



Algorithm Engineering Übung

Blatt 5, Aufgabe 1: 2-Zusammenhang

Christine Zarges, Thorsten Kerkhof

Universität Dortmund

19. Januar 2006

Literatur: R.E. Tarjan. Depth first search and linear graph algorithms, *SIAM Journal on Computing*, Vol. **1(2)** 146-160, 1972



Überblick

1. Einleitung
2. Algorithmus BICON
3. Laufzeit und Korrektheit



Kapitel 1

Einleitung

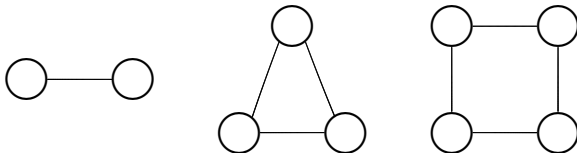
- 1.1 Definition: Schnittknoten und Zweizusammenhangskomponenten
- 1.2 Definition: Palm Tree
- 1.3 Definition: $\text{lowpt}(v)$



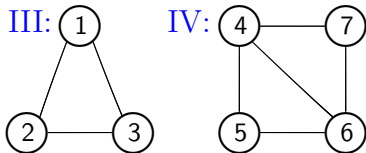
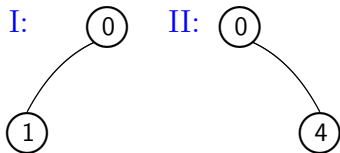
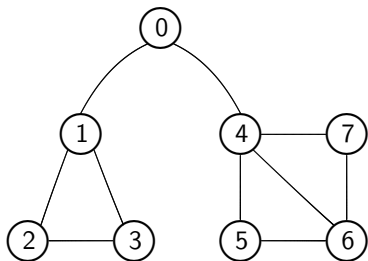
1.1 Definitionen

1. a ist **Schnittknoten** \Rightarrow nach Entfernen von a ist der Restgraph nicht mehr zusammenhängend.
2. Eine **Zweizusammenhangskomponente** enthält keinen Schnittknoten.

Beispiele:



Beispiel

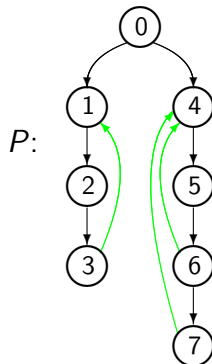
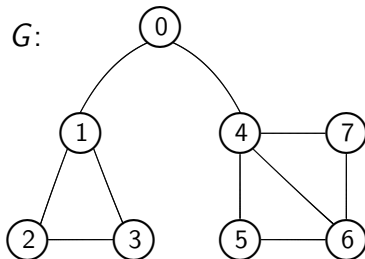


Graph G
(Schnittknoten: 0, 1, 4)

2-Zusammenhangskomponenten
von G

1.2 Definition: Palm Tree

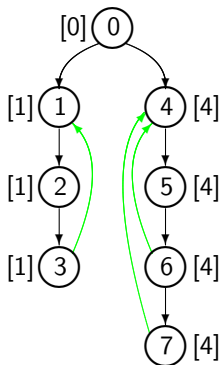
- ▶ Palm Tree von G : Gerichteter Graph P , der durch DFS-Suchlauf entsteht.
- ▶ Disjunkte Kantenmenge $v \rightarrow w$ und $v \hookrightarrow w$.
- ▶ $v \rightarrow w$ Kanten: Spannbaum T von P .
- ▶ $v \hookrightarrow w$ Kanten: Back-Kanten.





1.3 Definition: $\text{lowpt}(v)$

$$\text{lowpt}(v) = \min \left\{ \text{num}(v) \cup \{ \text{num}(w) \mid v \xrightarrow{*} \hookrightarrow w \} \right\}$$





Kapitel 2

Algorithmus BICON

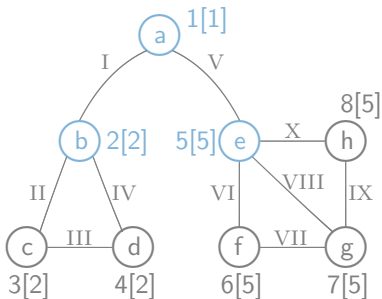


Algorithmus BICON

Eingabe: $G = (V, E)$, ungerichtet

Output: ZZKen von G

Beispiel:



Initialisierung:

- ▶ $i = 0$
- ▶ Stack $S = \emptyset$
- ▶ $\forall x \in V : \text{num}(x) := 0$

Main:

1. $\forall x \in V$ mit $\text{num}(x) = 0$: BICON(x, x)

BICON(v, u):

1. $i := i + 1$; $\text{num}(v) := i$; $\text{lowpt}(v) := i$
2. Für $w \in \text{Adj}(v)$ mit $w \neq u$:
3. Falls $\text{num}(w) = 0$:
 - a) $(v, w) \rightarrow S$
 - b) BICON(w, v)
 - c) $\text{lowpt}(v) := \min\{\text{lowpt}(v), \text{lowpt}(w)\}$
 - d) Falls $\text{lowpt}(w) \geq \text{num}(v)$: Entferne Kanten vom Stack bis einschließlich (v, w) und gebe Sie als ZZK aus.

Sonst: Falls $\text{num}(w) < \text{num}(v)$:

- a) $(v, w) \rightarrow S$
- b) $\text{lowpt}(v) := \min\{\text{lowpt}(v), \text{num}(w)\}$



Kapitel 3

Laufzeit und Korrektheit

- 3.1 Lineare Laufzeit
- 3.2 Korrektheitsbeweis



3.1 Lineare Laufzeit

Theorem: Der Algorithmus BICON benötigt Rechenzeit $O(|V|, |E|)$.

Beweis:

- ▶ BICON ist DFS und zusätzlich
 1. lowpt-Werte berechnen und
 2. Hinzufügen und Entfernen von Kanten zum bzw. vom Stack (beides genau einmal pro Kante).

⇒ lineare Laufzeit





Korrektheit

Theorem: Der Algorithmus BICON berechnet die Zweizusammenhangskomponenten eines Graphen G korrekt.

Beweis:

- ▶ Annahme: G ist zusammenhängend.
- ▶ Vollständige Induktion über Anzahl der Kanten $|E|$:
 - ▶ $|E| = 0$: G ist entweder leer oder besteht aus einem einzelnen Knoten. ✓



Korrektheit

Theorem: Der Algorithmus BICON berechnet die Zweizusammenhangskomponenten eines Graphen G korrekt.

Beweis:

- ▶ $|E| \rightarrow |E| + 1$: Jede Kante im Stack wird genau einmal vom Stack entfernt und einer Komponente hinzugefügt.
 1. Die Komponente enthält nicht alle Kanten
 - $\stackrel{\text{I.V.}}{\Rightarrow}$ Algo arbeitet korrekt für diese Komponente
 - ▶ Jede weitere Komponente kann nicht mehr alle Knoten enthalten
 - $\stackrel{\text{I.V.}}{\Rightarrow}$ Algo ist insgesamt korrekt. ✓



Korrektheit

Theorem: Der Algorithmus BICON berechnet die Zweizusammenhangskomponenten eines Graphen G korrekt.

Beweis:

2. Es wird nur eine Komponente gefunden.

▶ Annahme: Komponente ist nicht 2-zusammenhängend.

⇒ ∃ Schnittknoten a

Lemma

⇒ $\text{lowpt}(v) \geq \text{num}(a)$ für einen Sohn von a

⇒ Algo würde beim Untersuchen dieser Kante (Schritt 3.d)) eine neue ZZK erstellen. ⚡ □