Dijkstras Algorithmus

Definition

Ein *gewichteter Graph* ist ein Tripel $G = (V, E, f)$ mit:
(V, E) ist ein Graph und f eine *Gewichtsfunktion* $f : E \to \mathbb{R}_{\geq 0}$.

Definition

Es sei $G = (V, E, f)$ ein gewichteter Graph.

- Es sei $z = (v_0, \ldots, v_l)$ ein Kantenzug in G.
  $f(z) := \sum_{i=1}^{l} f(v_{i-1}v_i)$ heißt das *Gewicht* von z.
- Für alle $v, w \in V$ mit $v \sim w$ definieren wir die *Distanz* zwischen v und w als

$$d(v, w) := \min\{f(z) \mid z \text{ ist } v\text{-}w\text{-Pfad in } G\} \in \mathbb{R}_{\geq 0}.$$

- Für alle $v, w \in V$ mit $v \not\sim w$ wird $d(v, w) := \infty$ gesetzt.

Dijkstras Algorithmus (Forts.)

```
Dijkstra(Γ, w, f)
initialisiere array d[1, ..., n] mit allen Einträgen gleich ∞
initialisiere array p[1, ..., n] mit allen Einträgen gleich NIL
initialisiere priority queue Q mit Elementen 1, ..., n und
allen Prioritäten = ∞
d[w] ← 0
Insert(Q, w, d[w])
while Q nicht leer
do v ← ExtractMin(Q)
    for u ∈ Γ[v]
    do if d[v] + f(uv) < d[u]
          then d[u] ← d[v] + f(uv)
               p[u] ← v
               Insert(Q, u, d[u])
return d, p
```

Dijkstras Algorithmus (Forts.)

Kommentare (zum Algorithmus)

- Eingabe:
  - Γ: Adjazenzliste des Graphen $G = (V, E)$ mit $V = \underline{n}$
  - w: Knoten $w \in V$
  - f: Liste der Werte $f(e), e \in E$
- Der `array` $d[1, \ldots, n]$ enthält nach der Terminierung an Position v den Wert $d(w, v)$.
- Der `array` $p[1, \ldots, n]$ enthält nach der Terminierung an Position v einen Knoten u, der auf einem w-v-Pfad der Distanz $d(w, v)$ unmittelbar vor v kommt.

Dijkstras Algorithmus (Forts.)

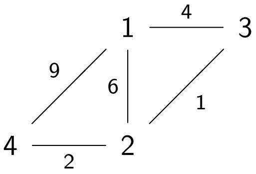
Kommentare (zum Algorithmus), Forts.

- `priority queue` ist eine *Vorrangwarteschlange*, bei der jedem ihrer Element ein *Prioritätswert* zugeordnet ist.
- Der Aufruf $\textsc{Insert}(Q, x, k)$ fügt das Element x in die Warteschlange ein und ordnet x die Priorität $k \geq 0$ zu.

  Falls x bereits in der Warteschlange enthalten ist, wird nur die Priorität neu auf k gesetzt.
- Der Aufruf $\textsc{ExtractMin}(Q)$ entnimmt das Element mit der niedrigsten Priorität.

Dijkstras Algorithmus (Forts.)

Beispiel



d	p	Q	v	$\Gamma(v)$	$d[v] + f(uv) < d[u]$
$[0, \infty, \infty, \infty]$	$[-, -, -, -]$	$\{1, 2, 3, 4\}$	1	$[2, 3, 4]$	$[2, 3, 4]$
$[0, 6, 4, 9]$	$[-, 1, 1, 1]$	$\{2, 3, 4\}$	3	$[1, 2]$	$[2]$
$[0, 5, 4, 9]$	$[-, 3, 1, 1]$	$\{2, 4\}$	2	$[1, 3, 4]$	$[4]$
$[0, 5, 4, 8]$	$[-, 3, 1, 2]$	$\{4\}$	4	$[1, 2]$	$[]$
$[0, 5, 4, 8]$	$[-, 3, 1, 2]$	$\{\}$			

Bäume und Wälder

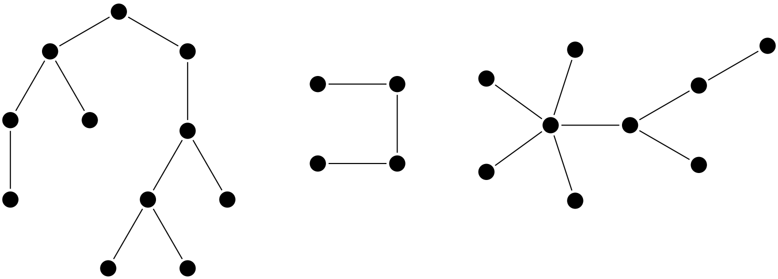
Es sei $G = (V, E)$ ein Graph mit $n_G > 0$.

Definition

- G heißt *kreisfrei* bzw. *Wald*, falls G keine Kreise enthält.
- Ein zusammenhängender Wald heißt *Baum*.
- Die Knoten eines Waldes mit Grad ≤ 1 heißen Blätter.

Wälder und Bäume (Forts.)

Beispiel



Wälder und Bäume (Forts.)

Es sei $G = (V, E)$ ein Graph mit $n_G > 0$.

Bemerkung

- G ist genau dann kreisfrei, wenn jede Kante eine Brücke ist.
- Ist G ein Baum mit $n_G \geq 2$, dann hat G mindestens zwei Blätter.
- Ist G ein Baum mit $n_G \geq 3$, dann hat G höchstens $n_G - 1$ Blätter.

Wälder und Bäume (Forts.)

Es sei $G = (V, E)$ ein Graph mit $n_G > 0$.

Erinnerung

- r_G: Anzahl der Zusammenhangskomponenten von G
- Es ist $r_G \geq n_G - m_G$.
- Es sei $e \in E$ und $G' := (V, E \setminus \{e\})$. Dann ist $r_{G'} \leq r_G + 1$. Weiter ist $r_{G'} = r_G + 1$ genau dann, wenn e eine Brücke ist.

Satz

Es gilt $r_G = n_G - m_G$ genau dann, wenn G kreisfrei ist.

Wälder und Bäume (Forts.)

Es sei $G = (V, E)$ ein Graph mit $n_G > 0$.

Folgerung

G ist genau dann ein Baum, wenn mindestens zwei der folgenden Bedingungen erfüllt sind.

- G ist kreisfrei.
- G ist zusammenhängend.
- $m_G = n_G - 1$.

Wälder und Bäume (Forts.)

Es sei $G = (V, E)$ ein Graph mit $n_G > 0$.

Erinnerung

- Ist G zusammenhängend, dann ist $m_G \geq n_G - 1$.
- Ist G kreisfrei, dann ist $m_G = n_G - r_G \leq n_G - 1$.

Bemerkung

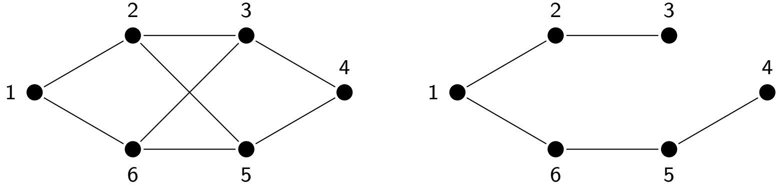
- Ein Baum ist ein zusammenhängender Graph mit minimal möglicher Kantenzahl.
- Ein Baum ist ein kreisfreier Graph mit maximal möglicher Kantenzahl.

Spannbäume

Es sei $G = (V, E)$ ein Graph mit $n_G > 0$.

Definition
Ein Teilgraph $G' = (V', E')$ von G heißt *Spannbaum* von G (engl. *spanning tree*), wenn G' ein Baum ist und $V' = V$.

Beispiel



Spannbäume (Forts.)

Satz

Jeder zusammenhängende Graph hat einen Spannbaum.

Beweis

Breitensuche.

Spannbäume (Forts.)

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

Algorithmus (Sukzessives Entfernen von Kanten)

- Initialisiere $B := E$.
- Entferne sukzessive solche Kanten aus B, die keine Brücken in (V, B) sind.
- Ist das nicht mehr möglich, dann ist (V, B) ein Spannbaum von G.

Spannbäume (Forts.)

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

Algorithmus (Sukzessives Hinzufügen von Kanten)

- Initialisiere $B := \emptyset$.
- Füge sukzessive solche Kanten zu B hinzu, deren Endknoten in verschiedenen Zusammenhangskomponenten von (V, B) liegen.
- Ist das nicht mehr möglich, dann ist (V, B) ein Spannbaum.

Spannbäume (Forts.)

Es sei $G = (V, E, f)$ ein gewichteter Graph.

Erinnerung

- (V, E) ist ein Graph, und $f : E \to \mathbb{R}_{\geq 0}$ eine Gewichtsfunktion.
- Für $T \subseteq E$ heißt $f(T) := \sum_{e \in T} f(e)$ das Gewicht von T.

Definition

Ein *minimaler Spannbaum* von G ist ein Spannbaum (V, B) von G mit minimalem Gewicht $f(B)$ unter allen Spannbäumen von G.

Spannbäume (Forts.)

Es sei $G = (V, E, f)$ ein gewichteter Graph.

Algorithmus (Kruskal)

- Initialisiere $B := \emptyset$.
- Füge sukzessive solche Kanten zu B hinzu, deren Endknoten in verschiedenen Zusammenhangskomponenten von (V, B) liegen, und unter allen solchen jeweils einen von minimalem Gewicht.
- Ist das nicht mehr möglich, dann ist (V, B) ein minimaler Spannbaum.

Spannbäume (Forts.)

Austauschlemma

Es seien (V, A) und (V, B) zwei Bäume mit derselben Knotenmenge V.

Für jedes $a \in A \setminus B$ gibt es ein $b \in B \setminus A$ so, dass $(V, B \cup \{a\} \setminus \{b\})$ auch ein Baum ist.