Graphen

als Mittel der Modellierung
und ihre Datenstruktur

Suche in Graphen

- Realsituation: Orte, Straßen, Autobahnen, Flüsse, Brücken, …
- Modellierung bedeutet Vereinfachung durch Reduzierung auf das für die Problemstellung Wichtige

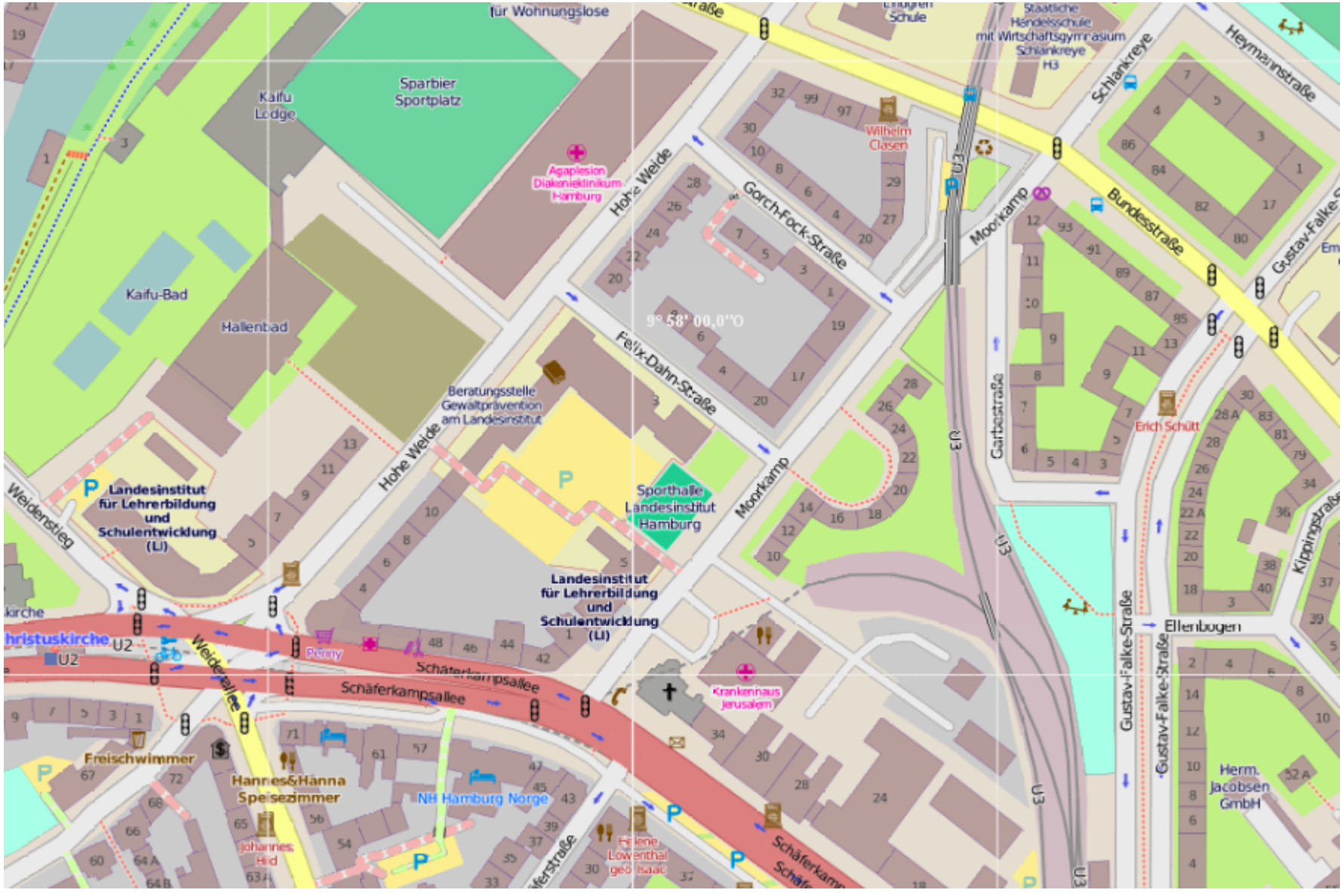
Suche in Graphen

Was ist wichtig?

- Name des Ortes?
- Lage des Ortes?
- Größe des Ortes?
- Name der Straße?
- Verlauf der Straße?
- Länge der Straße?
- Typ der Straße?
- Anzahl der Fahrbahnen?
- Einbahnstraßen?
- Verkehrsauf-kommen?
- Kreuzungen?
- Abbiegeverbote?

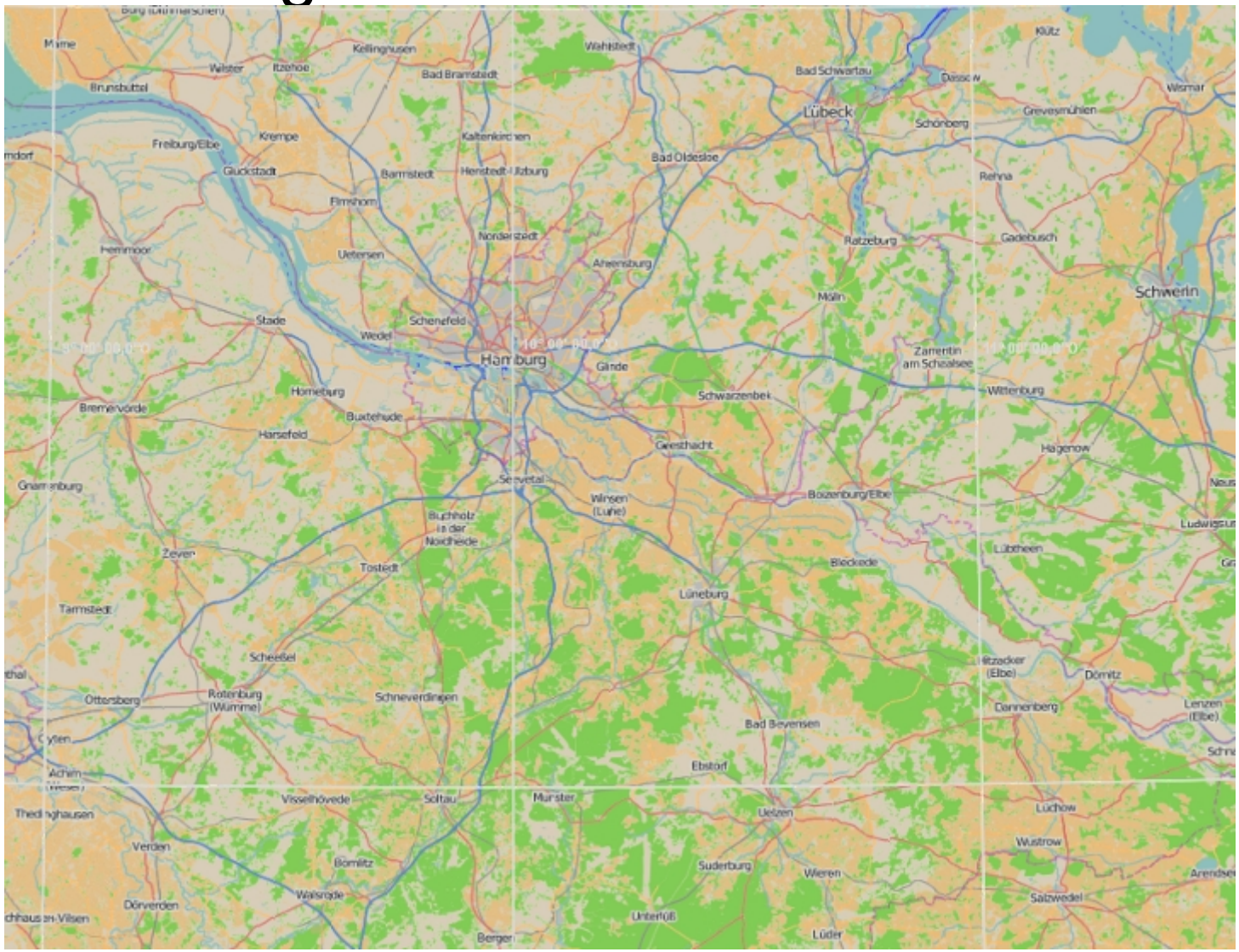
Suche in Graphen

- Karte zeigt Straßennamen



Suche in Graphen

- Karte zeigt Ortsnamen



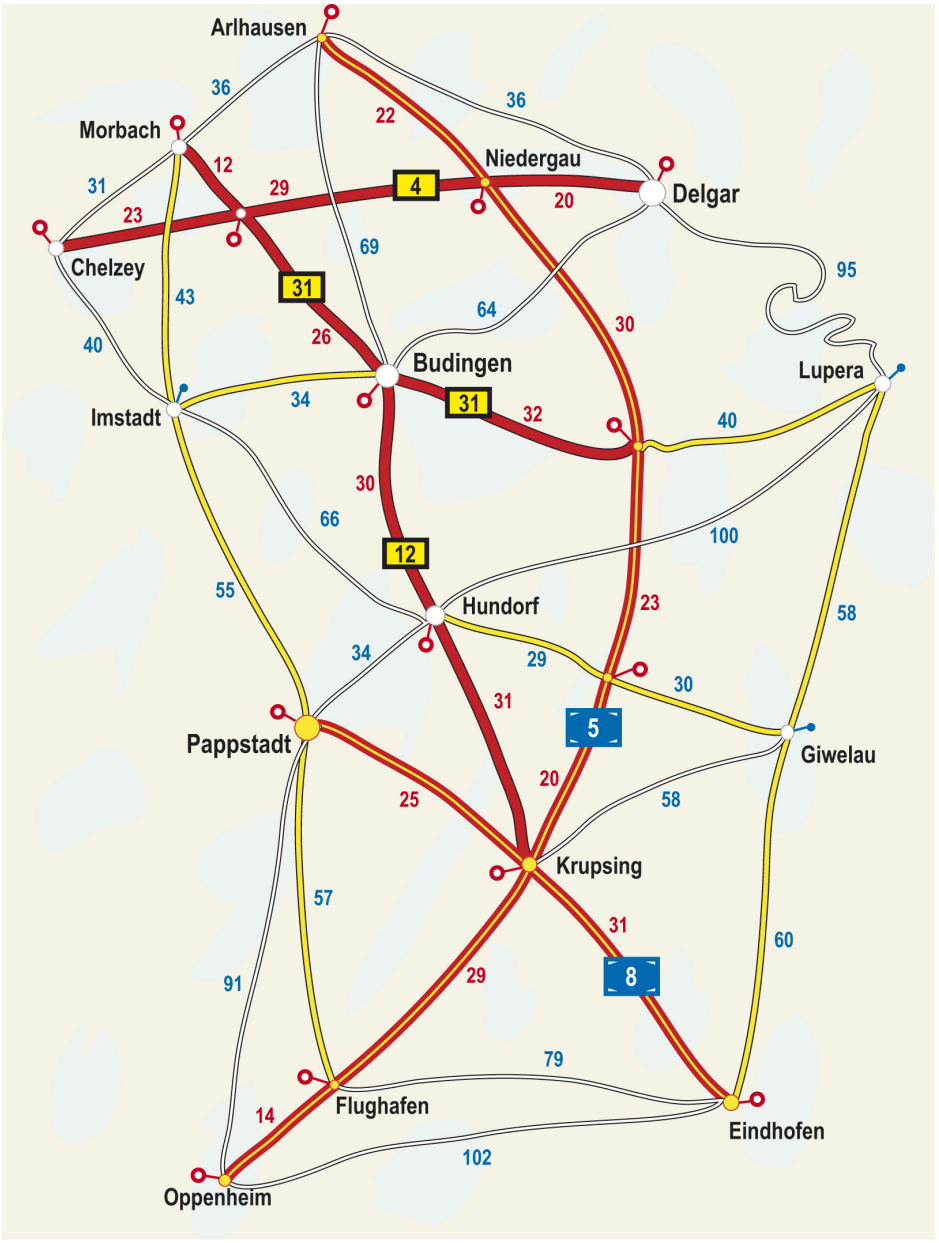
Bilder mit Marble erstellt

Suche in Graphen

Die beiden Beispiele zeigen zwei wesentliche Unterschiede bei der Darstellung, aber auch in der Ansicht über das Ziel der Darstellung:

- Es werden Straßennamen angegeben,
- oder es werden die Orte angegeben.

Suche in Graphen



Aus: Gallenbacher, *Abenteuer Informatik*, 2. Aufl.

← Quelle

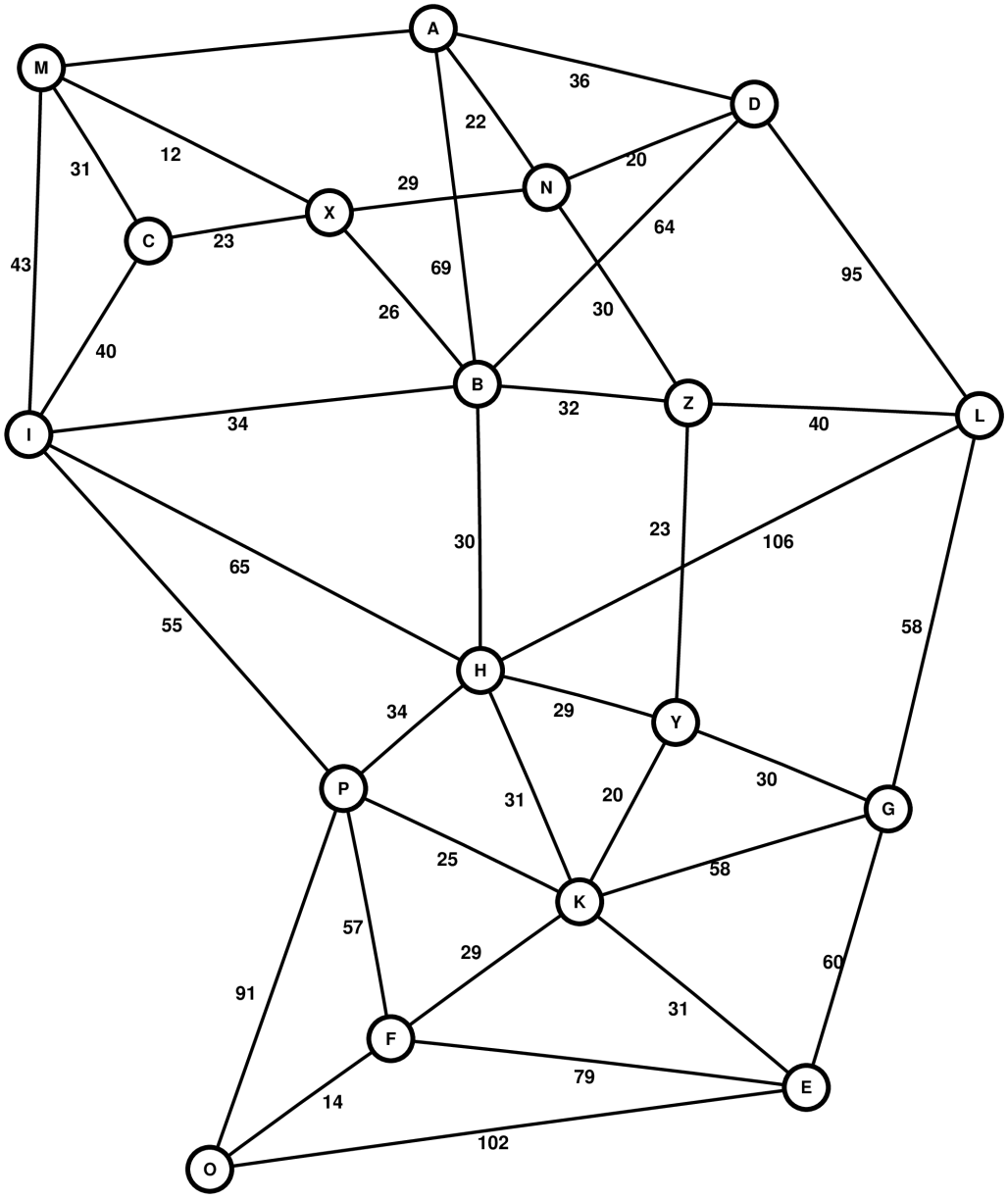
Suche in Graphen

Beispiel-Graph von Gallenbacher

- Realsituation: Orte, Straßen, Autobahnen, Flüsse, Brücken, …
- Modellierung bedeutet Vereinfachung durch Reduzierung auf das für die Problemstellung Wichtige

Suche in Graphen

Beispiel-Graph
nach Gallenbacher



Vorlage
Gallenbacher:
Abenteuer Informatik

Suche in Graphen

Der Beispiel-Graph von Gallenbacher zeigt einen vereinfachten Graphen mit folgenden Eigenschaften:

- Es werden keine Straßennamen angegeben, also nicht die Kanten benannt.
- Statt dessen werden die Orte mit Namen bezeichnet.
- Die Straßen werden mit einer einfachen Zahl bewertet.

Suche in Graphen

Elemente von Graphen

- Kanten
- Knoten (oder Ecken [Anm.])
- ggf. Kantenbewertungen

Graphen:

Knoten , Kanten, Kantenbewertungen

[Anm.] Ich vermeide den Begriff Ecke, da die englische Übersetzung edge sowohl für Ecken, als auch für Kanten verwendet wird.

Suche in Graphen

Welche Datenstruktur wählt man bei Python?

Listen (Tupel, ...) von ...

- … Kanten speichern mit den Orten, die sie verbinden und der Kantenbewertung (soweit vorhanden)
- … Knoten speichern mit den Zielknoten der Wege, die von ihnen wegführen und der Kantenbewertung (soweit vorhanden)

Suche in Graphen

Kanten speichern mit den Orten, die sie verbinden und der Kantenbewertung (soweit vorhanden)

- Bei diesem Beispiel wird eine Liste verwendet, die einfache Listen enthält:

```
[ [A, B, 69], [A, D, 36], … ]
```

Kantenliste

- Tupel sind auch sinnvoll, da die Elemente der Liste nicht verändert werden.

```
[ (A, B, 69), (A, D, 36), … ]
```

Suche in Graphen

Aber ACHTUNG!

A, B usw müssen als Objekte bekannt sein!
Alternativ also besser Strings **`'A'`**, **`'B'`**, *...*

- Liste, die einfache Listen enthält:

```
[ ['A', 'B', 69], ['A', 'D', 36], … ]
```

- Tupel:

```
[ ('A', 'B', 69), ('A', 'D', 36), … ]
```

Suche in Graphen

Knoten speichern mit den Zielknoten der Wege, die von ihnen wegführen und deren Kantenbewertung (soweit vorhanden)

- Im Beispiel wird ein **Dictionary** verwendet, das jedem Ausgangsknoten die Tupel aus den Zielknoten zusammen mit ihrer jeweiligen Kantenbewertung zuordnet:

```
{ 'A': [ ('B', 69), ('D', 36), ... ],
  'B': [ ('A', 69), ... ]
    ... }
```

Dictionary

Suche in Graphen

Knoten speichern mit den Zielknoten der Wege, die von ihnen wegführen und deren Kantenbewertung (soweit vorhanden)

- Im Beispiel wird ein Dictionary verwendet, das jedem Ausgangsknoten die Listen aus den Zielknoten zusammen mit ihrer jeweiligen Kantenbewertung zuordnet:

```
{ 'A': [ ['B', 69], ['D', 36], ... ],
  'B': [ ['A', 69], ... ]
    ... }
```

Dictionary

Suche in Graphen

Knoten speichern mit den Zielknoten der Wege, die von ihnen wegführen und deren Kantenbewertung (soweit vorhanden)

- Es geht natürlich auch mit tiefen Listen

```
[ ['A', [ ['B', 69], ['D', 36] ] ],
  ['B', [ ['A', 69] ] ],
  ]
```

tiefe Liste

Suche in Graphen

Knoten speichern mit den Zielknoten der Wege, die von ihnen wegführen und deren Kantenbewertung (soweit vorhanden)

- Es geht natürlich auch mit tiefen Listen, die Tupel enthalten

```
[ ['A', [ ('B', 69), ('D', 36) ] ],
  ['B', [ ('A', 69) ] ],
  ]
```

tiefe Liste mit Tupeln

Suche in Graphen

Das letzte Beispiel zeigt ein Problem:

- Speichert man überhaupt und wenn wie speichert man die Information, dass Wege in beiden Richtungen benutzt werden können?
- Sind überhaupt alle Wege in beiden Richtungen benutzbar?

→ Fachausdruck ***gerichteter Graph***