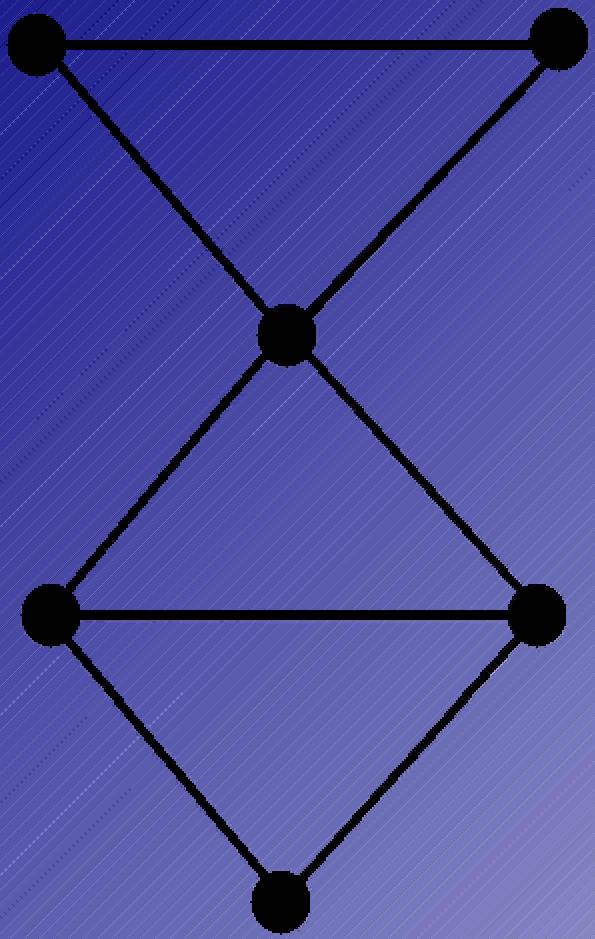
Gliederung

- Zusammenhang von Graphen
  - Stark Zusammenhängend
  - K-fach Zusammenhängend
- Brücken
  - Definition
  - Algorithmus zum Finden von Brücken
  - Anwendung
- Zusammenhangskomponente
  - Definition
  - Wichtige Aussagen und Sätze
  - Algorithmen zum Finden von Starken Zusammenhangskomponenten
    - Serieller Algorithmus von Warshall
    - Tripelalgorithmus
    - Tiefensuche
    - Algorithmus von Tarjan zur Bestimmung starker Zusammenhangskomponente
  - Anwendung/ Beispiele
- Fragen?

1. Zusammenhang von Graphen

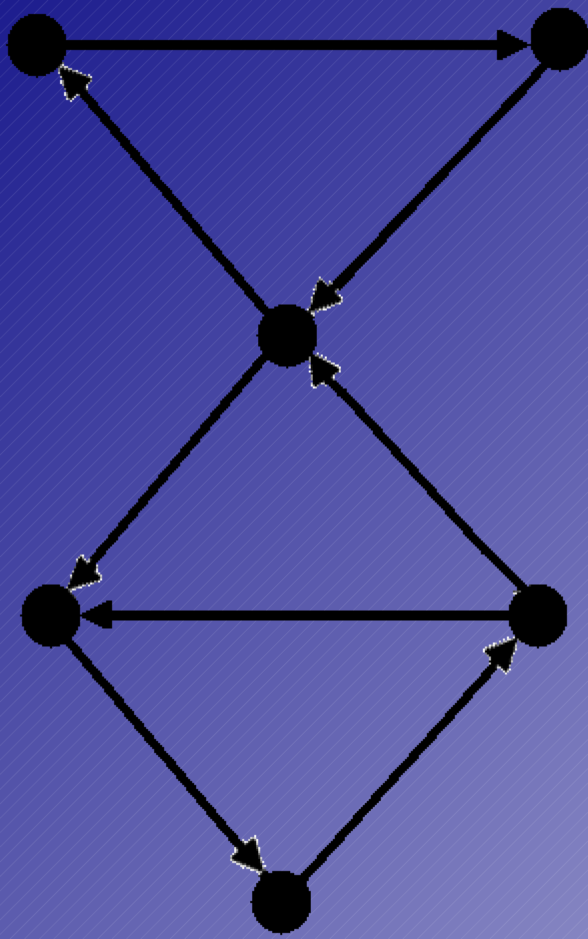


- Ungerichteter Graph

  Ein ungerichteter Graph G=(V,E) heißt zusammenhängend, falls es zu je zwei beliebigen Knoten v und w aus V einen ungerichteten Weg in G gibt, mit v als Startknoten und w als Endknoten.

  Falls G nicht zusammenhängend ist, nennt man G unzusammenhängend.

1. Zusammenhang von Graphen



Gerichteter Graph

Ein gerichteter Graph G=(V,E) heißt **zusammenhängend von einem Knoten** v **aus**, falls es zu jedem Knoten w aus V einen gerichteten Weg in G gibt, mit v als Startknoten und w als Endknoten.

G heißt **stark zusammenhängend**, falls G von jedem Knoten v aus V zusammenhängend ist.

Ein gerichteter Graph G=(V,E) heißt **zusammenhängend**, falls der zugehörige ungerichtete Graph (also der Graph, der entsteht, wenn man jede gerichtete Kante durch eine ungerichtete Kante ersetzt) zusammenhängend ist.

Algorithmus „zusammenhängend?“

Input: Graph G=(V,E)
output: ja, falls zusammenhängend

Starte mit einem beliebigen Knoten v, einer Knotenmenge S:={v}, und einem Vektor besucht(x) ($x \in V$) der zu Anfang überall den Eintrag 0 hat.

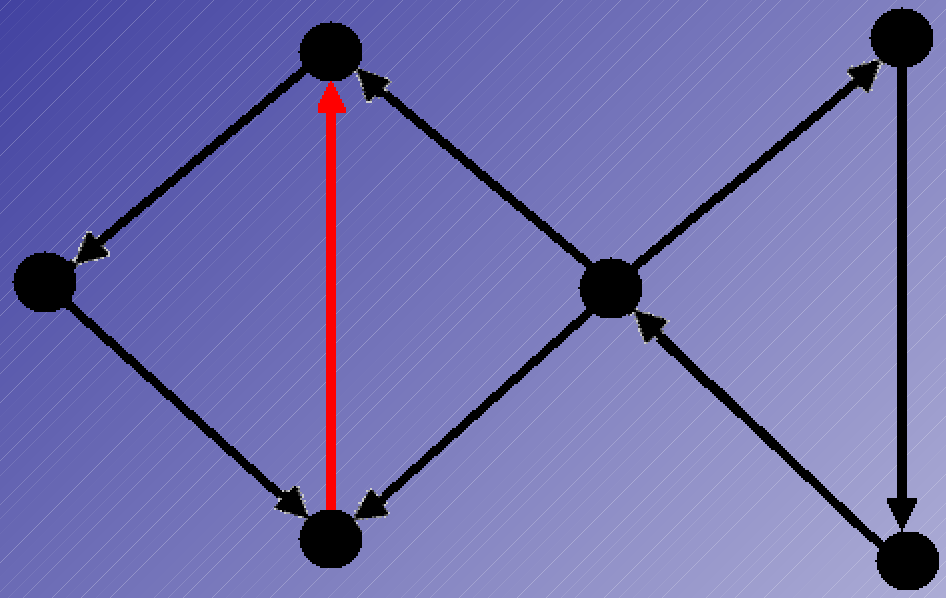
while $S \neq \emptyset$ **do**

entferne einen Knoten w aus S, setze besucht(w):=1;
für alle Kanten ws füge s zu S falls besucht(w)=0;

od

Aufwand: O(|V|+|E|)

2. Brücken



$e \in E$ heißt Brücke, wenn G-e mehr Zusammenhangskomponenten besitzt, als G.

(Eine Kante e=xy heißt Brücke, wenn sie x und y trennt, d.h. Wenn es ohne xy keinen Weg mehr von x nach y gibt. Der Graph wird also durch Entfernen von e in zwei unabhängige Teile geteilt.)

Algorithmus zur Bestimmung von Brücken

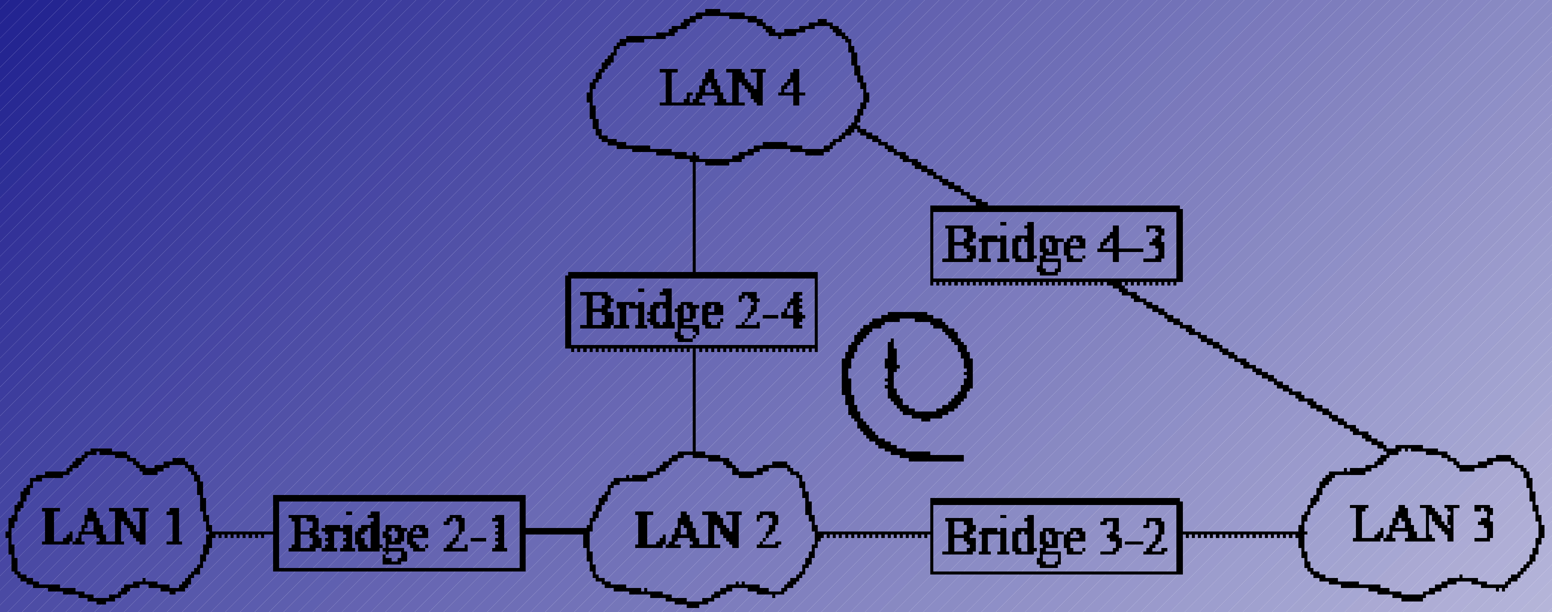
geg: Kante a von Knoten x nach y
ges: ist a Brücke in G=(V,E)

```
G'=G(V,E\a)                          //entfernen der Kante

//durchführen einer Tiefensuche beginnend mit Knoten x auf G'
if ( y∈dfs(G', x) ) then              //wenn y gefunden wird
      out << a ist keine Brücke       //ist a keine Brücke
else                                  //sonst
      out << a ist eine Brücke        //ist a eine Brücke
fi
```

Aufwand: $O(|V|+|E|)$
offensichtlich, da gleicher Aufwand, wie Tiefensuche

Anwendung



Netzwerktechnik:

Verbinden 2er Subnetze mittels einer „Bridge“
mehrerer Subnetze mittels eines „Switch“

Finden von Endlosschleifen

Stark frequentierte Pfade in Datennetzen haben bei eventuellen Brücken Schwachstellen

3. Zusammenhangskomponente

Zusammenhangskomponente:

Eine Zusammenhangskomponente von G ist ein maximal zusammenhängender Teilgraph von G.

starke Zusammenhangskomponente:

Ein induzierter Teilgraph K=(VK,EK) von G heißt starke Zusammenhangskomponente von G, falls K stark zusammenhängend ist und in G kein Knoten aus VK Vorgänger oder Nachfolger von einem Knoten aus der Differenzmenge V\VK ist.
Gelegentlich spricht man auch von streng zusammenhängend.

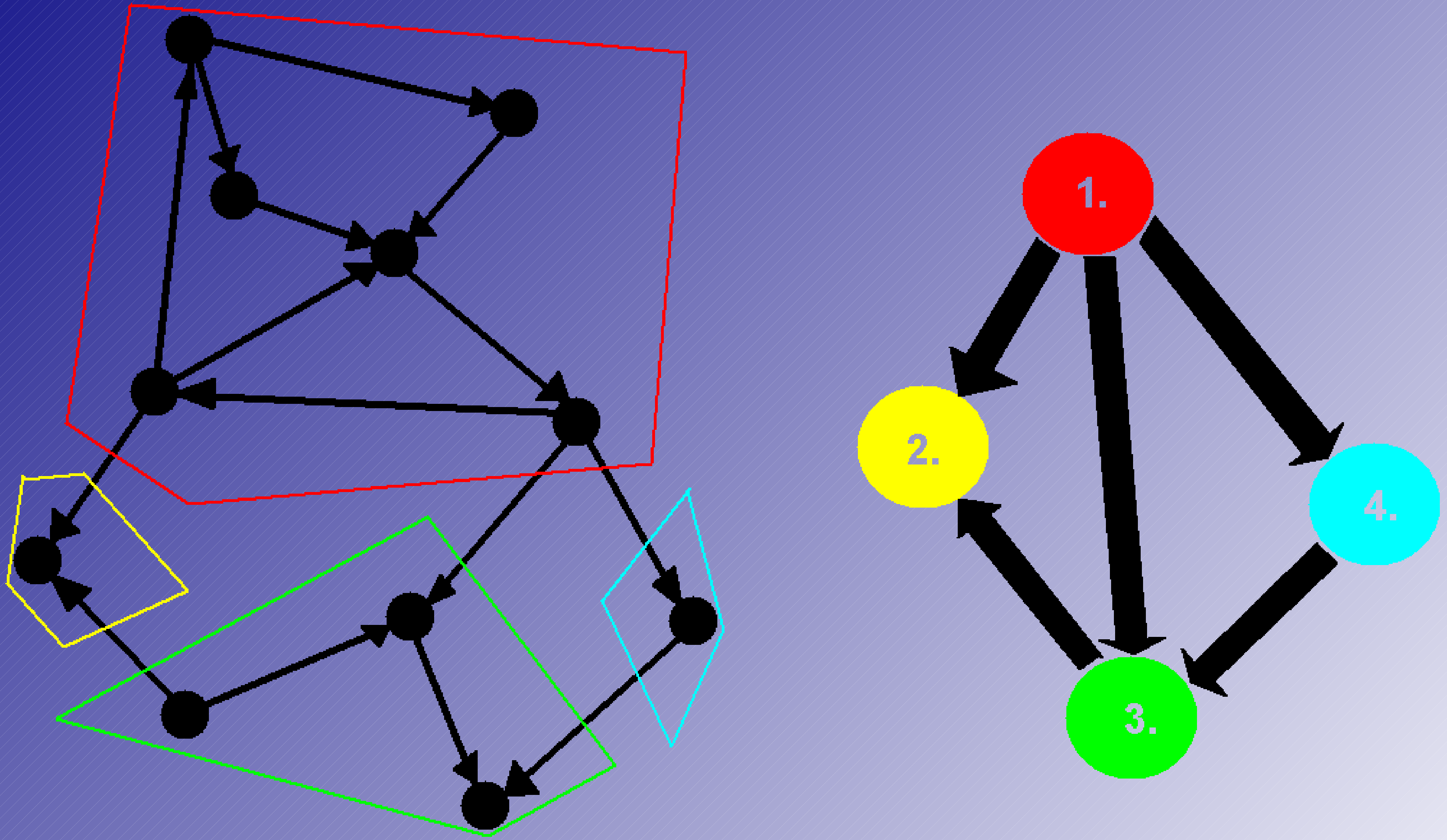
2-fache Zusammenhangskomponente:

u, v sind genau dann in der gleichen Zusammenhangskomponente, wenn es zwei knotendisjunkte Wege von u nach v in G gibt.

Eine 1-Zusammenhangskomponente nennt man auch einfach nur Zusammenhangskomponente.

Eine 2-Zusammenhangskomponente nennt man Block.

4 starke Zusammenhangskomponente



Wichtige Aussagen und Sätze I

- Ein ungerichteter Graph ist genau dann zusammenhängend, wenn er einen spannenden Baum enthält.
- Ein ungerichteter zusammenhängender Graph ist genau dann 2-zusammenhängend, wenn er keine Artikulation besitzt.
- G ist genau dann k-zusammenhängend, wenn G zwischen je zwei Ecken k disjunkter Wege enthält.
- G ist genau dann k-fach kantenzusammenhängend, wenn G zwischen je zwei Ecken k kantendisjunkte Wege enthält.
- Ist G=(V,E) ein ungerichteter Graph und sind A und B Teilmengen von V, so ist die kleinste Mächtigkeit einer A von B trenndenden Knotenmenge gleich der größten Mächtigkeit einer Menge disjunkter A-B-Wege in G. (Satz von Menger)

Wichtige Aussagen und Sätze II

- Ist B eine Teilmenge von V und a Element von V\B, so ist die kleinste Mächtigkeit einer a von B in G trennenden Teilmenge X von V\a gleich der größten Mächtigkeit eines a-B-Fächers. (Fächersatz)
- Sind a und b nicht benachbart, so ist die kleinste Mächtigkeit einer a von b trennenden Teilmenge von V\{a,b} gleich der gößten Mächtigkeit einer Menge disjunkter a-b-Wege in G.
- Die Knotenzusammenhangszahl ist höchstens so groß, wie die Kantenzusammenhangszahl und die höchstens so groß, wie der Minimalgrad
- Der Blockgraph G_B eines Graphen G ist ein Wald. G_B ist genau dann Baum (also zusammenhängend), wenn G zusammenhängend ist.

Algorithmen zum Finden von SZK

- **Serieller Algorithmus von Warshall**

  O(ln|V|)

  - Verwendet spezielle Adjazenzmatrizen, die multipliziert sogenannte Wegematrizen ergeben
- **Tripelalgorithmus**

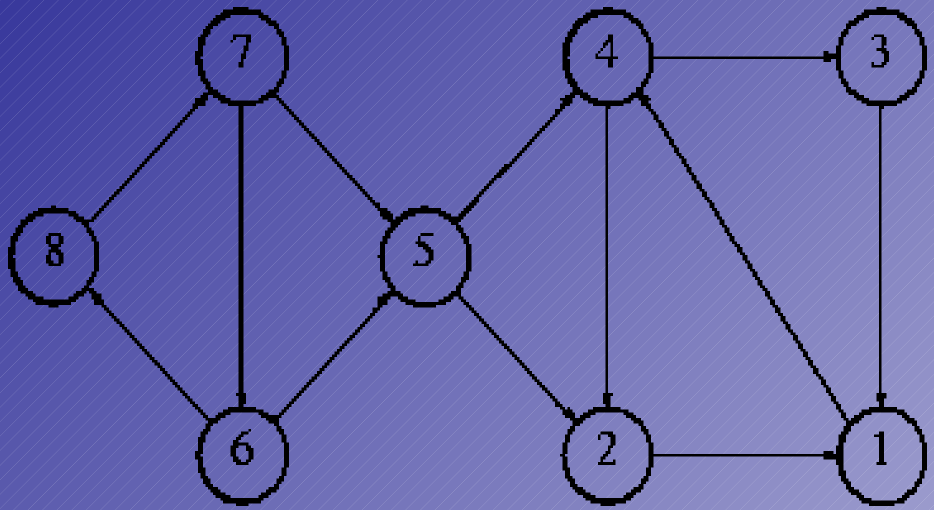
  (O(|V|³)

  - Wird hauptsächlich benutzt um kürzeste Wege in einem Graphen zu finden, bildet aber die transitive Hülle und kann so auch benutzt werden um SZK zu finden.
- **Tiefensuche**

  O(|V|+|E|)
- **Algorithmus von Tarjan zur Bestimmung von SZK**

  O(|V|+|E|)

Tiefensuche



1.

Führe eine Tiefensuche in G durch, wobei die Id eines Knotens vor dem Ende seiner Rekursion vergeben wird (nicht - wie ursprünglich - am Anfang).

2.

Bilde G' durch Umkehrung aller Kanten des Graphen G .

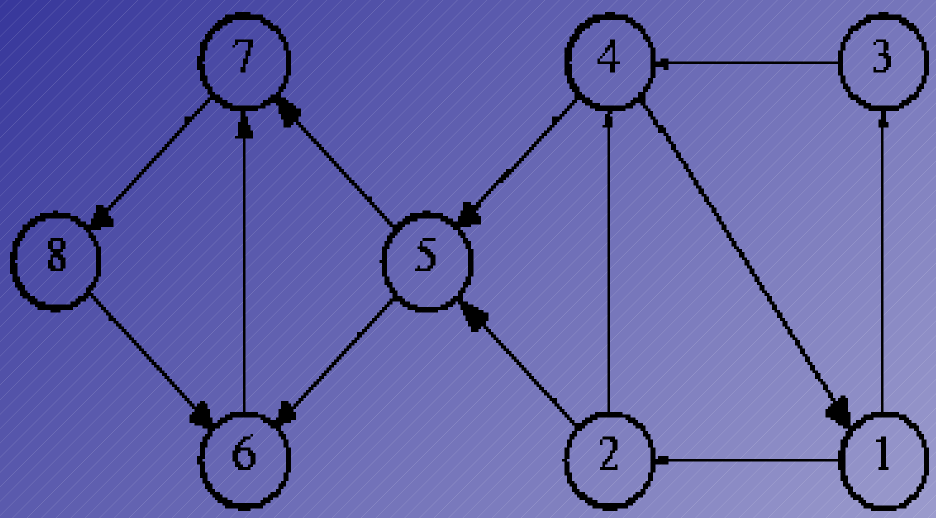
3.

Führe nun eine Tiefensuche in G' durch, wobei man jeweils mit den höchsten Ids aus Phase 1.) beginnt.

4.

Die Tiefensuchebäume aus Phase 3.) sind dann die sZHK vom Graphen G

Tiefensuche



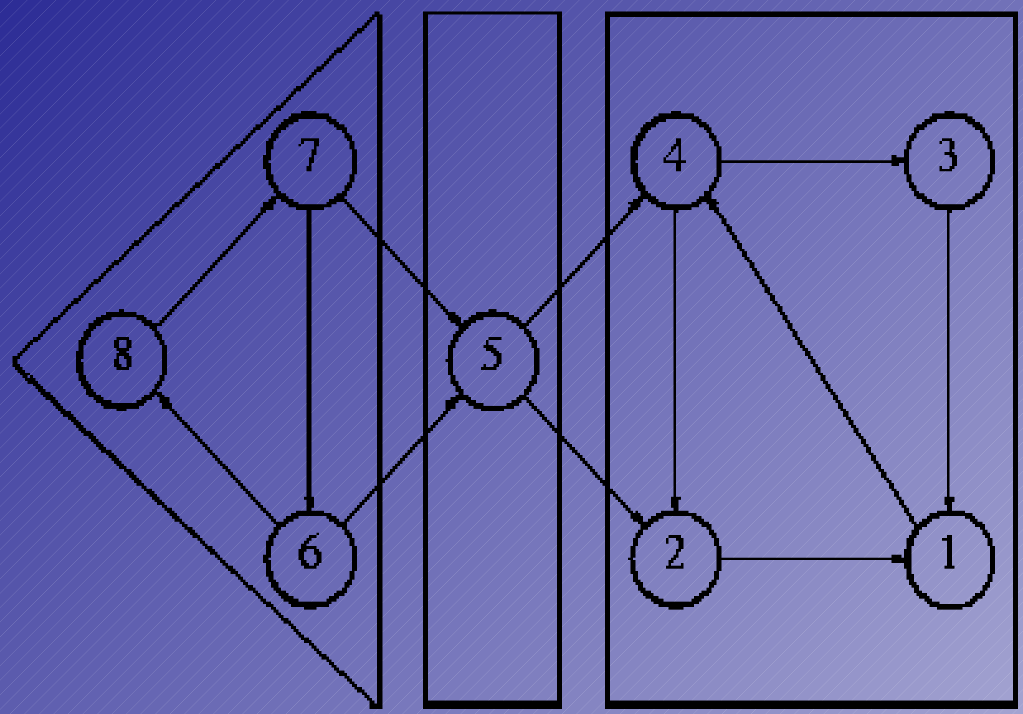
1.
Führe eine Tiefensuche in G durch, wobei die Id eines Knotens vor dem Ende seiner Rekursion vergeben wird (nicht - wie ursprünglich - am Anfang).

2.
Bilde G' durch Umkehrung aller Kanten des Graphen G .

3.
Führe nun eine Tiefensuche in G' durch, wobei man jeweils mit den höchsten Ids aus Phase 1.) beginnt.

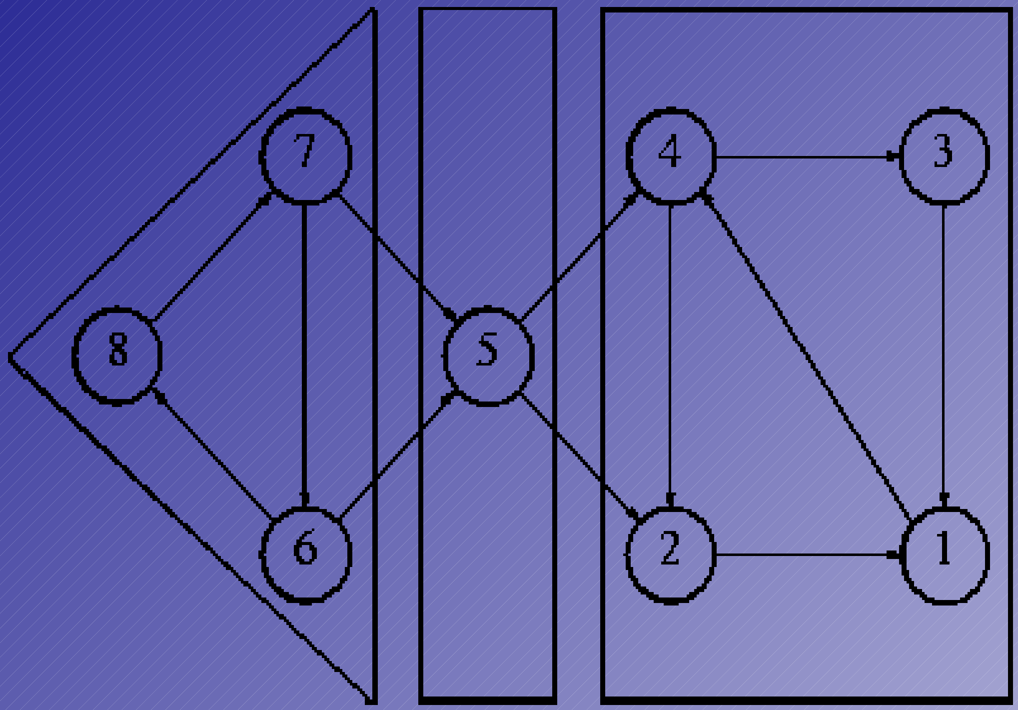
4.
Die Tiefensuchebäume aus Phase 3.) sind dann die sZHK vom Graphen G

Tiefensuche



1. Führe eine Tiefensuche in G durch, wobei die Id eines Knotens vor dem Ende seiner Rekursion vergeben wird (nicht - wie ursprünglich - am Anfang).
2. Bilde G' durch Umkehrung aller Kanten des Graphen G .
3. **Führe nun eine Tiefensuche in G' durch, wobei man jeweils mit den höchsten Ids aus Phase 1.) beginnt.**
4. Die Tiefensuchebäume aus Phase 3.) sind dann die sZHK vom Graphen G

Tiefensuche



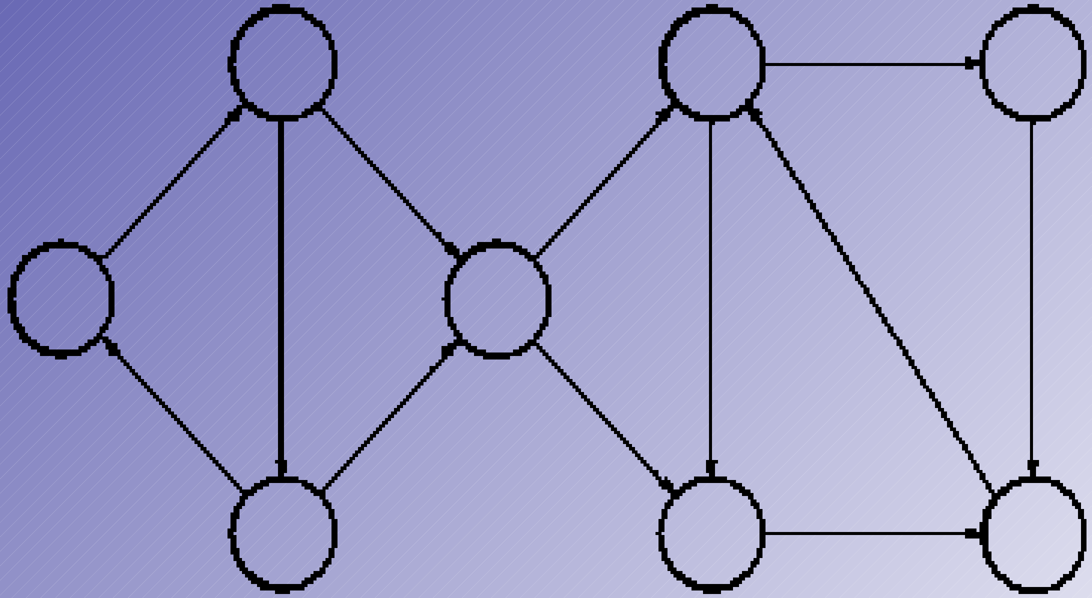
1.
   Führe eine Tiefensuche in G durch, wobei die Id eines Knotens vor dem Ende seiner Rekursion vergeben wird (nicht - wie ursprünglich - am Anfang).
2.
   Bilde G' durch Umkehrung aller Kanten des Graphen G .
3.
   Führe nun eine Tiefensuche in G' durch, wobei man jeweils mit den höchsten Ids aus Phase 1.) beginnt.
4.
   Die Tiefensuchebäume aus Phase 3.) sind dann die sZHK vom Graphen G

Algorithmus von Tarjan zur Bestimmung von SZK

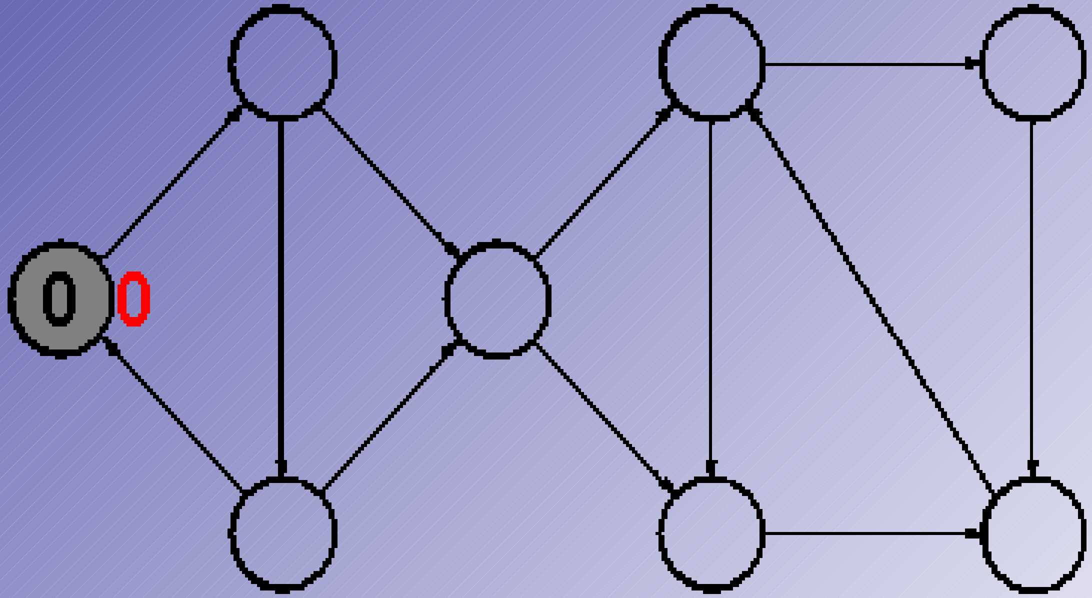
```
VAR Tarj: Stack von Knoten, maxdfs: Nat, weiss: Knotenmenge
(weiss := V), maxdfs = 0; Tarj := empty stack
dfs(v0);

dfs(v): v.dfs = v.lowlink = maxdfs; maxdfs += 1;
push(v,Tarj);(weiss := weiss \ {v})
FOR ALL v' ([v,v'] in E) DO
    IF v' in weiss THEN
        dfs(v')                                  //Baumkante
        v.lowlink = MIN(v.lowlink,v'.lowlink)
    ELSE
        IF v' on Tarj THEN
            v.lowlink = MIN(v.lowlink,v'.dfs)    //andere Kante
        END
    END
OD
IF v.lowlink = v.dfs THEN
    REPEAT
        v* = pop(Tarj)                           //eine SZK
    UNTIL v = v*
END
```

Algorithmus von Tarjan

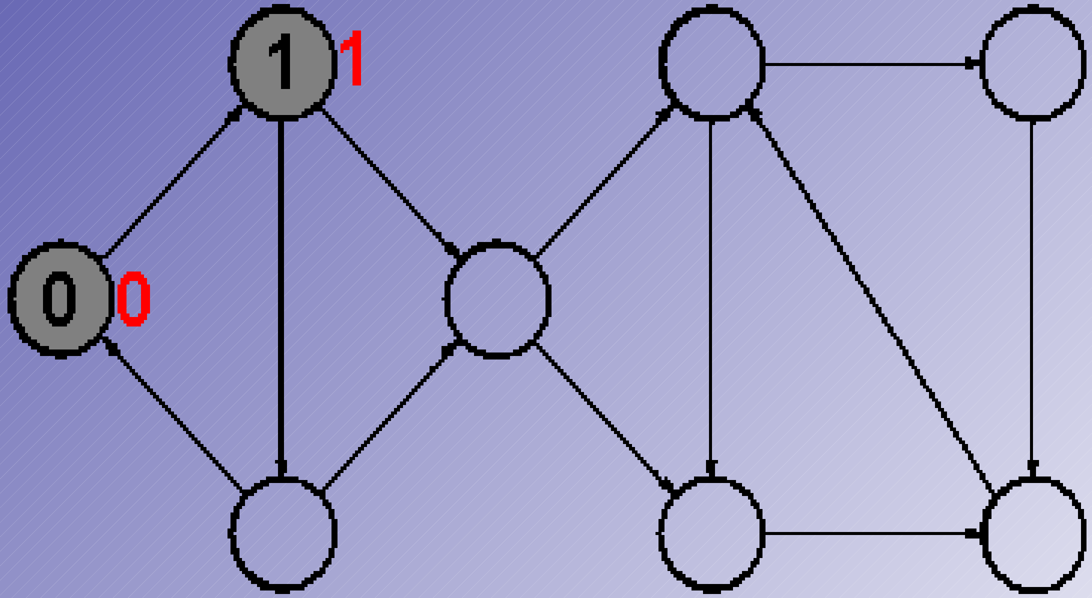


Algorithmus von Tarjan



0

Algorithmus von Tarjan



1

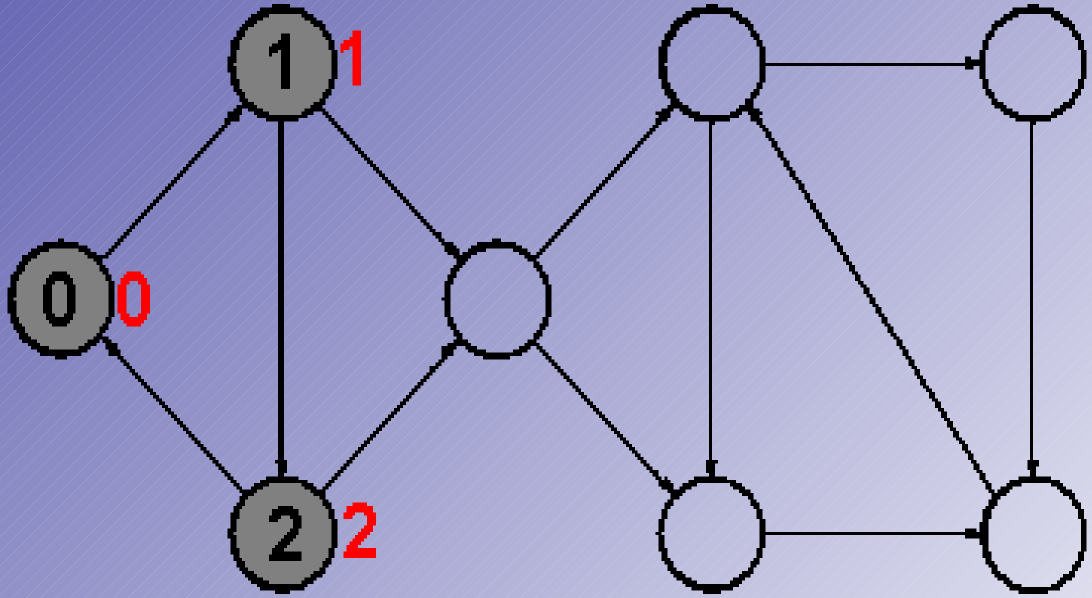
0

Algorithmus von Tarjan

2

1

0



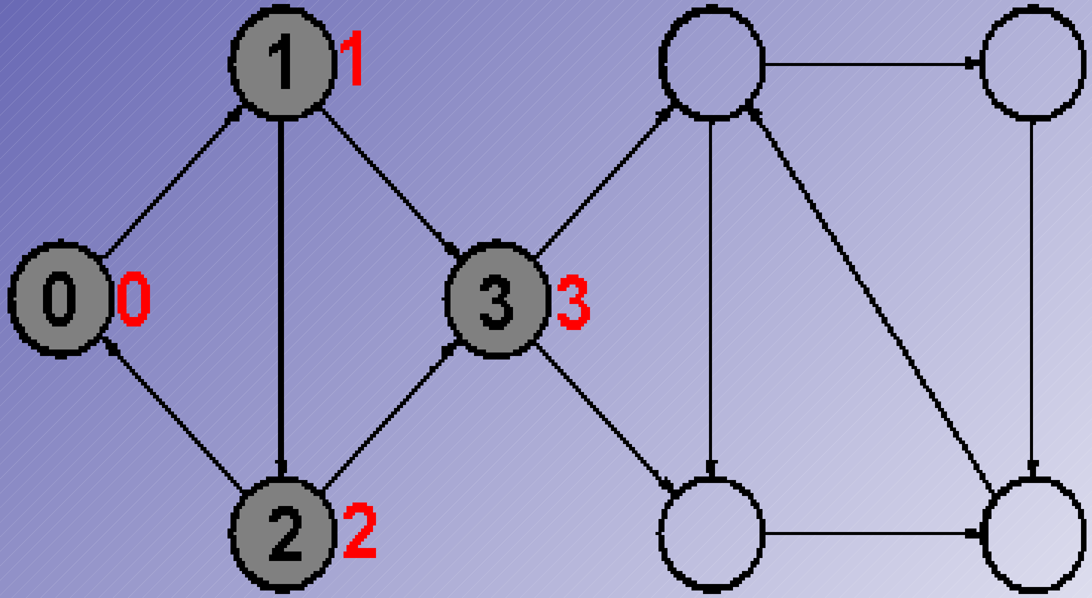
Algorithmus von Tarjan

3

2

1

0



Algorithmus von Tarjan

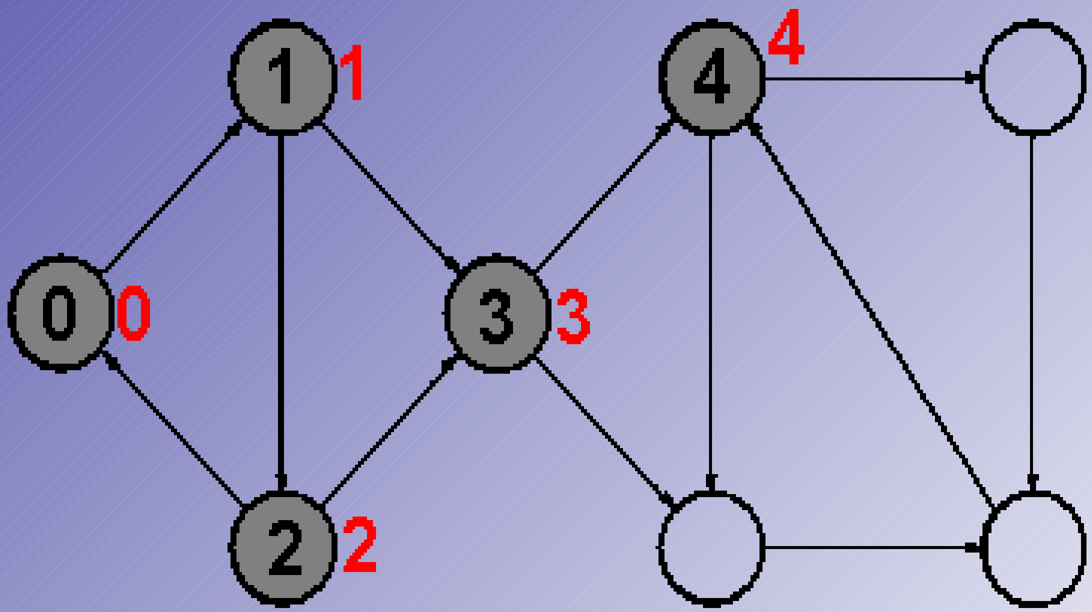
4

3

2

1

0



Algorithmus von Tarjan

5

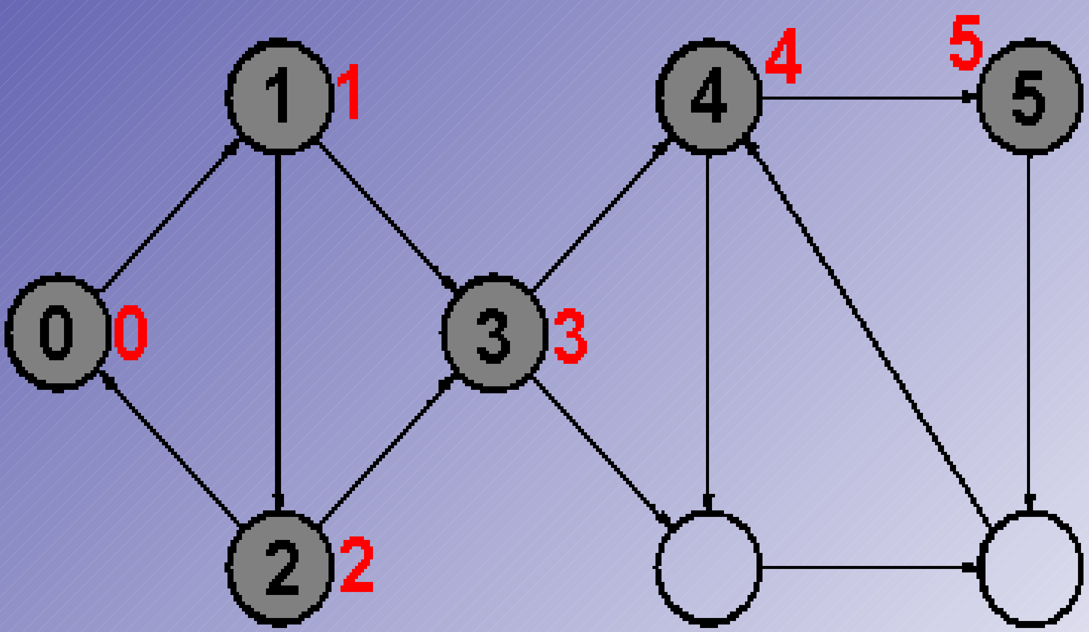
4

3

2

1

0



Algorithmus von Tarjan

6

5

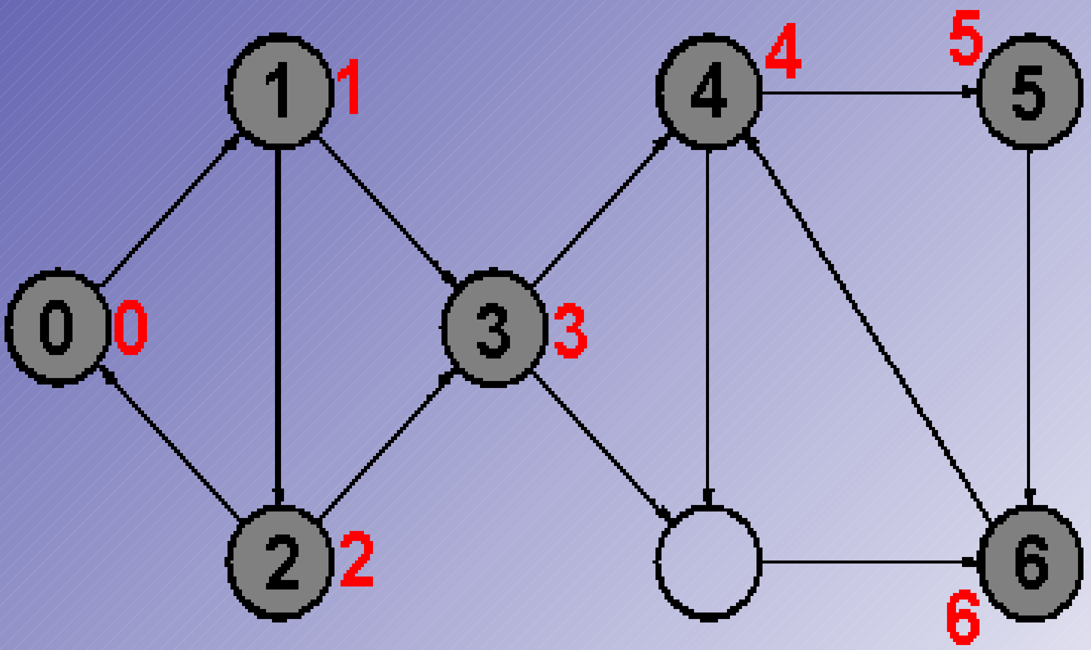
4

3

2

1

0



Algorithmus von Tarjan

6

5

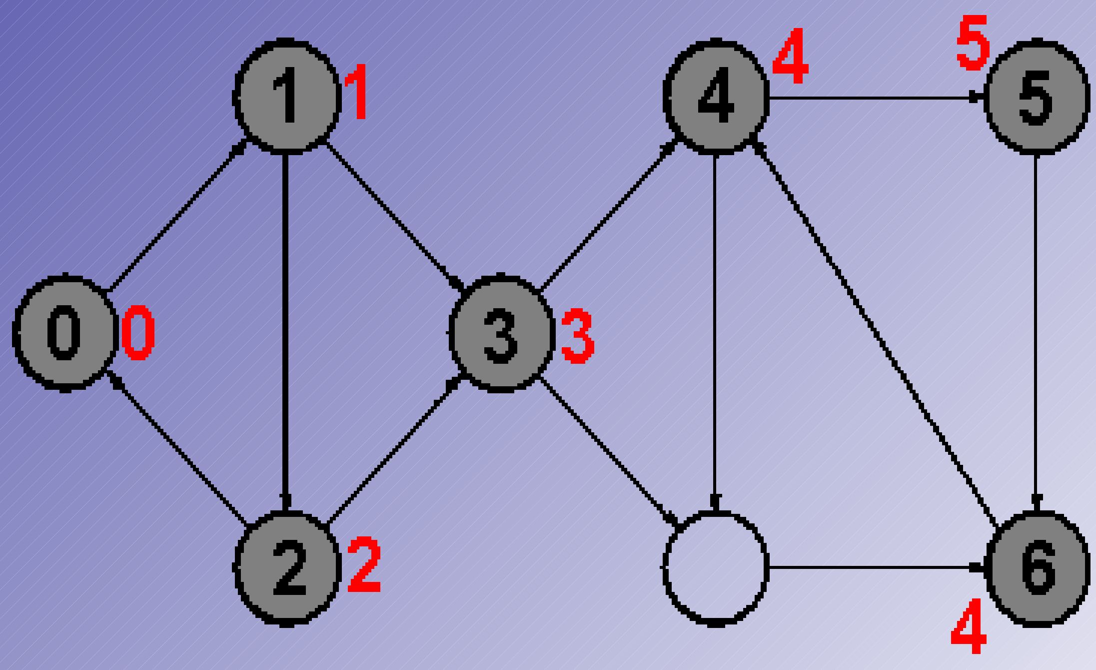
4

3

2

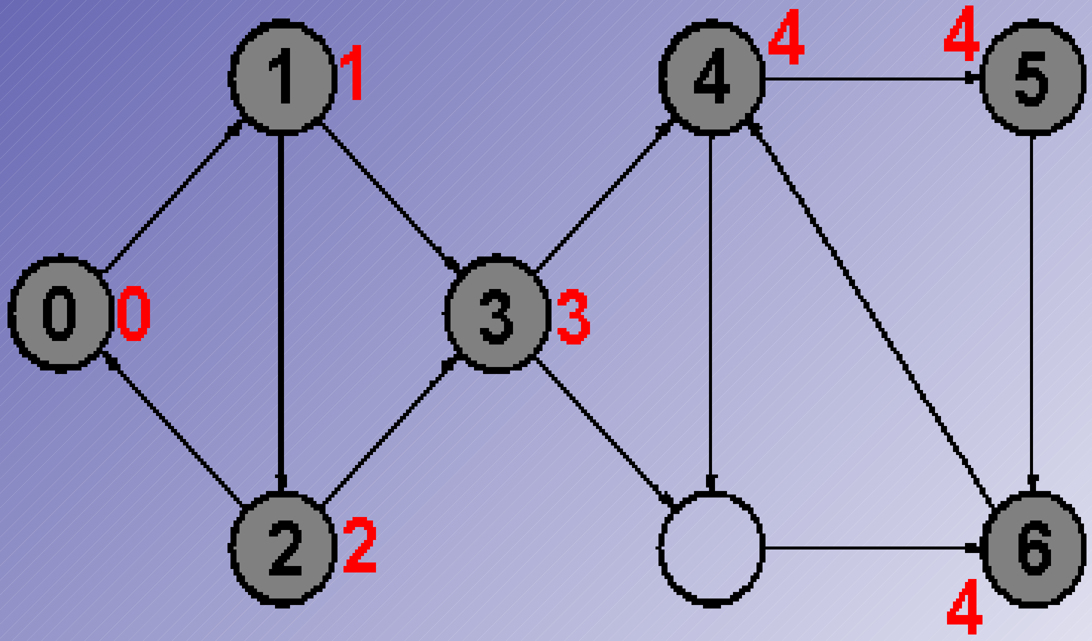
1

0



Algorithmus von Tarjan

6
5
4
3
2
1
0



Algorithmus von Tarjan

7

6

5

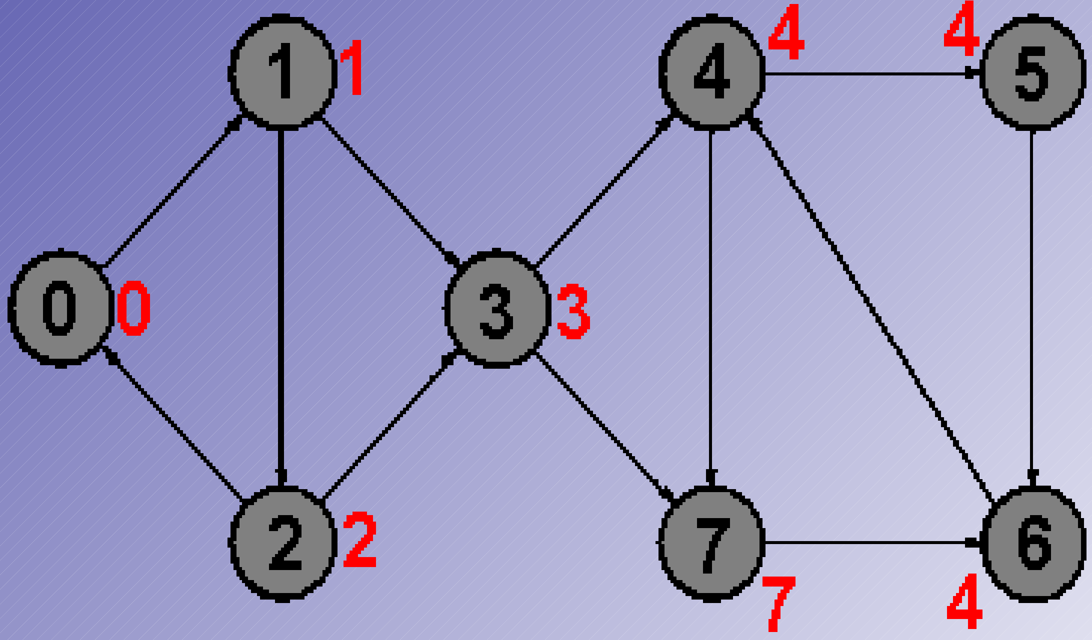
4

3

2

1

0



Algorithmus von Tarjan

7

6

5

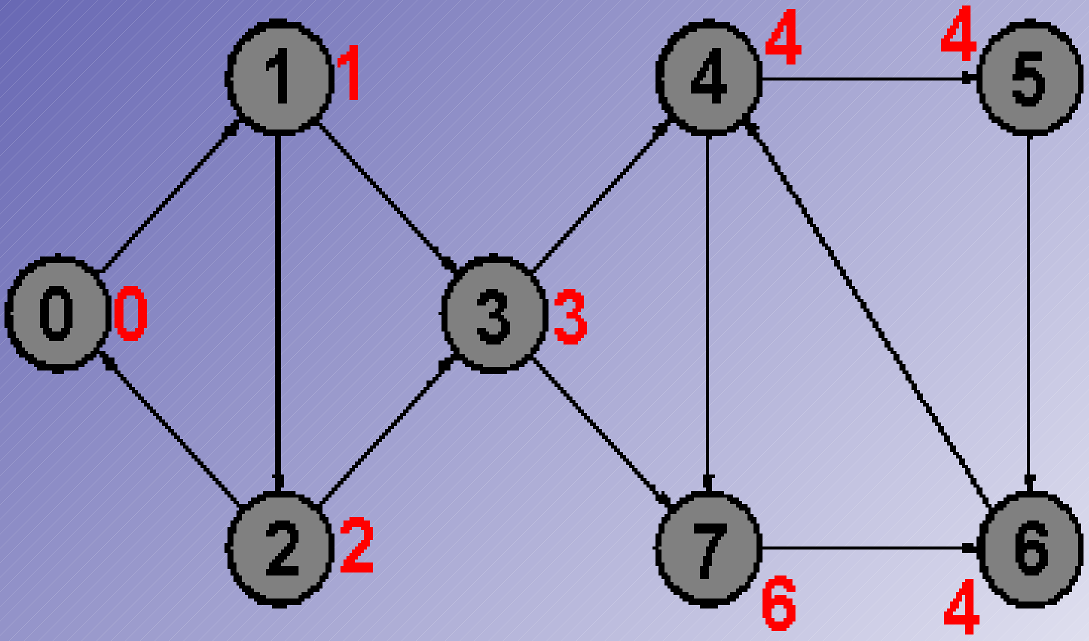
4

3

2

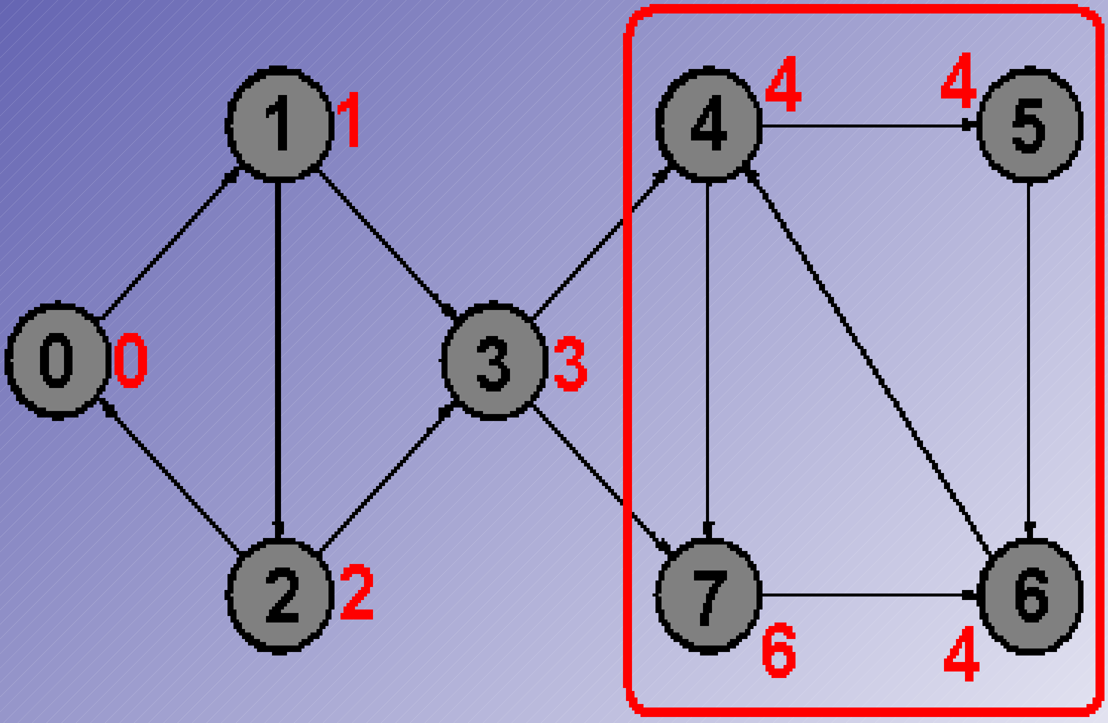
1

0



Algorithmus von Tarjan

7
6
5
4
3
2
1
0



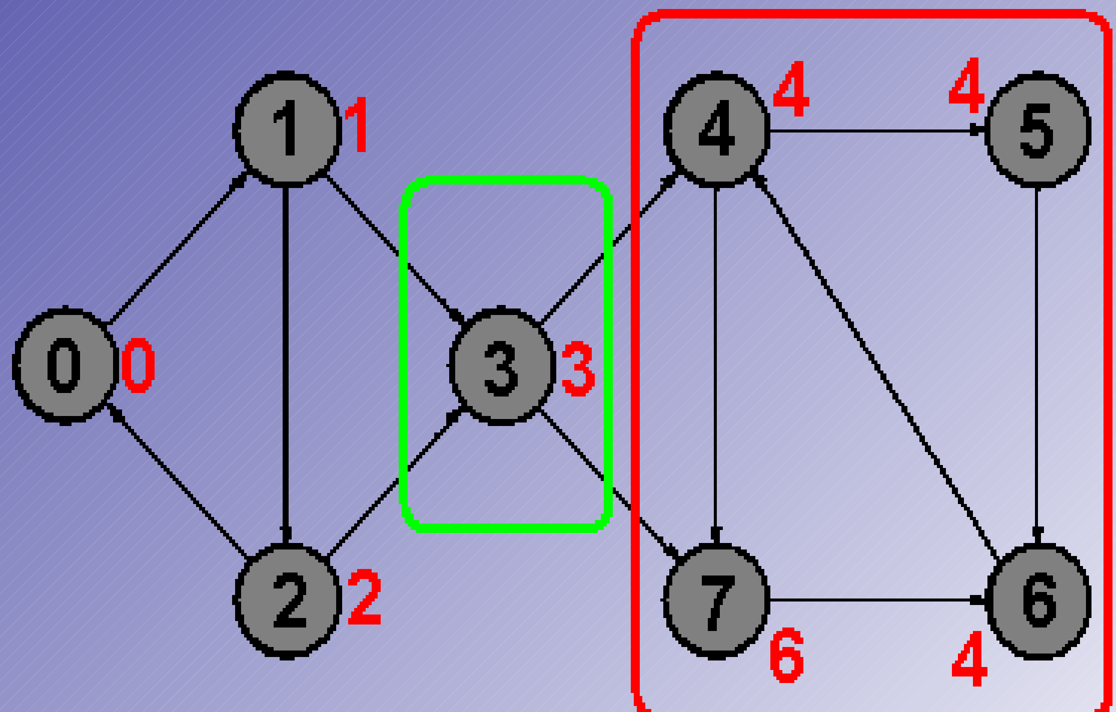
Algorithmus von Tarjan

3

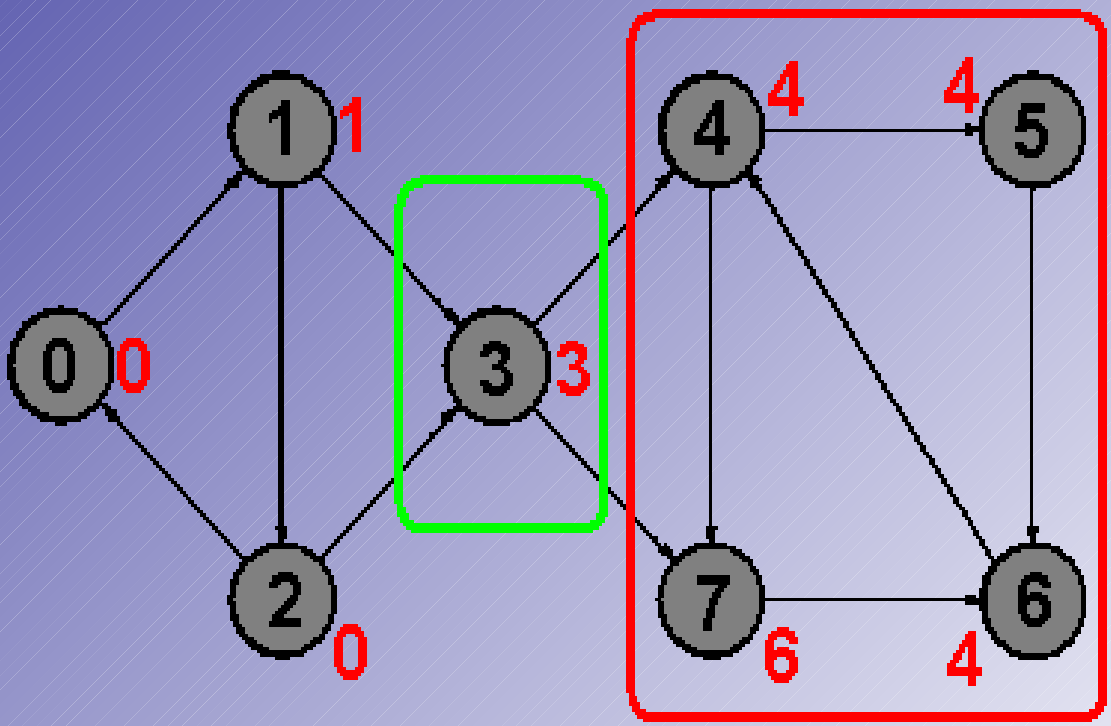
2

1

0



Algorithmus von Tarjan

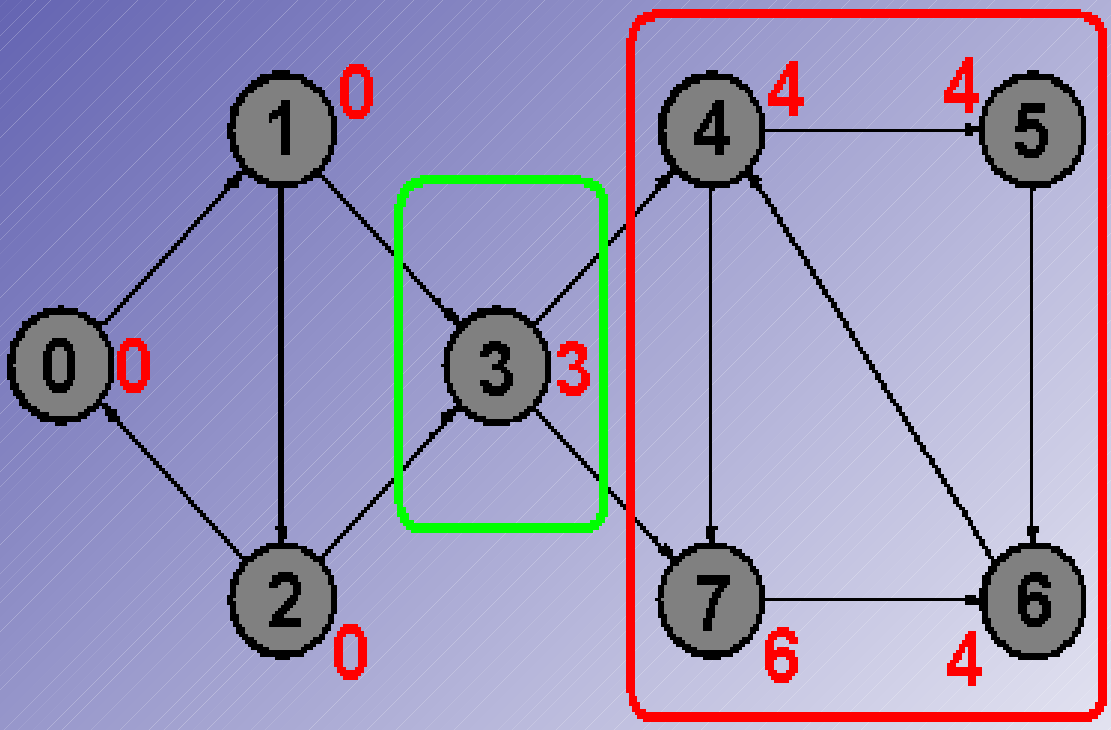


2

1

0

Algorithmus von Tarjan



2

1

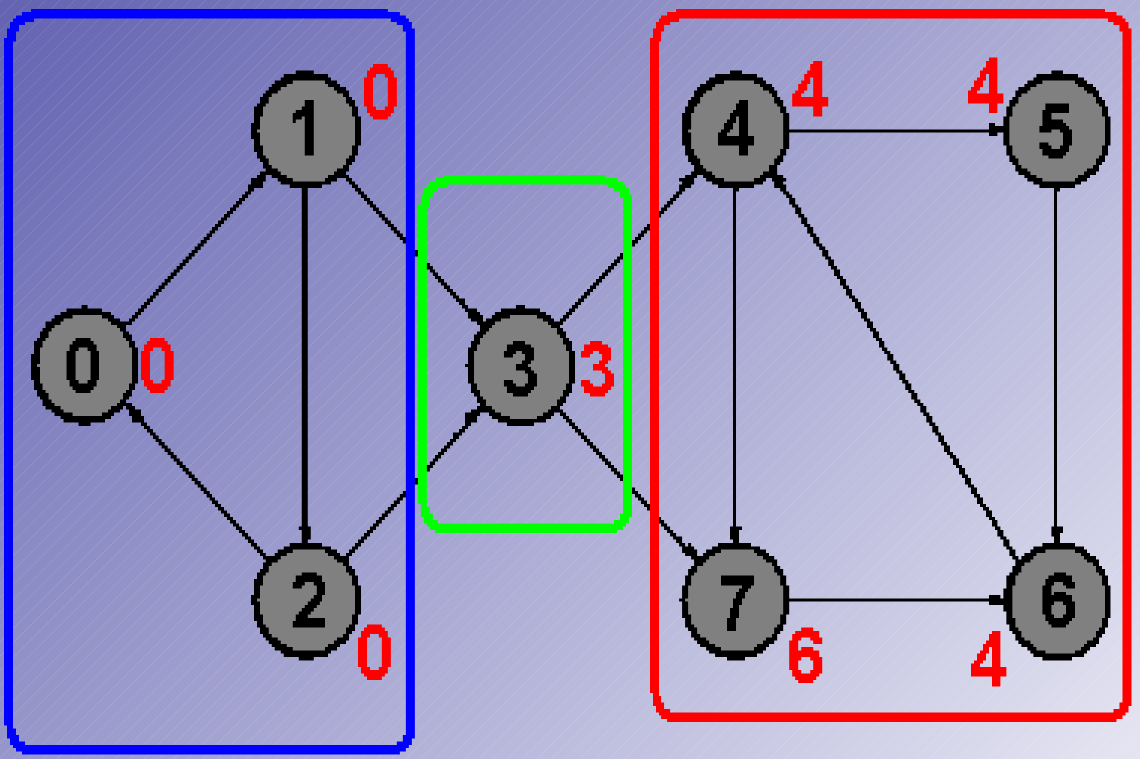
0

Algorithmus von Tarjan

2

1

0



Anwendungen

- KRUSKAL – Algorithmus
  - Das Problem, das mit diesem Algorithmus gelöst werden soll, ist die Suche nach einem einen Graphen aufspannenden Baum mit minmalem Gewicht
- Probleme bei Netzwerkanalysen
  - Frage: Wie lange benötigt man im Internet um von einer www-Side zu einer beliebigen andern zu kommen und dabei nur die Links auf der Seite zu benutzen
- Datenbankentwurf
  - Es existiert ein Algorithmus, der aus einem formulierten konzeptuellen Schema logische Strukturen gemäß dem Relationenmodell ableitet

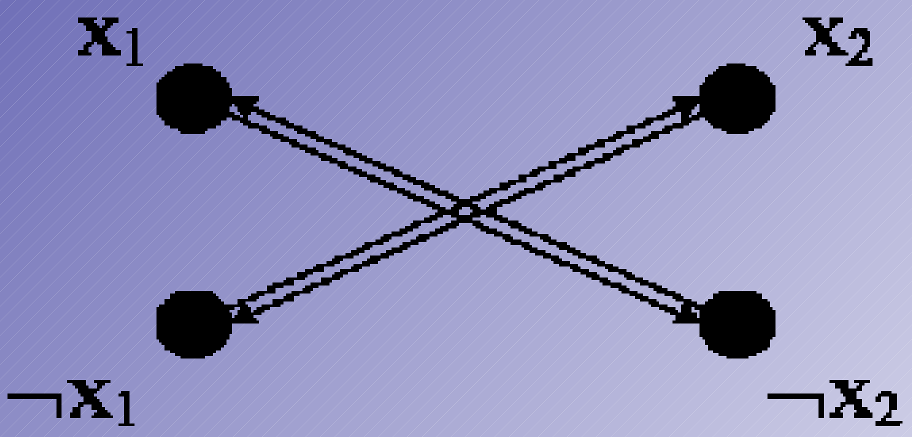
Anwendungen – 2 SAT Problem

1. Es ist ein 2-Sat Problem mit n Variablen x1, ..., xn gegeben.
2. Man definiert einen Graphen G = (V, E), wobei als Knoten x1, ¬x1, ..., xn, ¬xn herangezogen werden.
   Es liegen also 2n Knoten im Graphen G vor.
3. Die Elemente der Kantenmenge E wird folgendermaßen definiert:
   Enthält die Formel F des 2-Sat Problems die Klausel (xi ∨ xj), so enthält E die Kante (¬xi, xj) und (aufgrund der Kommutativität von ∨) die Kante (¬xj, xi).
4. Die Formel F des 2-Sat Problems ist genau dann erfüllbar, wenn es kein Literal xi gibt, so dass xi und ¬xi in derselben starken Zusammenhangskomponente liegen.

Beispiel - 2SAT

$F = (x1 \vee x2) \wedge (\neg x1 \vee \neg x2)$

daraus resultierender Graph:

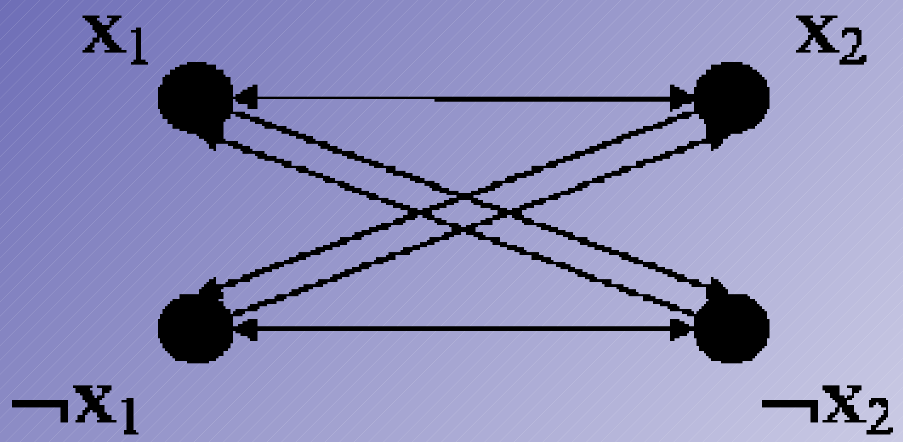


Man kann keinen Weg von x1 nach ¬x1 finden.
Ebenso kann man keinen Weg von x2 nach ¬x2. Finden.
Die zusammengehörigen Literale befinden sich also in keiner starken Zusammenhangskomponente.
-> Die Aussage ist erfüllbar!

Beispiel - 2SAT

$D = (\neg x1 \vee x2) \wedge (x1 \vee \neg x2) \wedge (\neg x1 \vee \neg x2) \wedge (x1 \vee x2)$

daraus resultierender Graph:



Man kann einen Weg von x1 über x2 nach ¬x1 erkennen.
Ebenso kann man von ¬x1 über x2 nach x1 gehen.
x1 und ¬x1 liegen also in einer starken Zusammenhangskomponente. Dasselbe gilt auch für x2 und ¬x2.

-> Die Aussage ist nicht erfüllbar!

4. Fragen?

Quellen

http://de.wikipedia.org/
http://www.jeckle.de/vorlesung/datenbanken/script.html

http://www-lehre.informatik.uni-osnabrueck.de/~graph/skript/skript.html
http://www.inf.fu-berlin.de/lehre/WS02/ALP3/material/alp3-27-Graph-3.pdf
http://www.tfh-berlin.de/~loopy/stud/grn/index.htm#zusammenhangskomponente
http://www.leda-tutorial.org/de/inoffiziell/ch05s03s02.html
http://www.vs.inf.ethz.ch
http://wwwipr.ira.uka.de

http://www.uni-ulm.de/~s_mmunz/studium/zushkomponenten.pdf
Michael Munz Universität Ulm
„Segmentierte parallele Präfixberechnung und Zusammenhangskomponenten eines Graphen“
27. Juli 2003
nverarbeitung
http://www.gup.uni-linz.ac.at/pgdv/slides/komponent.pdf
Parallele Graphische Datenverarbeitung (Kapitel 4: Zusammenhangskomponenten)
D. Kranzlmüller
GUP Linz Techn. Informatik und Telematik Joh. Kepler Universität Linz

Theoretische Informatik II (Übung zu Vorlesuang – Blatt 6)
Prof. Christoph Kreitz, Dipl. Math. Eva Richter
Universität Potsdam, Theoretische Informatik - Wintersemester 2003/04

Ausarbeitung über das
2-Sat Problem
Myriam Ezzedine, 0326943
Anton Ksernofontov, 0327064
Jürgen Platzer, 0025360
Nataliya Sokolovska, 0326991

Informatik II
V.Claus
12.-22.07.2004