5 Graphenalgorithmen

ungerichteter Graph $G = (V, E)$:

- V = endliche **Knotenmenge**
- E = **Kantenmenge** E
  dabei sei $e = \{u, v\} \in E$ **zweielementige Teilmenge** von V

ungerichteter Graph $G = (V, E)$:

- V = endliche **Knotenmenge**
- E = **Kantenmenge** E
  dabei sei $e = \{u, v\} \in E$ **zweielementige Teilmenge** von V

gerichteter Graph $G = (V, E)$:

- V = endliche **Knotenmenge**
- E = **Kantenmenge** E
  dabei sei $e = (u, v) \in E$ **geordnetes Paar** von Elementen aus V

ungerichteter Graph $G = (V, E)$:

- V = endliche **Knotenmenge**
- E = **Kantenmenge** E
  dabei sei $e = \{u, v\} \in E$ **zweielementige Teilmenge** von V

gerichteter Graph $G = (V, E)$:

- V = endliche **Knotenmenge**
- E = **Kantenmenge** E
  dabei sei $e = (u, v) \in E$ **geordnetes Paar** von Elementen aus V

Hinweis: Ist aus Kontext klar, dass G ungerichtet ist, schreiben wir oft (u, v) statt $\{u, v\}$.

ungerichteter Graph $G = (V, E)$:

- V = endliche **Knotenmenge**
- E = **Kantenmenge** E
  dabei sei $e = \{u, v\} \in E$ **zweielementige Teilmenge** von V

gerichteter Graph $G = (V, E)$:

- V = endliche **Knotenmenge**
- E = **Kantenmenge** E
  dabei sei $e = (u, v) \in E$ **geordnetes Paar** von Elementen aus V

Hinweis: Ist aus Kontext klar, dass G ungerichtet ist, schreiben wir oft (u, v) statt $\{u, v\}$.

Kantengewichte: Funktion $w\colon E \to \mathbb{R}$, die jeder Kante $e \in E$ ein *Gewicht* $w(e)$ zuweist.

ungerichteter Graph $G = (V, E)$:

Sei $G = (V, E)$ **ungerichtet**.

- $u, v \in V$ heißen **adjazent**, wenn $\{u, v\} \in E$.
- Knoten $u \in V$ und Kante $\{u, v\} \in E$ heißen **inzident**.
- Für $v \in V$: **Grad $d(v)$** = Anzahl zu v inzidenter Kanten.

ungerichteter Graph $G = (V, E)$:

Sei $G = (V, E)$ **ungerichtet**.

- $u, v \in V$ heißen **adjazent**, wenn $\{u, v\} \in E$.
- Knoten $u \in V$ und Kante $\{u, v\} \in E$ heißen **inzident**.
- Für $v \in V$: **Grad $d(v)$** = Anzahl zu v inzidenter Kanten.

gerichteter Graph $G = (V, E)$:

- Für $v \in V$ heißt $(u, v) \in E$ in v **eingehende Kante.**
- Für $v \in V$ heißt $(v, u) \in E$ von v **ausgehende Kante.**
- Für $(u, v) \in E$ heißt v **direkter Nachfolger** von u und u **direkter Vorgänger** von v.
- Für $v \in V$: **Outgrad $d^+(v)$** = Anzahl der von v ausgehenden Kanten.
- Für $v \in V$: **Ingrad $d^-(v)$** = Anzahl der in v eingehenden Kanten.

- Eine Folge $v_0, \ldots, v_\ell \in V$ heißt **Weg von v_0 nach v_ℓ der Länge $\ell \in \mathbb{N}_0$**, wenn $(v_{i-1}, v_i) \in E$ für alle $i \in \{1, \ldots, \ell\}$ gilt.
  Alternativ bezeichnen wir auch $(v_0, v_1), \ldots, (v_{\ell-1}, v_\ell) \in E$ als Weg.

- Eine Folge $v_0, \ldots, v_\ell \in V$ heißt **Weg von v_0 nach v_ℓ der Länge $\ell \in \mathbb{N}_0$**, wenn $(v_{i-1}, v_i) \in E$ für alle $i \in \{1, \ldots, \ell\}$ gilt.
  Alternativ bezeichnen wir auch $(v_0, v_1), \ldots, (v_{\ell-1}, v_\ell) \in E$ als Weg.
- Ein Weg $v_0, \ldots, v_\ell$ heißt **einfach**, wenn alle Knoten auf dem Weg paarweise verschieden sind.

- Eine Folge $v_0, \ldots, v_\ell \in V$ heißt **Weg von v_0 nach v_ℓ der Länge $\ell \in \mathbb{N}_0$**, wenn $(v_{i-1}, v_i) \in E$ für alle $i \in \{1, \ldots, \ell\}$ gilt.
  Alternativ bezeichnen wir auch $(v_0, v_1), \ldots, (v_{\ell-1}, v_\ell) \in E$ als Weg.
- Ein Weg $v_0, \ldots, v_\ell$ heißt **einfach**, wenn alle Knoten auf dem Weg paarweise verschieden sind.
- Ein Weg heißt **Kreis**, wenn $v_0 = v_\ell$ und $\ell \geq 1$. Er ist **einfach**, wenn $v_0 = v_\ell$ gilt und alle anderen Knoten paarweise verschieden und verschieden von $v_0 = v_\ell$ sind.

- Eine Folge $v_0, \ldots, v_\ell \in V$ heißt **Weg von v_0 nach v_ℓ der Länge $\ell \in \mathbb{N}_0$**, wenn $(v_{i-1}, v_i) \in E$ für alle $i \in \{1, \ldots, \ell\}$ gilt.
  Alternativ bezeichnen wir auch $(v_0, v_1), \ldots, (v_{\ell-1}, v_\ell) \in E$ als Weg.
- Ein Weg $v_0, \ldots, v_\ell$ heißt **einfach**, wenn alle Knoten auf dem Weg paarweise verschieden sind.
- Ein Weg heißt **Kreis**, wenn $v_0 = v_\ell$ und $\ell \geq 1$. Er ist **einfach**, wenn $v_0 = v_\ell$ gilt und alle anderen Knoten paarweise verschieden und verschieden von $v_0 = v_\ell$ sind.
- (Zusatzbedingung für Kreise in ungerichtete Graphen: Kreis ist einfach und $\ell \geq 3$.)

- Eine Folge $v_0, \ldots, v_\ell \in V$ heißt **Weg von v_0 nach v_ℓ der Länge $\ell \in \mathbb{N}_0$**, wenn $(v_{i-1}, v_i) \in E$ für alle $i \in \{1, \ldots, \ell\}$ gilt.
  Alternativ bezeichnen wir auch $(v_0, v_1), \ldots, (v_{\ell-1}, v_\ell) \in E$ als Weg.
- Ein Weg $v_0, \ldots, v_\ell$ heißt **einfach**, wenn alle Knoten auf dem Weg paarweise verschieden sind.
- Ein Weg heißt **Kreis**, wenn $v_0 = v_\ell$ und $\ell \geq 1$. Er ist **einfach**, wenn $v_0 = v_\ell$ gilt und alle anderen Knoten paarweise verschieden und verschieden von $v_0 = v_\ell$ sind.
- (Zusatzbedingung für Kreise in ungerichtete Graphen: Kreis ist einfach und $\ell \geq 3$.)
- Ein Graph, der keinen Kreis enthält, heißt **kreisfrei**, **azyklisch** oder **Wald**.

- Eine Folge $v_0, \ldots, v_\ell \in V$ heißt **Weg von v_0 nach v_ℓ der Länge $\ell \in \mathbb{N}_0$**, wenn $(v_{i-1}, v_i) \in E$ für alle $i \in \{1, \ldots, \ell\}$ gilt.
  Alternativ bezeichnen wir auch $(v_0, v_1), \ldots, (v_{\ell-1}, v_\ell) \in E$ als Weg.
- Ein Weg $v_0, \ldots, v_\ell$ heißt **einfach**, wenn alle Knoten auf dem Weg paarweise verschieden sind.
- Ein Weg heißt **Kreis**, wenn $v_0 = v_\ell$ und $\ell \geq 1$. Er ist **einfach**, wenn $v_0 = v_\ell$ gilt und alle anderen Knoten paarweise verschieden und verschieden von $v_0 = v_\ell$ sind.
- (Zusatzbedingung für Kreise in ungerichtete Graphen: Kreis ist einfach und $\ell \geq 3$.)
- Ein Graph, der keinen Kreis enthält, heißt **kreisfrei**, **azyklisch** oder **Wald**.
- Ein ungerichteter Graph heißt **zusammenhängend**, wenn es zwischen jedem Paar von Knoten einen Weg gibt. Ein zusammenhängender Graph, der keinen Kreis besitzt, heißt **Baum**.

Datenstrukturen für Graphen: Sei $G = (V, E)$ mit $V = \{v_1, \ldots, v_n\}$ und $m = |E|$.

- Bei der **Adjazenzmatrix** $\mathbf{A} = (\mathbf{a_{ij}})$ von G handelt es sich um eine $n \times n$-Matrix. Ist G ungewichtet oder gewichtet, gilt

$$a_{ij} = \begin{cases} 1 & \text{falls } (v_i, v_j) \in E, \\ 0 & \text{sonst,} \end{cases} \qquad \text{bzw.} \qquad a_{ij} = \begin{cases} w((v_i, v_j)) & \text{falls } (v_i, v_j) \in E, \\ \perp & \text{sonst.} \end{cases}$$

G ungerichtet $\Rightarrow$ A symmetrisch

Speicherplatz $\Theta(n^2)$.

Datenstrukturen für Graphen: Sei $G = (V, E)$ mit $V = \{v_1, \ldots, v_n\}$ und $m = |E|$.

- Bei der **Adjazenzmatrix** $\mathbf{A} = (\mathbf{a_{ij}})$ von G handelt es sich um eine $n \times n$-Matrix. Ist G ungewichtet oder gewichtet, gilt

$$a_{ij} = \begin{cases} 1 & \text{falls } (v_i, v_j) \in E, \\ 0 & \text{sonst,} \end{cases} \qquad \text{bzw.} \qquad a_{ij} = \begin{cases} w((v_i, v_j)) & \text{falls } (v_i, v_j) \in E, \\ \perp & \text{sonst.} \end{cases}$$

  G ungerichtet $\Rightarrow$ A symmetrisch Speicherplatz $\Theta(n^2)$.

- Eine **Adjazenzliste** ist ein Feld von n verketteten Listen. Liste von $v \in V$ enthält bei ungerichteten Graphen alle zu v adjazenten Knoten und bei gerichteten Graphen alle direkten Nachfolger von v.
  Speicherplatz $\Theta(n + m)$

5.1.1 Tiefensuche

Tiefensuche/Depth-First Search (DFS)

```
DFS(G = (V, E))
1    for each (u ∈ V) {
2         u.color = weiß;
3         u.π = null;
4    }
5    time = 0;
6    for each (u ∈ V)
7         if (u.color == weiß) DFS-VISIT(G, u);
```

Attribute eines Knotens:

- Farbe $u.\mathrm{color}$
- Vorgänger $u.\pi$
- Entdeckungszeitpunkt $u.d$
- Fertigstellungszeitpunkt $u.f$

Tiefensuche/Depth-First Search (DFS)

```
DFS(G = (V, E))
1    for each (u ∈ V) {
2         u.color = weiß;
3         u.π = null;
4    }
5    time = 0;
6    for each (u ∈ V)
7         if (u.color == weiß) DFS-VISIT(G, u);
```

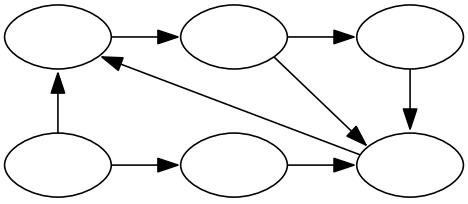
Attribute eines Knotens:

- Farbe $u.\text{color}$
- Vorgänger $u.\pi$
- Entdeckungszeitpunkt $u.d$
- Fertigstellungszeitpunkt $u.f$

```
DFS-VISIT(G = (V, E), u)
1    time++;
2    u.d = time;
3    u.color = grau;
4    for each ((u, v) ∈ E) {
5         if (v.color == weiß) {
6              v.π = u;
7              DFS-VISIT(G, v);
8         }
9    }
10   u.color = schwarz;
11   time++;
12   u.f = time;
```

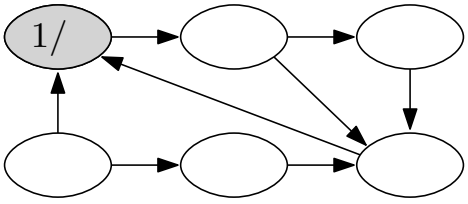
Beispiel für Tiefensuche

Knoten v ist mit $v.d/v.f$ beschriftet.



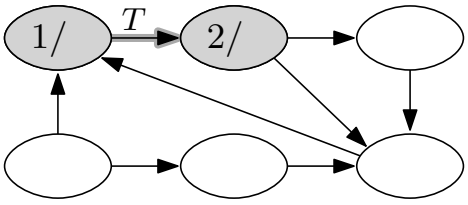
Beispiel für Tiefensuche

Knoten v ist mit $v.d/v.f$ beschriftet.



Beispiel für Tiefensuche

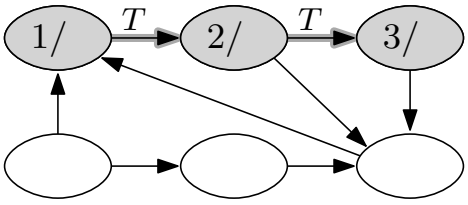
Knoten v ist mit $v.d/v.f$ beschriftet.



Ist Kante (u, v) mit T beschriftet, so gilt $v.\pi = u$.

Beispiel für Tiefensuche

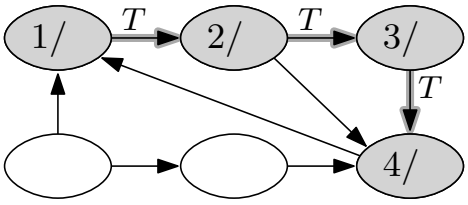
Knoten v ist mit $v.d/v.f$ beschriftet.



Ist Kante (u, v) mit T beschriftet, so gilt $v.\pi = u$.

Beispiel für Tiefensuche

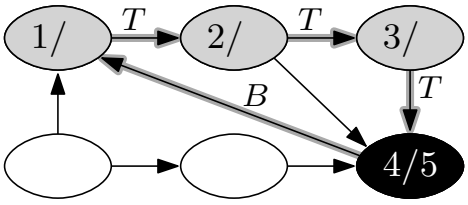
Knoten v ist mit $v.d/v.f$ beschriftet.



Ist Kante (u, v) mit T beschriftet, so gilt $v.\pi = u$.

Beispiel für Tiefensuche

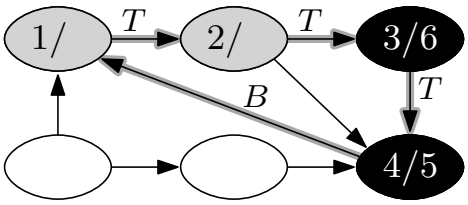
Knoten v ist mit $v.d/v.f$ beschriftet.



Ist Kante (u, v) mit T beschriftet, so gilt $v.\pi = u$.

Beispiel für Tiefensuche

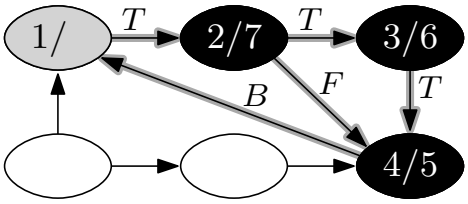
Knoten v ist mit $v.d/v.f$ beschriftet.



Ist Kante (u, v) mit T beschriftet, so gilt $v.\pi = u$.

Beispiel für Tiefensuche

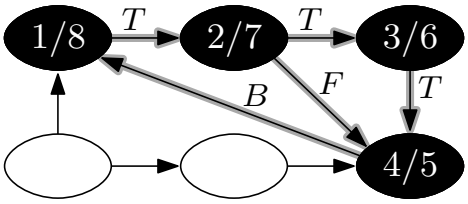
Knoten v ist mit $v.d/v.f$ beschriftet.



Ist Kante (u, v) mit T beschriftet, so gilt $v.\pi = u$.

Beispiel für Tiefensuche

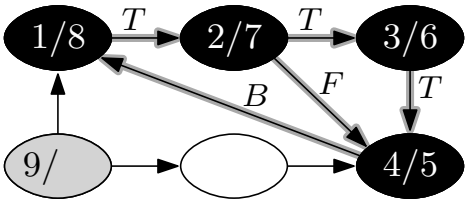
Knoten v ist mit $v.d/v.f$ beschriftet.



Ist Kante (u, v) mit T beschriftet, so gilt $v.\pi = u$.

Beispiel für Tiefensuche

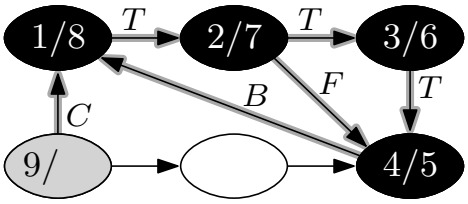
Knoten v ist mit $v.d/v.f$ beschriftet.



Ist Kante (u, v) mit T beschriftet, so gilt $v.\pi = u$.

Beispiel für Tiefensuche

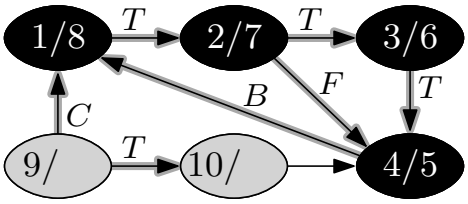
Knoten v ist mit $v.d/v.f$ beschriftet.



Ist Kante (u, v) mit T beschriftet, so gilt $v.\pi = u$.

Beispiel für Tiefensuche

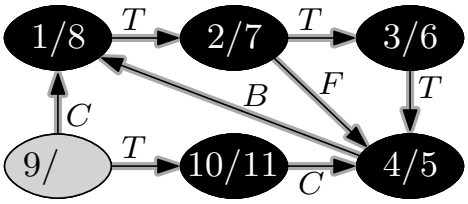
Knoten v ist mit $v.d/v.f$ beschriftet.



Ist Kante (u, v) mit T beschriftet, so gilt $v.\pi = u$.

Beispiel für Tiefensuche

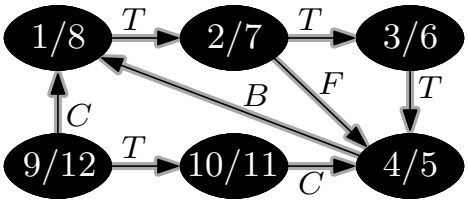
Knoten v ist mit $v.d/v.f$ beschriftet.



Ist Kante (u, v) mit T beschriftet, so gilt $v.\pi = u$.

Beispiel für Tiefensuche

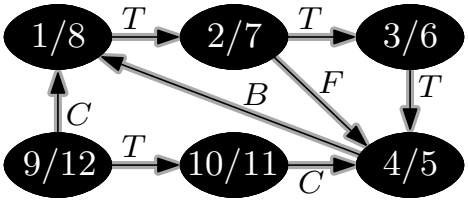
Knoten v ist mit $v.d/v.f$ beschriftet.



Ist Kante (u, v) mit T beschriftet, so gilt $v.\pi = u$.

Beispiel für Tiefensuche

Knoten v ist mit $v.d/v.f$ beschriftet.



Ist Kante (u, v) mit T beschriftet, so gilt $v.\pi = u$.

Ein Knoten $v \in V$ heißt **DFS-Nachfolger** von $u \in V$, wenn es Weg $u = v_0, v_1, \ldots, v_\ell = v$ gibt, sodass $v_i.\pi = v_{i-1}$ für alle $i \in \{1, \ldots, \ell\}$ gilt.

5.1.1 Tiefensuche

Lemma 5.1

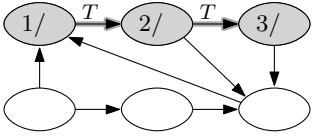
Ein Knoten $v \in V$ ist genau dann ein DFS-Nachfolger eines Knotens $u \in V$, wenn der Knoten u zum Zeitpunkt $v.d$ des ersten Besuches von v grau ist.

5.1.1 Tiefensuche

Lemma 5.1

Ein Knoten $v \in V$ ist genau dann ein DFS-Nachfolger eines Knotens $u \in V$, wenn der Knoten u zum Zeitpunkt $v.d$ des ersten Besuches von v grau ist.

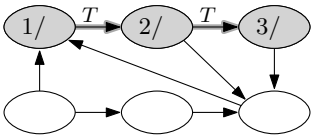
Beweis: Fixiere Zeitpunkt. Seien $u_1, \ldots, u_\ell$ die grauen Knoten mit $u_1.d < \ldots < u_\ell.d$.



Lemma 5.1

Ein Knoten $v \in V$ ist genau dann ein DFS-Nachfolger eines Knotens $u \in V$, wenn der Knoten u zum Zeitpunkt $v.d$ des ersten Besuches von v grau ist.

Beweis: Fixiere Zeitpunkt. Seien $u_1, \ldots, u_\ell$ die grauen Knoten mit $u_1.d < \ldots < u_\ell.d$.

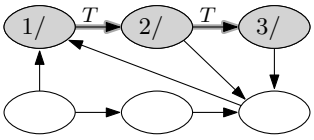


Zeige mit vollständiger Induktion: **$u_1, \ldots, u_\ell$ ist einfacher Weg** und Knoten u_i für $i \in \{1, \ldots, \ell\}$ besitzt genau die DFS-Vorgänger $u_1, \ldots, u_{i-1}$.

Lemma 5.1

Ein Knoten $v \in V$ ist genau dann ein DFS-Nachfolger eines Knotens $u \in V$, wenn der Knoten u zum Zeitpunkt $v.d$ des ersten Besuches von v grau ist.

Beweis: Fixiere Zeitpunkt. Seien $u_1, \ldots, u_\ell$ die grauen Knoten mit $u_1.d < \ldots < u_\ell.d$.



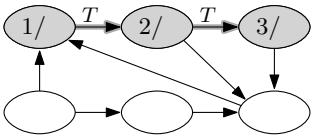
Zeige mit vollständiger Induktion: $u_1, \ldots, u_\ell$ **ist einfacher Weg** und Knoten u_i für $i \in \{1, \ldots, \ell\}$ besitzt genau die DFS-Vorgänger $u_1, \ldots, u_{i-1}$.

Zum aktuellen Zeitpunkt kann nur von u_ℓ aus ein neuer Knoten erreicht werden.

Lemma 5.1

Ein Knoten $v \in V$ ist genau dann ein DFS-Nachfolger eines Knotens $u \in V$, wenn der Knoten u zum Zeitpunkt $v.d$ des ersten Besuches von v grau ist.

Beweis: Fixiere Zeitpunkt. Seien $u_1, \ldots, u_\ell$ die grauen Knoten mit $u_1.d < \ldots < u_\ell.d$.



Zeige mit vollständiger Induktion: **$u_1, \ldots, u_\ell$ ist einfacher Weg** und Knoten u_i für $i \in \{1, \ldots, \ell\}$ besitzt genau die DFS-Vorgänger $u_1, \ldots, u_{i-1}$.

Zum aktuellen Zeitpunkt kann nur von u_ℓ aus ein neuer Knoten erreicht werden.

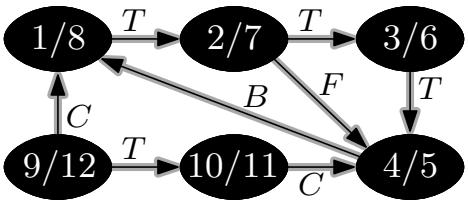
Wird von u_ℓ aus ein neuer Knoten v erreicht, so besitzt er genau die momentan grauen Knoten $u_1, \ldots, u_\ell$ als DFS-Vorgänger. □

5.1.1 Tiefensuche

Theorem 5.2

Für jeden (gerichteten oder ungerichteten) Graphen $G = (V, E)$ gilt für jedes Paar $u \in V$ und $v \in V$ von zwei verschiedenen Knoten genau eine der folgenden drei Aussagen.

1. Die Intervalle $[u.d, u.f]$ und $[v.d, v.f]$ sind disjunkt
   und weder u ist ein DFS-Nachfolger von v noch andersherum.
2. Es gilt $[u.d, u.f] \subseteq [v.d, v.f]$ und u ist ein DFS-Nachfolger von v.
3. Es gilt $[v.d, v.f] \subseteq [u.d, u.f]$ und v ist ein DFS-Nachfolger von u.



Beweis: Sei o. B. d. A. $u.d < v.d$.

Beweis: Sei o. B. d. A. $u.d < v.d$.

Da Knoten u vor Knoten v besucht wird, ist u kein DFS-Nachfolger von v.

Beweis: Sei o. B. d. A. $u.d < v.d$.

Da Knoten u vor Knoten v besucht wird, ist u kein DFS-Nachfolger von v.

Gilt $v.d > u.f$, so sind die Intervalle $[u.d, u.f]$ und $[v.d, v.f]$ disjunkt. In diesem Fall ist der Knoten u bereits schwarz, wenn der Knoten v erreicht wird. Gemäß Lemma 5.1 ist v demnach kein DFS-Nachfolger von u.

Beweis: Sei o. B. d. A. $u.d < v.d$.

Da Knoten u vor Knoten v besucht wird, ist u kein DFS-Nachfolger von v.

Gilt $v.d > u.f$, so sind die Intervalle $[u.d, u.f]$ und $[v.d, v.f]$ disjunkt. In diesem Fall ist der Knoten u bereits schwarz, wenn der Knoten v erreicht wird. Gemäß Lemma 5.1 ist v demnach kein DFS-Nachfolger von u.

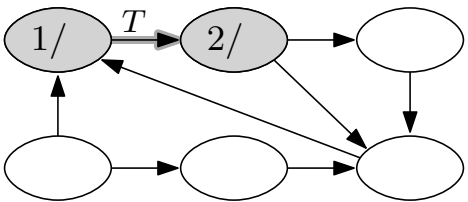
Gilt $v.d < u.f$, so ist der Knoten u zu dem Zeitpunkt, zu dem v erreicht wird, grau. Gemäß Lemma 5.1 ist v damit ein DFS-Nachfolger von u. Außerdem gilt $v.f < u.f$, denn die Tiefensuche arbeitet erst alle von v ausgehenden Kanten ab, bevor ein Backtracking zu u erfolgt. □

5.1.1 Tiefensuche

Einteilung der Kantenmenge: Situation bei erster Betrachtung von (u, v) in DFS-VISIT.

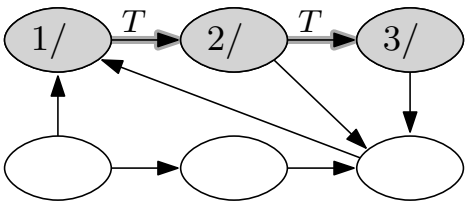
Einteilung der Kantenmenge: Situation bei erster Betrachtung von (u, v) in DFS-VISIT.

1. Ist $v.\mathrm{color} = \text{weiß}$, so ist (u, v) eine **Tree-** oder **T-Kante**.



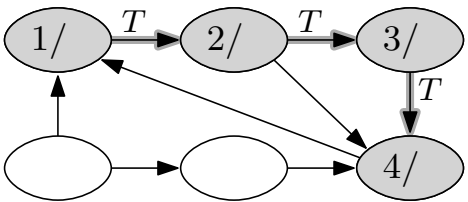
Einteilung der Kantenmenge: Situation bei erster Betrachtung von (u, v) in DFS-VISIT.

1. Ist $v.\mathrm{color} = \text{weiß}$, so ist (u, v) eine **Tree-** oder **T-Kante**.



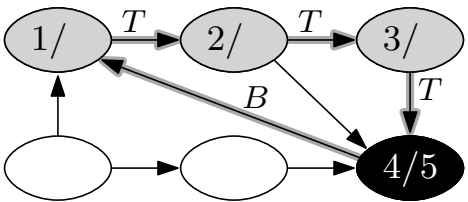
Einteilung der Kantenmenge: Situation bei erster Betrachtung von (u, v) in DFS-VISIT.

1. Ist $v.\mathrm{color} = \text{weiß}$, so ist (u, v) eine **Tree-** oder **T-Kante**.
2. Ist $v.\mathrm{color} = \text{grau}$, so ist (u, v) eine **Back-** oder **B-Kante**.



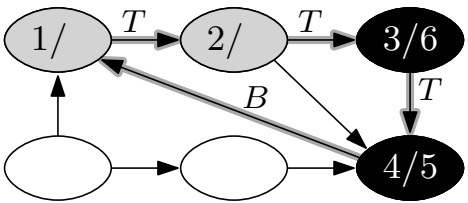
Einteilung der Kantenmenge: Situation bei erster Betrachtung von (u, v) in DFS-VISIT.

1. Ist $v.\mathrm{color} =$ weiß, so ist (u, v) eine **Tree-** oder **T-Kante**.
2. Ist $v.\mathrm{color} =$ grau, so ist (u, v) eine **Back-** oder **B-Kante**.



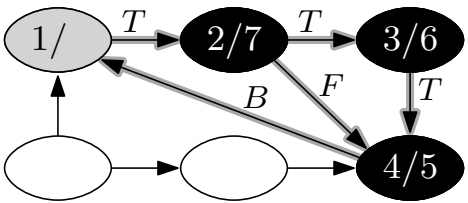
Einteilung der Kantenmenge: Situation bei erster Betrachtung von (u, v) in DFS-VISIT.

1. Ist $v.\mathrm{color} =$ weiß, so ist (u, v) eine **Tree-** oder **T-Kante**.
2. Ist $v.\mathrm{color} =$ grau, so ist (u, v) eine **Back-** oder **B-Kante**.
3. Ist $v.\mathrm{color} =$ schwarz und $v.d > u.d$, so ist (u, v) eine **Forward-** oder **F-Kante**.



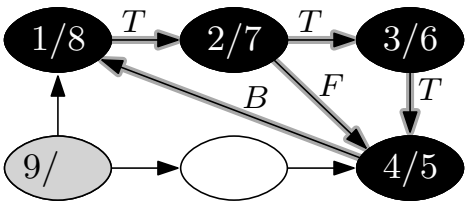
Einteilung der Kantenmenge: Situation bei erster Betrachtung von (u, v) in DFS-VISIT.

1. Ist $v.\mathrm{color} =$ weiß, so ist (u, v) eine **Tree-** oder **T-Kante**.
2. Ist $v.\mathrm{color} =$ grau, so ist (u, v) eine **Back-** oder **B-Kante**.
3. Ist $v.\mathrm{color} =$ schwarz und $v.d > u.d$, so ist (u, v) eine **Forward-** oder **F-Kante**.



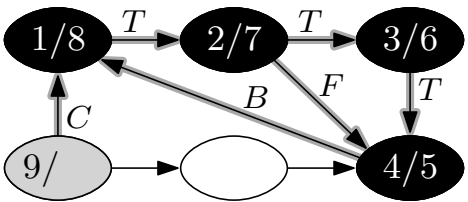
Einteilung der Kantenmenge: Situation bei erster Betrachtung von (u, v) in DFS-VISIT.

1. Ist $v.\text{color} =$ weiß, so ist (u, v) eine **Tree-** oder **T-Kante**.
2. Ist $v.\text{color} =$ grau, so ist (u, v) eine **Back-** oder **B-Kante**.
3. Ist $v.\text{color} =$ schwarz und $v.d > u.d$, so ist (u, v) eine **Forward-** oder **F-Kante**.
4. Ist $v.\text{color} =$ schwarz und $v.d < u.d$, so ist (u, v) eine **Cross-** oder **C-Kante**.



Einteilung der Kantenmenge: Situation bei erster Betrachtung von (u, v) in DFS-VISIT.

1. Ist $v.\mathrm{color} =$ weiß, so ist (u, v) eine **Tree-** oder **T-Kante**.
2. Ist $v.\mathrm{color} =$ grau, so ist (u, v) eine **Back-** oder **B-Kante**.
3. Ist $v.\mathrm{color} =$ schwarz und $v.d > u.d$, so ist (u, v) eine **Forward-** oder **F-Kante**.
4. Ist $v.\mathrm{color} =$ schwarz und $v.d < u.d$, so ist (u, v) eine **Cross-** oder **C-Kante**.



5.1.1 Tiefensuche

Lemma 5.3

In einem zusammenhängenden ungerichteten Graphen erzeugt die Tiefensuche nur T- und B-Kanten.

Lemma 5.3

In einem zusammenhängenden ungerichteten Graphen erzeugt die Tiefensuche nur T- und B-Kanten.

Beweis:

Es sei $\{u, v\}$ eine beliebige Kante des Graphen. O. B. d. A. gelte $u.d < v.d$.

5.1.1 Tiefensuche

Lemma 5.3

In einem zusammenhängenden ungerichteten Graphen erzeugt die Tiefensuche nur T- und B-Kanten.

Beweis:

Es sei $\{u, v\}$ eine beliebige Kante des Graphen. O. B. d. A. gelte $u.d < v.d$.

Bevor u komplett abgearbeitet ist, wird $\{u, v\}$ betrachtet. Passiert dies zuerst ausgehend von u, so ist v zu diesem Zeitpunkt noch weiß und $\{u, v\}$ wird eine T-Kante.

Lemma 5.3

In einem zusammenhängenden ungerichteten Graphen erzeugt die Tiefensuche nur T- und B-Kanten.

Beweis:

Es sei $\{u, v\}$ eine beliebige Kante des Graphen. O. B. d. A. gelte $u.d < v.d$.

Bevor u komplett abgearbeitet ist, wird $\{u, v\}$ betrachtet. Passiert dies zuerst ausgehend von u, so ist v zu diesem Zeitpunkt noch weiß und $\{u, v\}$ wird eine T-Kante.

Wird v von u aus nicht direkt über die Kante $\{u, v\}$, sondern über Zwischenknoten erreicht, so wird die Kante $\{u, v\}$ zuerst von v ausgehend betrachtet. Sie wird dann eine B-Kante, da u zu diesem Zeitpunkt noch grau ist. □

5.1.1 Tiefensuche

Theorem 5.4

Ein ungerichteter oder gerichteter Graph G ist genau dann kreisfrei, wenn bei der Tiefensuche keine B-Kante erzeugt wird.

5.1.1 Tiefensuche

Theorem 5.4

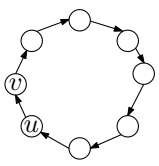
Ein ungerichteter oder gerichteter Graph G ist genau dann kreisfrei, wenn bei der Tiefensuche keine B-Kante erzeugt wird.

Beweis: Erzeugt DFS B-Kante (u, v), so ist v grau, wenn u erreicht wird. Gemäß Lemma 5.1 ist u DFS-Nachfolger von v. Also gibt es Weg von v nach u, der zusammen mit der Kante (u, v) einen Kreis bildet.

Theorem 5.4

Ein ungerichteter oder gerichteter Graph G ist genau dann kreisfrei, wenn bei der Tiefensuche keine B-Kante erzeugt wird.

Beweis: Erzeugt DFS B-Kante (u, v), so ist v grau, wenn u erreicht wird. Gemäß Lemma 5.1 ist u DFS-Nachfolger von v. Also gibt es Weg von v nach u, der zusammen mit der Kante (u, v) einen Kreis bildet.



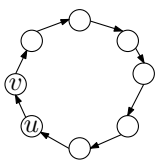
Annahme: G gerichtet.
Sei C ein Kreis und $v \in C$ der erste Knoten, der von DFS erreicht wird.
Sei u Vorgänger von v auf C, also $(u, v) \in C$.

5.1.1 Tiefensuche

Theorem 5.4

Ein ungerichteter oder gerichteter Graph G ist genau dann kreisfrei, wenn bei der Tiefensuche keine B-Kante erzeugt wird.

Beweis: Erzeugt DFS B-Kante (u, v), so ist v grau, wenn u erreicht wird. Gemäß Lemma 5.1 ist u DFS-Nachfolger von v. Also gibt es Weg von v nach u, der zusammen mit der Kante (u, v) einen Kreis bildet.

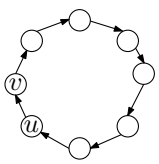


Annahme: G gerichtet.
Sei C ein Kreis und $v \in C$ der erste Knoten, der von DFS erreicht wird.
Sei u Vorgänger von v auf C, also $(u, v) \in C$.
Wenn v erreicht wird, sind alle Knoten auf C noch weiß.

Theorem 5.4

Ein ungerichteter oder gerichteter Graph G ist genau dann kreisfrei, wenn bei der Tiefensuche keine B-Kante erzeugt wird.

Beweis: Erzeugt DFS B-Kante (u, v), so ist v grau, wenn u erreicht wird. Gemäß Lemma 5.1 ist u DFS-Nachfolger von v. Also gibt es Weg von v nach u, der zusammen mit der Kante (u, v) einen Kreis bildet.

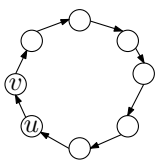


Annahme: G gerichtet.
Sei C ein Kreis und $v \in C$ der erste Knoten, der von DFS erreicht wird.
Sei u Vorgänger von v auf C, also $(u, v) \in C$.
Wenn v erreicht wird, sind alle Knoten auf C noch weiß.
Alle Knoten aus C werden besucht, bevor v abgearbeitet ist.

Theorem 5.4

Ein ungerichteter oder gerichteter Graph G ist genau dann kreisfrei, wenn bei der Tiefensuche keine B-Kante erzeugt wird.

Beweis: Erzeugt DFS B-Kante (u, v), so ist v grau, wenn u erreicht wird. Gemäß Lemma 5.1 ist u DFS-Nachfolger von v. Also gibt es Weg von v nach u, der zusammen mit der Kante (u, v) einen Kreis bildet.



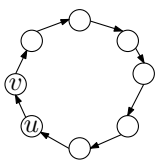
Annahme: G gerichtet.
Sei C ein Kreis und $v \in C$ der erste Knoten, der von DFS erreicht wird.
Sei u Vorgänger von v auf C, also $(u, v) \in C$.
Wenn v erreicht wird, sind alle Knoten auf C noch weiß.
Alle Knoten aus C werden besucht, bevor v abgearbeitet ist.
Somit wird u besucht, während v grau ist.

5.1.1 Tiefensuche

Theorem 5.4

Ein ungerichteter oder gerichteter Graph G ist genau dann kreisfrei, wenn bei der Tiefensuche keine B-Kante erzeugt wird.

Beweis: Erzeugt DFS B-Kante (u, v), so ist v grau, wenn u erreicht wird. Gemäß Lemma 5.1 ist u DFS-Nachfolger von v. Also gibt es Weg von v nach u, der zusammen mit der Kante (u, v) einen Kreis bildet.



Annahme: G gerichtet.
Sei C ein Kreis und $v \in C$ der erste Knoten, der von DFS erreicht wird.
Sei u Vorgänger von v auf C, also $(u, v) \in C$.
Wenn v erreicht wird, sind alle Knoten auf C noch weiß.
Alle Knoten aus C werden besucht, bevor v abgearbeitet ist.
Somit wird u besucht, während v grau ist.
Von u aus wird die Kante (u, v) betrachtet und als B-Kante markiert. □

5.1.1 Tiefensuche

$G = (V, E)$ ungerichtet: $v \rightsquigarrow u :\iff$ Es gibt Weg von v nach u in G.

Äquivalenzklassen von $\rightsquigarrow$ heißen **Zusammenhangskomponenten** von G.

G heißt **zusammenhängend** wenn er nur eine Zusammenhangskomponente besitzt.

Theorem 5.5

In einem ungerichteten Graphen bilden die T-Kanten einen Wald, dessen Zusammenhangskomponenten genau die Zusammenhangskomponenten des Graphen sind.

5.1.1 Tiefensuche

$G = (V, E)$ ungerichtet: $v \rightsquigarrow u :\iff$ Es gibt Weg von v nach u in G.

Äquivalenzklassen von $\rightsquigarrow$ heißen **Zusammenhangskomponenten** von G.

G heißt **zusammenhängend** wenn er nur eine Zusammenhangskomponente besitzt.

Theorem 5.5

In einem ungerichteten Graphen bilden die T-Kanten einen Wald, dessen Zusammenhangskomponenten genau die Zusammenhangskomponenten des Graphen sind.

Beweis: T-Kanten können keinen Kreis bilden.

5.1.1 Tiefensuche

$G = (V, E)$ ungerichtet: $v \rightsquigarrow u : \Longleftrightarrow$ Es gibt Weg von v nach u in G.

Äquivalenzklassen von $\rightsquigarrow$ heißen **Zusammenhangskomponenten** von G.

G heißt **zusammenhängend** wenn er nur eine Zusammenhangskomponente besitzt.

Theorem 5.5

In einem ungerichteten Graphen bilden die T-Kanten einen Wald, dessen Zusammenhangskomponenten genau die Zusammenhangskomponenten des Graphen sind.

Beweis: T-Kanten können keinen Kreis bilden.

Zwischen zwei Knoten aus derselben Zusammenhangskomponente gibt es Weg über T-Kanten: Sei eine beliebige Zusammenhangskomponenten fixiert und sei u der erste Knoten aus dieser Komponente, der von der Tiefensuche besucht wird. Wie im Beweis von Theorem 5.4 kann man argumentieren, dass jeder Knoten v, der von u aus erreichbar ist, ein DFS-Nachfolger von u wird. □

5.1.1 Tiefensuche

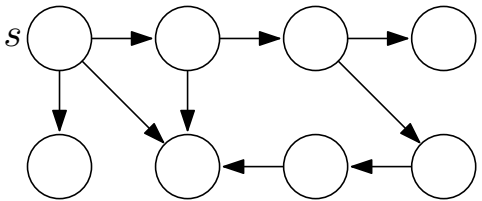
Theorem 5.6

Die Laufzeit von Tiefensuche auf einem Graphen $G = (V, E)$ beträgt $\Theta(|V| + |E|)$, wenn der Graph als Adjazenzliste gegeben ist.

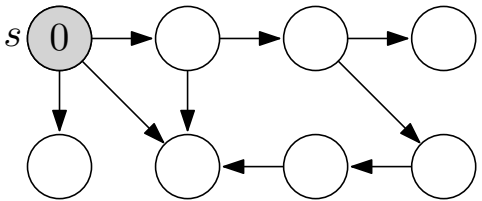
Breitensuche/Breadth-First Search (BFS)

```
BFS(G = (V, E),s)
 1   for each (u ∈ V \ {s}) {
 2       u.color = weiß; u.π = null; u.d = ∞;
 3   }
 4   s.color = grau; s.π = null; s.d = 0;
 5   Q = ∅;
 6   Q.enqueue(s);
 7   while (Q ≠ ∅) {
 8       u = Q.dequeue();
 9       for each ((u, v) ∈ E) {
10           if (v.color == weiß) {
11               v.color = grau; v.π = u; v.d = u.d + 1;
12               Q.enqueue(v);
13           }
14       }
15       u.color = schwarz;
16   }
```

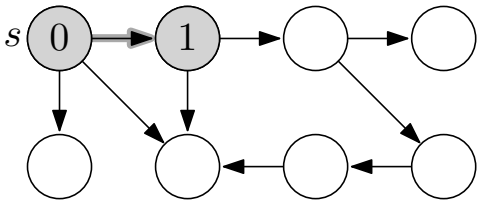
Beispiel für Breitensuche



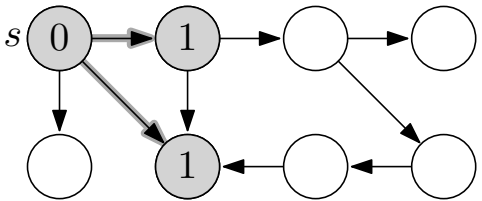
Beispiel für Breitensuche



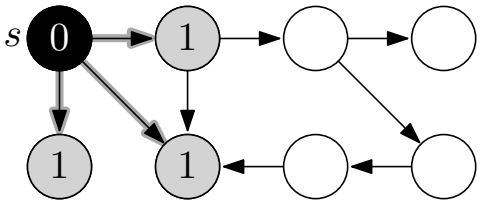
Beispiel für Breitensuche



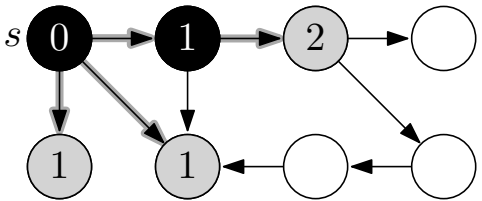
Beispiel für Breitensuche



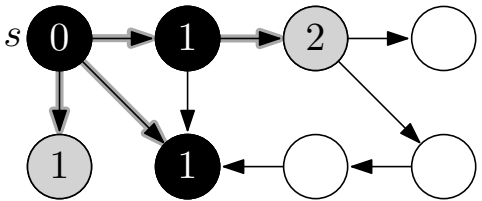
Beispiel für Breitensuche



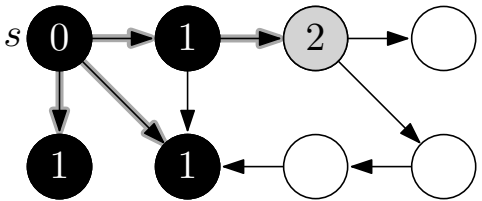
Beispiel für Breitensuche



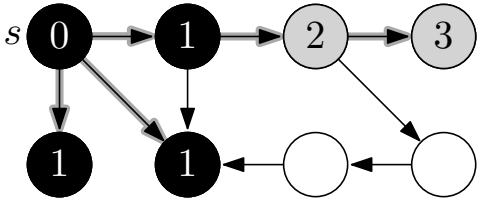
Beispiel für Breitensuche



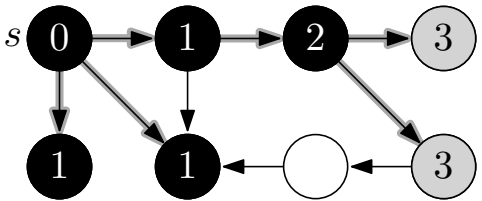
Beispiel für Breitensuche



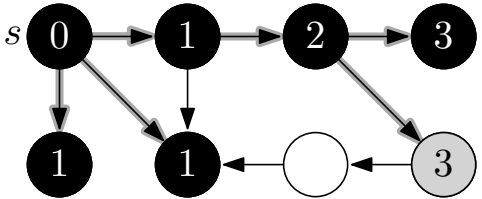
Beispiel für Breitensuche



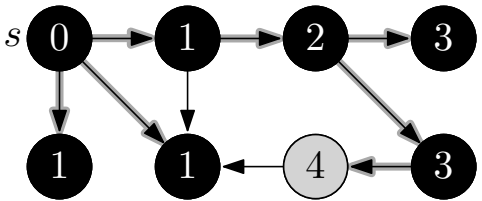
Beispiel für Breitensuche



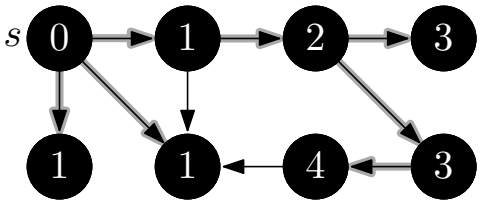
Beispiel für Breitensuche



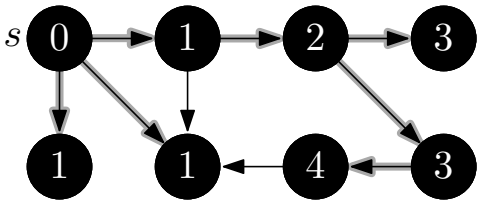
Beispiel für Breitensuche



Beispiel für Breitensuche



Beispiel für Breitensuche



Theorem 5.7

Die Laufzeit von Breitensuche auf einem Graphen $G = (V, E)$ beträgt $O(|V| + |E|)$, wenn der Graph als Adjazenzliste gegeben ist.

5.1.2 Breitensuche

Definition: $\delta(u, v) =$ Länge des kürzesten Weges von u nach v.

5.1.2 Breitensuche

Definition: $\delta(u, v) =$ Länge des kürzesten Weges von u nach v.

Theorem 5.8

Sei $G = (V, E)$ ein beliebiger Graph und sei $s \in V$ ein beliebiger Knoten. Nachdem die Breitensuche $\mathrm{BFS}(G, s)$ abgeschlossen ist, gilt $u.d = \delta(s, u)$ für jeden Knoten $u \in V$.

5.1.2 Breitensuche

Definition: $\delta(u, v) =$ Länge des kürzesten Weges von u nach v.

Theorem 5.8

Sei $G = (V, E)$ ein beliebiger Graph und sei $s \in V$ ein beliebiger Knoten. Nachdem die Breitensuche BFS(G, s) abgeschlossen ist, gilt $u.d = \delta(s, u)$ für jeden Knoten $u \in V$.

Für jeden Knoten $u \in V$ mit $\delta(s, u) < \infty$ kann ein kürzester Weg von s zu u rückwärts von u aus konstruiert werden, indem man vom aktuellen Knoten v stets zu seinem Vorgänger $v.\pi$ geht.

5.1.2 Breitensuche

Lemma 5.9

Es sei $G = (V, E)$ ein beliebiger Graph und es sei $s \in V$ ein beliebiger Knoten. Für jede Kante $(u, v) \in E$ gilt $\delta(s, v) \leq \delta(s, u) + 1$.

5.1.2 Breitensuche

Lemma 5.9

Es sei $G = (V, E)$ ein beliebiger Graph und es sei $s \in V$ ein beliebiger Knoten. Für jede Kante $(u, v) \in E$ gilt $\delta(s, v) \leq \delta(s, u) + 1$. Ist die Kante $(u, v) \in E$ in einem kürzesten Weg von s nach v enthalten, so gilt sogar $\delta(s, v) = \delta(s, u) + 1$.

Lemma 5.9

Es sei $G = (V, E)$ ein beliebiger Graph und es sei $s \in V$ ein beliebiger Knoten. Für jede Kante $(u, v) \in E$ gilt $\delta(s, v) \leq \delta(s, u) + 1$. Ist die Kante $(u, v) \in E$ in einem kürzesten Weg von s nach v enthalten, so gilt sogar $\delta(s, v) = \delta(s, u) + 1$.

Beweis:

Sei $\delta(s, u) < \infty$. Hänge an kürzesten s-u-Weg die Kante $(u, v) \in E$ an.
Dies ergibt s-v-Weg der Länge $\delta(s, u) + 1$.

Lemma 5.9

Es sei $G = (V, E)$ ein beliebiger Graph und es sei $s \in V$ ein beliebiger Knoten. Für jede Kante $(u, v) \in E$ gilt $\delta(s, v) \leq \delta(s, u) + 1$. Ist die Kante $(u, v) \in E$ in einem kürzesten Weg von s nach v enthalten, so gilt sogar $\delta(s, v) = \delta(s, u) + 1$.

Beweis:

Sei $\delta(s, u) < \infty$. Hänge an kürzesten s-u-Weg die Kante $(u, v) \in E$ an.
Dies ergibt s-v-Weg der Länge $\delta(s, u) + 1$.

Sei P ein kürzester s-v-Weg, der die Kante (u, v) enthält.
Dann ist $P' = P \setminus \{(u, v)\}$ ein kürzester s-u-Weg.
Also $\delta(s, v) = \delta(s, u) + 1$. □

5.1.2 Breitensuche

Lemma 5.10

Sei $G = (V, E)$ ein beliebiger Graph und sei $s \in V$ ein beliebiger Knoten. Nach Ausführung von Zeile 3 in BFS(G, s), gilt $u.d \geq \delta(s, u)$ für jeden Knoten $u \in V$.

5.1.2 Breitensuche

Lemma 5.10

Sei $G = (V, E)$ ein beliebiger Graph und sei $s \in V$ ein beliebiger Knoten. Nach Ausführung von Zeile 3 in BFS(G, s), gilt $u.d \geq \delta(s, u)$ für jeden Knoten $u \in V$.

Beweis:

Induktion über die Anzahl an Knoten, die bislang in die Queue eingefügt wurden:

Lemma 5.10

Sei $G = (V, E)$ ein beliebiger Graph und sei $s \in V$ ein beliebiger Knoten. Nach Ausführung von Zeile 3 in BFS(G, s), gilt $u.d \geq \delta(s, u)$ für jeden Knoten $u \in V$.

Beweis:

Induktion über die Anzahl an Knoten, die bislang in die Queue eingefügt wurden:

Induktionsanfang: Nach Einfügen von s: $s.d = \delta(s, s) = 0$;

$\forall u \in V \setminus \{s\} : u.d = \infty \geq \delta(s, u)$.

Lemma 5.10

Sei $G = (V, E)$ ein beliebiger Graph und sei $s \in V$ ein beliebiger Knoten. Nach Ausführung von Zeile 3 in BFS(G, s), gilt $u.d \geq \delta(s, u)$ für jeden Knoten $u \in V$.

Beweis:

Induktion über die Anzahl an Knoten, die bislang in die Queue eingefügt wurden:

Induktionsanfang: Nach Einfügen von s: $s.d = \delta(s, s) = 0$;
$\forall u \in V \setminus \{s\} : u.d = \infty \geq \delta(s, u)$.

Induktionsschritt: Während der Bearbeitung von u wird v der Queue hinzugefügt.

Lemma 5.10

Sei $G = (V, E)$ ein beliebiger Graph und sei $s \in V$ ein beliebiger Knoten. Nach Ausführung von Zeile 3 in BFS(G, s), gilt $u.d \geq \delta(s, u)$ für jeden Knoten $u \in V$.

Beweis:

Induktion über die Anzahl an Knoten, die bislang in die Queue eingefügt wurden:

Induktionsanfang: Nach Einfügen von s: $s.d = \delta(s, s) = 0$;
$\forall u \in V \setminus \{s\} : u.d = \infty \geq \delta(s, u)$.

Induktionsschritt: Während der Bearbeitung von u wird v der Queue hinzugefügt.
Dann $(u, v) \in E$ und es wird $v.d = u.d + 1$ gesetzt.

Lemma 5.10

Sei $G = (V, E)$ ein beliebiger Graph und sei $s \in V$ ein beliebiger Knoten. Nach Ausführung von Zeile 3 in BFS(G, s), gilt $u.d \geq \delta(s, u)$ für jeden Knoten $u \in V$.

Beweis:

Induktion über die Anzahl an Knoten, die bislang in die Queue eingefügt wurden:

Induktionsanfang: Nach Einfügen von s: $s.d = \delta(s, s) = 0$;
$\forall u \in V \setminus \{s\} : u.d = \infty \geq \delta(s, u)$.

Induktionsschritt: Während der Bearbeitung von u wird v der Queue hinzugefügt.
Dann $(u, v) \in E$ und es wird $v.d = u.d + 1$ gesetzt.
Lemma 5.9 besagt $\delta(s, v) \leq \delta(s, u) + 1$.

Lemma 5.10

Sei $G = (V, E)$ ein beliebiger Graph und sei $s \in V$ ein beliebiger Knoten. Nach Ausführung von Zeile 3 in BFS(G, s), gilt $u.d \geq \delta(s, u)$ für jeden Knoten $u \in V$.

Beweis:

Induktion über die Anzahl an Knoten, die bislang in die Queue eingefügt wurden:

Induktionsanfang: Nach Einfügen von s: $s.d = \delta(s, s) = 0$;

$\forall u \in V \setminus \{s\} : u.d = \infty \geq \delta(s, u)$.

Induktionsschritt: Während der Bearbeitung von u wird v der Queue hinzugefügt.

Dann $(u, v) \in E$ und es wird $v.d = u.d + 1$ gesetzt.

Lemma 5.9 besagt $\delta(s, v) \leq \delta(s, u) + 1$.

Induktionsannahme impliziert $u.d \geq \delta(s, u)$.

Lemma 5.10

Sei $G = (V, E)$ ein beliebiger Graph und sei $s \in V$ ein beliebiger Knoten. Nach Ausführung von Zeile 3 in BFS(G, s), gilt $u.d \geq \delta(s, u)$ für jeden Knoten $u \in V$.

Beweis:

Induktion über die Anzahl an Knoten, die bislang in die Queue eingefügt wurden:

Induktionsanfang: Nach Einfügen von s: $s.d = \delta(s, s) = 0$;
$\forall u \in V \setminus \{s\} : u.d = \infty \geq \delta(s, u)$.

Induktionsschritt: Während der Bearbeitung von u wird v der Queue hinzugefügt.
Dann $(u, v) \in E$ und es wird $v.d = u.d + 1$ gesetzt.
Lemma 5.9 besagt $\delta(s, v) \leq \delta(s, u) + 1$.
Induktionsannahme impliziert $u.d \geq \delta(s, u)$.

$$\Rightarrow v.d = u.d + 1 \geq \delta(s, u) + 1 \geq \delta(s, v). \quad \square$$

5.1.2 Breitensuche

Lemma 5.11

Enthält die Queue Q während der Ausführung von $\mathrm{BFS}(G, s)$ zu einem Zeitpunkt die Knoten $v_1, \ldots, v_r$ in dieser Reihenfolge (wobei v_1 von diesen Elementen das erste ist, das eingefügt wurde), so gilt $v_i.d \leq v_{i+1}.d$ für alle $i \in \{1, \ldots, r-1\}$ und $v_r.d \leq v_1.d + 1$.

Lemma 5.11

Enthält die Queue Q während der Ausführung von BFS(G, s) zu einem Zeitpunkt die Knoten $v_1, \ldots, v_r$ in dieser Reihenfolge (wobei v_1 von diesen Elementen das erste ist, das eingefügt wurde), so gilt $v_i.d \leq v_{i+1}.d$ für alle $i \in \{1, \ldots, r-1\}$ und $v_r.d \leq v_1.d + 1$.

Beweis:

Induktion über die Anzahl an dequeue-Operationen:

Lemma 5.11

Enthält die Queue Q während der Ausführung von BFS(G, s) zu einem Zeitpunkt die Knoten $v_1, \ldots, v_r$ in dieser Reihenfolge (wobei v_1 von diesen Elementen das erste ist, das eingefügt wurde), so gilt $v_i.d \leq v_{i+1}.d$ für alle $i \in \{1, \ldots, r-1\}$ und $v_r.d \leq v_1.d + 1$.

Beweis:

Induktion über die Anzahl an dequeue-Operationen:

Induktionsanfang: Trivial, da nur s in der Queue.

5.1.2 Breitensuche

Lemma 5.11

Enthält die Queue Q während der Ausführung von BFS(G, s) zu einem Zeitpunkt die Knoten $v_1, \ldots, v_r$ in dieser Reihenfolge (wobei v_1 von diesen Elementen das erste ist, das eingefügt wurde), so gilt $v_i.d \leq v_{i+1}.d$ für alle $i \in \{1, \ldots, r-1\}$ und $v_r.d \leq v_1.d + 1$.

Beweis:

Induktion über die Anzahl an dequeue-Operationen:

Induktionsanfang: Trivial, da nur s in der Queue.

Induktionsschritt: Betrachte eine dequeue-Operation mit der Knoten v_1 aus der Queue entfernt wird. Danach ist v_2 der neue erste Knoten in der Queue und die Aussagen des Lemmas sind noch immer erfüllt, denn es gilt $v_r.d \leq v_1.d + 1 \leq v_2.d + 1$.

Lemma 5.11

Enthält die Queue Q während der Ausführung von BFS(G, s) zu einem Zeitpunkt die Knoten $v_1, \ldots, v_r$ in dieser Reihenfolge (wobei v_1 von diesen Elementen das erste ist, das eingefügt wurde), so gilt $v_i.d \leq v_{i+1}.d$ für alle $i \in \{1, \ldots, r-1\}$ und $v_r.d \leq v_1.d + 1$.

Beweis:

Induktion über die Anzahl an dequeue-Operationen:

Induktionsanfang: Trivial, da nur s in der Queue.

Induktionsschritt: Betrachte eine dequeue-Operation mit der Knoten v_1 aus der Queue entfernt wird. Danach ist v_2 der neue erste Knoten in der Queue und die Aussagen des Lemmas sind noch immer erfüllt, denn es gilt $v_r.d \leq v_1.d + 1 \leq v_2.d + 1$.

Zu $u = v_1$ adjazente Knoten $v_{r+1}, \ldots, v_{r+\ell}$ werden an das Ende der Queue angefügt. Es gilt $v_{r+1}.d = \ldots = v_{r+\ell}.d = v_1.d + 1$.

Lemma 5.11

Enthält die Queue Q während der Ausführung von BFS(G, s) zu einem Zeitpunkt die Knoten $v_1, \ldots, v_r$ in dieser Reihenfolge (wobei v_1 von diesen Elementen das erste ist, das eingefügt wurde), so gilt $v_i.d \leq v_{i+1}.d$ für alle $i \in \{1, \ldots, r-1\}$ und $v_r.d \leq v_1.d + 1$.

Beweis:

Induktion über die Anzahl an dequeue-Operationen:

Induktionsanfang: Trivial, da nur s in der Queue.

Induktionsschritt: Betrachte eine dequeue-Operation mit der Knoten v_1 aus der Queue entfernt wird. Danach ist v_2 der neue erste Knoten in der Queue und die Aussagen des Lemmas sind noch immer erfüllt, denn es gilt $v_r.d \leq v_1.d + 1 \leq v_2.d + 1$.

Zu $u = v_1$ adjazente Knoten $v_{r+1}, \ldots, v_{r+\ell}$ werden an das Ende der Queue angefügt. Es gilt $v_{r+1}.d = \ldots = v_{r+\ell}.d = v_1.d + 1$. Also $v_{r+i}.d = v_1.d + 1 \leq v_2.d + 1$ für alle i.

Lemma 5.11

Enthält die Queue Q während der Ausführung von BFS(G, s) zu einem Zeitpunkt die Knoten $v_1, \ldots, v_r$ in dieser Reihenfolge (wobei v_1 von diesen Elementen das erste ist, das eingefügt wurde), so gilt $v_i.d \leq v_{i+1}.d$ für alle $i \in \{1, \ldots, r-1\}$ und $v_r.d \leq v_1.d + 1$.

Beweis:

Induktion über die Anzahl an dequeue-Operationen:

Induktionsanfang: Trivial, da nur s in der Queue.

Induktionsschritt: Betrachte eine dequeue-Operation mit der Knoten v_1 aus der Queue entfernt wird. Danach ist v_2 der neue erste Knoten in der Queue und die Aussagen des Lemmas sind noch immer erfüllt, denn es gilt $v_r.d \leq v_1.d + 1 \leq v_2.d + 1$.

Zu $u = v_1$ adjazente Knoten $v_{r+1}, \ldots, v_{r+\ell}$ werden an das Ende der Queue angefügt. Es gilt $v_{r+1}.d = \ldots = v_{r+\ell}.d = v_1.d + 1$. Also $v_{r+i}.d = v_1.d + 1 \leq v_2.d + 1$ für alle i. Ferner gilt aufgrund der Induktionsannahme $v_r.d \leq v_1.d + 1 = v_{r+i}.d$. □

Beweis von Theorem 5.8: Annahme: Es gibt Knoten $u \in V$ mit $u.d \neq \delta(s, u)$.

Beweis von Theorem 5.8: Annahme: Es gibt Knoten $u \in V$ mit $u.d \neq \delta(s, u)$.

Sei P kürzester s-u-Weg und sei u' erster solcher Knoten entlang P.

Beweis von Theorem 5.8: Annahme: Es gibt Knoten $u \in V$ mit $u.d \neq \delta(s, u)$.

Sei P kürzester s-u-Weg und sei u' erster solcher Knoten entlang P.

Sei w Vorgänger von u' auf P. Nach Wahl von u' gilt $w.d = \delta(s, w)$.

Beweis von Theorem 5.8: Annahme: Es gibt Knoten $u \in V$ mit $u.d \neq \delta(s, u)$.

Sei P kürzester s-u-Weg und sei u' erster solcher Knoten entlang P.

Sei w Vorgänger von u' auf P. Nach Wahl von u' gilt $w.d = \delta(s, w)$.

Gemäß Lemma 5.9 und Lemma 5.10 gilt $u'.d > \delta(s, u') = \delta(s, w) + 1$.

5.1.2 Breitensuche

Beweis von Theorem 5.8: Annahme: Es gibt Knoten $u \in V$ mit $u.d \neq \delta(s, u)$.

Sei P kürzester s-u-Weg und sei u' erster solcher Knoten entlang P.

Sei w Vorgänger von u' auf P. Nach Wahl von u' gilt $w.d = \delta(s, w)$.

Gemäß Lemma 5.9 und Lemma 5.10 gilt $u'.d > \delta(s, u') = \delta(s, w) + 1$.

Betrachte Farbe von u', zum Zeitpunkt, dass w entfernt wird:

1) u' ist grau oder schwarz: dann ist u' von einem Knoten x vorher erreicht worden.

5.1.2 Breitensuche

Beweis von Theorem 5.8: Annahme: Es gibt Knoten $u \in V$ mit $u.d \neq \delta(s, u)$.

Sei P kürzester s-u-Weg und sei u' erster solcher Knoten entlang P.

Sei w Vorgänger von u' auf P. Nach Wahl von u' gilt $w.d = \delta(s, w)$.

Gemäß Lemma 5.9 und Lemma 5.10 gilt $u'.d > \delta(s, u') = \delta(s, w) + 1$.

Betrachte Farbe von u', zum Zeitpunkt, dass w entfernt wird:

1) u' ist grau oder schwarz: dann ist u' von einem Knoten x vorher erreicht worden.

Vor w nur Knoten x mit $x.d \leq w.d = \delta(s, w)$ aus der Queue entfernt (Lemma 5.11).

Beweis von Theorem 5.8: Annahme: Es gibt Knoten $u \in V$ mit $u.d \neq \delta(s,u)$.

Sei P kürzester s-u-Weg und sei u' erster solcher Knoten entlang P.

Sei w Vorgänger von u' auf P. Nach Wahl von u' gilt $w.d = \delta(s,w)$.

Gemäß Lemma 5.9 und Lemma 5.10 gilt $u'.d > \delta(s,u') = \delta(s,w) + 1$.

Betrachte Farbe von u', zum Zeitpunkt, dass w entfernt wird:

1) u' ist grau oder schwarz: dann ist u' von einem Knoten x vorher erreicht worden.

Vor w nur Knoten x mit $x.d \leq w.d = \delta(s,w)$ aus der Queue entfernt (Lemma 5.11).

$$u'.d \leq x.d + 1 \leq w.d + 1 = \delta(s,w) + 1 = \delta(s,u')$$

Beweis von Theorem 5.8: Annahme: Es gibt Knoten $u \in V$ mit $u.d \neq \delta(s,u)$.

Sei P kürzester s-u-Weg und sei u' erster solcher Knoten entlang P.

Sei w Vorgänger von u' auf P. Nach Wahl von u' gilt $w.d = \delta(s,w)$.

Gemäß Lemma 5.9 und Lemma 5.10 gilt $u'.d > \delta(s,u') = \delta(s,w) + 1$.

Betrachte Farbe von u', zum Zeitpunkt, dass w entfernt wird:

1) u' ist grau oder schwarz: dann ist u' von einem Knoten x vorher erreicht worden.

Vor w nur Knoten x mit $x.d \leq w.d = \delta(s,w)$ aus der Queue entfernt (Lemma 5.11).

$$u'.d \leq x.d + 1 \leq w.d + 1 = \delta(s,w) + 1 = \delta(s,u')$$

Mit Lemma 5.10 folgt daraus $u'.d = \delta(s,u')$, was im Widerspruch zur Wahl von u' steht.

Beweis von Theorem 5.8: Annahme: Es gibt Knoten $u \in V$ mit $u.d \neq \delta(s,u)$.

Sei P kürzester s-u-Weg und sei u' erster solcher Knoten entlang P.

Sei w Vorgänger von u' auf P. Nach Wahl von u' gilt $w.d = \delta(s,w)$.

Gemäß Lemma 5.9 und Lemma 5.10 gilt $u'.d > \delta(s,u') = \delta(s,w) + 1$.

Betrachte Farbe von u', zum Zeitpunkt, dass w entfernt wird:

1) u' ist grau oder schwarz: dann ist u' von einem Knoten x vorher erreicht worden.

Vor w nur Knoten x mit $x.d \leq w.d = \delta(s,w)$ aus der Queue entfernt (Lemma 5.11).

$$u'.d \leq x.d + 1 \leq w.d + 1 = \delta(s,w) + 1 = \delta(s,u')$$

Mit Lemma 5.10 folgt daraus $u'.d = \delta(s,u')$, was im Widerspruch zur Wahl von u' steht.

2) u' ist weiss: dann wird u' bei Betrachtung von w in die Queue eingefügt:

$$u'.d = w.d + 1 = \delta(s,w) + 1 = \delta(s,u').$$

Beweis von Theorem 5.8: Annahme: Es gibt Knoten $u \in V$ mit $u.d \neq \delta(s,u)$.
Sei P kürzester s-u-Weg und sei u' erster solcher Knoten entlang P.
Sei w Vorgänger von u' auf P. Nach Wahl von u' gilt $w.d = \delta(s,w)$.
Gemäß Lemma 5.9 und Lemma 5.10 gilt $u'.d > \delta(s,u') = \delta(s,w) + 1$.
Betrachte Farbe von u', zum Zeitpunkt, dass w entfernt wird:
1) u' ist grau oder schwarz: dann ist u' von einem Knoten x vorher erreicht worden.
Vor w nur Knoten x mit $x.d \leq w.d = \delta(s,w)$ aus der Queue entfernt (Lemma 5.11).

$$u'.d \leq x.d + 1 \leq w.d + 1 = \delta(s,w) + 1 = \delta(s,u')$$

Mit Lemma 5.10 folgt daraus $u'.d = \delta(s,u')$, was im Widerspruch zur Wahl von u' steht.
2) u' ist weiss: dann wird u' bei Betrachtung von w in die Queue eingefügt:

$$u'.d = w.d + 1 = \delta(s,w) + 1 = \delta(s,u').$$

Dies ist ebenfalls Widerspruch zur Wahl von u'.

Beweis von Theorem 5.8:

Für jede Kante der Form $(v.\pi, v)$ gilt $v.d = v.\pi.d + 1$.

Beweis von Theorem 5.8:

Für jede Kante der Form $(v.\pi, v)$ gilt $v.d = v.\pi.d + 1$.

$\Rightarrow$ Folgen wir Kanten der Form $(v.\pi, v)$ von s zu $u \in V$, so erhalten wir Weg der Länge $u.d = \delta(s, u)$. $\square$