

Algorithmen und Datenstrukturen

Sommersemester 2020

Stefan Näher

Universität Trier

naeher@uni-trier.de

Vorlesung 15

17. Juni 2020

Eine genauere Untersuchung der Tiefensuche

DFS als rekursive Prozedur

```
1. dfs(v)
2. besucht[v] ← true;
3. forall  $w \in V$  mit  $(v, w) \in E$  do
4.   if  $\neg$ besucht[w] then
5.     dfs(w);
6.   fi
7. od
```

Hauptprogramm (Initialisierung)

```
forall  $v \in V$  do besucht[v] ← false; od
```

Partitionierung der Kanten in T,F,B,C

T: Baumkanten oder Tree-Edges

Eine Kante (v, w) , die in Zeile 4 der dfs-Funktion zu einem unbesuchten Knoten w führt, heißt **Baumkante** oder Tree-Edge.

Alle anderen Kanten (v, w) , die also in Zeile 4 zu einem **bereits besuchten Knoten** w führen, werden auf die Klassen F, B und C verteilt.

F: Vorwärtskanten oder Forward-Edges

$(v, w) \in F$, wenn ein Pfad aus Baumkanten der Länge > 0 von v nach w existiert.

$$v \xrightarrow[T]{+} w$$

B: Rückwärtskanten oder Back-Edges

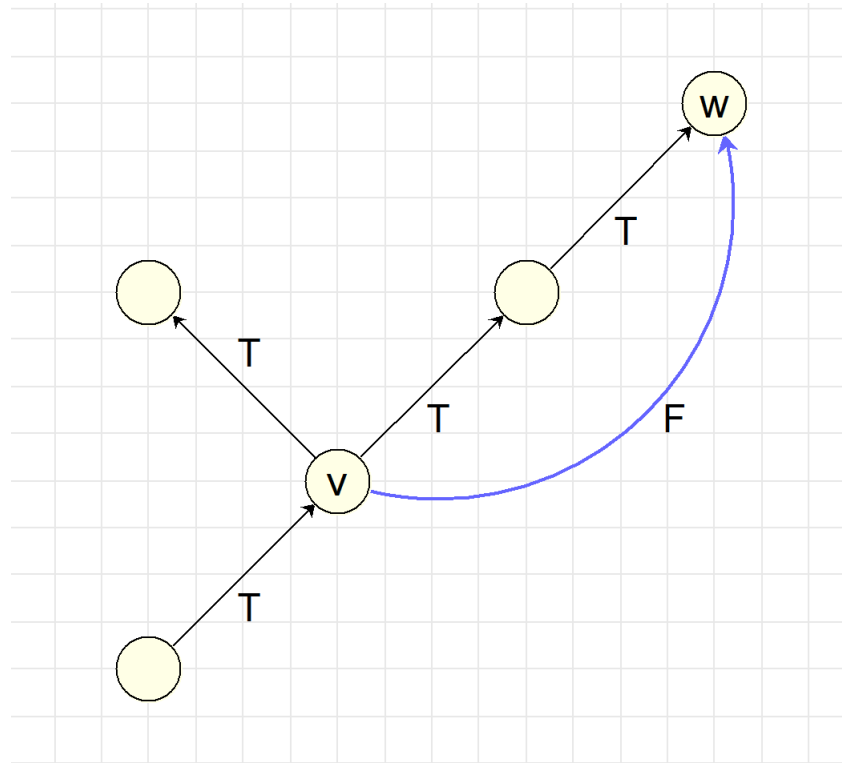
$(v, w) \in B$, wenn ein beliebiger Pfad aus Baumkanten von w nach v existiert (auch wenn $v = w$).

$$w \xrightarrow[T]{*} v$$

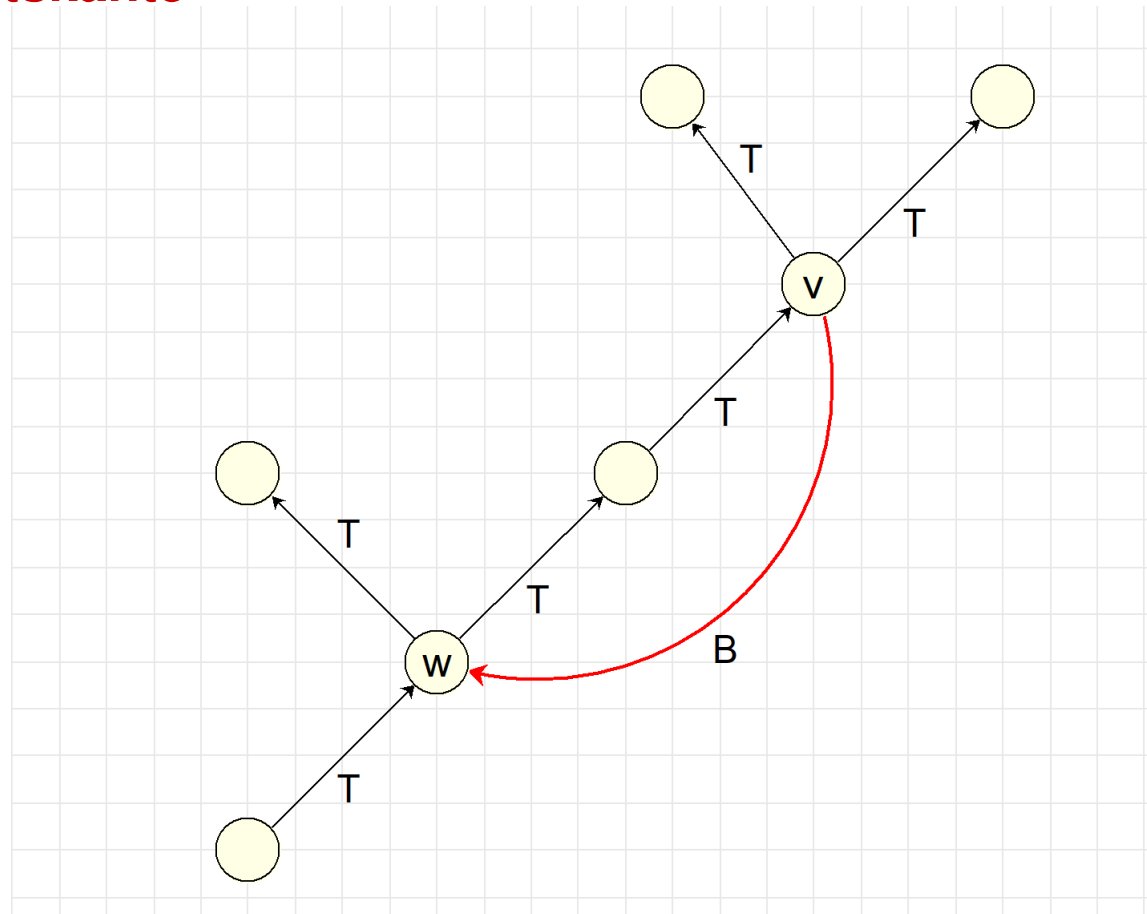
C: Querkanten oder Cross-Edges

$(v, w) \in C$, wenn es weder einen Baum-Pfad von v nach w noch von w nach v gibt (d.h. alle übrigen Kanten).

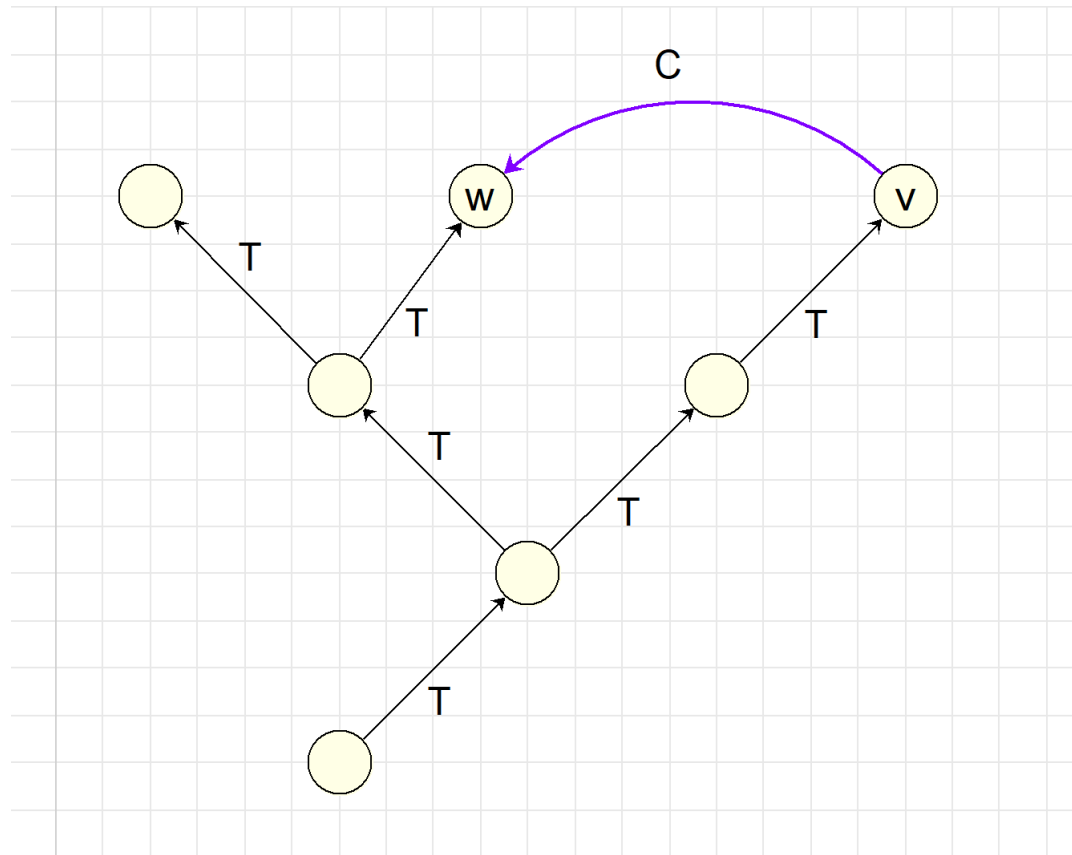
Eine Vorwärtskante



Eine Rückwärtskante



Eine Querkante



Erweiterung des Algorithmus

1. Nummerierung der Knoten

- $\text{dfsnum}[v]$ Reihenfolge der rekursiven Aufrufe von $\text{dfs}(v)$
- $\text{compum}[v]$ Reihenfolge der Abschlüsse der dfs -Aufrufe

2. Zwei Zähler count1 , count2

3. Die 4 Kantenmengen T, F, B, C

dfs(v) (grüne Zeilen sind konzeptuell)

```
1. besucht[v] ← true;
2. dfsnum[v] ← ++count1;
3. forall  $w \in V$  mit  $(v, w) \in E$  do
4.   if  $\neg \text{besucht}[w]$ 
5.   then  $T \leftarrow T \cup \{ (v, w) \};$ 
6.     dfs(w);
7.   else if  $v \xrightarrow[T]{+} w$ 
8.   then  $F \leftarrow F \cup \{ (v, w) \};$ 
9.   else if  $w \xrightarrow[T]{*} v$ 
10.    then  $B \leftarrow B \cup \{ (v, w) \};$ 
11.    else  $C \leftarrow C \cup \{ (v, w) \};$ 
12.    fi
13.  fi
14. fi
15. od
16. compnum ← ++count2;
```

DFS Hauptprogramm

Ein DFS-Lauf auf dem Graphen G

$\text{count1}, \text{count2} \leftarrow 0;$

$T, F, B, C \leftarrow \emptyset;$

forall $v \in V$ **do** $\text{besucht}[v] = \text{false};$ **od**

forall $v \in V$ **do**

if $\neg \text{besucht}[v]$ **then**

$\text{dfs}(v);$

fi

od

Laufzeitanalyse

Lemma 1

Ein DFS-Lauf (Hauptprogramm) auf einem Graphen mit n Knoten und m Kanten hat Laufzeit $\mathcal{O}(n + m)$.

Beweis:

Die Ausführung des Rumpfes eines Aufrufs von $\text{dfs}(v)$ *ohne* die darin verschachtelten rekursiven Aufrufe hat Laufzeit $\mathcal{O}(1 + \text{outdeg}(v))$. Ferner wird $\text{dfs}(v)$ für jeden Knoten v genau einmal aufgerufen (wegen des besucht-Feldes).

Damit ergibt sich eine Gesamtlaufzeit von

$$\mathcal{O} \left(\sum_{v \in V} (1 + \text{outdeg}(v)) \right) = \mathcal{O}(n + m)$$

Klassifizierung der Kanten durch dfsnum und compnum

Lemma 2:

Sei $G = (V, E)$ ein gerichteter Graph und T, F, B, C , dfsnum , compnum durch einen DFS-Lauf auf G berechnet, dann gilt:

- a) T, F, B, C bilden eine Partition der Kantenmenge E
- b) T entspricht dem Baum der rekursiven dfs-Aufrufe.
- c) $v \xrightarrow[T]{*} w \iff \text{dfsnum}[v] \leq \text{dfsnum}[w] \text{ und } \text{compnum}[v] \geq \text{compnum}[w]$
- d) $(v, w) \in T \cup F \iff \text{dfsnum}[v] < \text{dfsnum}[w]$
- e) $(v, w) \in B \iff \text{dfsnum}[v] \geq \text{dfsnum}[w] \text{ und } \text{compnum}[v] \leq \text{compnum}[w]$
- f) $(v, w) \in C \iff \text{dfsnum}[v] > \text{dfsnum}[w] \text{ und } \text{compnum}[v] > \text{compnum}[w]$

Folgerungen

Sei $G = (V, E)$ ein **azyklischer Graph**, dann gilt

1. Es gibt keine Rückwärtskanten, d.h. $B = \emptyset$.
 2. Aus Lemma 2 folgt dann: $\text{compnum}(v) > \text{compnum}(w)$ für alle $(v, w) \in E$.
 3. Dann ist $\text{ord} : V \longrightarrow \{1, \dots, n\}$ mit $\text{ord}(v) = n - \text{compnum}(v) + 1$ eine topologische Sortierung von G .
- alternativer Algorithmus zur Berechnung einer topologischen Sortierung mit Laufzeit $\mathcal{O}(n + m)$.