Algorithmische
Graphentheorie

12.Vorlesung
12.05.2015
Anmerkung

Falls nicht alle Gewichte unterschiedlich sind, wähle die lexicographische Ordnung der Kanten als Tie-breaker.

\[ E = (a, b) < E' = (a', b') \implies \]

- \[ w(e) < w(e') \]

oder \[ w(e) = w(e') \] und \[ (a, b) <_{\text{lex}} (a', b') \]

Ablauf MST-Verifikation:

1.) Konsumiere \( T_B \) von \( T \)
   - \( \tau \) läuft in \( O(|V|) \) durch, zu jeder Zeit habe ich einen Baum bei Ausführung Borůvka
   - \( \Rightarrow \) Aufwand = \( T(n) \leq T(n/2) + O(n) \)
   - \( T(n) = O(n) \)

2.) Für jede Kante \( e \in \text{typus } E \) teste, ob schwere Kante in \( x \leftrightarrow y \in T_B \) leichter ist (oder gleich) \( w(e) \)
Finden von schwersten Kanten in Bäumen

Eingabe: Baum $T_i$, Anfragen $(v_1,v_1) \ (v_2,v_2) \ ... \ (v_m,v_m)$, wobei $v_i,v_i$ Blätter in $T_i$

Frage: Gib für jedes $1 \leq i \leq m$ die schwere Kante auf $v_i \rightarrow v_i$ Pfad aus

Anwendung: Dies kann für den 2) Schritt in Verifikationsalg. benutzt werden

Notation: Tiefe eines Knotens = Abstand zur Wurzel

$\textbf{LCA}(u,v) = \text{Tiefe gleichkamerter Vorfahr von } u \text{ und } v$

$\textbf{LA}(u,i) = \text{Vorfahr von } u \text{ auf Ebene } i$

$p(u) = \text{Vater von } u$
- Originalfragen: \((u_i, v_i)\)

  kann übersetzt werden in

  \[
  (u_i, \text{LCA}(u_i, v_i)) \quad + \quad 1 \times \text{Vergleich}
  \]

  \[
  8 \times (v_i, \text{LCA}(u_i, v_i))
  \]

  von nun an nur solche Anfragen!

- für jedes Blatt \(u\) muss ich also eine Reihe von Anfragen
  \((u, v_1), (u, v_2), (u, v_3), \ldots, (u, v_k)\)
  beantragen

- ObdA. \(d[u, v_1] < d[u, v_2] < d[u, v_3] \ldots < d[u, v_k]\)

- Wir speichern als \(Q(m) = \langle d[u, v_1], d[u, v_2], \ldots, d[u, v_k] \rangle\)

- Für jedes \(Q(v)\) möchten wir eine Sequenz \(A(v) = \langle a_1, a_2, \ldots, a_k \rangle\) berechnen mit:

  \[\text{LA}(u, a_i)\text{ ist der untere Knoten}
  \]

  der Schwestern Kante auf

  Pfad \(u \to \text{LA}(u, d_i)\)
Beobachtung: $A(\cdot)$ ist schwach steigend.

Zur Arbeit mit den Sequenzen $A(\cdot)$ und $Q(\cdot)$ benötige ich folgende Operationen:

- $\text{unite}(A, B)$, vereinigt Sequenzen $A, B$ (sortiert)
  zu einer sortierten Sequenz ohne Duplicates
  z.B. $\text{unite}((1,2,3), (2,4,7)) = (1, 2, 3, 4)$

- $\text{remove}(A, a)$, entfernt $a$ aus Sequenz $A$
  z.B. $\text{remove}((1,2,3,4), 2) = (1,3,4)$

- $\text{add}(A, a)$, fügt $a$ zu $A$ hinzu
  Ergebnis bleibt sortiert, ohne Duplicates
  z.B. $\text{add}((1,3,4), 6) = (1,3,4,6)$
\text{extract}(A, i) \) liefert das \( i \)-te Element von \( A \)
\[ \text{2.} \text{B. } \text{extract}(\langle 1,3,4,6 \rangle, 2) = 3 \]

\( \text{replace-suffix}(A, a, j) \) liefert Sequenz \( A' \), wobei die \( a \)-ten Einträge \( (a; b) \) durch \( a \) ersetzt werden
\[ \text{2.} \text{B. } \text{replace-suffix}(\langle 1,3,4,6 \rangle, 3, 8) = \langle 1,3,8,8 \rangle \]

\( \text{Subseq}(A, B_1, B_2) \): finde die Positionen der Einträge aus \( B_1 \) die auch in \( B_2 \) auftauchen
\rightarrow übernehme diese Stellen von \( A \)
\[ \text{2.} \text{B. } \text{Subseq}(\langle 1,3,8,8 \rangle, \langle 2,4,6,16 \rangle, \langle 2,3,6 \rangle) = \langle 1,8,8 \rangle \]
Algorithmus

1. \( r \leftarrow \text{Wurzel von } T \)
   \( A(v,v) \text{ minier. Knoten} : Q(v) = \langle \rangle \)
   for alle \( v \in V \) (bottom-up)
   \[ L \quad Q(p(v)) = \text{unite}(Q(p(v)), \text{remove}(Q(v), d[v]-1)) \]

2. \( A(r) = \langle \rangle \)
   foreach \( v \neq r \) (top-down)
   \[ A(v) = \text{subseq}(A(p(v)), Q(p(v)), Q(v)) \]
   \[ j \leftarrow \text{binary-search}(v) \]
   \[ \text{replace-suffix}(A(v), d[v], j) \]
   if \( d[v]-1 \in Q(v) \) then \( A(v) = \text{append}(A(v), d[v]) \)

RETURN A

Teil 1 Initialisierung, d.h. Sehe alle Sequenzen \( Q(\cdot) \)

Teil 2 Berechne Top-Down die Sequenzen \( A(\cdot) \)

wobei

\( \text{binary-search}(v) \) liefert die Index \( j \)

für aktuelles \( Q(\cdot) \), sodass

\[ w(m_{j-1}, P(m_{j-1})) = w(v, p(v)) = w(m_j, P(m_j)) \]

für \( w(m_0, \ldots) = +\infty \) \& \( w(m_{k+1}, \ldots) = -\infty \)
Erklärung

Teil (a)

Die "interessanten" Knoten von Knoten u sind die interessanten Knoten der Kinder - Kinderer Bsp

Bestimme Ringen Q bottom-up

Teil (b)

1.) Übernehmen die Stellen aus \( A(p(u)) \), die für \( A(v) \) relevant sind
   (d.h. die zu \( Q(v) \) zugehören sind)

2.B. \( Q(p(u)) = \{0, 2, 3, 5\} \)

\( Q(v) = \{0, 2, 6\} \)

\( A(p(u)) = \{2, 4, 4, 6\} \)

\[ \Rightarrow \]

relevant: \( \langle 2, 4 \rangle \)

Für \( A(v) \)

Aber bisher ist Kante \( \langle u, p(v) \rangle \)
noch nicht berücksichtigt worden.
- die "neue" Kante \((u, p(u))\) kann die schwersten Kanten, d.h. \(A(v)\) verändern.

\[\text{Bsp}\]

\[\text{D.h. schwerste Kanten waren vorher von Gewicht } \underline{4,4,2} \text{ jetzt } \underline{4,4,3,3}\]

\[\text{sortiert}\]

\[\text{Also: finde Position, ab welcher Gewicht der neuen Kante größer als das alle Max. Gewicht ist}\]

\[\text{(- Binäre Suche da } \langle w(u), w(v) \rangle \text{ relevant)}\]

\[\text{Ab dieser Stelle ist das neue Maximalgewicht durch die neue Kante } (u, p(u)) \text{ gegeben!}\]

\[\text{- evtl. füge neue Anfrage ein wenn } |\partial^+(p(u))| \geq |\partial(v)|\]