5.5 Minimale Spannbäume und Gerüste

5.5.1 Bäume und Wälder

- Ein zusammenhängender Graph $G = (V, E)$ heißt *Baum*, wenn $|E| = |V| - 1$ gilt. (Zu jedem Knoten außer der Wurzel gibt es genau eine Kante, die ihn mit seinem übergeordneten Knoten verbindet. Je nach Art des Baums, können diese Kanten ungerichtet, „von oben nach unten“ gerichtet oder „von unten nach oben“ gerichtet sein. Bei einem ungerichteten Baum kann jeder Knoten die Rolle des Wurzelknotens spielen.)

- Eine Menge von Bäumen heißt *Wald*.

- *Anmerkung:* Für jeden zusammenhängenden Graphen gilt: $|E| \geq |V| - 1$.
  Damit ist ein Baum ein zusammenhängender Graph mit möglichst wenig Kanten.

Alternative Definition

- Ein Graph ohne Schlingen heißt Baum, wenn es zwischen je zwei Knoten genau eine Verbindung gibt.

- Ein Graph ohne Schlingen heißt Wald, wenn es zwischen je zwei Knoten höchstens eine Verbindung gibt.

Anmerkung

- Da ein ungerichteter Graph mit mindestens einer Kante $\{u, v\} = \{v, u\}$ gemäß § 5.1.3 immer Zyklen u, v, u und v, u, v enthält, ist die Charakterisierung eines Baums als zusammenhängender, azyklischer Graph (vgl. Wikipedia) für ungerichtete Graphen falsch.

- Um solche „unerwünschten“ Zyklen zu vermeiden, könnte man zusätzlich verlangen, dass die Kanten auf einem Zyklus paarweise verschieden sind.

- Dann bestünde jedoch ein feiner Unterschied zwischen einem ungerichteten Graphen und dem entsprechenden gerichteten Graphen, in dem es zu jeder Kante auch die entgegengesetzte Kante gibt: Der ungerichtete Graph könnte dann azyklisch sein, der gerichtete Graph wäre jedoch immer zyklisch (sofern es mindestens eine Kante gibt).

Lemma

1. Wenn man aus einem Baum eine Kante entfernt, zerfällt er in zwei Bäume.

2. Wenn man zwei Bäume durch eine Kante verbindet, entsteht ein einziger Baum.

3. Wenn man zu einem Baum eine Kante von einem Knoten u zu einem Knoten v hinzufügt und gleichzeitig eine Kante auf der Verbindung von u nach v entfernt, entsteht wieder ein Baum.

4. Wenn man zu einem Wald eine Kante von einem Knoten u zu einem Knoten v hinzufügt, die bereits miteinander verbunden sind, und gleichzeitig eine Kante auf dieser Verbindung entfernt, entsteht wieder ein Wald mit gleich vielen Bäumen.

Beweis jeweils mit der Definition, dass es in einem Baum zwischen je zwei Knoten genau eine Verbindung gibt.

5.5.2 Spannbäume und Gerüste

- Ein *Gerüst* oder *Spannwald* (spanning forest) eines ungerichteten Graphen $G = (V, E)$ ist ein Wald (V, F) mit $F \subseteq E$, der die gleichen Zusammenhangskomponenten wie G besitzt.

- Das heißt:
  - Ein Gerüst eines Graphen enthält alle Knoten des Graphen und eine möglichst kleine Teilmenge seiner Kanten.
  - Jeder Zusammenhangskomponente des Graphen entspricht ein Baum im Gerüst.
  - Insbesondere gibt es für jede Kante $\{u, v\} \in E$ des Graphen einen Weg im Gerüst von u nach v.

- Abkürzend wird auch die Kantenmenge F eines Gerüsts (V, F) als Gerüst bezeichnet.

- Wenn der Graph G zusammenhängend ist, ist jedes Gerüst von G ein Baum, der dann als *Spannbaum* (spanning tree) von G bezeichnet wird.

5.5.3 Minimale Spannbäume und Gerüste

- Ein *gewichteter Graph* $G = (V, E, \rho)$ ist ein Graph (V, E) mit einer zugehörigen *Gewichtsfunktion* $\rho: E \to \mathbb{R}$, die jeder Kante $e \in E$ ein *Gewicht* $\rho(e) \in \mathbb{R}$ zuordnet.

- Abkürzend wird auch $\rho(u, v)$ statt $\rho((u, v))$ (Gewicht der gerichteten Kante (u, v)) bzw. $\rho(\{u, v\})$ (Gewicht der ungerichteten Kante $\{u, v\}$) geschrieben.

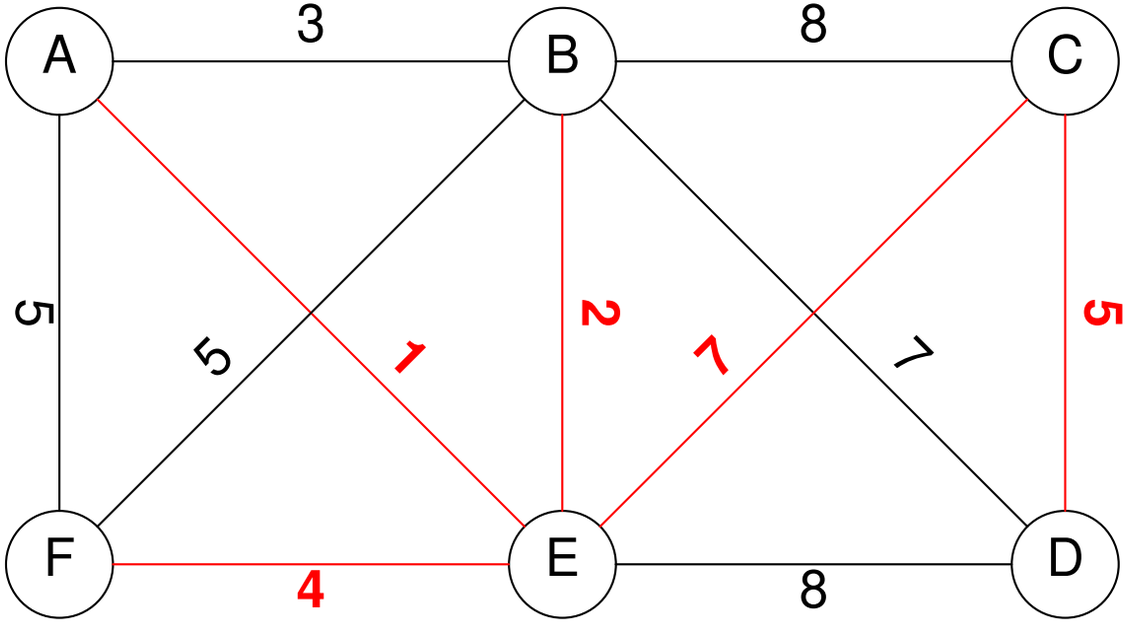
- Sofern nichts anderes gesagt wird, sind prinzipiell auch negative Kantengewichte zulässig.

- Das Gewicht einer Kantenmenge $F \subseteq E$ ist die Summe $\rho(F) = \sum_{e \in F} \rho(e)$ der Gewichte aller Kanten $e \in F$.

- Ein *Minimalgerüst* (oder *minimales Gerüst*, *minimaler Spannwald*; minimum spanning forest) eines ungerichteten, gewichteten Graphen $G = (V, E, \rho)$ ist ein Gerüst F des Graphen mit minimalem Gewicht, d. h. $\rho(F) \leq \rho(F')$ für jedes Gerüst F' von G.

- Wenn der Graph zusammenhängend ist, ist jedes Minimalgerüst ein Baum, der dann als *minimaler Spannbaum* (minimum spanning tree, MST) des Graphen bezeichnet wird.

5.5.4 Beispiel



Die roten Kanten bilden einen von zwei möglichen minimalen Spannbäumen des Graphen.

5.5.5 Anwendungsbeispiele

- Verbindung elektrischer Bauteile mit möglichst wenig Draht
- Approximationsverfahren zur Lösung des Problems des Handlungsreisenden (vgl. § 5.7.3)

5.5.6 Schleifeninvarianten

Definition

- Eine *Schleifeninvariante* ist eine Aussage, die an folgenden Stellen gilt:
  - Vor Beginn einer Schleife
  - Am Anfang und am Ende jedes Schleifendurchlaufs
  - Nach Beendigung der Schleife

Verwendung in Beweisen

- Um zu beweisen, dass eine bestimmte Aussage tatsächlich eine Schleifeninvariante darstellt, genügt es (ähnlich wie bei vollständiger Induktion), zwei Dinge zu zeigen (unter der Annahme, dass die Auswertung der Schleifenbedingung keine Nebeneffekte verursacht):
  - *Initialisierung:*
    Die Invariante gilt vor Beginn der Schleife.
    (Dann gilt sie auch am Anfang des ersten Durchlaufs, sofern es überhaupt einen Durchlauf gibt.)
  - *Aufrechterhaltung:*
    Wenn die Invariante am Anfang eines Durchlaufs gilt, dann gilt sie auch am Ende dieses Durchlaufs (und damit auch am Anfang des nächsten Durchlaufs, sofern es einen solchen gibt).
    (Zusammen mit der Initialisierung folgt dann durch vollständige Induktion, dass die Invariante nach *jedem* Durchlauf gilt.)
- Daraus folgt dann automatisch:
  - *Terminierung:*
    Wenn die Schleife terminiert (was ggf. anderweitig gezeigt werden muss, sofern es nicht offensichtlich ist), dann gilt die Invariante auch nach ihrer Beendigung.

Beispiel

```
int pow (int x, int n) {
  int p = 1, k = 0;
  while (k < n) {
    p = p * x;
    k = k + 1;
  }
  return p;
}
```

- Um zu zeigen, dass diese Funktion für $n \geq 0$ tatsächlich x^n berechnet, kann die Schleifeninvariante $p = x^k$ verwendet werden:

- Initialisierung:
  Vor Beginn der Schleife gilt: $p = 1 = x^0 = x^k$
- Aufrechterhaltung:
  Wenn die Aussage $p = x^k$ am Anfang eines Durchlaufs gilt, dann gilt am Ende dieses Durchlaufs: $p' = p \cdot x = x^k \cdot x = x^{k+1} = x^{k'}$.
  Dabei bezeichnen p und k die Werte der entsprechenden Variablen am Anfang des Durchlaufs, p' und k' ihre Werte am Ende des Durchlaufs.
- Terminierung:
  Nach Beendigung der Schleife ist $k = n$ (sofern $n \geq 0$ ist) und somit gilt:
  $p = x^k = x^n$.

Anmerkung

- for-Schleifen sind für derartige Beweise aus mehreren Gründen schlecht geeignet:
  - Die Veränderung der Laufvariablen erfolgt (je nach konkreter syntaktischer Form) mehr oder weniger „versteckt“.
  - Vor Beginn und nach Beendigung der Schleife existiert die Laufvariable meist gar nicht.

  Deshalb sollten for-Schleifen in äquivalente while-Schleifen umgeschrieben werden, wenn die Laufvariable in der Schleifeninvariante gebraucht wird.

5.5.7 Algorithmus von Kruskal

Gegeben

- Ungerichteter, gewichteter Graph $G = (V, E, \rho)$

Algorithmus

1. Setze $F = \emptyset$.
2. Sortiere die Kantenmenge E nach aufsteigenden Gewichten.
3. Für jede Kante $\{u, v\} \in E$ in dieser sortierten Reihenfolge:

   Wenn die Knoten u und v noch nicht durch Kanten aus F verbunden sind:

   Füge die Kante $\{u, v\}$ zur Menge F hinzu.

Ergebnis

- Nach Ausführung des Algorithmus ist die Kantenmenge F ein Minimalgerüst von G (das aus $|V| - |F|$ Bäumen besteht).

- Wenn G zusammenhängend ist (d. h. wenn $|V| - |F| = 1$ ist), ist F folglich ein minimaler Spannbaum von G.

Beispiel

❒ Siehe § 5.5.4.

Korrektheit

1. Nachdem eine Kante $\{u, v\} \in E$ in Schritt 3 des Algorithmus verarbeitet wurde, sind die Knoten u und v durch Kanten aus F verbunden.

2. Die Kantenmenge F ist zu jedem Zeitpunkt Teilmenge eines Minimalgerüsts von G, d. h. es gibt immer ein Minimalgerüst M mit $F \subseteq M$.

3. Nach Ausführung des Algorithmus ist die Kantenmenge F ein Minimalgerüst von G.

Beweis von Aussage 1:

❒ Wenn u und v vor der Verarbeitung der Kante $\{u, v\}$ bereits durch Kanten aus F verbunden sind, ist nichts zu zeigen.

❒ Andernfalls wird die Kante zur Menge F hinzugefügt, sodass die Behauptung anschließend erfüllt ist.

Beweis von Aussage 2 als Schleifeninvariante:

- Initialisierung:
  Vor Beginn der Schleife in Schritt 3 des Algorithmus ist $F = \emptyset$, und somit gilt trivialerweise $F \subseteq M$ für jedes Minimalgerüst M.

- Aufrechterhaltung:
  - Bevor in Schritt 3 eine Kante $e = \{u, v\}$ zur Menge F hinzugefügt wird, ist die ursprüngliche Menge F aufgrund der Invariante Teilmenge eines Minimalgerüsts M.
  - Da M ein Gerüst von G ist, gibt es in M einen Weg von u nach v.
  - Da u und v noch nicht durch Kanten aus F verbunden sind, muss es auf diesem Weg mindestens eine Kante $e' = \{u', v'\}$ geben, deren Knoten u' und v' ebenfalls noch nicht durch Kanten aus F verbunden sind.
    Insbesondere muss $e' \notin F$ und somit $F \subseteq M \setminus \{e'\}$ gelten.
  - Deshalb gilt für die neue Menge $F' = F \cup \{e\}$: $F' \subseteq M \setminus \{e'\} \cup \{e\} =: M'$, wobei M' aufgrund des Lemmas in § 5.5.1 ebenfalls ein Gerüst von G ist.
  - Da u' und v' noch nicht durch Kanten aus F verbunden sind, wurde die Kante e' wegen Aussage 1 vom Algorithmus noch nicht verarbeitet.
    Da die Kanten nach aufsteigendem Gewicht verarbeitet werden, gilt folglich $\rho(e) \leq \rho(e')$ und somit $\rho(M') = \rho(M) - \rho(e') + \rho(e) \leq \rho(M)$.

- Da M bereits ein Minimalgerüst von G ist, muss somit auch M' ein Minimalgerüst sein.
- Wegen $F' \subseteq M'$ ist die neue Menge F' also wiederum Teilmenge eines Minimalgerüsts von G.

Beweis von Aussage 3:

- Nach Aussage 2 gilt nach Ausführung des Algorithmus:
  $F \subseteq M$ für ein Minimalgerüst M von G.
- Für jede Kante $\{u, v\} \in M$ gilt:
  - Nach Aussage 1 sind die Knoten u und v durch Kanten aus F verbunden.
  - Wegen $F \subseteq M$ sind u und v in M durch dieselben Kanten verbunden.
  - Da es im Gerüst M höchstens eine Verbindung von u und v gibt, kann diese Verbindung nur aus der Kante $\{u, v\}$ bestehen.
  - Also muss die Kante $\{u, v\}$ auch zur Menge F gehören.
- Also gilt $F = M$, d. h. F ist ein Minimalgerüst von G.

Hilfsdatenstruktur zur effizienten Implementierung

- Um effizient überprüfen zu können, ob zwei Knoten u und v bereits durch Kanten aus F verbunden sind, werden die Knoten des Graphen sukzessive zu Bäumen zusammengefasst (deren Kanten jedoch nichts mit den Kanten des Graphen G bzw. des vom Algorithmus konstruierten Minimalgerüsts F zu tun haben).

- Für jeden Knoten $v \in V$ werden $\pi(v)$, $\tau(v)$ und $\delta(v)$ hierfür wie folgt definiert:

- Wenn v der Wurzelknoten eines Baums ist, ist $\pi(v) = \perp$.
  Andernfalls ist $\pi(v)$ der Vorgänger von v im entsprechenden Baum.

- Durch Verfolgen der Vorgängerkette findet man zu einem Knoten v den zugehörigen Wurzelknoten $\tau(v) = \begin{cases} v, & \text{falls } \pi(v) = \perp \\ \tau(\pi(v)) & \text{sonst} \end{cases}$

- Für einen Knoten v ist $\delta(v)$ die Tiefe des Teilbaums mit Wurzel v.

- Initialisierung der Hilfsdatenstruktur:
  Zu Beginn des Algorithmus wird $\pi(v) = \perp$ und $\delta(v) = 0$ für jeden Knoten $v \in V$ gesetzt, d. h. jeder Knoten stellt einen einelementigen Baum dar.

- Immer, wenn in Schritt 3 des Algorithmus eine Kante $\{u, v\}$ zur Menge F hinzugefügt wird, werden die Bäume, zu denen u und v gehören, zu einem einzigen Baum zusammengefasst, indem einer der beiden (im Zweifelsfall der mit der geringeren Tiefe) in den anderen eingehängt wird (auf diese Weise bleibt die Tiefe der Bäume möglichst klein), das heißt:
  - Seien $u' = \tau(u)$ und $v' = \tau(v)$ die Wurzelknoten der beiden Bäume.
  - Wenn $\delta(u') < \delta(v')$: Setze $\pi(u') = v'$,
    d. h. hänge den Baum mit Wurzel u' in den Baum mit Wurzel v' ein,
    dessen Tiefe dabei unverändert bleibt.
  - Wenn $\delta(u') > \delta(v')$: Setze $\pi(v') = u'$ (also gerade umgekehrt).
  - Wenn $\delta(u') = \delta(v')$: Setze $\pi(u') = v'$ und $\delta(v') = \delta(v') + 1$,
    d. h. hänge den Baum mit Wurzel u' in den Baum mit Wurzel v' ein,
    dessen Tiefe dabei um eins größer wird.

- Dann gilt offensichtlich zu jedem Zeitpunkt:
  Zwei Knoten u und v sind genau dann durch Kanten in F verbunden, wenn $\tau(u) = \tau(v)$ gilt, d. h. wenn u und v zum selben Baum gehören.

Optimierungsmöglichkeit

- Jedesmal, wenn mittels $\tau(v)$ der zu einem Knoten v gehörende Wurzelknoten r bestimmt wurde, wird für alle dabei durchlaufenen Knoten u ihr Vorgänger $\pi(u)$ auf r gesetzt, d. h. diese Knoten werden direkt in den Wurzelknoten r eingehängt.

- Damit werden anschließende Berechnungen von $\tau(u)$ für diese Knoten u. U. erheblich beschleunigt.

Laufzeit der Operationen auf der Hilfsdatenstruktur

Behauptung:

- Ein Baum der Hilfsdatenstruktur mit Tiefe k enthält mindestens 2^k Knoten.

Beweis durch vollständige Induktion nach k:

- Induktionsanfang $k = 0$:
  - Ein Baum mit Tiefe $k = 0$ enthält genau $1 = 2^0 = 2^k$ Knoten.
- Induktionsschritt $k \to k + 1$:
  - Ein Baum mit Tiefe $k + 1$ ist ursprünglich aus zwei Bäumen mit Tiefe k entstanden (Fall $\delta(u') = \delta(v')$ beim Zusammenfassen zweier Bäume), zu dem später eventuell weitere Bäume mit Tiefe $\leq k$ hinzugefügt wurden (Fälle $\delta(u') < \delta(v')$ und $\delta(u') > \delta(v')$).
  - Deshalb enthält ein Baum mit Tiefe $k + 1$ nach Induktionsvoraussetzung mindestens $2 \cdot 2^k = 2^{k+1}$ Knoten.

Daraus folgt:

- Ein Baum mit N Knoten hat höchstens Tiefe $\log_2 N$.
- Die Länge der Vorgängerkette eines Knotens ist $O(\log |V|)$.
- Sowohl der Test $\tau(u) = \tau(v)$ zur Überprüfung, ob u und v bereits durch Kanten in F verbunden sind, als auch das Zusammenfassen zweier Bäume hat Laufzeit $O(\log |V|)$.

Laufzeit des Algorithmus

- Sortieren der Kantenmenge E mit einem geeigneten Verfahren:
  $O(|E| \log |E|) = O\left(|E| \log |V|^2\right) = O(|E| \, 2 \log |V|) = O(|E| \log |V|)$
  (Für jeden Graphen gilt: $|E| \leq |V|^2$.)
- Operationen auf der Hilfsdatenstruktur:
  $|E|$-mal Test $\tau(u) = \tau(v)$ und höchstens $|E|$-mal Zusammenfassen zweier Bäume, insgesamt also $O(|E| \log |V|)$.
- Gesamtlaufzeit somit: $O(|E| \log |V|)$

5.5.8 Algorithmus von Prim

Gegeben

- Ungerichteter, gewichteter Graph $G = (V, E, \rho)$

Algorithmus

1. Für jeden Knoten $v \in V$:
   1. Füge v mit Priorität $\delta(v) = \infty$ in eine Minimum-Vorrangwarteschlange Q ein.
   2. Setze $\pi(v) = \perp$.
2. Solange Q nicht leer ist:
   1. Entnimm einen Knoten u mit minimaler Priorität.
   2. Für jeden Nachfolger v von u:

      Wenn $v \in Q$ und $\rho(u, v) < \delta(v)$:
      1. Erniedrige die Priorität $\delta(v)$ auf $\rho(u, v)$.
      2. Setze $\pi(v) = u$.

Laufzeit

- Operationen auf der Vorrangwarteschlange:
  - $|V|$-mal Einfügen eines Knotens
  - $|V|$-mal Test, ob die Warteschlange leer ist
  - $|V|$-mal Entnehmen eines Knotens mit minimaler Priorität
  - $|E|$-mal Test, ob ein Knoten enthalten ist
    (Der Rumpf der inneren Schleife wird insgesamt $|E|$-mal ausgeführt.)
  - Maximal $|E|$-mal Erniedrigen der Priorität eines Knotens

  Insgesamt: $O(|V| + |E|)$

- Laufzeit jeder solchen Operation: $O(\log |V|)$,
  da die Warteschlange maximal $|V|$ Einträge enthält.

- Gesamtlaufzeit somit: $O((|V| + |E|) \log |V|)$

- Für zusammenhängende Graphen: $O(|E| \log |V|)$, da für sie $|E| \geq |V| - 1$ gilt.

Ergebnis

- Nach Ausführung des Algorithmus ist die Kantenmenge $F = \{\{\pi(v), v\} \mid \pi(v) \neq \perp\}$ ein Minimalgerüst von G (das aus $|V| - |F|$ Bäumen besteht).

- Wenn G zusammenhängend ist (d. h. wenn $|V| - |F| = 1$ ist), ist F folglich ein minimaler Spannbaum von G.

- Andernfalls kann der Algorithmus dies bei Bedarf feststellen:
  Wenn mehr als einmal ein Knoten u mit $\delta(u) = \infty$ entnommen wird, ist der Graph nicht zusammenhängend.

Beispiel

- Siehe § 5.5.4.

Korrektheit

- Bezeichnungen (jeweils am Anfang eines Schleifendurchlaufs):
  - Q = Menge aller Knoten, die sich noch in der Vorrangwarteschlange befinden
  - $P = V \setminus Q$ = Menge aller Knoten, die bereits entnommen wurden
  - $PQ = \{\{p, q\} \in E \mid p \in P, q \in Q\} =$
    Menge aller Kanten, die Knoten aus P mit Knoten aus Q verbinden
  - $F = \{\{\pi(v), v\} \mid v \in P, \pi(v) \neq \perp\}$

- Schleifeninvariante für Schritt 2 des Algorithmus:
  1. Für alle Knoten $q \in Q$ gilt entweder $\pi(q) = \perp$ und $\delta(q) = \infty$
     oder $\pi(q) \in P$, $\{\pi(q), q\} \in E$ und $\delta(q) = \rho(\pi(q), q)$
  2. Für alle Kanten $\{p, q\} \in PQ$ gilt: $\delta(q) \leq \rho(p, q)$
  3. Die Kantenmenge F ist Teilmenge eines Minimalgerüsts von G,
     d. h. es gibt ein Minimalgerüst M mit $F \subseteq M$
  4. Für alle Knoten $p_1, p_2 \in P$ gilt:
     Wenn p_1 und p_2 zur gleichen Zusammenhangskomponente von G gehören,
     dann sind sie durch Kanten in F verbunden.

- Initialisierung: Vor Beginn der Schleife gilt:
  1. $\pi(q) = \perp$ und $\delta(q) = \infty$ für alle $q \in Q$
  2. $P = \emptyset$ und somit auch $PQ = \emptyset$
  3. $F = \emptyset$ und somit $F \subseteq M$ für jedes Minimalgerüst M
  4. $P = \emptyset$

- Aufrechterhaltung: Wenn die Invariante am Anfang eines Schleifendurchlaufs gilt, gilt am Ende dieses Durchlaufs:
  1. Wenn $\pi(v)$ und $\delta(v)$ in Schritt 2.2 verändert werden, bleibt Teil 1 der Invariante jeweils erhalten.
  2. Weil der Knoten u zur Menge P hinzukommt, muss Teil 2 der Invariante zusätzlich für alle Kanten $\{u, v\}$ mit $v \in Q$ gelten.
     Dies wird durch Schritt 2.2 sichergestellt.

     Für alle anderen Kanten $\{p, q\} \in PQ$ gilt Teil 2 der Invariante weiterhin, weil $\delta(q)$ während eines Schleifendurchlaufs nur kleiner, aber niemals größer werden kann.
  3. Wenn $\pi(u) = \perp$ ist, ist nichts zu zeigen, weil die Menge F in diesem Fall unverändert bleibt.

     Wenn $\pi(u) \neq \perp$ ist, kommt die Kante $e = \{\pi(u), u\}$ zur Menge F hinzu, und es gilt:

- Aufgrund von Teil 3 der Invariante ist die ursprüngliche Menge F Teilmenge eines Minimalgerüsts M von G.
- Da M ein Gerüst von G ist, gibt es in M einen Weg von $\pi(u) \in P$ nach $u \in Q$, der mindestens eine Kante $e' = \{ p, q \} \in PQ$ enthalten muss.
  (Beachte: u gehört noch zur ursprünglichen Menge Q.)
- Es gilt: $\rho(e) = \rho(\pi(u), u) \underset{\text{(a)}}{=} \delta(u) \underset{\text{(b)}}{\leq} \delta(q) \underset{\text{(c)}}{\leq} \rho(p, q) = \rho(e')$, denn:
  a) Teil 1 der Invariante
  b) Es wird ein Knoten u mit minimaler Priorität $\delta(u)$ entnommen
  c) Teil 2 der Invariante
- Deshalb gilt für $M' := M \setminus \{ e' \} \cup \{ e \}$, das aufgrund des Lemmas in § 5.5.1 ebenfalls ein Gerüst von G ist:
  $\rho(M') = \rho(M) - \rho(e') + \rho(e) \leq \rho(M)$.
- Da M bereits ein Minimalgerüst von G ist, muss somit auch M' ein Minimalgerüst sein.
- Wegen $q \notin P$ gilt $e' = \{ p, q \} \notin F$ (F enthält nur Kanten $\{ \pi(v), v \}$ mit $v \in P$ und $\pi(v) \in P$) und somit $F' = F \cup \{ e \} = F \setminus \{ e' \} \cup \{ e \} \subseteq M \setminus \{ e' \} \cup \{ e \} = M'$, d. h. die neue Menge F' ist wiederum Teilmenge eines Minimalgerüsts von G.

4. Weil der Knoten u zur Menge P hinzukommt, muss Teil 4 der Invariante zusätzlich für $p_1 = u$ und jeden Knoten $p_2 = p \in P$ gelten.

   Wenn $\delta(u) = \infty$ ist:

   - Weil ein Knoten u mit minimaler Priorität $\delta(u)$ entnommen wird, ist $\delta(q) = \infty$ für alle Knoten $q \in Q$.
   - Deshalb gibt es wegen Teil 2 der Invariante zu diesem Zeitpunkt keine Kanten in PQ und deshalb keinen Weg von $p \in P$ nach $u \in Q$, d. h. p und u gehören nicht zur gleichen Zusammenhangskomponente.

   Wenn $\delta(u) < \infty$ ist:

   - Nach Teil 1 der Invariante gilt: $\pi(u) \in P$ und $\{ \pi(u), u \} \in E$.
   - Wenn p und u zur gleichen Zusammenhangskomponente gehören, gehört wegen $\{ \pi(u), u \} \in E$ auch $\pi(u) \in P$ zu dieser Zusammenhangskomponente, das aufgrund von Teil 4 der Invariante bereits durch Kanten in F mit p verbunden ist.
   - Da die Kante $\{ \pi(u), u \}$ zu F hinzukommt, sind dann auch p und u durch Kanten in F verbunden.

   Für alle übrigen Knoten $p_1, p_2 \in P$ gilt Teil 4 der Invariante weiterhin, weil die Kantenmenge F während eines Schleifendurchlaufs nur größer, aber niemals kleiner werden kann.

- Terminierung: Nach Beendigung der Schleife gilt:

  3. Die endgültige Kantenmenge F ist Teilmenge eines Minimalgerüsts von G.
  4. Für alle Knoten $p_1, p_2 \in P = V$ gilt:
     Wenn p_1 und p_2 zur gleichen Zusammenhangskomponente von G gehören, dann sind sie durch Kanten in F verbunden.

     Damit ist F ein vollständiges Minimalgerüst von G.

Modifizierter Algorithmus mit vorgegebenem Startknoten

Gegeben

- Ungerichteter, gewichteter Graph $G = (V, E, \rho)$
- Startknoten $s \in V$

Algorithmus

1 Für jeden Knoten $v \in V \setminus \{ s \}$:
   1 Füge v mit Priorität $\delta(v) = \infty$ in eine Minimum-Vorrangwarteschlange Q ein.
   2 Setze $\pi(v) = \perp$.
2 Setze $\pi(s) = \perp$.
3 Setze $u = s$.
4 Solange Q nicht leer ist:
   1 Für jeden Nachfolger v von u:

     Wenn $v \in Q$ und $\rho(u, v) < \delta(v)$:
      1 Erniedrige die Priorität $\delta(v)$ auf $\rho(u, v)$.
      2 Setze $\pi(v) = u$.
   2 Entnimm einen Knoten u mit minimaler Priorität.