LUG	BIE	THU	CHT	LCF	SCH	FRB	CHR	QNC	VNR	USR	SIR
ZRH		225	74	174	96	21	41	63	156	102	98	99	134	36	123	96	129	228	14	156
GVA	225		187	51	129	245	190	281	218	134	130	126	113	252	103	269	106	4	234	94
BSL	74	187		137	68	87	78	135	201	53	89	72	77	80	90	167	81	189	88	148
LAU	174	51	137		78	195	140	232	188	83	81	75	67	201	52	225	57	54	184	65
BRN	96	129	68	78		117	66	156	157	25	26	3	50	123	26	160	39	132	107	80
WIN	21	245	87	195	117		61	48	166	121	118	120	152	24	143	94	149	248	17	176
LZN	41	190	78	140	66	61		92	126	81	60	69	112	77	91	96	104	193	46	116
QGL	63	281	135	232	156	48	92		160	164	152	159	196	64	182	65	192	285	50	203
LUG	156	218	201	188	157	166	126	160		182	132	156	204	190	164	103	190	222	150	126
BIE	102	134	53	83	25	121	81	164	182		51	27	32	122	37	177	29	136	114	101
THU	98	130	89	81	26	118	60	152	132	51		25	72	129	36	146	59	133	106	62
CHT	99	126	72	75	3	120	69	159	156	27	25		49	126	23	162	38	129	110	77
LCF	134	113	77	67	50	152	112	196	204	32	72	49		152	42	208	15	115	146	105
SCH	36	252	80	201	123	24	77	64	190	122	129	126	152		149	116	151	255	40	189
FRB	123	103	90	52	26	143	91	182	164	37	36	23	42	149		181	27	106	133	66
CHR	96	269	167	225	160	94	96	65	103	177	146	162	208	116	181		199	273	83	180
QNC	129	106	81	57	39	149	104	192	190	29	59	38	15	151	27	199		108	141	91
VNR	228	4	189	54	132	248	193	285	222	136	133	129	115	255	106	273	108		237	98
USR	14	234	88	184	107	17	46	50	150	114	106	110	146	40	133	83	141	237		161
SIR	156	94	148	65	80	176	116	203	126	101	62	77	105	189	66	180	91	98	161	

Anhang A

Beweise

A.1 Algorithmus von Kruskal

Satz A.1.1 Der Algorithmus von Kruskal konstruiert einen minimalen Spannbaum.

BEWEIS: Der Beweis besteht aus zwei Teilen. Zuerst zeigen wir, dass der Algorithmus von Kruskal einen Spannbaum produziert, danach, dass dieser Spannbaum minimal ist.

1. **Spannbaum:** Angenommen, Graph $W = (V, E')$ ist das Resultat einer Ausführung des Algorithmus von Kruskal auf dem Graphen $G = (V, E)$. Dann hat der Graph W keinen geschlossenen Weg, da die Kanten im Algorithmus per Definition so gewählt werden, dass kein geschlossener Weg entsteht. Wenn wir beweisen können, dass Graph W auch zusammenhängend ist, folgt daraus, dass W ein Spannbaum ist. Um dies zu beweisen, nehmen wir an, dass W aus zwei getrennten Komponenten W_1 und W_2 besteht. Da G zusammenhängend ist, existiert in G mindestens eine Kante von einem Knoten aus W_1 zu einem Knoten aus W_2. Dies heisst jedoch, dass der Algorithmus noch nicht fertig ist, da eine solche Kante noch nicht in W liegt, und auch mit den Kanten von E' keinen geschlossenen Weg bildet.

2. **Minimalität:** Dieser Beweis folgt durch Induktion. Wir betrachten jeden Schritt k zwischen 0 und der Anzahl Kanten des Spannbaums. Wir zeigen dann, dass der Graph $W_k = (V, E')$, den wir nach dem k-ten Schritt des Algorithmus erhalten, ein Subgraph eines minimalen Spannbaums ist. Da wir oben bewiesen haben, dass wir nach dem letzten Schritt des Algorithmus

einen Spannbaum erhalten, würde dies bedeuten, dass der erhaltene Spannbaum minimal ist. Den Induktionsanfang setzen wir bei $W_0 = (V, \{\})$. Dieser Graph ist offensichtlich ein Subgraph eines minimalen Spannbaums, da er mit dem minimalen Spannbaum knotengleich ist, und die Kantenmenge von W_0 leer ist, und somit Teilmenge jeder beliebigen Kantenmenge. Für den Induktionsschritt nehmen wir an, dass $W_k = (V, E'_k)$ ein Subgraph des minimalen Spannbaum T_k ist. Schauen wir jetzt an, was geschieht, wenn wir im $k+1$-ten Schritt W um die Kante e_{k+1} zu $W_{k+1} = W_k \cup \{e_{k+1}\}$ erweitern. Falls e_{k+1} in T_k vorkommt, lässt sich auch W_{k+1} zu T_k erweitern, und wir sind fertig mit dem Induktionsschritt. Falls e_{k+1} nicht in T_k vorkommt, hat $T_k \cup \{e_{k+1}\}$ einen geschlossenen Weg. In diesem geschlossenen Weg existiert eine Kante $f \neq e_{k+1}$, welche in W_k nicht vorkommt (Ansonsten würde e_{k+1} mit W_k einen geschlossenen Weg formen). Da e_{k+1} und nicht f im $k+1$-ten Schritt des Algorithmus ausgewählt wurde, muss das Kantengewicht von e_{k+1} kleiner oder gleich dem Kantengewicht von f sein. $T_k \cup \{e_{k+1}\} \setminus \{f\}$ ist somit ein Spannbaum mit einem höchstens so hohen Kantengewicht wie T_k. Also muss $T_k \cup \{e_{k+1}\} \setminus \{f\}$ auch ein minimaler Spannbaum sein. Da $W_{k+1} = W_k \cup \{e_{k+1}\}$ ein Subgraph von $T_k \cup \{e_{k+1}\} \setminus \{f\}$ ist, haben wir die Aussage bewiesen.

A.2 Algorithmus von Prim

Satz A.2.1 Der Algorithmus von Prim konstruiert einen minimalen Spannbaum.

BEWEIS: Es ist leicht zu sehen, dass der Algorithmus einen Spannbaum T konstruiert: In jedem Schritt wird V_B um genau einen Knoten und E_B um genau eine Kante erweitert, die den Knoten mit dem bestehenden Graphen verbindet. Der Algorithmus konstruiert also einen zusammenhängenden Graphen mit $V_B = V$ und (Anzahl Kanten) $=$ (Anzahl Knoten) $- 1$, was der Definition eines Spannbaums entspricht.

Es bleibt noch zu zeigen, dass T minimal ist: Wir wissen, dass mindestens ein minimaler Spannbaum, nennen wir ihn T', existieren muss. Wenn $T' \neq T$, wählen wir die erste Kante $e_k = \{u, v\}$ aus Schritt 2.1 des Algorithmus aus, die nicht in T' enthalten ist. Es muss also in T' einen anderen Weg von u nach v geben, und daraus wählen wir eine Kante aus, die die bereits in vorherigen Schritten gewählten Knoten V_B^{k-1} mit den noch nicht gewählten Knoten verbindet: $e' = \{u', v'\}, u' \in V_B^{k-1}, v' \in (V \setminus V_B^{k-1})$. Das Gewicht von e' kann nicht kleiner sein als das von e_k, weil sonst im Schritt k die Kante e' gewählt worden wäre. Wenn wir also e' durch e_k ersetzen, wird das Gewicht des resultierenden Baums nicht grösser, und diese Ersetzungen können wiederholt werden, bis $T' = T$.