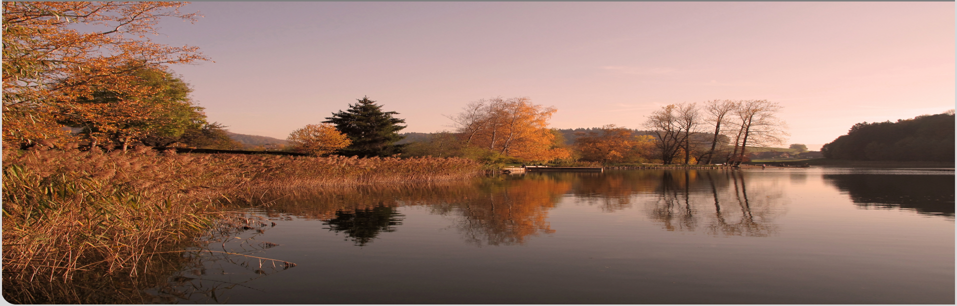
Übungen zu Algorithmentechnik Wintersemester 09/10

3. Sitzung

Thomas Pajor | 19. November 2009



Aufgabe 1

Minimal spannende Bäume – ein bisschen Theorie

Sei $G = (V, E)$ ein ungerichteter Graph mit Gewichten $w : E \to \mathbb{R}$.

Aufgabe 1

Minimal spannende Bäume – ein bisschen Theorie

Sei $G = (V, E)$ ein ungerichteter Graph mit Gewichten $w : E \to \mathbb{R}$.

(a) Zeigen Sie: Wenn in G für jeden Schnitt die den Schnitt kreuzende Kante minimalen Gewichts eindeutig ist, dann hat G einen eindeutigen spannenden Baum minimalen Gewichts.

Aufgabe 1

Minimal spannende Bäume – ein bisschen Theorie

Sei $G = (V, E)$ ein ungerichteter Graph mit Gewichten $w : E \to \mathbb{R}$.

(a) Zeigen Sie: Wenn in G für jeden Schnitt die den Schnitt kreuzende Kante minimalen Gewichts eindeutig ist, dann hat G einen eindeutigen spannenden Baum minimalen Gewichts.

(b) Gilt die Umkehrung von (a)?

Aufgabe 1

Minimal spannende Bäume – ein bisschen Theorie

Sei $G = (V, E)$ ein ungerichteter Graph mit Gewichten $w : E \to \mathbb{R}$.

(a) Zeigen Sie: Wenn in G für jeden Schnitt die den Schnitt kreuzende Kante minimalen Gewichts eindeutig ist, dann hat G einen eindeutigen spannenden Baum minimalen Gewichts.

(b) Gilt die Umkehrung von (a)?

(c) Zeigen oder widerlegen Sie: Wenn in G mit alle Kanten paarweise verschiedene Gewichte haben, dann hat G einen eindeutigen spannenden Baum minimalen Gewichts.

Aufgabe 1

Minimal spannende Bäume – ein bisschen Theorie

Sei $G = (V, E)$ ein ungerichteter Graph mit Gewichten $w : E \rightarrow \mathbb{R}$.

(a) Zeigen Sie: Wenn in G für jeden Schnitt die den Schnitt kreuzende Kante minimalen Gewichts eindeutig ist, dann hat G einen eindeutigen spannenden Baum minimalen Gewichts.

(b) Gilt die Umkehrung von (a)?

(c) Zeigen oder widerlegen Sie: Wenn in G mit alle Kanten paarweise verschiedene Gewichte haben, dann hat G einen eindeutigen spannenden Baum minimalen Gewichts.

(d) Zeigen oder widerlegen Sie die Umkehrung der Aussage in (c).

Aufgabe 2

Zweitbeste minimal spannende Bäume

Gegeben: Ungerichteter Graph $G = (V, E)$ mit paarweise verschiedenenen Kantengewichten $w(e)$.

Sei $\mathcal{B}$ Menge aller Spannbäume und B_{min} der minimal spannende Baum, dann ist ein *zweitbester minimal spannender Baum* (2MST) B definiert durch

$$B \in \underset{B \in \mathcal{B} \setminus \{B_{\text{min}}\}}{\operatorname{argmin}} \{w(B)\}$$

Aufgabe 2

Zweitbeste minimal spannende Bäume

Gegeben: Ungerichteter Graph $G = (V, E)$ mit paarweise verschiedenenen Kantengewichten $w(e)$.

Sei $\mathcal{B}$ Menge aller Spannbäume und B_{min} der minimal spannende Baum, dann ist ein *zweitbester minimal spannender Baum* (2MST) B definiert durch

$$B \in \underset{B \in \mathcal{B} \setminus \{B_{\text{min}}\}}{\operatorname{argmin}} \{w(B)\}$$

(a) Zeigen oder widerlegen Sie: Ist der minimal spannende Baum eindeutig, so ist auch der zweitbeste minimal spannende Baum eindeutig.

Aufgabe 2

Zweitbeste minimal spannende Bäume

Gegeben: Ungerichteter Graph $G = (V, E)$ mit paarweise verschiedenenen Kantengewichten $w(e)$.

Sei $\mathcal{B}$ Menge aller Spannbäume und B_{min} der minimal spannende Baum, dann ist ein *zweitbester minimal spannender Baum* (2MST) B definiert durch

$$B \in \operatorname*{argmin}_{B \in \mathcal{B} \setminus \{B_{\text{min}}\}} \{w(B)\}$$

(a) Zeigen oder widerlegen Sie: Ist der minimal spannende Baum eindeutig, so ist auch der zweitbeste minimal spannende Baum eindeutig.

(b) Sei B_{min} der minimal spannende Baum in G. Zeigen Sie: Es gibt Kanten $e \in B_{\text{min}}$ und $e' \in E \setminus B_{\text{min}}$ so, dass $B_{\text{min}} \setminus \{e\} \cup \{e'\}$ ein zweitbester minimal spannender Baum von G ist.

Aufgabe 2

Zweitbeste minimal spannende Bäume

Gegeben: Ungerichteter Graph $G = (V, E)$ mit paarweise verschiedenenen Kantengewichten $w(e)$.

Sei $\mathcal{B}$ Menge aller Spannbäume und B_{min} der minimal spannende Baum, dann ist ein *zweitbester minimal spannender Baum* (2MST) B definiert durch

$$B \in \underset{B \in \mathcal{B} \setminus \{B_{\text{min}}\}}{\operatorname{argmin}} \{w(B)\}$$

(c) Sei B ein spannender Baum in G. Für zwei Knoten $u, v \in V$ werde mit $\text{maxedge}(u, v)$ die Kante *maximalen* Gewichts auf dem eindeutigen u-v-Pfad in B bezeichnet. Geben Sie einen Algorithmus mit Laufzeit $\mathcal{O}(|V|^2)$ an, der $\text{maxedge}(u, v)$ für alle Knotenpaare berechnet.

Hinweis: Starten Sie von jedem Knoten eine modifizierte Breitensuche eingeschränkt auf B.

Aufgabe 2

Zweitbeste minimal spannende Bäume

Gegeben: Ungerichteter Graph $G = (V, E)$ mit paarweise verschiedenenen Kantengewichten $w(e)$.

Sei $\mathcal{B}$ Menge aller Spannbäume und B_{min} der minimal spannende Baum, dann ist ein *zweitbester minimal spannender Baum* (2MST) B definiert durch

$$B \in \underset{B \in \mathcal{B} \setminus \{B_{\text{min}}\}}{\operatorname{argmin}} \{w(B)\}$$

(d) Basierend auf Aufgabe (b) und (c), geben Sie einen Algorithmus mit Laufzeit $\mathcal{O}(|V|^2)$ an, der einen zweitbesten minimal spannenden Baum berechnet.

MaxEdge-Algorithmus

Eingabe : Graph $G = (V, E)$, Spannender Baum B.
Seiteneffekte : maxedge(u, v) für alle paare $u, v \in V$.

maxedge$(u, v) \leftarrow \bot$ für alle $u, v \in V$
für alle $u \in V$ **tue**
    $Q \leftarrow \emptyset$
    `Enqueue`(Q, u)
    solange $Q \neq \emptyset$ **tue**
        $x \leftarrow$ `Dequeue`(Q)
        für alle $(x, v) \in B$ **tue**
            wenn maxedge$(u, v) = \bot$ **und** $u \neq v$ **dann**
                wenn $w(x, v) > w($maxedge$(u, x))$ **oder** $x = u$ **dann**
                    maxedge$(u, v) \leftarrow (x, v)$
                sonst
                    maxedge$(u, v) \leftarrow$ maxedge(u, x)
                `Enqueue` (Q, v)

MaxEdge-Algorithmus

Eingabe : Graph $G = (V, E)$, Spannender Baum B.
Seiteneffekte : maxedge(u, v) für alle paare $u, v \in V$.

```
maxedge(u, v) ← ⊥ für alle u, v ∈ V
für alle u ∈ V tue
    Q ← ∅
    Enqueue(Q, u)
    solange Q ≠ ∅ tue
        x ← Dequeue(Q)
        für alle (x, v) ∈ B tue
            wenn maxedge(u, v) = ⊥ und u ≠ v dann
                wenn w(x, v) > w(maxedge(u, x)) oder x = u dann
                    maxedge(u, v) ← (x, v)
                sonst
                    maxedge(u, v) ← maxedge(u, x)
                Enqueue(Q, v)
```

MaxEdge-Algorithmus

Eingabe : Graph $G = (V, E)$, Spannender Baum B.
Seiteneffekte : maxedge(u, v) für alle paare $u, v \in V$.

maxedge$(u, v) \leftarrow \perp$ für alle $u, v \in V$
für alle $u \in V$ **tue**
    $Q \leftarrow \emptyset$
    `Enqueue` (Q, u)
    solange $Q \neq \emptyset$ **tue**
        $x \leftarrow$ `Dequeue` (Q)
        für alle $(x, v) \in B$ **tue**
            wenn maxedge$(u, v) = \perp$ **und** $u \neq v$ **dann**
                wenn $w(x, v) > w(\text{maxedge}(u, x))$ **oder** $x = u$ **dann**
                    maxedge$(u, v) \leftarrow (x, v)$
                sonst
                    maxedge$(u, v) \leftarrow$ maxedge(u, x)
                `Enqueue` (Q, v)

MAXEDGE-Algorithmus

Eingabe : Graph $G = (V, E)$, Spannender Baum B.
Seiteneffekte : maxedge(u, v) für alle paare $u, v \in V$.

```
maxedge(u, v) ← ⊥ für alle u, v ∈ V
für alle u ∈ V tue
    Q ← ∅
    Enqueue(Q, u)
    solange Q ≠ ∅ tue
        x ← Dequeue(Q)
        für alle (x, v) ∈ B tue
            wenn maxedge(u, v) = ⊥ und u ≠ v dann
                wenn w(x, v) > w(maxedge(u, x)) oder x = u dann
                    maxedge(u, v) ← (x, v)
                sonst
                    maxedge(u, v) ← maxedge(u, x)
                Enqueue(Q, v)
```

MaxEdge-Algorithmus

Eingabe : Graph $G = (V, E)$, Spannender Baum B.
Seiteneffekte : maxedge(u, v) für alle paare $u, v \in V$.

maxedge$(u, v) \leftarrow \perp$ für alle $u, v \in V$
für alle $u \in V$ **tue**
    $\mathrm{Q} \leftarrow \emptyset$
    `Enqueue` (Q, u)
    solange $\mathrm{Q} \neq \emptyset$ **tue**
        $x \leftarrow$ `Dequeue` (Q)
        für alle $(x, v) \in B$ **tue**
            wenn maxedge$(u, v) = \perp$ **und** $u \neq v$ **dann**
                wenn $w(x, v) > w(\text{maxedge}(u, x))$ **oder** $x = u$ **dann**
                    maxedge$(u, v) \leftarrow (x, v)$
                sonst
                    maxedge$(u, v) \leftarrow$ maxedge(u, x)
                `Enqueue` (Q, v)

MaxEdge-Algorithmus

Eingabe : Graph $G = (V, E)$, Spannender Baum B.
Seiteneffekte : maxedge(u, v) für alle paare $u, v \in V$.

maxedge$(u, v) \leftarrow \perp$ für alle $u, v \in V$
für alle $u \in V$ **tue**
    $Q \leftarrow \emptyset$
    `Enqueue`(Q, u)
    solange $Q \neq \emptyset$ **tue**
        $x \leftarrow$ `Dequeue`(Q)
        für alle $(x, v) \in B$ **tue**
            wenn maxedge$(u, v) = \perp$ **und** $u \neq v$ **dann**
                wenn $w(x, v) > w($maxedge$(u, x))$ **oder** $x = u$ **dann**
                    maxedge$(u, v) \leftarrow (x, v)$
                sonst
                    maxedge$(u, v) \leftarrow$ maxedge(u, x)
                `Enqueue` (Q, v)

MaxEdge-Algorithmus

Eingabe : Graph $G = (V, E)$, Spannender Baum B.
Seiteneffekte : maxedge(u, v) für alle paare $u, v \in V$.

```
maxedge(u, v) ← ⊥ für alle u, v ∈ V
für alle u ∈ V tue
    Q ← ∅
    Enqueue(Q, u)
    solange Q ≠ ∅ tue
        x ← Dequeue(Q)
        für alle (x, v) ∈ B tue
            wenn maxedge(u, v) = ⊥ und u ≠ v dann
                wenn w(x, v) > w(maxedge(u, x)) oder x = u dann
                    maxedge(u, v) ← (x, v)
                sonst
                    maxedge(u, v) ← maxedge(u, x)
                Enqueue(Q, v)
```

2MST-Algorithmus

Eingabe: Graph $G = (V, E)$ und minimal spannender Baum B_{min} von G.
Ausgabe: Zweitbester minimal spannder Baum in G.

2MST-Algorithmus

Eingabe: Graph $G = (V, E)$ und minimal spannender Baum B_{min} von G.
Ausgabe: Zweitbester minimal spannder Baum in G.

- Berechne maxedge(u, v) für alle $u, v \in V$.

2MST-Algorithmus

Karlsruhe Institute of Technology

Eingabe: Graph $G = (V, E)$ und minimal spannender Baum B_{min} von G.
Ausgabe: Zweitbester minimal spannder Baum in G.

- Berechne maxedge(u, v) für alle $u, v \in V$.
- Berechne für jede Kante $e = (u, v) \in E \setminus B_{\text{min}}$ mithilfe von maxedge(u, v) die Kante $e' \in B_{\text{min}}$ die

$$w(B) = w(B_{\text{min}}) + w(e) - w(e')$$

minimiert. Merke das global minimierende Paar (e, e').

2MST-Algorithmus

Eingabe: Graph $G = (V, E)$ und minimal spannender Baum B_{min} von G.
Ausgabe: Zweitbester minimal spannder Baum in G.

- Berechne maxedge(u, v) für alle $u, v \in V$.
- Berechne für jede Kante $e = (u, v) \in E \setminus B_{\text{min}}$ mithilfe von maxedge(u, v) die Kante $e' \in B_{\text{min}}$ die

  $$w(B) = w(B_{\text{min}}) + w(e) - w(e')$$

  minimiert. Merke das global minimierende Paar (e, e').
- Gib $B := B_{\text{min}} \setminus \{e'\} \cup \{e\}$ aus.

2MST-Algorithmus

Eingabe: Graph $G = (V, E)$ und minimal spannender Baum B_{min} von G.
Ausgabe: Zweitbester minimal spannder Baum in G.

- Berechne maxedge(u, v) für alle $u, v \in V$. – $\mathcal{O}(|V|^2)$
- Berechne für jede Kante $e = (u, v) \in E \setminus B_{\text{min}}$ mithilfe von maxedge(u, v) die Kante $e' \in B_{\text{min}}$ die

  $$w(B) = w(B_{\text{min}}) + w(e) - w(e')$$

  minimiert. Merke das global minimierende Paar (e, e').
- Gib $B := B_{\text{min}} \setminus \{e'\} \cup \{e\}$ aus.

2MST-Algorithmus

Eingabe: Graph $G = (V, E)$ und minimal spannender Baum B_{min} von G.
Ausgabe: Zweitbester minimal spannder Baum in G.

- Berechne maxedge(u, v) für alle $u, v \in V$. – $\mathcal{O}(|V|^2)$
- Berechne für jede Kante $e = (u, v) \in E \setminus B_{\text{min}}$ mithilfe von maxedge(u, v) die Kante $e' \in B_{\text{min}}$ die

$$w(B) = w(B_{\text{min}}) + w(e) - w(e')$$

  minimiert. Merke das global minimierende Paar (e, e'). – $\mathcal{O}(|V|^2)$
- Gib $B := B_{\text{min}} \setminus \{e'\} \cup \{e\}$ aus.

2MST-Algorithmus

Eingabe: Graph $G = (V, E)$ und minimal spannender Baum B_{min} von G.
Ausgabe: Zweitbester minimal spannder Baum in G.

- Berechne maxedge(u, v) für alle $u, v \in V$. – $\mathcal{O}(|V|^2)$
- Berechne für jede Kante $e = (u, v) \in E \setminus B_{\text{min}}$ mithilfe von maxedge(u, v) die Kante $e' \in B_{\text{min}}$ die

  $$w(B) = w(B_{\text{min}}) + w(e) - w(e')$$

  minimiert. Merke das global minimierende Paar (e, e'). – $\mathcal{O}(|V|^2)$
- Gib $B := B_{\text{min}} \setminus \{e'\} \cup \{e\}$ aus. – $\mathcal{O}(1)$

2MST-Algorithmus

Eingabe: Graph $G = (V, E)$ und minimal spannender Baum B_{min} von G.
Ausgabe: Zweitbester minimal spannder Baum in G.

- Berechne maxedge(u, v) für alle $u, v \in V$. – $\mathcal{O}(|V|^2)$
- Berechne für jede Kante $e = (u, v) \in E \setminus B_{\text{min}}$ mithilfe von maxedge(u, v) die Kante $e' \in B_{\text{min}}$ die

  $$w(B) = w(B_{\text{min}}) + w(e) - w(e')$$

  minimiert. Merke das global minimierende Paar (e, e'). – $\mathcal{O}(|V|^2)$
- Gib $B := B_{\text{min}} \setminus \{e'\} \cup \{e\}$ aus. – $\mathcal{O}(1)$

$\sum$: $\mathcal{O}(|V|^2)$

Unabhängigkeitssysteme u. Matroide

Unabhängigkeitssystem

Ein Tupel $(M, \mathcal{U})$ mit $\mathcal{U} \subseteq 2^M$ über einer endlichen Menge M heißt *Unabhängigkeitssystem*, wenn gilt

(i) $\emptyset \in \mathcal{U}$

(ii) Wenn $I_1 \in \mathcal{U}$ und $I_2 \subseteq I_1$, dann ist auch $I_2 \in \mathcal{U}$

Mengen $I \in \mathcal{U}$ heißen *unabhängig*, Mengen $I \in 2^M \setminus \mathcal{U}$ *abhängig*.

Unabhängigkeitssysteme u. Matroide

Unabhängigkeitssystem

Ein Tupel $(M, \mathcal{U})$ mit $\mathcal{U} \subseteq 2^M$ über einer endlichen Menge M heißt *Unabhängigkeitssystem*, wenn gilt

(i) $\emptyset \in \mathcal{U}$

(ii) Wenn $I_1 \in \mathcal{U}$ und $I_2 \subseteq I_1$, dann ist auch $I_2 \in \mathcal{U}$

Mengen $I \in \mathcal{U}$ heißen *unabhängig*, Mengen $I \in 2^M \setminus \mathcal{U}$ *abhängig*.

Basis eines Unabhängigkeitssystems

Zu $(M, \mathcal{U})$ heißt eine bezüglich $\subseteq$ maximale Menge $B \in \mathcal{U}$ *Basis*. D. h.

$$B \in \mathcal{U} \text{ ist Basis} :\Leftrightarrow \text{Für alle } B' \in \mathcal{U} \text{ mit } B \subseteq B' \text{ gilt } B' = B.$$

Die Menge aller Basen $\mathcal{B} \subseteq \mathcal{U}$ heißt *Basissystem* von $(M, \mathcal{U})$.

Unabhängigkeitssysteme u. Matroide

Optimierungsproblem über Unabhängigkeitssystem

Gegeben: Zu $(M, \mathcal{U})$ sei $w : M \to \mathbb{R}$ eine Gewichtsfunktion.

Gesucht: Basis $B \in \mathcal{U}$ mit $w(B)$ minimal (Minimerungsproblem) oder mit $w(B)$ maximal (Maximierungsproblem).

Unabhängigkeitssysteme u. Matroide

Optimierungsproblem über Unabhängigkeitssystem

Gegeben: Zu $(M,\mathcal{U})$ sei $w : M \to \mathbb{R}$ eine Gewichtsfunktion.

Gesucht: Basis $B \in \mathcal{U}$ mit $w(B)$ minimal (Minimerungsproblem) oder mit $w(B)$ maximal (Maximierungsproblem).

Greedy-Methode auf Unabhängigkeitssystemen

Sortiere M aufsteigend (absteigend). Die Sortierung sei $l_1, l_2, \ldots, l_{|M|}$.

$B \leftarrow \emptyset$
für $i \leftarrow 1, \ldots, |M|$ **tue**
  wenn $B \cup \{l_i\} \in \mathcal{U}$ **dann**
    $B \leftarrow B \cup \{l_i\}$

gibaus B

Unabhängigkeitssysteme u. Matroide

Karlsruhe Institute of Technology

Matroid

Ein Unabhängigkeitssystem $(M, \mathcal{U})$ heißt Matroid, wenn

(iii) Für alle $U_1, U_2 \in \mathcal{U}$ mit $|U_1| < |U_2|$ existiert $l \in U_2 \setminus U_1$, so dass $U_1 \cup \{l\} \in \mathcal{U}$.

Unabhängigkeitssysteme u. Matroide

Matroid

Ein Unabhängigkeitssystem $(M, \mathcal{U})$ heißt Matroid, wenn

(iii) Für alle $U_1, U_2 \in \mathcal{U}$ mit $|U_1| < |U_2|$ existiert $l \in U_2 \setminus U_1$, so dass $U_1 \cup \{l\} \in \mathcal{U}$.

Matroide und die Greedy-Methode

Satz 2.16 und 2.17.

Für ein Unabhängigkeitssystem $(M, \mathcal{U})$ sind äquivalent:

(a) $(M, \mathcal{U})$ ist ein Matroid

(b) Die Greedy-Methode liefert die Optimallösung für das Minimierungs- bzw. Maximierungsproblem über $(M, \mathcal{U})$ zu einer beliebigen Gewichtsfunktion $w : M \to \mathbb{R}$.

(c) Sind $B_1, B_2 \in \mathcal{U}$ Basen, so gilt $|B_1| = |B_2|$.

Dijkstra's Algorithmus

Eingabe : Graph $G = (V, E)$ mit Gewichten $w : E \to \mathbb{R}^+$, Startknoten $s \in V$
Seiteneffekte : Kürzeste Wege und Distanzen zu allen Knoten $v \in V$

$S \leftarrow V$
$P \leftarrow \emptyset$
$\mathrm{pre}(u) \leftarrow \perp$ für alle $u \in V$
$\mathrm{dist}(u) \leftarrow \infty$ für alle $u \in V$

$\mathrm{dist}(s) \leftarrow 0$

solange $S \neq \emptyset$ **tue**
  $u \leftarrow \mathrm{argmin}_{u \in S}\{\mathrm{dist}(u)\}$
  $S \leftarrow S \setminus \{u\}$
  $P \leftarrow P \cup \{(\mathrm{pre}(u), u)\}$

  für alle $e = (u, v) \in E$ **tue**
    wenn $\mathrm{dist}(v) > \mathrm{dist}(u) + w(e)$ **dann**
      $\mathrm{dist}(v) \leftarrow \mathrm{dist}(u) + w(e)$
      $\mathrm{pre}(v) \leftarrow u$

Dijkstra's Algorithmus

Eingabe : Graph $G = (V, E)$ mit Gewichten $w : E \to \mathbb{R}^+$, Startknoten $s \in V$
Seiteneffekte : Kürzeste Wege und Distanzen zu allen Knoten $v \in V$

$S \leftarrow V$
$P \leftarrow \emptyset$
$\mathrm{pre}(u) \leftarrow \perp$ für alle $u \in V$
$\mathrm{dist}(u) \leftarrow \infty$ für alle $u \in V$

$\mathrm{dist}(s) \leftarrow 0$

solange $S \neq \emptyset$ **tue**
    $u \leftarrow \mathrm{argmin}_{u \in S}\{\mathrm{dist}(u)\}$
    $S \leftarrow S \setminus \{u\}$
    $P \leftarrow P \cup \{(\mathrm{pre}(u), u)\}$

    für alle $e = (u, v) \in E$ **tue**
        wenn $\mathrm{dist}(v) > \mathrm{dist}(u) + w(e)$ **dann**
            $\mathrm{dist}(v) \leftarrow \mathrm{dist}(u) + w(e)$
            $\mathrm{pre}(v) \leftarrow u$

Dijkstra's Algorithmus

Karlsruhe Institute of Technology

Eingabe : Graph $G = (V, E)$ mit Gewichten $w : E \to \mathbb{R}^+$, Startknoten $s \in V$
Seiteneffekte : Kürzeste Wege und Distanzen zu allen Knoten $v \in V$

$S \leftarrow V$
$P \leftarrow \emptyset$
$\mathrm{pre}(u) \leftarrow \perp$ für alle $u \in V$
$\mathrm{dist}(u) \leftarrow \infty$ für alle $u \in V$

$\mathrm{dist}(s) \leftarrow 0$

solange $S \neq \emptyset$ **tue**
  $u \leftarrow \mathrm{argmin}_{u \in S}\{\mathrm{dist}(u)\}$
  $S \leftarrow S \setminus \{u\}$
  $P \leftarrow P \cup \{(\mathrm{pre}(u), u)\}$

  für alle $e = (u, v) \in E$ **tue**
    wenn $\mathrm{dist}(v) > \mathrm{dist}(u) + w(e)$ **dann**
      $\mathrm{dist}(v) \leftarrow \mathrm{dist}(u) + w(e)$
      $\mathrm{pre}(v) \leftarrow u$

Dijkstra's Algorithmus

Eingabe : Graph $G = (V, E)$ mit Gewichten $w : E \to \mathbb{R}^+$, Startknoten $s \in V$
Seiteneffekte : Kürzeste Wege und Distanzen zu allen Knoten $v \in V$

$S \leftarrow V$
$P \leftarrow \emptyset$
$\text{pre}(u) \leftarrow \perp$ für alle $u \in V$
$\text{dist}(u) \leftarrow \infty$ für alle $u \in V$

$\text{dist}(s) \leftarrow 0$

solange $S \neq \emptyset$ **tue**
  $u \leftarrow \text{argmin}_{u \in S}\{\text{dist}(u)\}$
  $S \leftarrow S \setminus \{u\}$
  $P \leftarrow P \cup \{(\text{pre}(u), u)\}$

  für alle $e = (u, v) \in E$ **tue**
    wenn $\text{dist}(v) > \text{dist}(u) + w(e)$ **dann**
      $\text{dist}(v) \leftarrow \text{dist}(u) + w(e)$
      $\text{pre}(v) \leftarrow u$

Dijkstra's Algorithmus

Eingabe : Graph $G = (V, E)$ mit Gewichten $w : E \to \mathbb{R}^+$, Startknoten $s \in V$
Seiteneffekte : Kürzeste Wege und Distanzen zu allen Knoten $v \in V$

$S \leftarrow V$
$P \leftarrow \emptyset$
$\mathrm{pre}(u) \leftarrow \perp$ für alle $u \in V$
$\mathrm{dist}(u) \leftarrow \infty$ für alle $u \in V$

$\mathrm{dist}(s) \leftarrow 0$

solange $S \neq \emptyset$ **tue**
    $u \leftarrow \mathrm{argmin}_{u \in S}\{\mathrm{dist}(u)\}$
    $S \leftarrow S \setminus \{u\}$
    $P \leftarrow P \cup \{(\mathrm{pre}(u), u)\}$

    für alle $e = (u, v) \in E$ **tue**
        wenn $\mathrm{dist}(v) > \mathrm{dist}(u) + w(e)$ **dann**
            $\mathrm{dist}(v) \leftarrow \mathrm{dist}(u) + w(e)$
            $\mathrm{pre}(v) \leftarrow u$

Aufgabe 3

Matroide und kürzeste Wege

Gegeben: Gerichteter Graph $G = (V, E)$ mit Gewichten $w : E \rightarrow \mathbb{R}^+$.
Kürzeste s-t-Wege für $s, t \in V$ sind eindeutig.

Ziel: Beweis der Korrektheit von Dijkstra's Algorithmus durch Matroid-Struktur.

Aufgabe 3

Matroide und kürzeste Wege

Gegeben: Gerichteter Graph $G = (V, E)$ mit Gewichten $w : E \rightarrow \mathbb{R}^+$.
Kürzeste s-t-Wege für $s, t \in V$ sind eindeutig.

Ziel: Beweis der Korrektheit von Dijkstra's Algorithmus durch Matroid-Struktur.

(a) Begründen Sie, warum Dijkstra's Algorithmus ein Greedy-Verfahren ist.

Aufgabe 3

Matroide und kürzeste Wege

Gegeben: Gerichteter Graph $G = (V, E)$ mit Gewichten $w : E \rightarrow \mathbb{R}^+$.
Kürzeste s-t-Wege für $s, t \in V$ sind eindeutig.

Ziel: Beweis der Korrektheit von Dijkstra's Algorithmus durch Matroid-Struktur.

(a) Begründen Sie, warum Dijkstra's Algorithmus ein Greedy-Verfahren ist.

$S \leftarrow V$
...
$\text{dist}(s) \leftarrow 0$
solange $S \neq \emptyset$ **tue**
  $u \leftarrow \text{argmin}_{u \in S}\{\text{dist}(u)\}$
  $S \leftarrow S \setminus \{u\}$
  ...

Aufgabe 3

Matroide und kürzeste Wege

Gegeben: Gerichteter Graph $G = (V, E)$ mit Gewichten $w : E \to \mathbb{R}^+$.
Kürzeste s-t-Wege für $s, t \in V$ sind eindeutig.

Ziel: Beweis der Korrektheit von Dijkstra's Algorithmus durch Matroid-Struktur.

(a) Begründen Sie, warum Dijkstra's Algorithmus ein Greedy-Verfahren ist.

(b) Definieren Sie ein geeignetes unabhängiges Mengensystem $(M, \mathcal{U})$.
Dabei sei M die Menge aller azyklischen Pfade in G, die bei s anfangen.

Aufgabe 3

Matroide und kürzeste Wege

Gegeben: Gerichteter Graph $G = (V, E)$ mit Gewichten $w : E \to \mathbb{R}^+$.
Kürzeste s-t-Wege für $s, t \in V$ sind eindeutig.

Ziel: Beweis der Korrektheit von Dijkstra's Algorithmus durch Matroid-Struktur.

(a) Begründen Sie, warum Dijkstra's Algorithmus ein Greedy-Verfahren ist.

(b) Definieren Sie ein geeignetes unabhängiges Mengensystem $(M, \mathcal{U})$.
Dabei sei M die Menge aller azyklischen Pfade in G, die bei s anfangen.

(c) Zeigen Sie, dass $(M, \mathcal{U})$ ein Matroid ist.

Aufgabe 3

Matroide und kürzeste Wege

Gegeben: Gerichteter Graph $G = (V, E)$ mit Gewichten $w : E \to \mathbb{R}^+$.
Kürzeste s-t-Wege für $s, t \in V$ sind eindeutig.

Ziel: Beweis der Korrektheit von Dijkstra's Algorithmus durch Matroid-Struktur.

(a) Begründen Sie, warum Dijkstra's Algorithmus ein Greedy-Verfahren ist.

(b) Definieren Sie ein geeignetes unabhängiges Mengensystem $(M, \mathcal{U})$.
Dabei sei M die Menge aller azyklischen Pfade in G, die bei s anfangen.

(c) Zeigen Sie, dass $(M, \mathcal{U})$ ein Matroid ist.

(d) Geben Sie eine Kostenfunktion c für die Elemente $I \in M$ an.

Aufgabe 3

Matroide und kürzeste Wege

Gegeben: Gerichteter Graph $G = (V, E)$ mit Gewichten $w : E \to \mathbb{R}^+$.
Kürzeste s-t-Wege für $s, t \in V$ sind eindeutig.

Ziel: Beweis der Korrektheit von Dijkstra's Algorithmus durch Matroid-Struktur.

(a) Begründen Sie, warum Dijkstra's Algorithmus ein Greedy-Verfahren ist.

(b) Definieren Sie ein geeignetes unabhängiges Mengensystem $(M, \mathcal{U})$.
Dabei sei M die Menge aller azyklischen Pfade in G, die bei s anfangen.

(c) Zeigen Sie, dass $(M, \mathcal{U})$ ein Matroid ist.

(d) Geben Sie eine Kostenfunktion c für die Elemente $I \in M$ an.

(e) Zeigen Sie, dass die Greedy-Methode über $(M, \mathcal{U})$ mit Dijkstra's Algorithmus übereinstimmt.