Breitensuche BFS (Breadth First Search)

Algorithmus BREITENSUCHE

EINGABE: $G = (V, E)$ als Adjazenzliste, Startknoten $s \in V$

1. Für alle $v \in V$
   1. If $(v = s)$ then $d[v] \leftarrow 0$ else $d[v] \leftarrow \infty$;
   2. pred$[v] \leftarrow$ nil;
2. $Q \leftarrow$ new Queue; Q.Enqueue(s);
3. While (Q.Isempty() $\neq$ TRUE)
   1. $v \leftarrow Q$.Dequeue(Q);
   2. Für alle $u \in \Gamma(v)$
      1. If $d[u] = \infty$ then $d[u] \leftarrow d[v] + 1$; pred$[u] \leftarrow$ v; Q.Enqueue(u);

AUSGABE: Arrays $d[v]$, pred$[v]$ für alle $v \in V$

Kürzeste u-v Pfade

Satz Kürzeste u-v Pfade

BREITENSUCHE berechnet bei Eingabe $G = (V, E)$, v für jeden Knoten $u \in V$ einen kürzesten u-v Pfad $p = (u, \text{pred}[u], \text{pred}[\text{pred}[u]], \ldots, v)$ in Zeit $\mathcal{O}(|V| + |E|)$.

Beweis: Korrektheit

- p ist ein Pfad, denn kein Knoten wird zweimal in Q eingefügt.
- Wegen $d[\text{pred}[u]] = d[u] - 1, \ldots, d[v] = 0$ besitzt p Länge $d[u]$.
- **Annahme:** $\exists$ Pfad $p' = (u = v_0, \ldots, v = v_k)$ mit Länge $k < d[u]$.
- Für jede Kante $\{u', v'\} \in E$ gilt $d[u'] \leq d[v'] + 1$. Damit folgt

$$d[u] \leq d[v_1] + 1 \leq \ldots \leq d[v_k] + k = k.$$

(Widerspruch zu $k < d[u]$.)

Laufzeit Breitensuche

Beweis: Laufzeit

- Schritt 1: $\mathcal{O}(|V|)$, Schritt 2: $\mathcal{O}(1)$.
- Schritt 3: Sei s in der ZHK $V' \subseteq V$.
- Für alle $v \in V'$ werden alle Nachbarn $u \in \Gamma(v)$ besucht, d.h.

$$\sum_{v \in V'} |\Gamma(v)| \leq \sum_{v \in V} \deg(v) = 2|E|.$$

  Damit ist die Laufzeit von Schritt 3 $\mathcal{O}(|E|)$.

Spannbaum mit kürzesten u-v Pfaden

Satz Spannbaum mit kürzesten u-v Pfaden

BREITENSUCHE berechnet bei Eingabe eines zusammenhängenden $G = (V, E)$ und $v \in V$ in Zeit $\mathcal{O}(|V| + |E|)$ einen Spannbaum $T = (V, E')$, $E' = \{\{u, \text{pred}(u)\} \mid u \in V \setminus \{v\}\}$ mit kürzesten u-v Pfaden.

Beweis:

- Kürzeste u-v Pfade in G sind von der Form $(u, \text{pred}[u], \ldots, v)$.
- Wegen $E' = \{\{u, \text{pred}(u)\} \mid u \in V \setminus \{v\}\}$ gilt $|E'| = n - 1$.
- Zeigen nun, dass G zusammenhängend ist. Damit folgt zusammen mit $|E'| = n - 1$, dass T ein Baum ist.
- Für alle $u, w \in V$ sind $p_u = (u, \text{pred}[u], \ldots, v)$ und $p_w = (w, \text{pred}[w], \ldots, v)$ ein u-v bzw. ein w-v Pfad.

Fall 1: p_u, p_w sind bis auf v knotendisjunkt. Dann ist $p = (u, \text{pred}[u], \ldots, v_k, \ldots, \text{pred}[w], w)$ ein u-w Pfad.

Fall 2: p_u, p_w sind nicht knotendisjunkt. Dann ist p aus Fall 1 ein u-w Weg. Dieser kann zu einem u-w Pfad verkürzt werden.

Zusammenhangskomponenten

Algorithmus VOLLSTÄNDIGE BREITENSUCHE

EINGABE: $G = (V, E)$

1. Setze $i \leftarrow 1$.
2. While $V \neq \emptyset$
   1. Starte BREITENSUCHE in beliebigem Startknoten $s \in V$.
   2. $V_i \leftarrow \{v \in V \mid d[v] < \infty\}$.
   3. $V \leftarrow V \setminus V_i$; $G \leftarrow G[V]$; $i \leftarrow i + 1$;

AUSGABE: ZHKs $G[V_1], \ldots, G[V_k]$

Laufzeit für ZHKs

Satz Berechnung von ZHKs

VOLLSTÄNDIGE BREITENSUCHE berechnet bei Eingabe $G = (V, E)$ die ZHKs von G in Zeit $\mathcal{O}(|V| + |E|)$.

Beweis:

- **Laufzeit:** Jeder Knoten wird genau einmal in Q eingefügt. Laufzeit: $\mathcal{O}(|V|)$.
- Jede Kante $\{u, v\}$ wird einmal bei den Nachbarn von u und einmal bei den Nachbarn von v betrachtet. Laufzeit: $\mathcal{O}(|E|)$.
- **Korrektheit:** Folgt aus den Sätzen zuvor.

Tiefensuche DFS (Depth First Search)

Algorithmus TIEFENSUCHE

EINGABE: $G = (V, E)$ als Adjazenzliste, Startknoten $s \in V$

1. Für alle $v \in V$
   1. pred[v] $\leftarrow$ nil;
2. $S \leftarrow$ new Stack; S.Push(s);
3. While (S.Isempty() $\neq$ TRUE)
   1. $v \leftarrow S$.Pop();
   2. If ($\exists u \in \Gamma(v) \setminus \{s\}$ mit $\mathrm{pred}[u] =$nil) then
      1. S.Push(v); S.Push(u);
      2. $\mathrm{pred}[u] \leftarrow v$;

AUSGABE: $\mathrm{pred}[v]$ für alle $v \in V$

Rekursive Version von DFS

Algorithmus REKURSIVE TIEFENSUCHE

EINGABE: $G = (V, E)$ als Adjazenzliste, Startknoten $s \in V$

1. Für alle $v \in V$
   1. pred[v] $\leftarrow$ nil;
2. DFS-rekursiv(s);

Funktion DFS-rekursiv(v)

1. While ($\exists u \in \Gamma(v) \setminus \{s\}$ mit $\mathrm{pred}[u] =$nil)
   1. $\mathrm{pred}[u] \leftarrow v$;
   2. DFS-rekursiv(u);

AUSGABE: $\mathrm{pred}[v]$ für alle $v \in V$

Anmerkung:

- Die rekursiven Aufrufe simulieren den Stack S.

Berechnung eines Spannbaums

Satz Berechnung eines Spannbaums

TIEFENSUCHE berechnet bei Eingabe eines zusammenhängenden $G = (V, E)$ und $v \in V$ in Zeit $\mathcal{O}(|E|)$ einen Spannbaum $T = (V, E')$ mit $E' = \{\{u, \text{pred}[u]\} \mid u \in V \setminus \{v\}\}$.

Beweis:

- **Korrektheit:** Es gilt $|E'| = n - 1$.
- Falls T zusammenhängend ist, so ist T ein Baum.
- Für jeden Knoten $u \in V \setminus \{v\}$ ist $(u, \text{pred}[u], \text{pred}[\text{pred}[u]], \ldots, v)$ ein u-v Pfad.
- Konstruktion von u-w Pfaden für alle $u, w \in V$ analog zu BFS.
- **Laufzeit:** Analog zur Analyse bei BFS.

Hamiltonscher Kreis

Definition Hamiltonscher Kreis

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

1. Ein *Hamiltonscher Kreis* in G ist ein Kreis, der alle Knoten enthält.
2. Graphen mit Hamiltonkreis bezeichnet man als *hamiltonsch*.

Bsp:

- 5 Personen setzen sich an einen runden Tisch.
- Frage: Gibt es zwei Konstellationen, bei denen jeder andere Nachbarn besitzt?
- Modellierung als Graphproblem: u, v sind Nachbarn, falls $\{u, v\}$ Kante eines Hamiltonkreises ist.
- D.h. wir suchen zwei Konstellationen mit kantendisjunkten Hamiltonkreisen.
- Der K_5 besitzt zwei kantendisjunkte Hamiltonkreise.