Algorithmen und Wahrscheinlichkeit

Rasmus Kyng, Angelika Steger, Emo Welzl

Institut für Theoretische Informatik

Kapitel 1.3

Zusammenhang

Zusammenhang

Definition: Sei $G = (V, E)$ ein Graph.

G heisst zusammenhängend, wenn

$\forall u, v \in V, u \neq v$ gilt: es gibt einen u-v-Pfad in G.

Heute:

Gegeben ein zusammenhängender Graph.

Wie (sehr) zusammenhängend ist dieser Graph?

Separatoren und k-Zusammenhang

Definition: Sei $G = (V, E)$ ein Graph, $u, v \in V$ und $X \subseteq V \setminus \{u, v\}$.

X heisst u-v-Separator, wenn u und v in verschiedenen Zusammenhangskomponenten von $G[V \setminus X]$ liegen.

G heisst k-zusammenhängend, wenn gilt:

- $|V| \geq k + 1$ und
- $\forall u, v \in V$: Jeder u-v-Separator X hat Grösse $|X| \geq k$.

Definition: Sei $G = (V, E)$ ein Graph, $u, v \in V$ und $X \subseteq E$.

X heisst u-v-Kanten-Separator, wenn u und v in verschiedenen Zusammenhangskomponenten von $G' = (V, E \setminus X)$ liegen.

G heisst k-kanten-zusammenhängend, wenn gilt:

- $\forall u, v \in V$: Jeder u-v-Kantenseparator X hat Grösse $|X| \geq k$.

Heute:

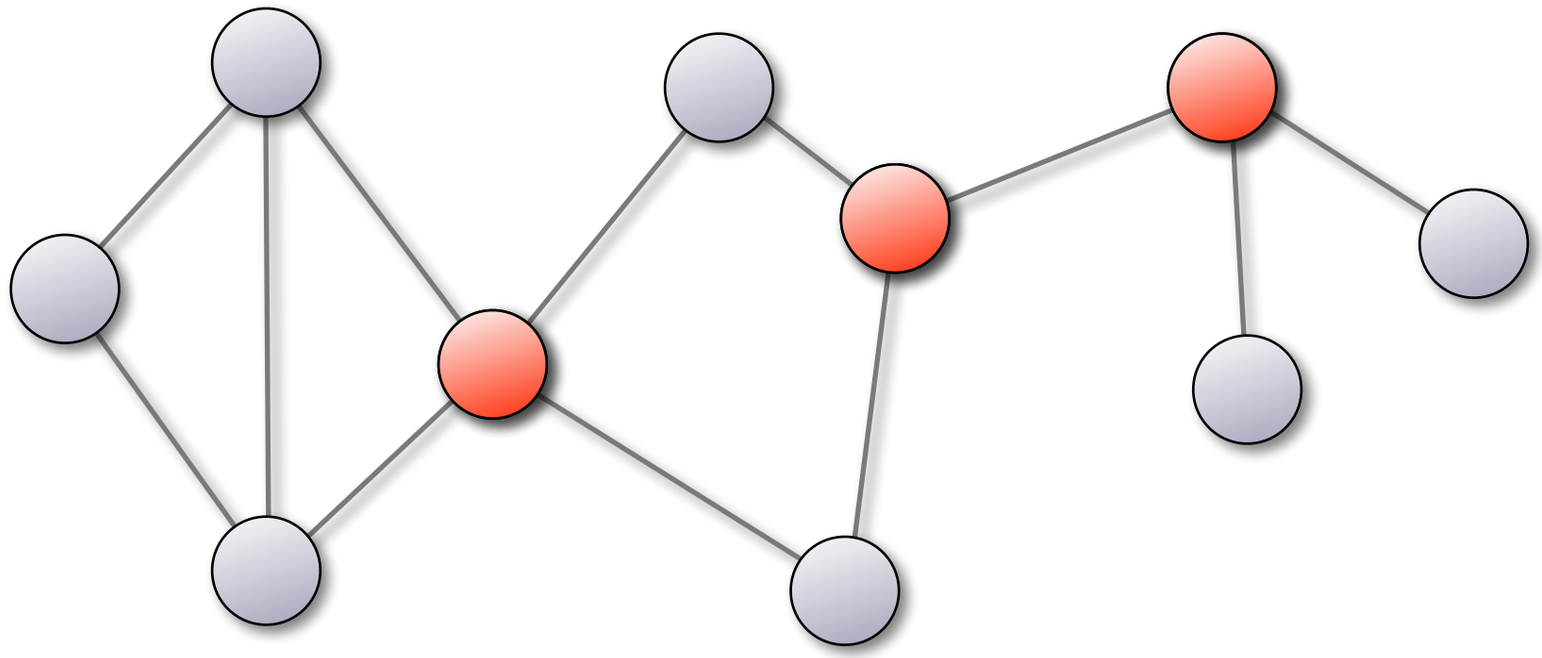
Spezialfall k=1

Artikulationsknoten

Definition: Sei $G = (V, E)$ ein zusammenhängender Graph.

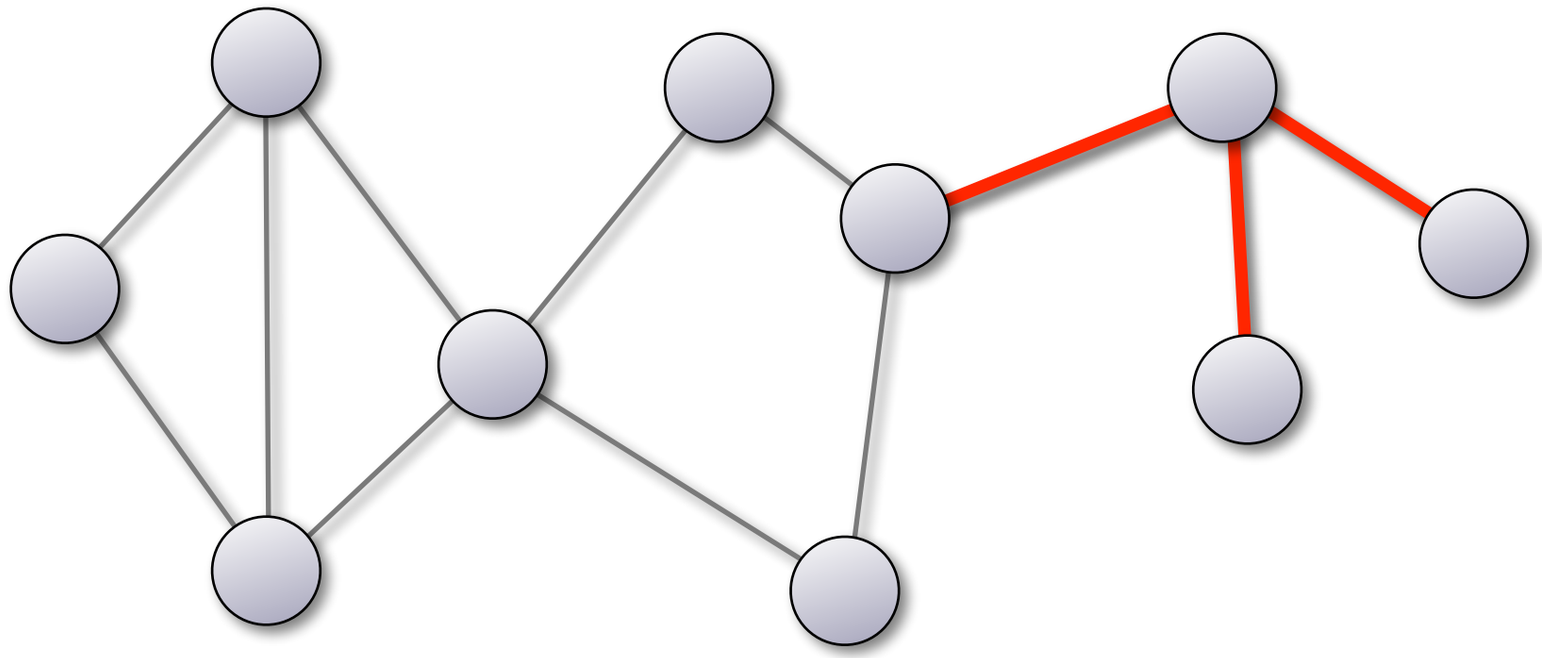
Ein Knoten $v \in V$ heisst *Artikulationsknoten* (engl. cut vertex)

gdw. $G[V \setminus \{v\}]$ nicht zusammenhängend ist



Brücken

Definition: Sei $G = (V, E)$ ein zusammenhängender Graph.
Ein Kante $e \in E$ heisst *Brücke* (engl. cut edge)
gdw. $G - e$ nicht zusammenhängend ist



Wie findet man Artikulationsknoten
bzw Brücken effizient ?

Exkurs

(Wdh aus dem letzten Semester)

BFS-Visit-Iterative(G, v)

```
1  Q ← ∅
2  Markiere v als aktiv
3  Enqueue(Q, v)
4  while Q ≠ ∅ do
5      w ← Dequeue(Q)
6      Markiere w als besucht
7      for each (w, x) ∈ E do
8          if x nicht aktiv und x noch nicht besucht then
9              Markiere x als aktiv
10             Enqueue(Q, x)
```

DFS-Visit-Iterative(G, v)

```
1  S ← ∅
2  Push(S, v)
3  while S ≠ ∅ do
4      w ← Pop(S)
5      if w noch nicht besucht then
6          Markiere w als besucht
7          for each (w, x) ∈ E in reverse order do
8              if x noch nicht besucht then
9                  Push(S, x)
```

Breiten- und Tiefensuche

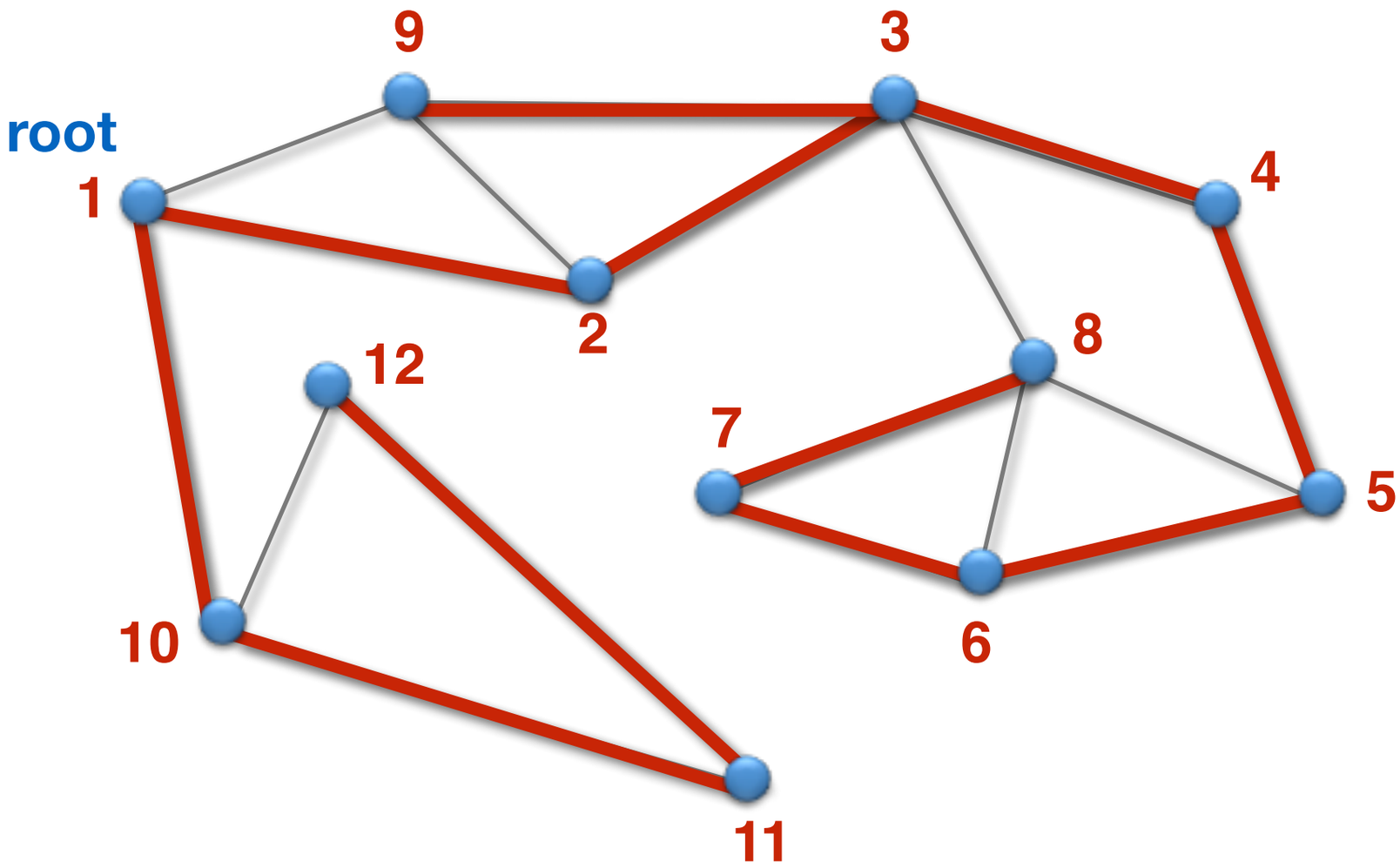
BFS-Visit-Iterative(G, v)

```
1  Q ← ∅
2  Markiere v als aktiv
3  Enqueue(Q, v)
4  while Q ≠ ∅ do
5      w ← Dequeue(Q)
6      Markiere w als besucht
7      for each (w, x) ∈ E do
8          if x nicht aktiv und x noch nicht besucht then
9              Markiere x als aktiv
10             Enqueue(Q, x)
```

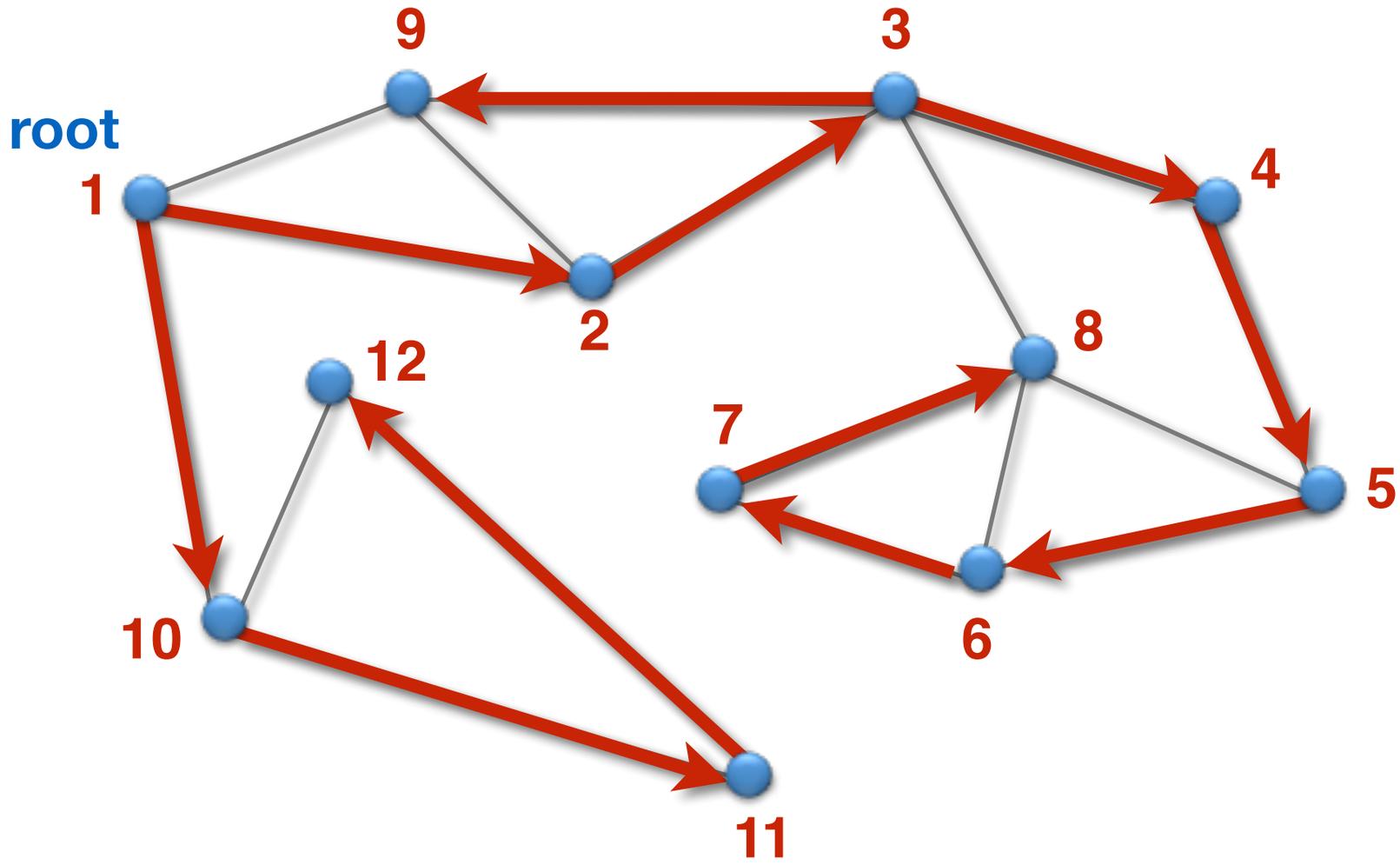
DFS-Visit-Iterative(G, v)

```
1  S ← ∅
2  Push(S, v)
3  while S ≠ ∅ do
4      w ← Pop(S)
5      if w noch nicht besucht then
6          Markiere w als besucht
7          for each (w, x) ∈ E in reverse order do
8              if x noch nicht besucht then
9                  Push(S, x)
```

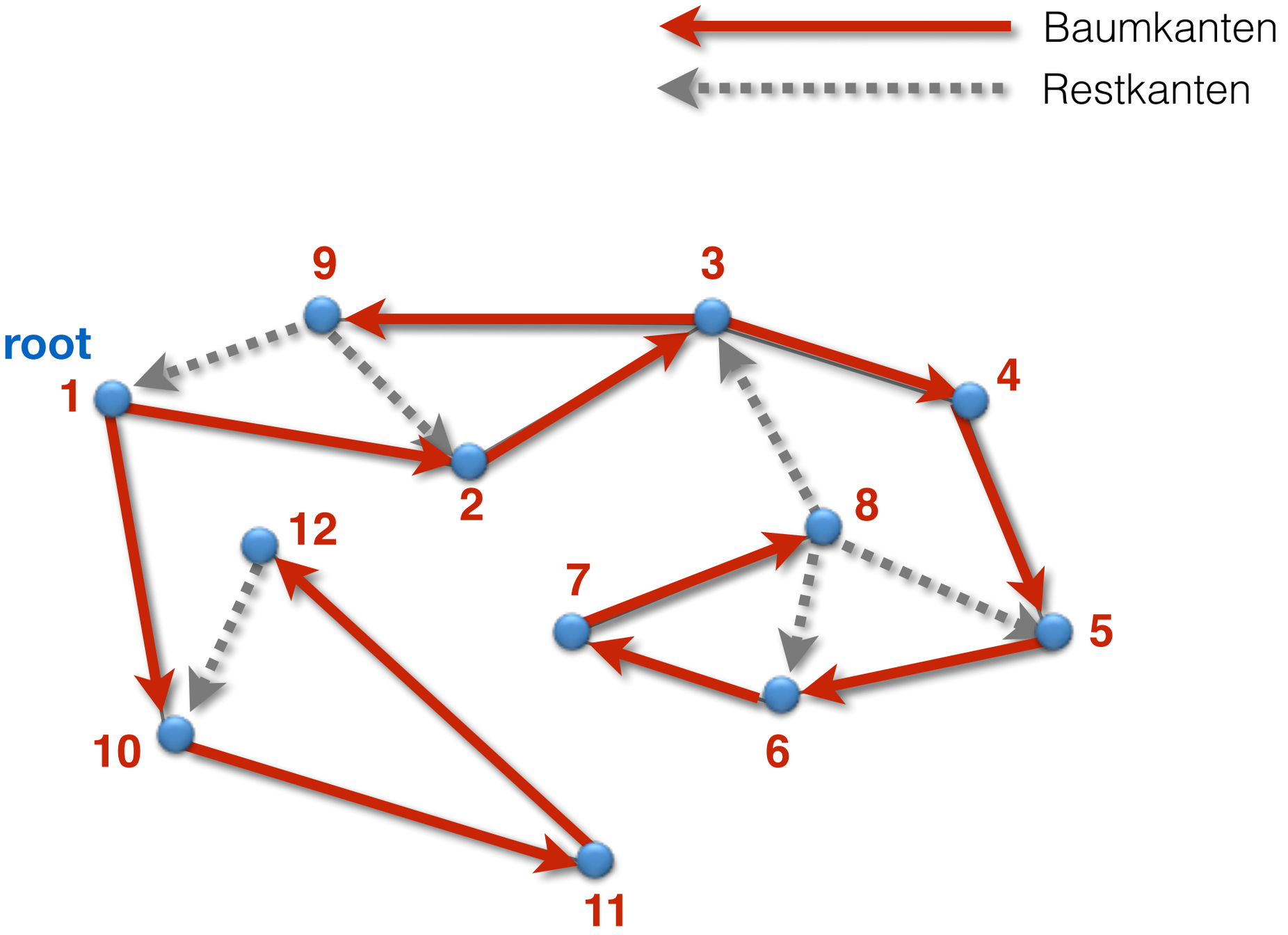
Tiefensuche



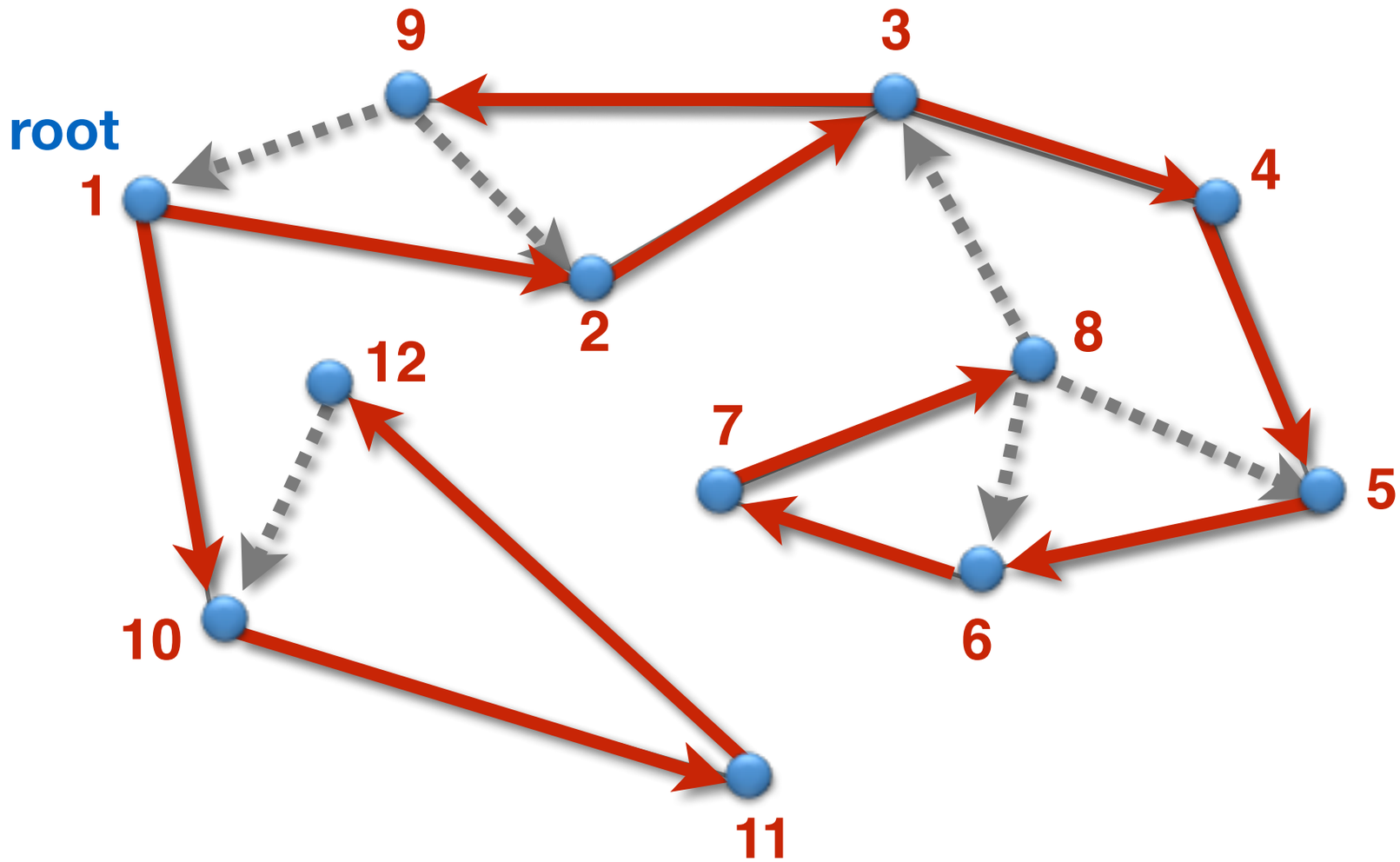
Tiefensuche



Tiefensuche

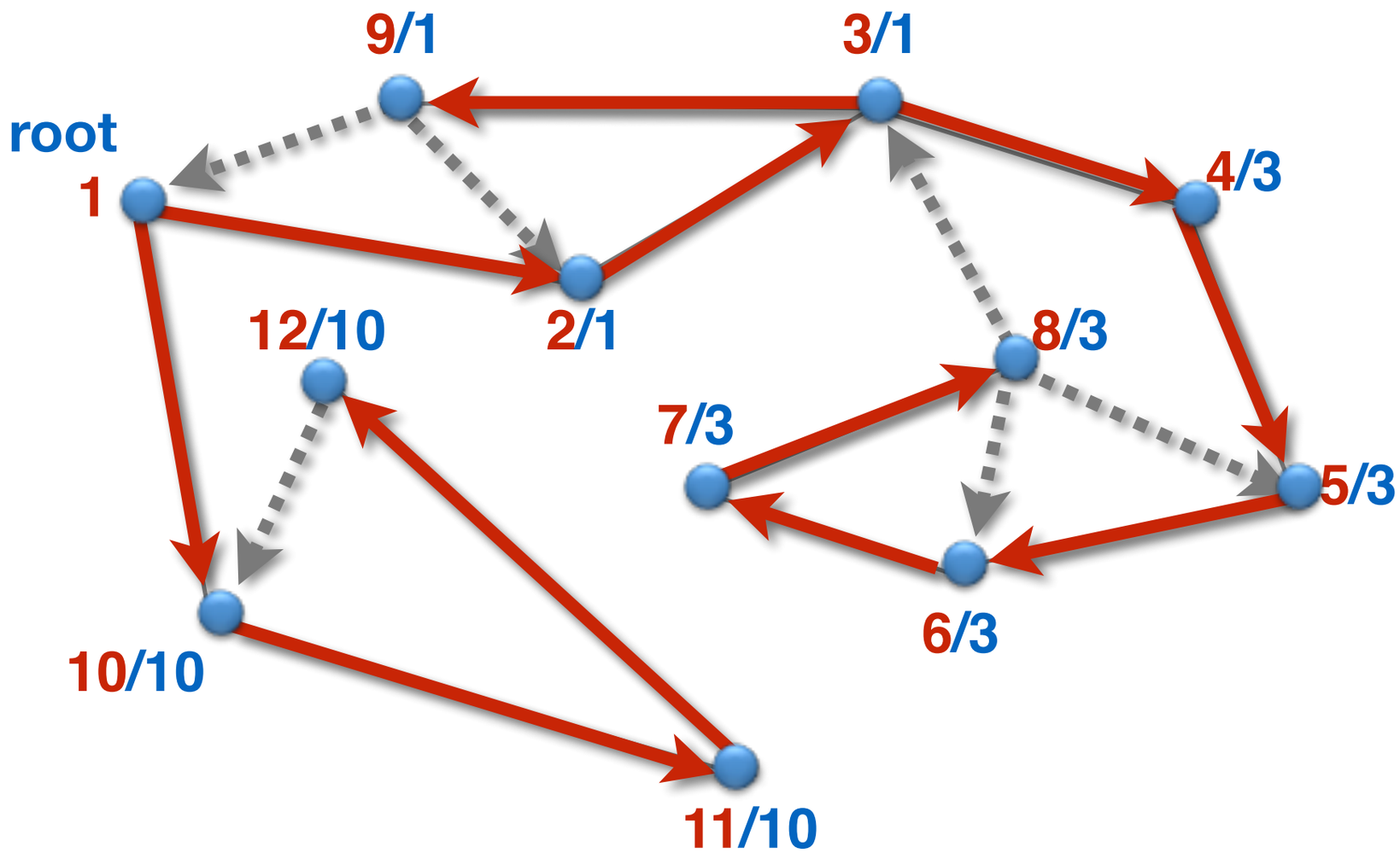


Tiefensuche



$\text{low}[v]$:= kleinste dfs-Nummer, die man von v aus durch einen gerichteten Pfad aus (beliebig vielen) Baumkanten und maximal einer Restkante erreichen kann.

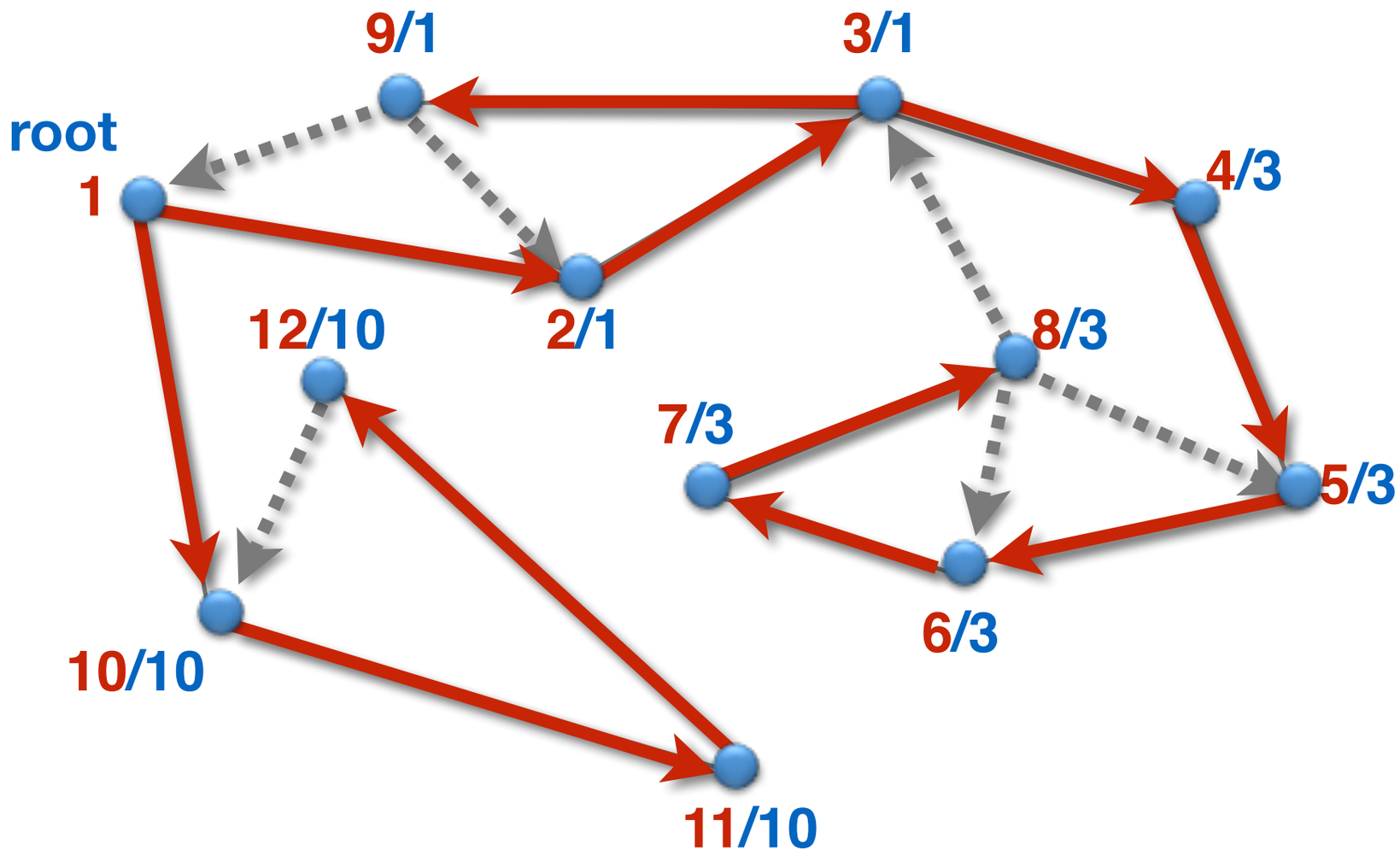
Tiefensuche



$\text{low}[v]$:= kleinste dfs-Nummer, die man von v aus durch einen gerichteten Pfad aus (beliebig vielen) Baumkanten und maximal einer Restkante erreichen kann.

Tiefensuche

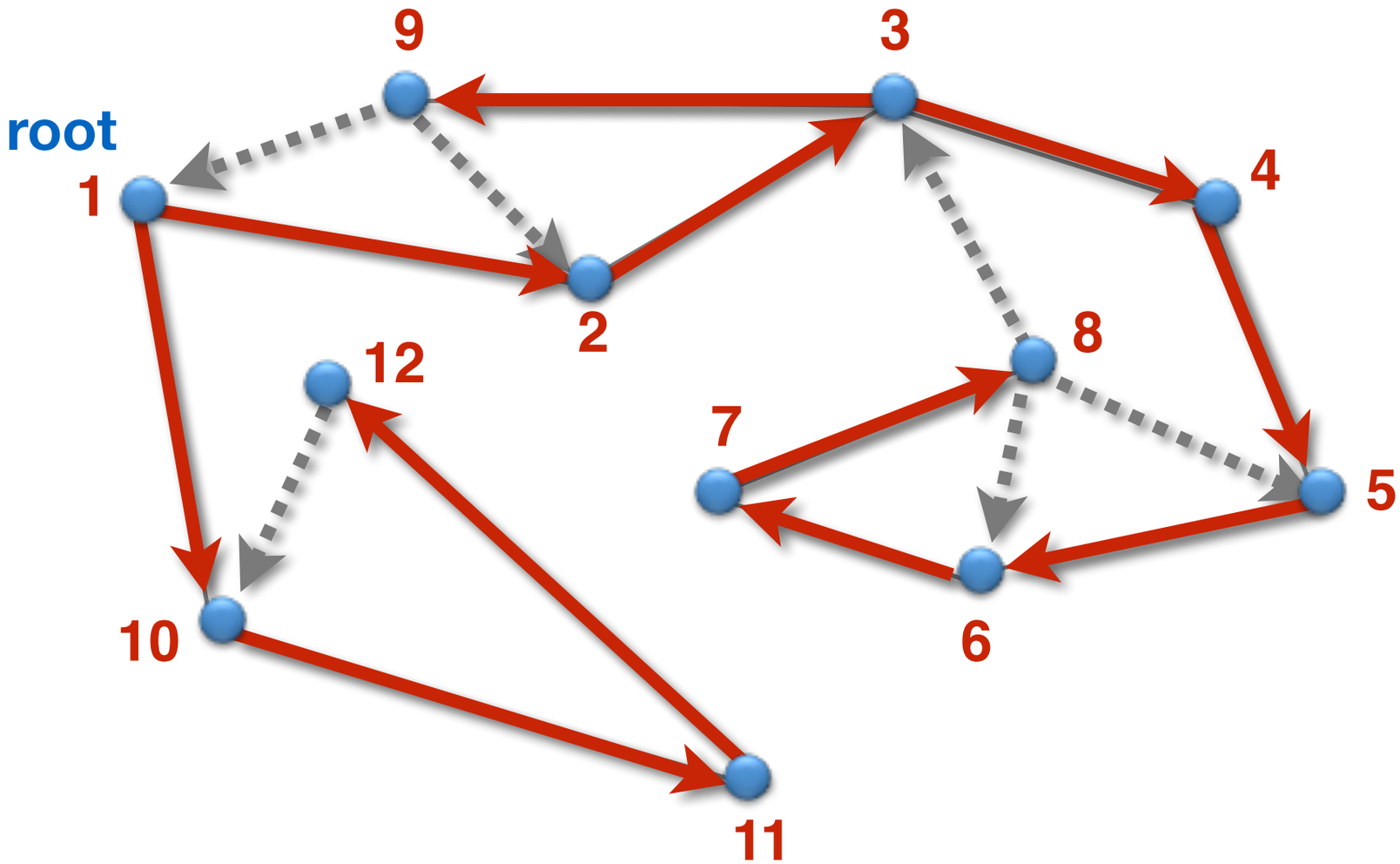
dfs / low



Alle low-Werte sind mit DP in Zeit O(|V|+|E|) berechenbar:

$$\text{low}[v] = \min\left(\text{dfs}[v],\quad \min_{(v,w)\in E}\begin{cases}\text{dfs}[w], & \text{falls } (v,w) \text{ Restkante}\\ \text{low}[w], & \text{falls } (v,w) \text{ Baumkante}\end{cases}\right)$$

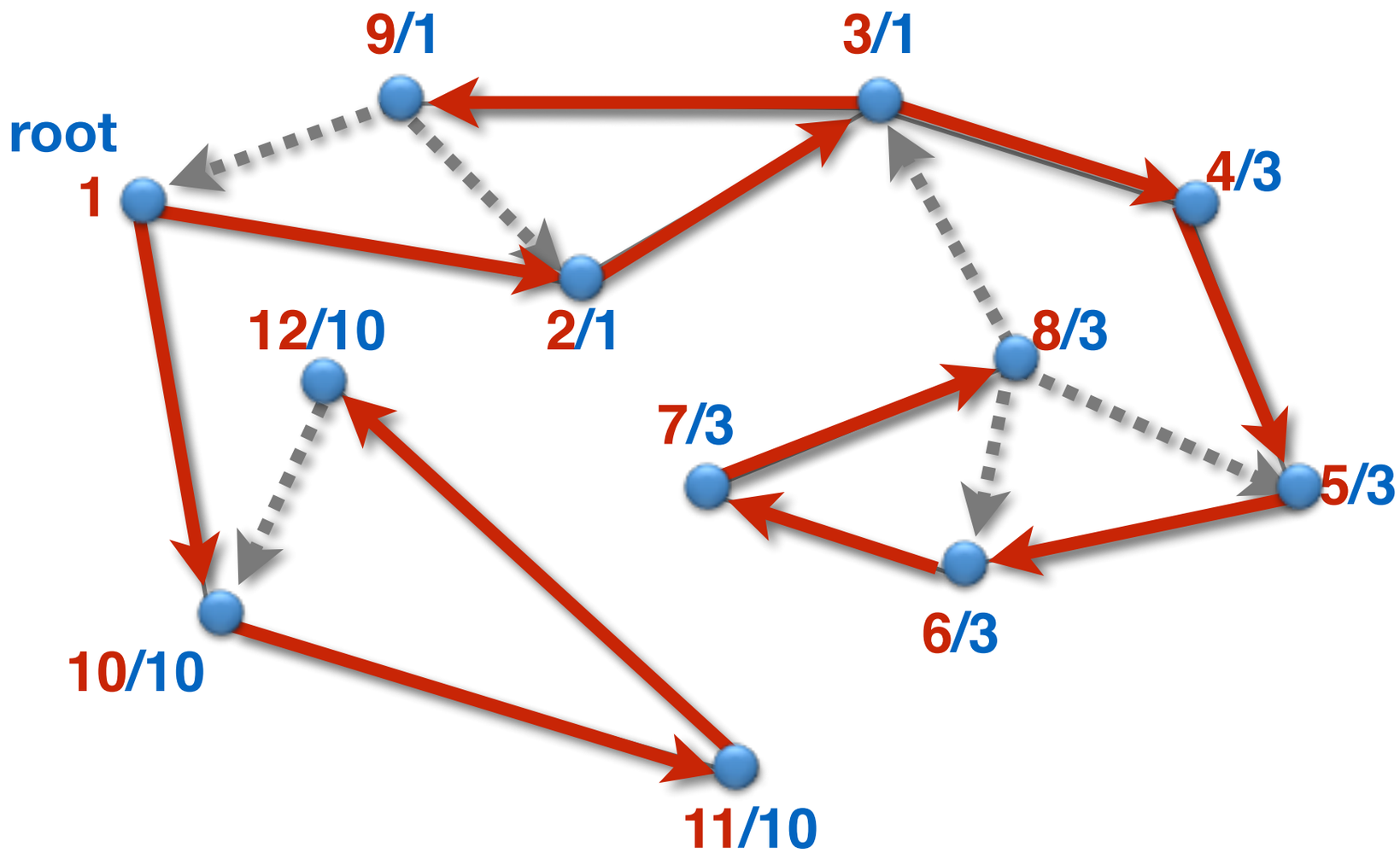
Tiefensuche



Alle low-Werte sind mit DP in Zeit O(|V|+|E|) berechenbar:

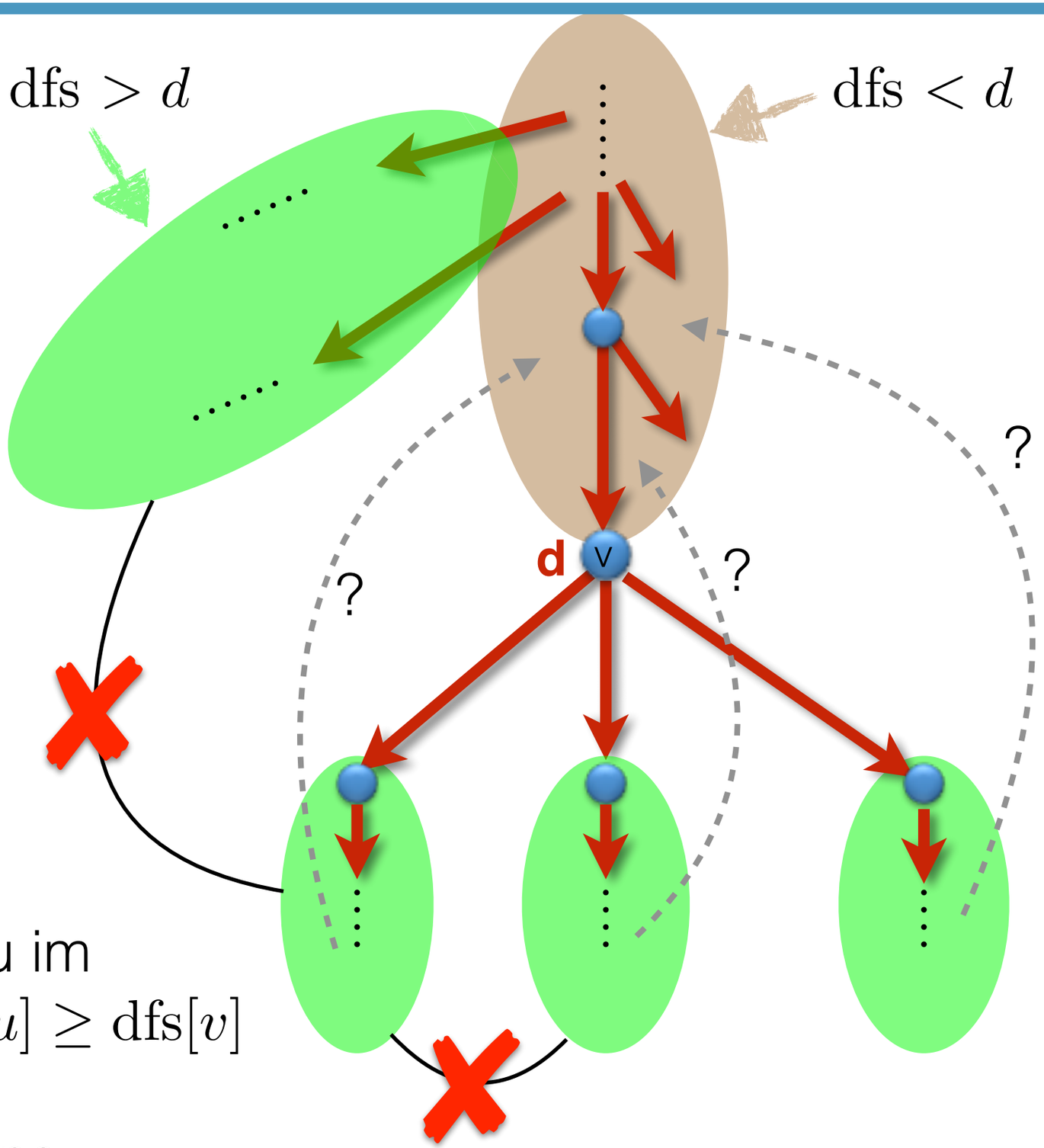
$$\text{low}[v] = \min\left(\text{dfs}[v],\quad \min_{(v,w)\in E}\begin{cases}\text{dfs}[w], & \text{falls } (v,w) \text{ Restkante}\\ \text{low}[w], & \text{falls } (v,w) \text{ Baumkante}\end{cases}\right)$$

Tiefensuche



$\text{low}[v]$:= kleinste dfs-Nummer, die man von v aus durch einen gerichteten Pfad aus (beliebig vielen) Baumkanten und maximal einer Restkante erreichen kann.

Tiefensuche



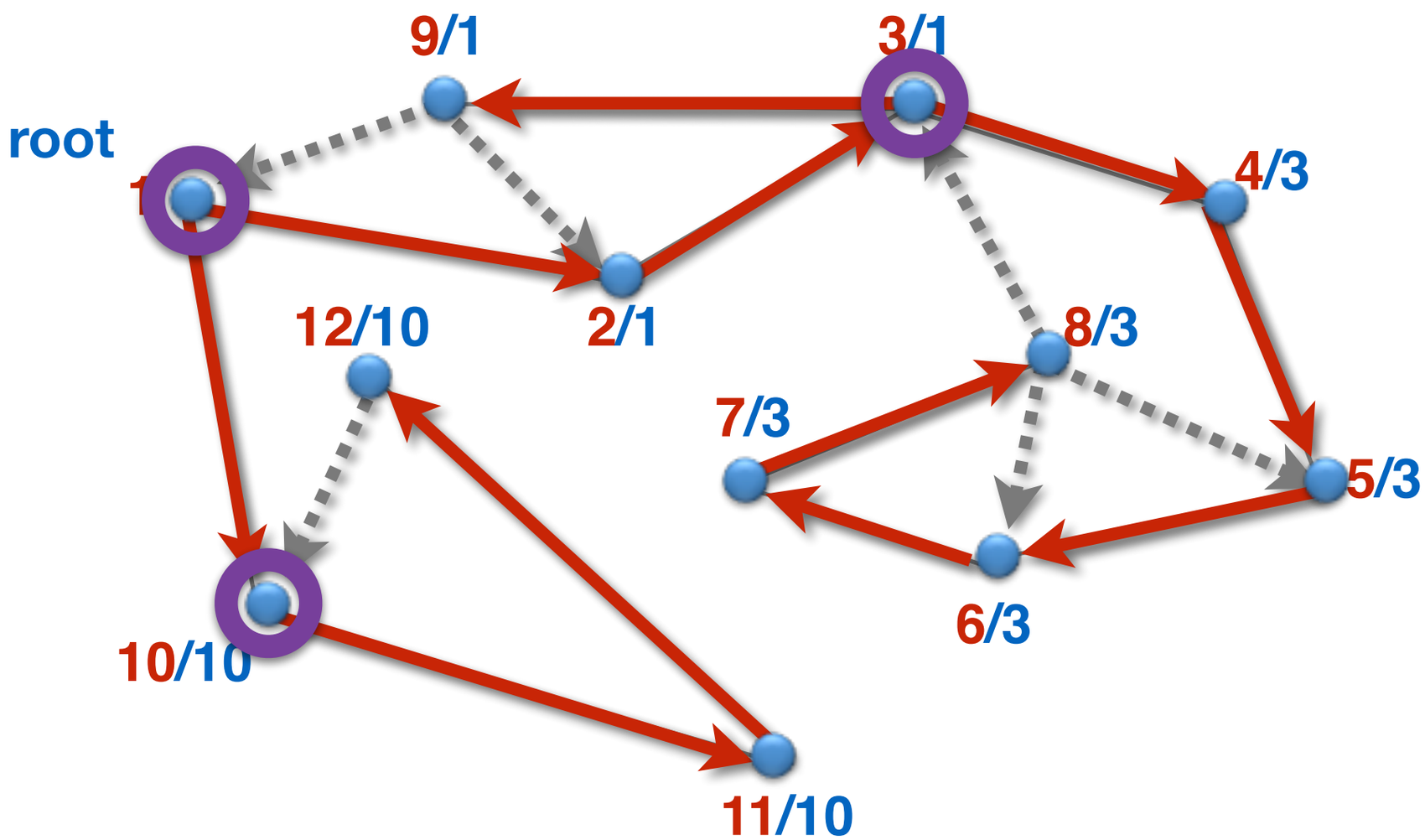
v ist genau dann Artikulations-knoten, wenn

1) v ≠ root, und v hat ein Kind u im DFS-Baum mit $\mathrm{low}[u] \geq \mathrm{dfs}[v]$

oder

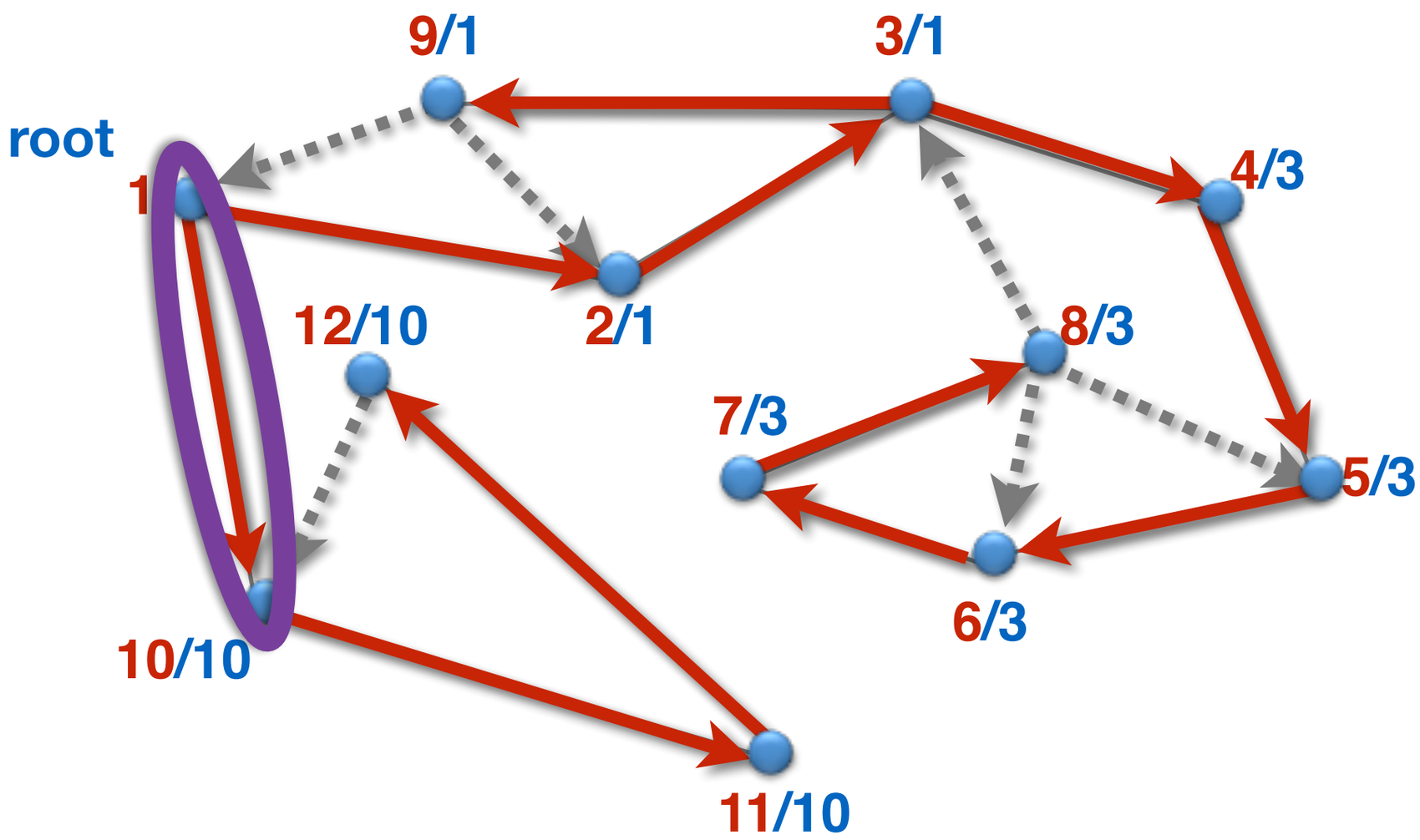
2) v = root, und v hat mindestens zwei Kinder im DFS-Baum.

Tiefensuche



Ein Knoten v ist genau dann ein Artikulationsknoten, wenn
1) v ≠ root, und v hat ein Kind u im DFS-Baum mit $\mathrm{low}[u] \geq \mathrm{dfs}[v]$,
oder
2) v = root, und v hat mindestens zwei Kinder im DFS-Baum.

Tiefensuche



Eine Baumkante e = (v,w) (v Elternknoten, w Kindknoten) ist genau dann eine Brücke, wenn $\mathrm{low}[w] > \mathrm{dfs}[v]$.

Restkanten sind niemals Brücken.

Artikulationsknoten und Brücken

Satz: Die um die Berechnung von **low[]** ergänzte Tiefensuche berechnet in einem zusammenhängenden Graphen alle Artikulationsknoten und Brücken in Zeit O(m).

Kapitel 1.3

Kreise

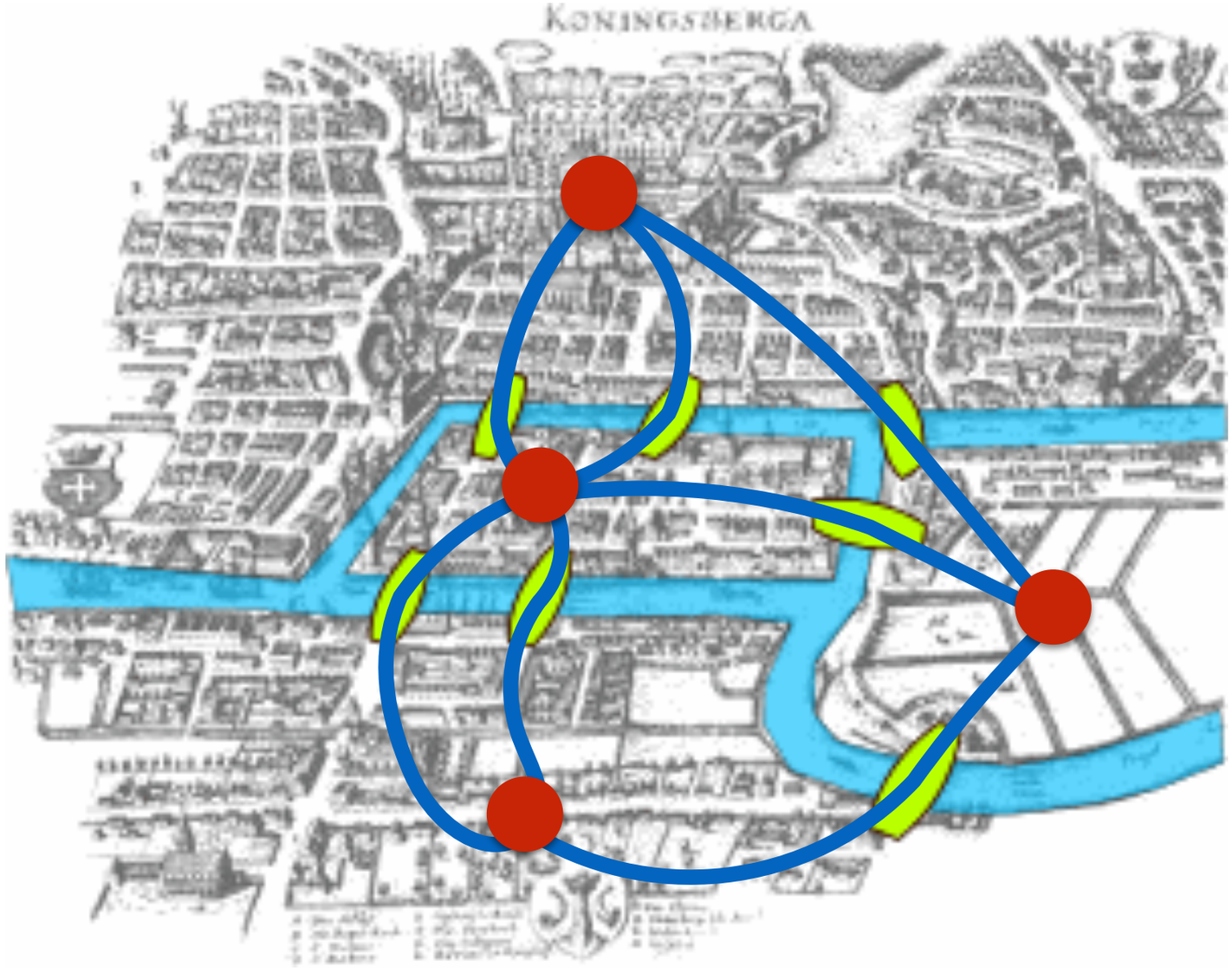
Hamiltonkreise und Eulertouren

- Sei G = (V, E) ein Graph.

- **Hamiltonkreis:**
  - Ein Kreis in G, der jeden Knoten genau einmal enthält.

- **Eulertour:**
  - Ein geschlossener Weg in G, der jede Kante genau einmal enthält.

Königsberger Brückenproblem



Leonard Euler
(1707 - 1783)

Gesucht: **Eulertour**

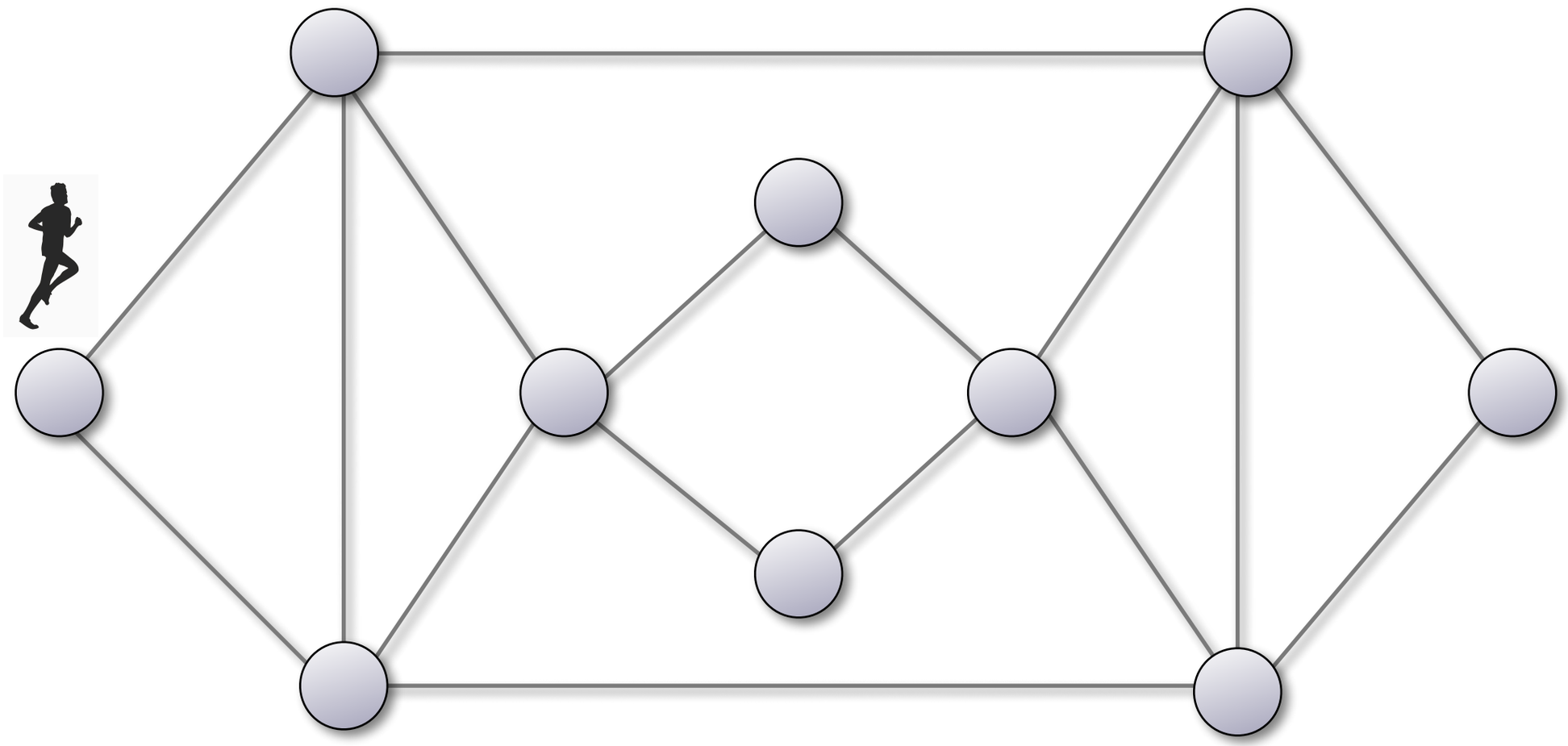
Eulertour: Charakterisierung

Satz: Ein zusammenhängender Graph G = (V, E) enthält eine Eulertour

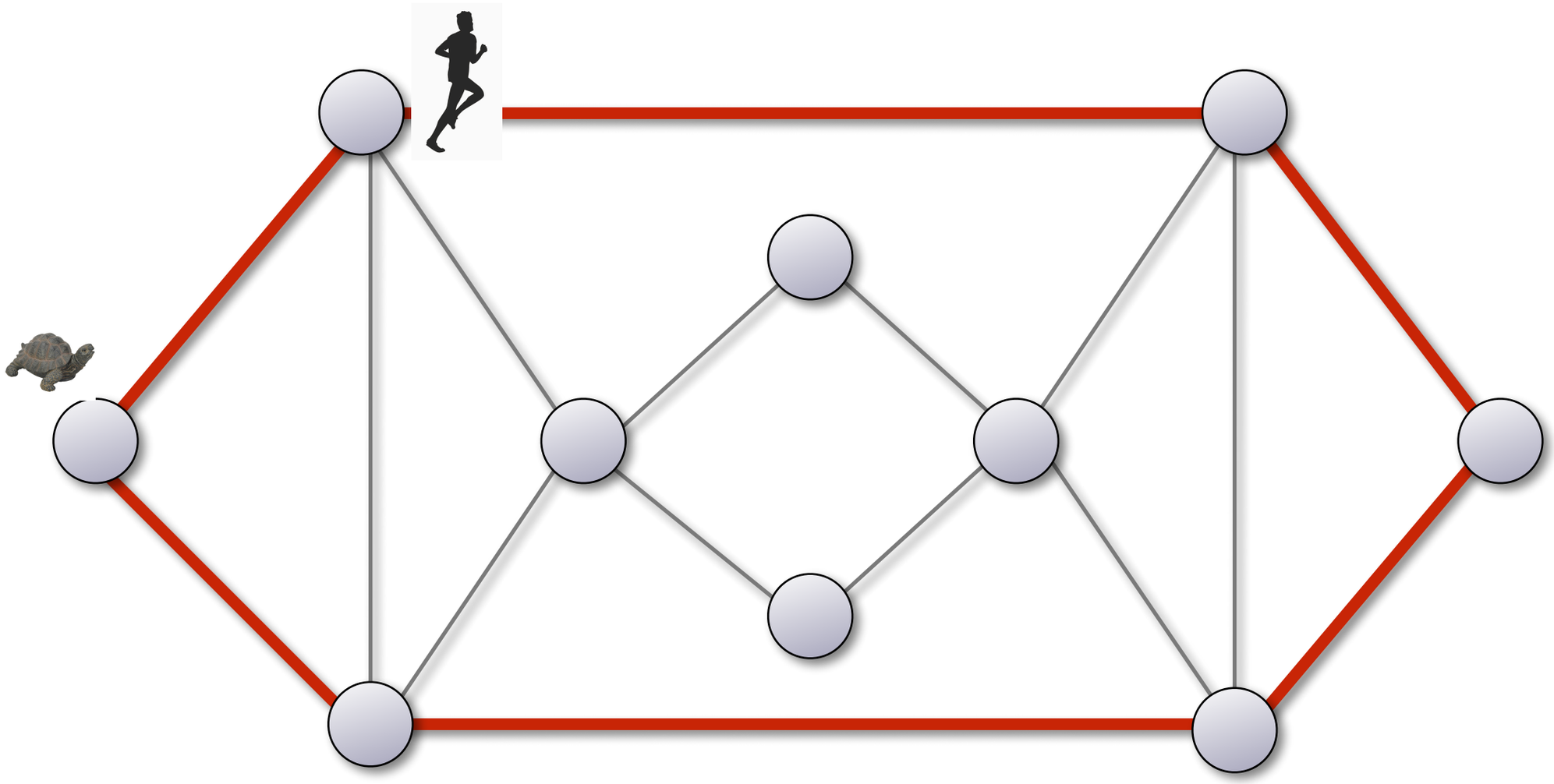
gdw. der Grad jedes Knotens gerade ist.

… und eine solche kann man in O(|E|) Zeit finden

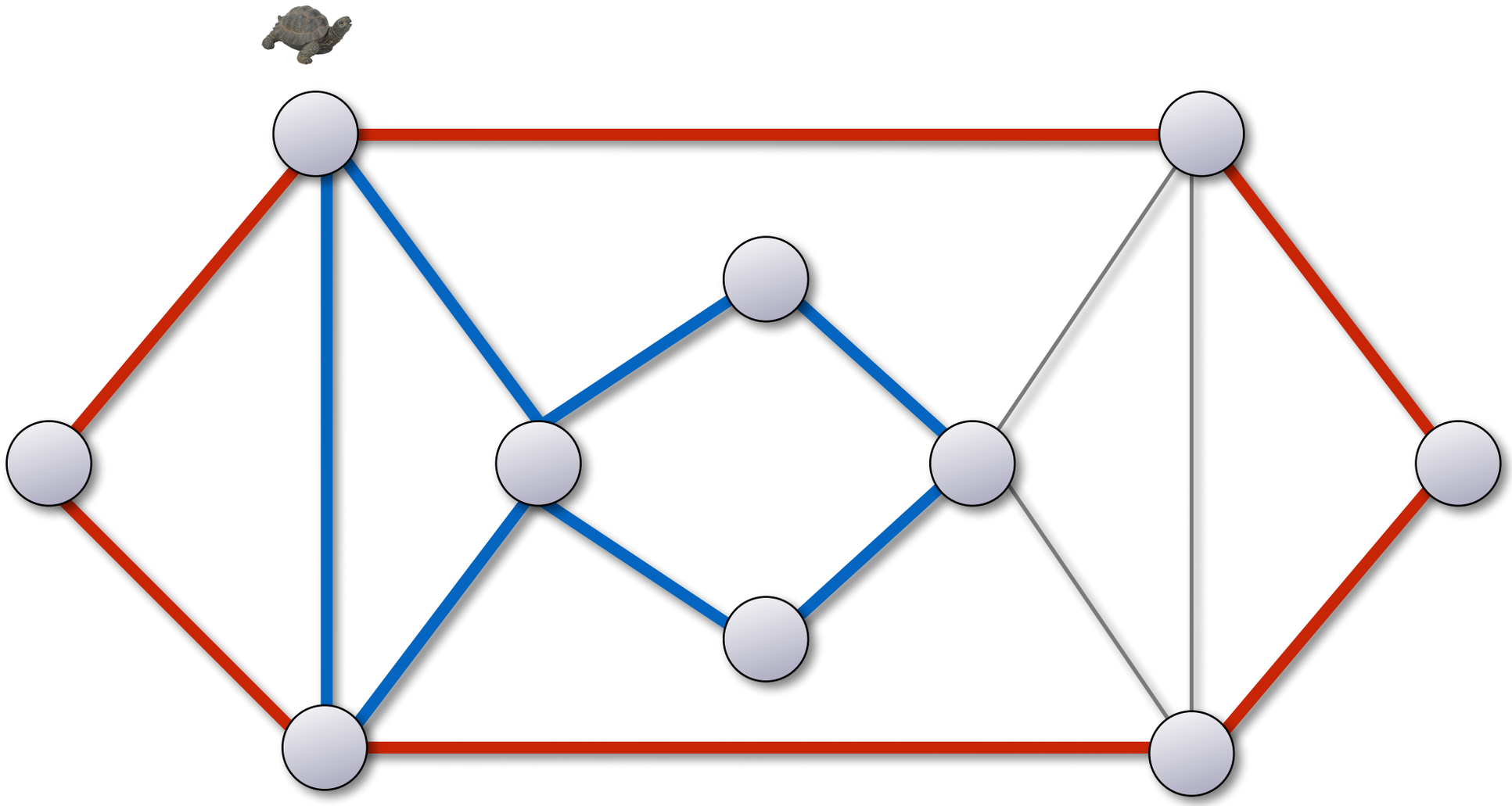
Idee des Algorithmus



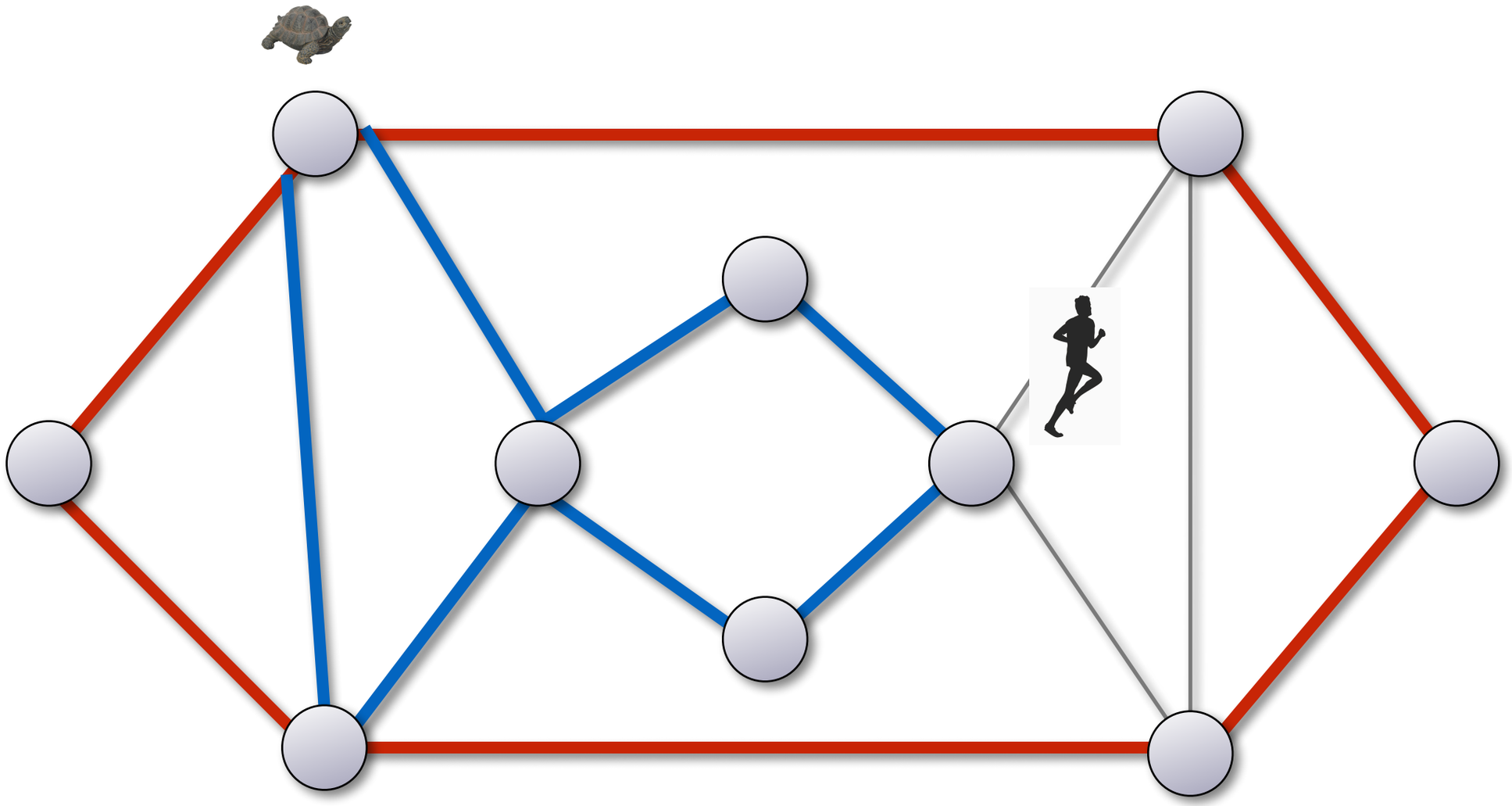
Idee des Algorithmus



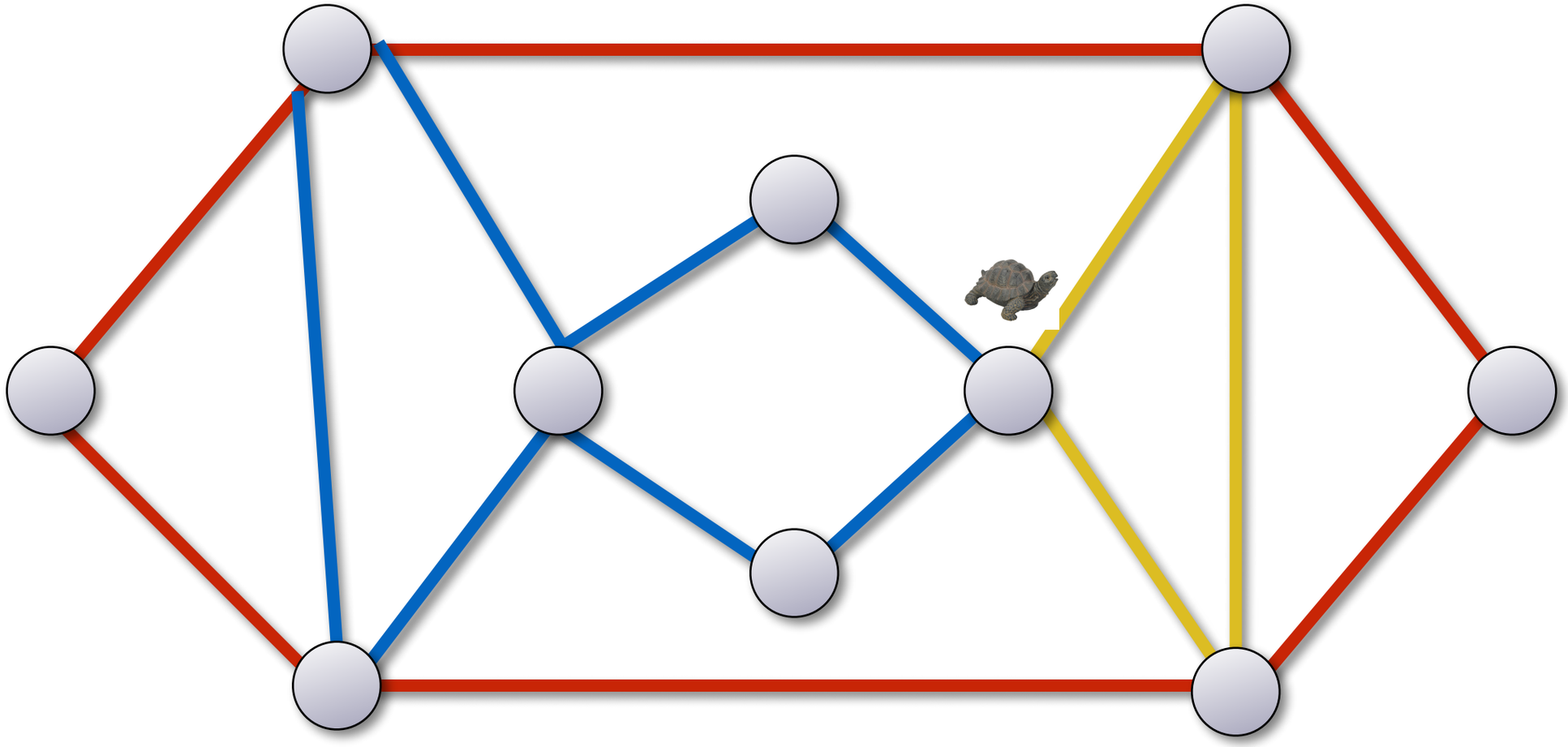
Idee des Algorithmus



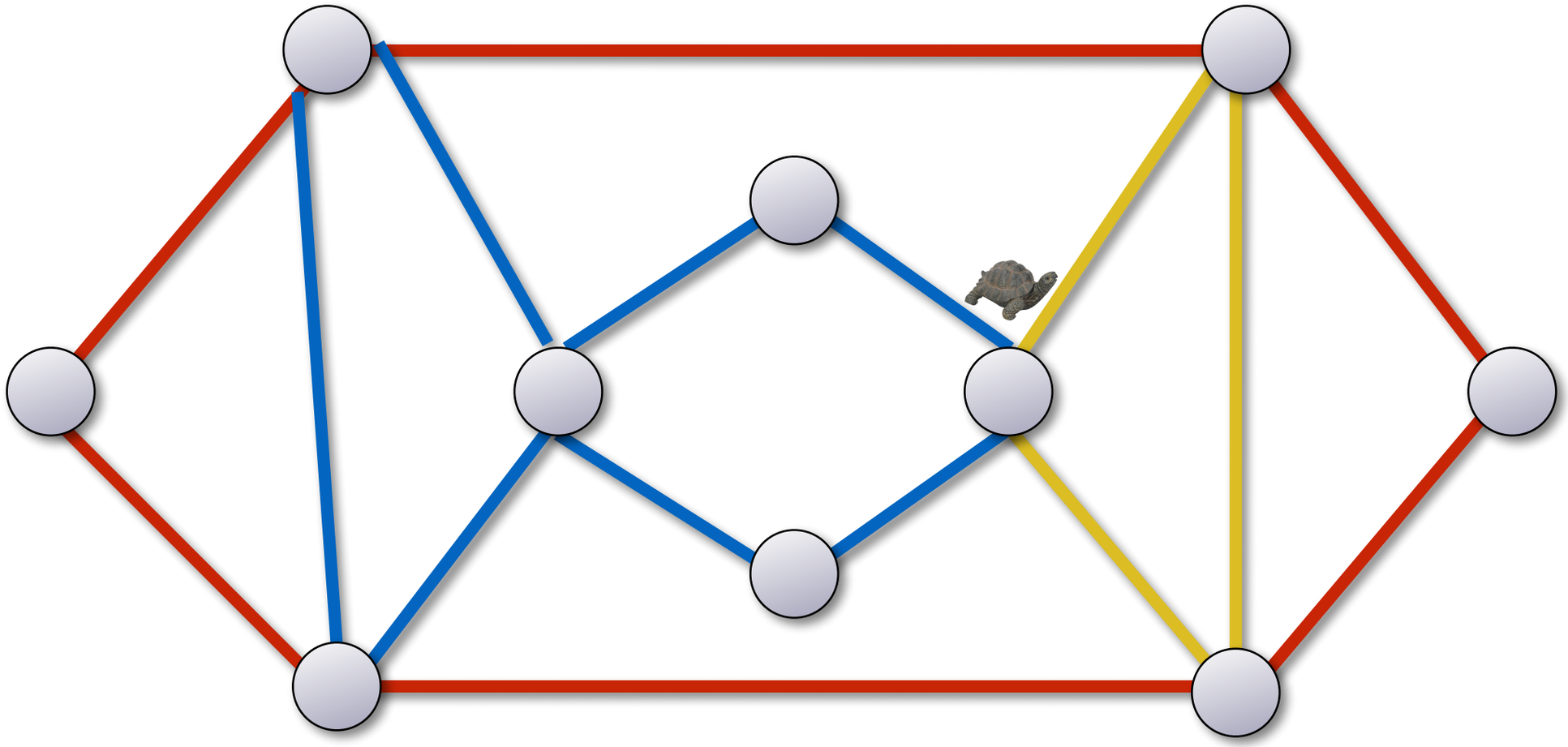
Idee des Algorithmus



Idee des Algorithmus



Idee des Algorithmus



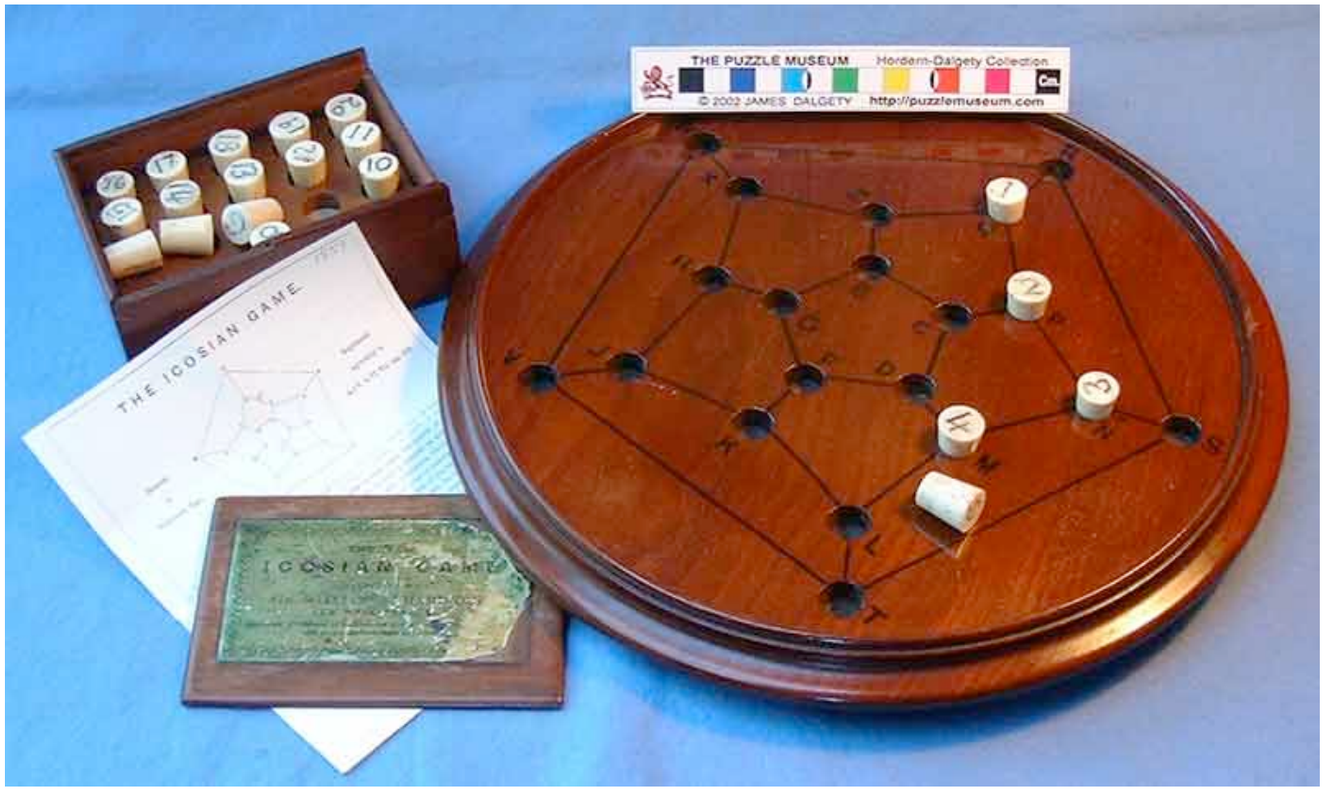
Hamiltonkreise und Eulertouren

- Sei G = (V, E) ein Graph.

- **Hamiltonkreis:**
  - Ein Kreis in G, der jeden Knoten genau einmal enthält.

- **Eulertour:**
  - Ein geschlossener Weg in G, der jede Kante genau einmal enthält.

Hamiltonkreis

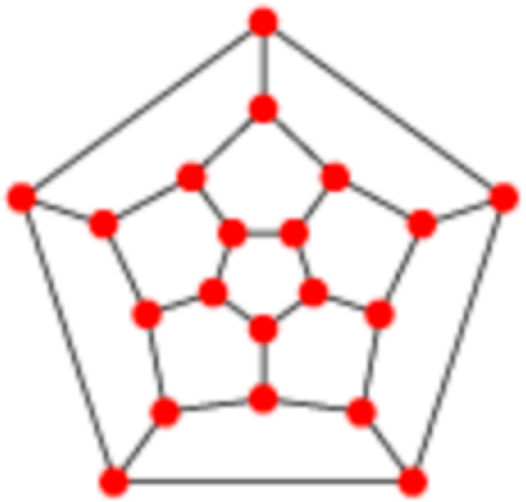


Ikosaeder Spiel

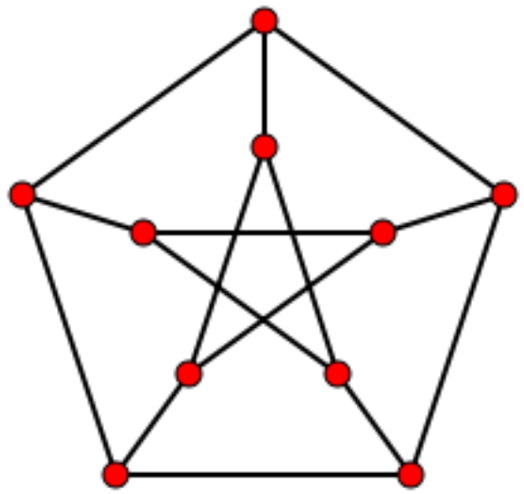
Sir William Hamilton
(1805 - 1865)

Gesucht: **Hamiltonkreis**

Hamiltonkreis - Beispiele

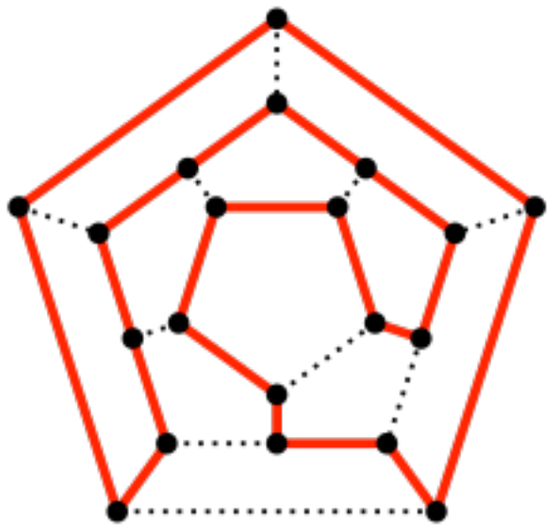


Ikosaeder

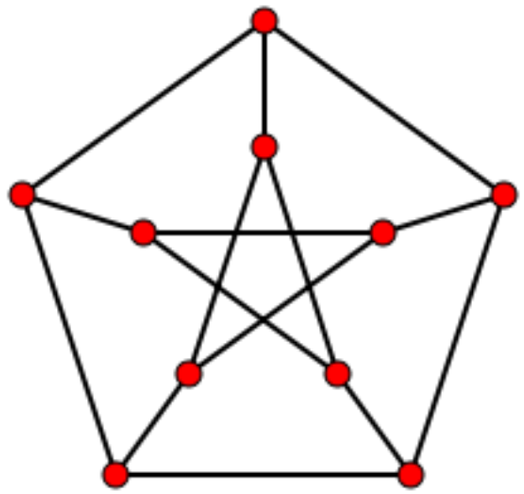


Petersengraph

Hamiltonkreis - Beispiele

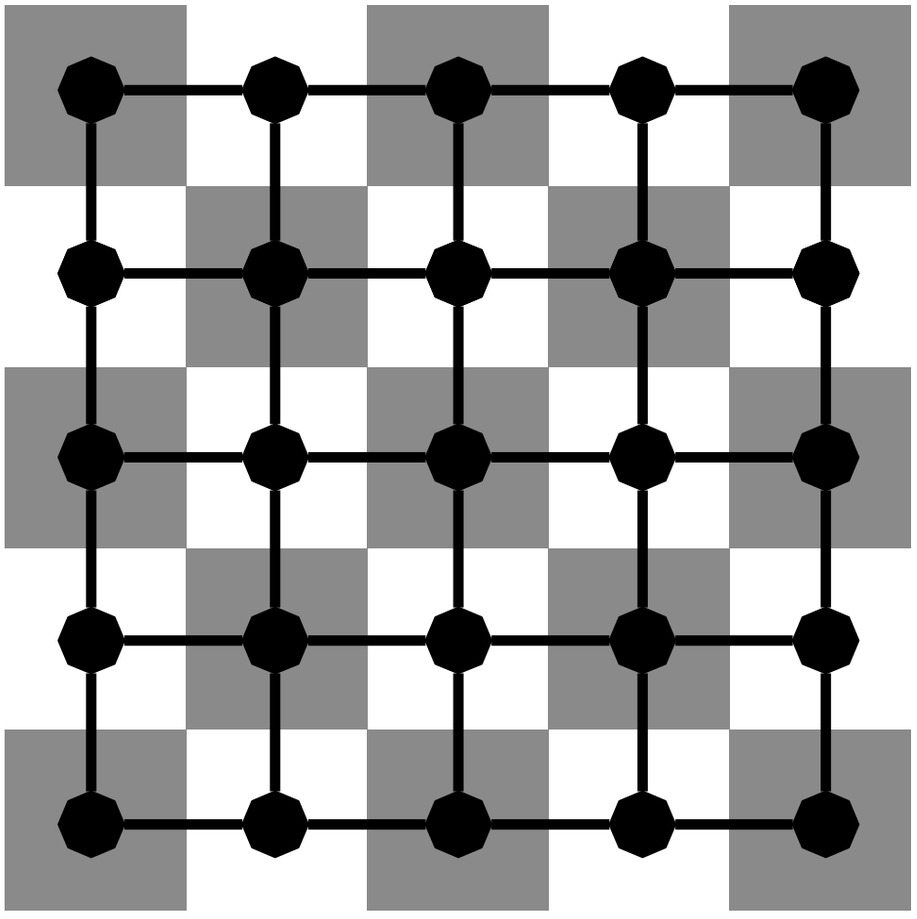


Ikosaeder



Petersengraph

Hamiltonkreis - Beispiele



Satz: Seien m,n ≥ 2.

Ein n×m Gitter enthält einen Hamiltonkreis **gdw** n*m gerade ist.

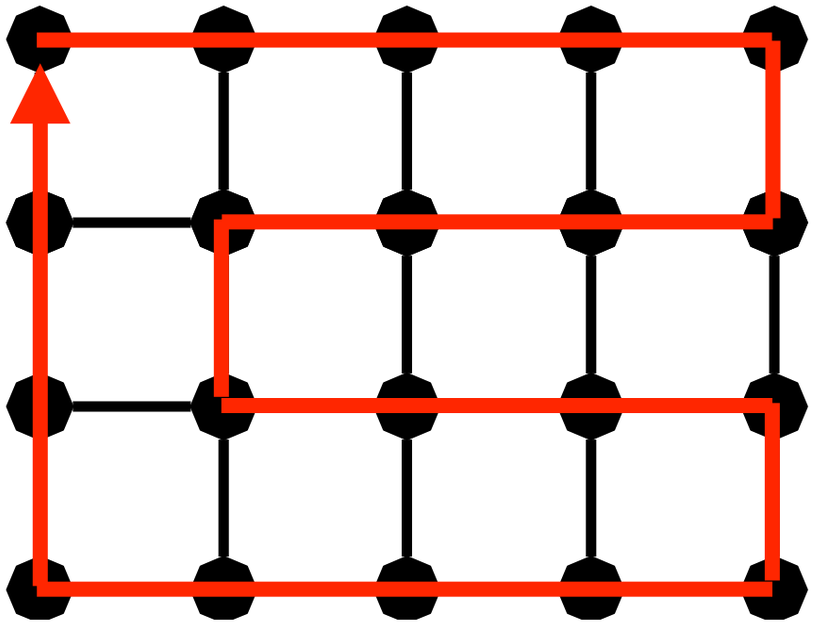
Hamiltonkreis - Beispiele

Satz: Seien m,n ≥ 2.

Ein n×m Gitter enthält einen Hamiltonkreis **gdw** n*m gerade ist.

Beweis:

„⇐“ siehe Skizze:



Hamiltonkreis - Beispiele

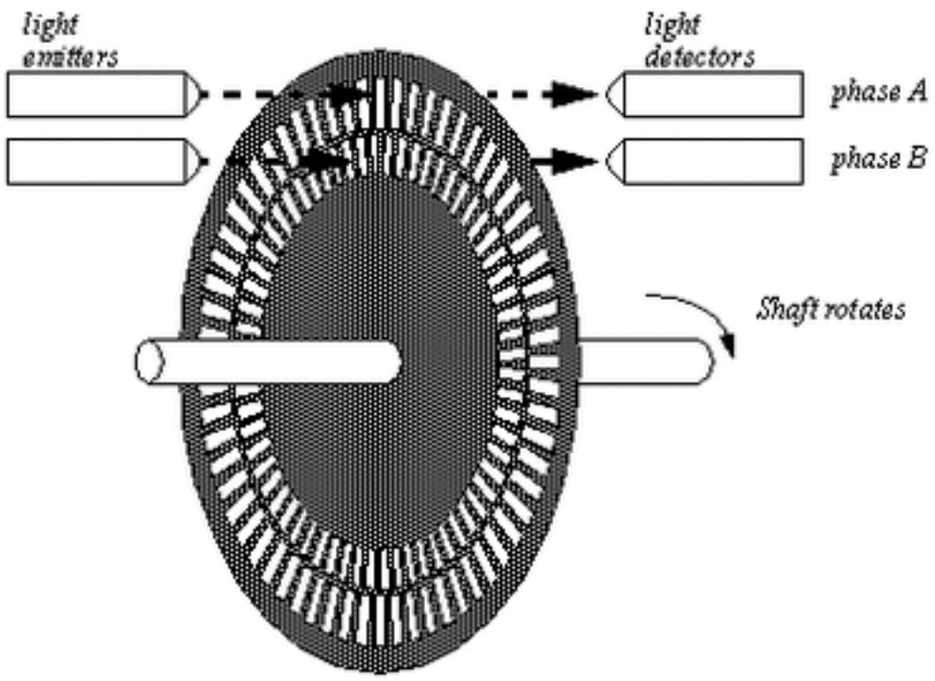
Rotary Encoder - Illuminated (RGB)

COM-10982 ROHS ✔

★★★½☆ 8

Description: Rotary encoders can be used similarly to potentiometers. The difference being that an encoder has full rotation without limits (It just goes round and round). This is a quadrature encoder and outputs similar to 2-bit gray code so that you can tell how much and in which direction the encoder has been turned. They're great for navigating menu screens and things like that.

$3.95



n Sensoren

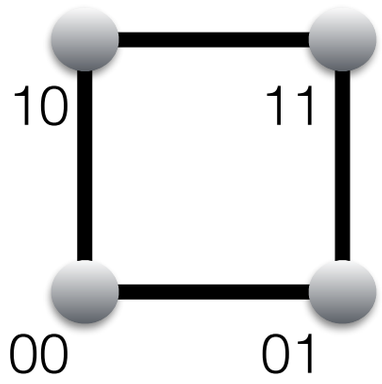
Hamiltonkreis - Beispiele

d-dimensionaler Hyperwürfel H_d

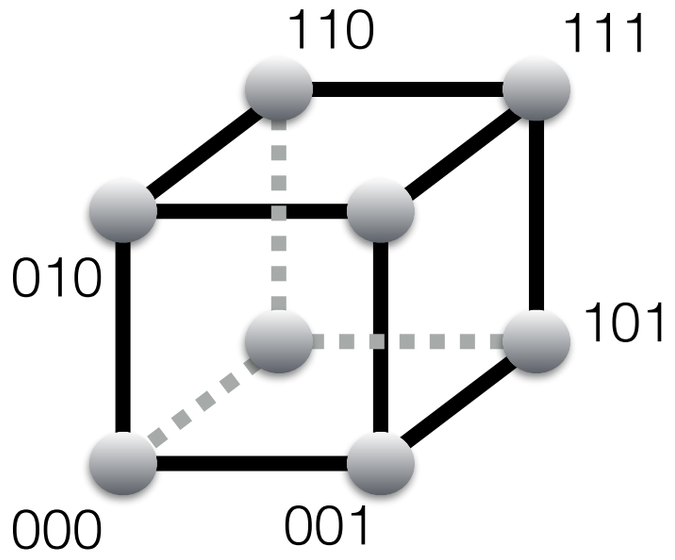
Knotenmenge: $\{0,1\}^d$

Kantenmenge: alle Knotenpaare, die sich in genau einer Koordinate unterscheiden

d=2:



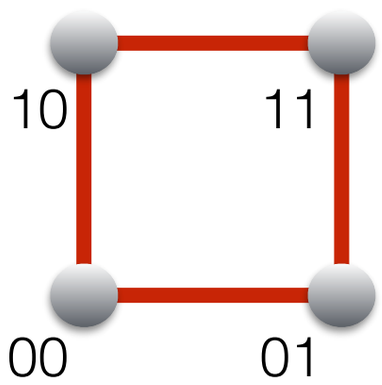
d=3:



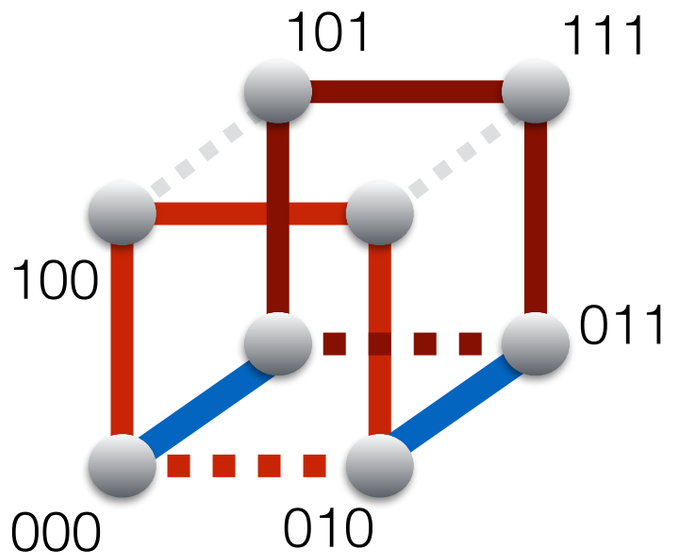
Enthält H_d einen Hamiltonkreis ?

Hamiltonkreis im Hyperwürfel: Gray-Code

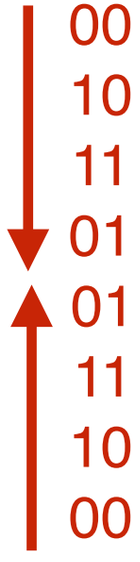
d=2:



d=3:



00
10
11
01



analog für d ≧ 4