

Berechnung der 3-Zusammenhangskomponenten in Linearzeit

Carsten Gutwenger



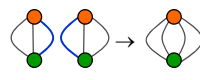
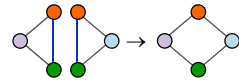
Vorlesung
Graphenalgorithmen
WS 09/10
20. Januar 2010

tu technische universität dortmund
fi department of computer science

Split-Komponenten

Theorem. Man erhält die 3-Zusammenhangskomponenten aus den Split-Komponenten, indem man

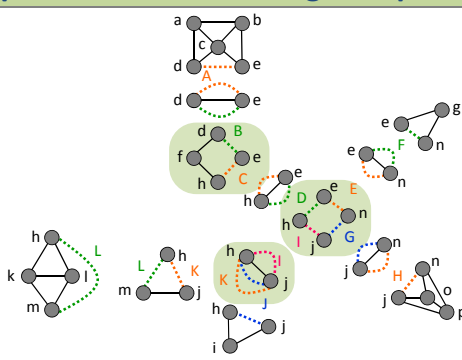
- Dreiecke zu maximalen Polygonen, und
- 3-er Bündel zu maximalen Bündeln verschmelzt.

Merge-Operation

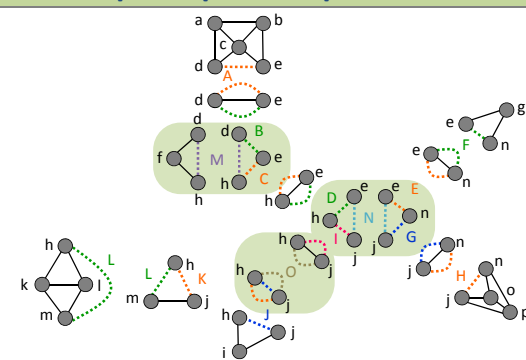
Graphenalgorithmen • Berechnung der 3-Zusammenhangskomponenten in Linearzeit • 2

Beispiel: 3-Zusammenhangskomponenten



Graphenalgorithmen • Berechnung der 3-Zusammenhangskomponenten in Linearzeit • 3

Beispiel: Split-Komponenten



Graphenalgorithmen • Berechnung der 3-Zusammenhangskomponenten in Linearzeit • 4

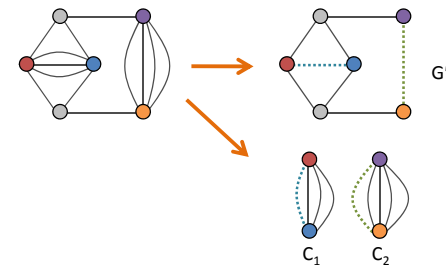
Der Algorithmus

1. Ersetze Multi-Kanten durch virtuelle Kanten
→ Graph G' und Bündel C_1, \dots, C_p
2. Bestimme Split-Komponenten C_{p+1}, \dots, C_m von G'
3. Verschmelze in C_1, \dots, C_m (solange wie möglich) je zwei Polygone bzw. Bündel, die die gleiche virtuelle Kante enthalten.
→ 3-Zusammenhangskomponenten von G

Graphenalgorithmen • Berechnung der 3-Zusammenhangskomponenten in Linearzeit • 5

Ersetzen von Multi-Kanten

zu 1.:



Graphenalgorithmen • Berechnung der 3-Zusammenhangskomponenten in Linearzeit • 6

Maximale Bonds und Polygone

zu 3.:

```

for i := 1,...,m do
  if  $C_i \neq \emptyset$  and type( $C_i$ ) ist Bündel oder Polygon then
    for all  $e = (u,v,\ell) \in C_i$  do
      if ex.  $j \neq i$  mit  $e \in C_j$  und type( $C_i$ ) = type( $C_j$ ) then
         $C_i := (C_i \cup C_j) \setminus \{e\}$ 
         $C_j := \emptyset$ 
  
```

Markiere C_i als gelöscht

Verschmelze C_i und C_j

Graphenalgorithmen • Berechnung der 3-Zusammenhangskomponenten in Linearzeit • 7

Segmente

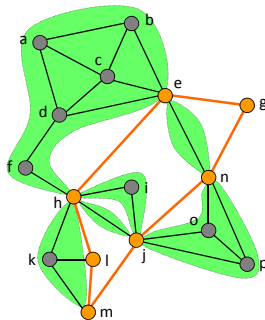
Betrachte einen Kreis c im Graphen G .

Ein **Segment** relativ zu c ist ein Untergraph S von G , der

- aus einer einzelnen Kante $e=(u,v)$ mit $u,v \in c$ und $e \notin c$ besteht, **oder**
- eine Zusammenhangskomponente K von $G \setminus c$ ist, plus alle Kanten, die K mit c verbinden.

Graphenalgorithmen • Berechnung der 3-Zusammenhangskomponenten in Linearzeit • 8

Segmente (2)



Graphenalgorithmen • Berechnung der 3-Zusammenhangskomponenten in Linearzeit • 9

Separationspaare

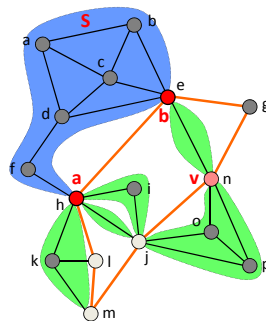
Lemma. Seien S_1, \dots, S_n die Segmente relativ zu c . Ist $\{a, b\}$ ein Separationspaar von G , dann gilt:

- Die Knoten a und b liegen beide auf c , oder beide liegen im gleichen Segment.
- Falls a und b auf c liegen, dann trifft (mind.) einer der folgenden Fälle zu:

Graphenalgorithmen • Berechnung der 3-Zusammenhangskomponenten in Linearzeit • 10

Typ-1

Ein Segment S mit mind. 2 Kanten hat nur a und b mit c gemeinsam und ein Knoten v liegt nicht S .

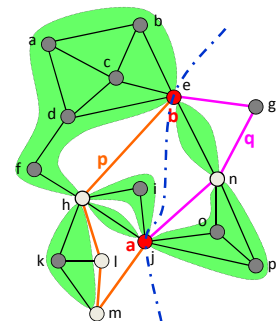


Graphenalgorithmen • Berechnung der 3-Zusammenhangskomponenten in Linearzeit • 11

Typ-2

Seien p und q die beiden Pfade, in die c durch a und b geteilt wird.

- Kein Segment enthält einen internen Knoten von p **und** einen von q .
- p und q enthalten beide einen internen Knoten.



Graphenalgorithmen • Berechnung der 3-Zusammenhangskomponenten in Linearzeit • 12

DFS-Baum

Betrachte DFS Baum („palm tree“) von G:

$\text{num}(v)$ DFS-Nummer von v

$v \rightarrow w$ Baumkante

$v \leftarrow w$ Rückwärtskante

$D(v) = \{ w \mid v \rightarrow^* w \}$ Menge der Nachfolger von v

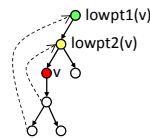
Graphenalgorithmen • Berechnung der 3-Zusammenhangskomponenten in Linearzeit • 13

Lowpt-Werte

bel. viele Baumkanten gefolgt von einer Rückwärtskante

$\text{lowpt1}(v) = \min \{ \text{num}(v) \} \cup \{ \text{num}(w) \mid v \rightarrow^* w \}$

$\text{lowpt2}(v) = \min \{ \text{num}(v) \} \cup \{ \text{num}(w) \mid v \rightarrow^* w \} \setminus \text{lowpt1}(v)$



Graphenalgorithmen • Berechnung der 3-Zusammenhangskomponenten in Linearzeit • 14

Nummerierung

Aber: Wir benötigen eine spezielle Nummerierung $\text{num}(v)$:

Sei $\text{Adj}(v)$ die Adjazenzliste von v .

(P1) Die Wurzel hat Nummer 1.

(P2) Seien w_1, \dots, w_n die Kinder von v gemäß $\text{Adj}(v)$.

Dann ist

$\text{num}(w_i) = \text{num}(v) + |D(w_1+1) \cup \dots \cup D(w_n)| + 1$

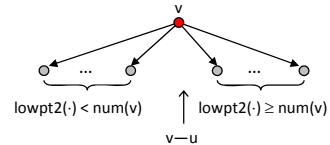
Graphenalgorithmen • Berechnung der 3-Zusammenhangskomponenten in Linearzeit • 15

Nummerierung (2)

(P3) Die Kanten e in $\text{Adj}(v)$ sind aufsteigend sortiert nach

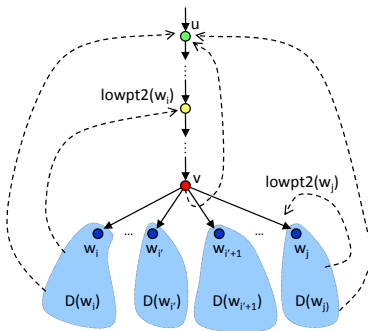
$\begin{cases} \text{lowpt1}(w) & \text{falls } e = v \rightarrow w \\ \text{num}(w) & \text{falls } e = v \leftarrow w \end{cases}$

Und: Knoten w_1, \dots, w_j mit gleichem lowpt1 -Wert u sind wie folgt gemäß lowpt2 -Wert sortiert:



Graphenalgorithmen • Berechnung der 3-Zusammenhangskomponenten in Linearzeit • 16

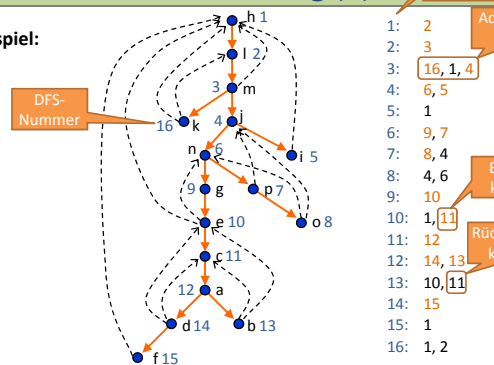
Nummerierung (3)



Graphenalgorithmen • Berechnung der 3-Zusammenhangskomponenten in Linearzeit • 17

Nummerierung (4)

Beispiel:



1:	2	DFS-Nummer
2:	3	Adjazenzliste
3:	16, 1, 4	
4:	6, 5	
5:	1	
6:	9, 7	
7:	8, 4	
8:	4, 6	Baumkante
9:	10	
10:	1, 11	
11:	12	
12:	14, 13	Rückwärtskante
13:	10, 11	
14:	15	
15:	1	
16:	1, 2	

Graphenalgorithmen • Berechnung der 3-Zusammenhangskomponenten in Linearzeit • 18

Nummerierung (5)

Wie kann man diese Nummerierung effizient bestimmen?

- Sortiere Adjazenzlisten aufsteigend gemäß

$$\Phi(e) = \begin{cases} 3 \text{ lowpt1}(w) & \text{falls } e = v \rightarrow w \text{ und } \text{lowpt2}(w) < \text{num}(v) \\ 3w + 1 & \text{falls } e = v - w \\ 3 \text{ lowpt1}(w) + 2 & \text{falls } e = v \rightarrow w \text{ und } \text{lowpt2}(w) \geq \text{num}(v) \end{cases}$$

→ Bucket-Sort

- Bestimme damit Nummerierung gemäß (P1) und (P2).
- lowpt1 und lowpt2 müssen neu berechnet werden!

Graphenalgorithmen • Berechnung der 3-Zusammenhangskomponenten in Linearzeit • 19

Nummerierung: Konvention

Wir identifizieren im Folgenden einen Knoten mit seiner DFS-Nummer:

z.B.: $v \neq 1$ „v ist nicht Wurzel“
 $a \leq b$ „num(a) ≤ num(b)“

Graphenalgorithmen • Berechnung der 3-Zusammenhangskomponenten in Linearzeit • 20

Pfade

Betrachte DFS-Traversierung gemäß Adj(v):

→ Menge von Pfaden der Form $v \rightarrow^* w - u$

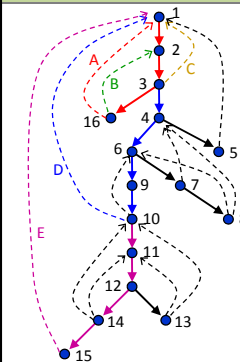
Jeder Pfad

- endet am Knoten mit der kleinsten möglichen Nummer;
- hat nur Anfangs- und Endknoten mit vorangehenden Pfaden gemeinsam.

→ erhalten Kreise $u \rightarrow^* v \rightarrow^* w - u$

Graphenalgorithmen • Berechnung der 3-Zusammenhangskomponenten in Linearzeit • 21

Pfade (2)



- A: $1 \rightarrow 2 \rightarrow 3 \rightarrow 16 - 1$
- B: $16 - 2$
- C: $3 - 1$
- D: $3 \rightarrow 4 \rightarrow 6 \rightarrow 9 \rightarrow 10 - 1$
- E: $10 \rightarrow 11 \rightarrow 12 \rightarrow 14 \rightarrow 15 - 1$
- F: $14 - 10$
- G: $14 - 11$
- H: $12 \rightarrow 13 - 10$
- I: $13 - 11$
- J: $10 - 6$
- K: $6 \rightarrow 7 \rightarrow 8 - 4$
- L: $8 - 6$
- M: $7 - 4$
- N: $4 \rightarrow 5 - 1$

Graphenalgorithmen • Berechnung der 3-Zusammenhangskomponenten in Linearzeit • 22

Separationspaare: Typ-1

$\{a, b\}$ mit $\text{num}(a) < \text{num}(b)$ ist Separationspaar, falls

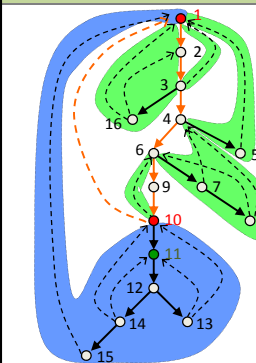
Typ-1:

Ex. $r \neq a, b$ und $s \neq a, b$ mit

- $b \rightarrow r$
- $\text{lowpt1}(r) = a$
- $\text{lowpt2}(r) \geq b$
- $s \notin D(r)$

Graphenalgorithmen • Berechnung der 3-Zusammenhangskomponenten in Linearzeit • 23

Separationspaare: Typ-1 (2)



- $a = 1$
- $b = 10$
- $r = 11$
- $s = 2$
- $\text{lowpt1}(r) = 1$
- $\text{lowpt2}(r) = 10 \geq b$

Graphenalgorithmen • Berechnung der 3-Zusammenhangskomponenten in Linearzeit • 24

Separationspaare: Typ-2

$\{a, b\}$ mit $a \neq 1$ und $a < b$ ist Separationspaar, falls

Typ-2:

Ex. $r \neq b$ mit

- $a \rightarrow r \rightarrow^* b$
- b ist ein „erster“ Nachfahre von r
- Für alle $x - y$ mit $r \leq x < b$ gilt: $y \geq a$
- Für alle $x - y$ mit $a < y < b$ und $b \rightarrow w \rightarrow^* x$ gilt: $\text{lowpt1}(w) \geq a$

Graphenalgorithmen • Berechnung der 3-Zusammenhangskomponenten in Linearzeit • 25

Separationspaare: Typ-2

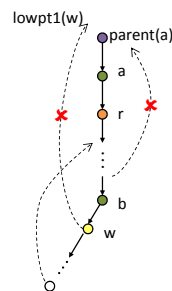
Intuitiv:

- Löschen von a und b trennt die Knoten $\text{parent}(a)$ und r voneinander.
- Die Bedingungen für Rückwärtskanten bedeuten:
 - Es darf keine Rückwärtskante geben, die aus dem Bereich zwischen r und b über a hinausläuft, denn diese würde $\text{parent}(a)$ mit r verbinden.
 - Gibt es ein Kind w von b , aus dessen Teilbaum eine Rückwärtskante in den Bereich zwischen r und x läuft, dann ist der Teilbaum an w auch nach dem Löschen von a und b mit r verbunden. Daher muss ausgeschlossen werden, dass eine Rückwärtskante aus diesem Teilbaum in den Bereich über a führt, also wird $\text{lowpt1}(w) \geq a$ gefordert.

Graphenalgorithmen • Berechnung der 3-Zusammenhangskomponenten in Linearzeit • 26

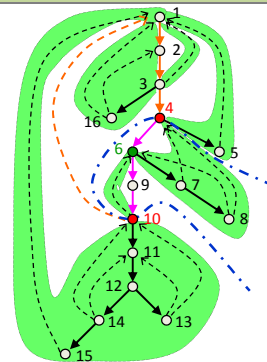
Separationspaare: Typ-2 (2)

Löschen von a und b trennt die Knoten $\text{parent}(a)$ und r voneinander.



Graphenalgorithmen • Berechnung der 3-Zusammenhangskomponenten in Linearzeit • 27

Separationspaare: Typ-2 (2)

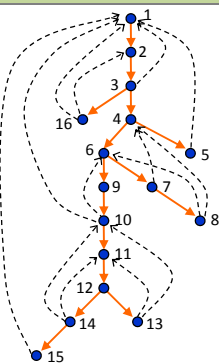


$a = 4$
 $b = 10$
 $r = 6$

relevante Rückwärtskanten:
8 - 6
8 - 4
7 - 4

Graphenalgorithmen • Berechnung der 3-Zusammenhangskomponenten in Linearzeit • 28

Separationspaare



Typ-1:

$\{1,4\}, \{1,6\}, \{4,6\}$
 $\{1,10\}, \{1,3\}$

Typ-2:

$\{6,10\}, \{4,10\}, \{10,15\}$
 $\{10,14\}$

Graphenalgorithmen • Berechnung der 3-Zusammenhangskomponenten in Linearzeit • 29

Test auf Separationspaar

Typ-1:

→ Test einer einfachen Bedingung für jede Baumkante.

Typ-2:

→ Einfach, falls $a \rightarrow v \rightarrow b$ und $\text{deg}(v)=2$.

Sonst: Benutze TSTACK

Graphenalgorithmen • Berechnung der 3-Zusammenhangskomponenten in Linearzeit • 30

Test auf Separationspaar (2)

TSTACK

- Enthält Tripel (h, a, b)
 - h ist höchste Nummer eines Knotens in zugehöriger Split-Komponente
 - $\{a, b\}$ ist potentiell Typ-2 Separationspaar
- $(h_1, a_1, b_1), \dots, (h_k, a_k, b_k)$ aktueller Knoten
→ $a_k \leq a_{k-1} \leq \dots \leq a_1 \leq v_i \leq b_1 \leq \dots \leq b_k$
- Alle a_i, b_i sind auf aktuellem Kreis

Graphenalgorithmen • Berechnung der 3-Zusammenhangskomponenten in Linearzeit • 31

Test auf Separationspaar (3)

Nach Bearbeitung von $v_i \rightarrow v_{i+1}$:

- Sei (h, a, b) oberstes Element auf TSTACK.
Dann ist $\{a, b\}$ Typ-2 Separationspaar, falls
 - $a = v_i$
 - $v_i \neq 1$
 - $a \neq \text{parent}(b)$
- zugehörige Split-Komponente: speichert „aktuelle“ Kanten
 - $e=(x, y)$ auf ESTACK mit $a \leq x \leq h$ und $a \leq y \leq h$
 - $e=(a, b) \rightarrow$ Multi-Kante
- Eigenschaften bzgl. Rückwärtskanten werden durch Update von TSTACK sichergestellt!

Graphenalgorithmen • Berechnung der 3-Zusammenhangskomponenten in Linearzeit • 32

Test auf Separationspaar (4)

Multi-Kanten:

- Können beim Berechnen der Split-Komponenten auftreten.
- Werden beim Erzeugen einer virtuellen Kante erkannt.
- Wichtig: Ordnung der Adjazenzlisten!

Graphenalgorithmen • Berechnung der 3-Zusammenhangskomponenten in Linearzeit • 33

Split-Komponenten

Berechnung der Split-Komponenten:

- Verwalte aktuellen Graphen und DFS-Baum.
→ Update beim Abspalten von Split-Komponenten.
- Abspalten = Löschen einiger Kanten und ersetzen durch neue virtuelle Kante.
→ ESTACK

Graphenalgorithmen • Berechnung der 3-Zusammenhangskomponenten in Linearzeit • 34

Split-Komponenten (2)

Verwaltung von ESTACK:

- Bearbeitung von $v \rightarrow w$:
→ Push auf ESTACK nach rekursivem Aufruf für w .
- Bearbeitung von $v \leftarrow w$:
→ Push auf ESTACK falls keine Multi-Kante.

Graphenalgorithmen • Berechnung der 3-Zusammenhangskomponenten in Linearzeit • 35

Korrekturen

- Funktion Φ zum Sortieren der Adjazenzlisten
→ Erkennen von Multi-Kanten
- Erzeugen der letzten Split-Komponente
- Update von TSTACK
- Test auf Typ-2 Separationspaare
- Verschiedene Updates: A1(v), DEGREE(v), HIGHPT(v)

Graphenalgorithmen • Berechnung der 3-Zusammenhangskomponenten in Linearzeit • 36

Literatur

J. E. Hopcroft, R. E. Tarjan, *Dividing a graph into triconnected components*, SIAM Journal on Computing 2, 1973, pp. 135-158.

C. Gutwenger, P. Mutzel, *A linear time implementation of SPQR-trees*, in: J. Marks (ed.), Graph Drawing (GD 2000), LNCS 1984, pp. 77-90, Springer-Verlag, 2001.