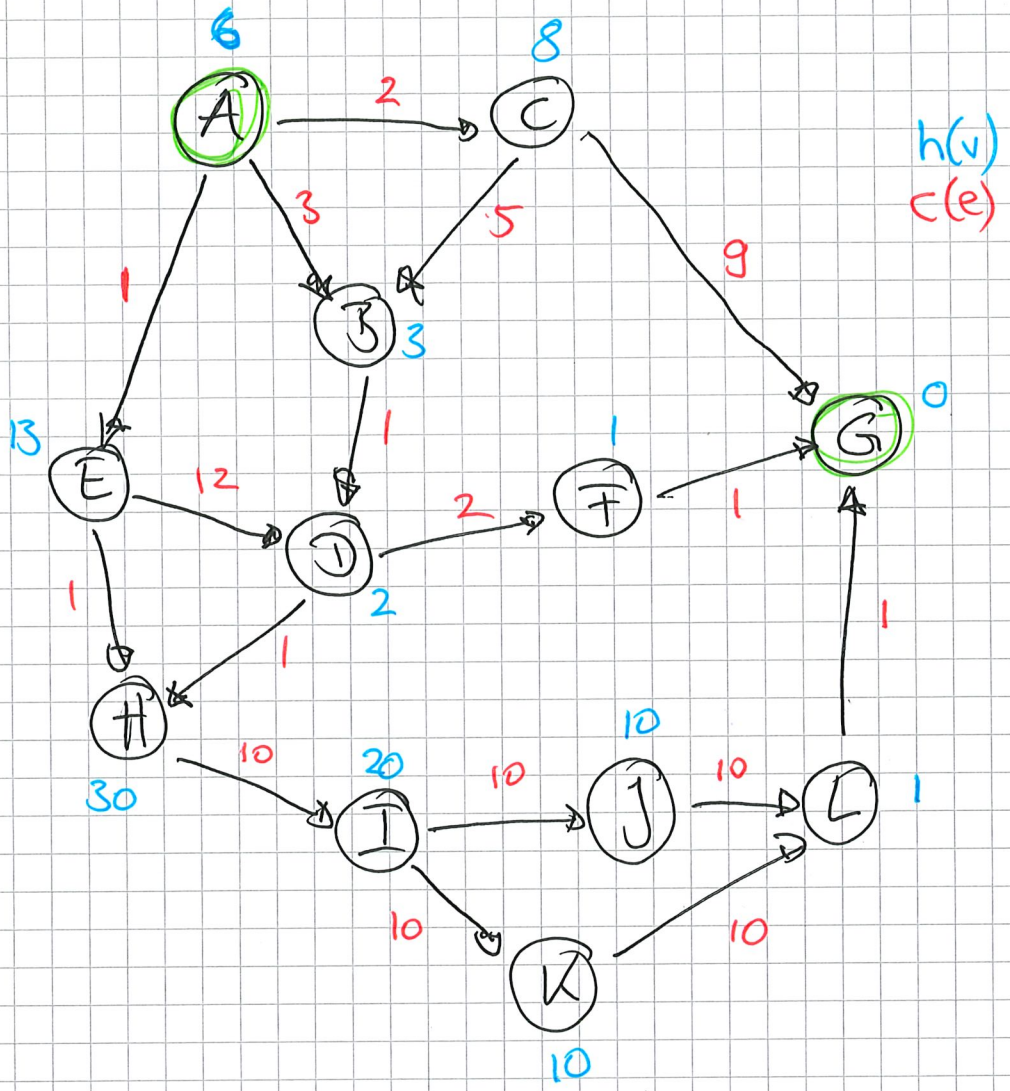
Beispiel zum A*-Algorithmus



1. E, $p(E) = A$, $g(E) = 1$, $f(E) = 14$
   B, $p(B) = A$, $g(B) = 3$, $f(B) = 6$ ←
   C, $p(C) = A$, $g(C) = 2$, $f(C) = 10$

2. D, $p(D) = B$, $g(D) = 4$, $f(D) = 6$ ←

3. F, $p(F) = D$, $g(F) = 6$, $f(F) = 7$ ←
   H, $p(H) = D$, $g(H) = 5$, $f(H) = 35$

4. G, $p(G) = F$, $g(G) = 7$, $f(G) = 7$ ←

Zusammenhangskomp in ungerichteten Graphen?

• Starke Zusammenhangskomponenten

→ gerichtete Kreise in gerichteten Graphen

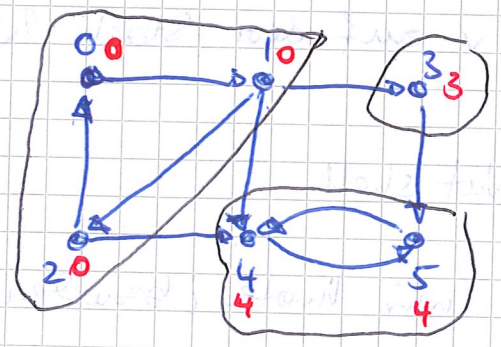
→ Starke Zshgs-Komp sind eindeutig

→ es gibt keinen Weg aus einer Komponente raus und wieder hinein...

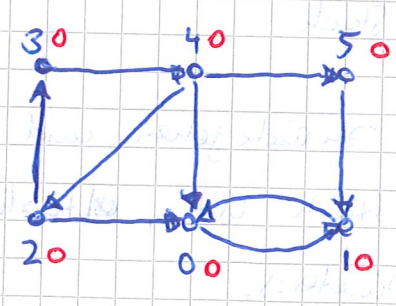
Konzept "low-link value":

→ kleinste Id die von Knoten erreichbar ist (geschlossen!)

→ DFS zum Labeln der Knoten



jeder Knoten mit selbem low-link value gehört zur selben SCC... ?!



abhängig davon wo DFS startet und welche Knoten verwendet werden, können die low-link values 50 nicht verwendet werden...

→ Invariante aufrecht erhalten, dass low-link values verschiedener SCCs nicht "interagieren" / sich beeinflussen

"falsch"

- Von welchen Knoten low-link value übernehmen?

→ Stack ... • füge Knoten zu Stack hinzu wenn sie das erste Mal besucht wurden
• entferne Knoten, wenn SCC gefunden

low-link update Bedingung:

Seien u, v Knoten und wir finden gerade u, dann ~~wenn~~ aktualisieren wir u's low-link zu v's low-link, wenn es einen Pfad von u zu v gibt und v auf dem Stack liegt...

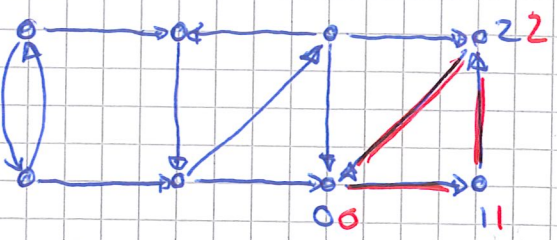
Tajans Algorithmus in a Nut shell

1. Starte DFS. Wenn wir Knoten i besuchen geben wir ihm id = i und low-link = i und packen ihn auf einen Stack

2. Wenn wir im DFS-Baum zurück gehen und der vorherige Knoten auf dem Stack ist, aktualisieren wir den low link des Knotens.

3. Wenn alle Nachbarn besucht wurden und der aktuelle Knoten eine SCC gefunden hat, schreiben wir alle Knoten vom Stack die zur SCC gehören.

$lowlink(v_i) = id(v_i)$



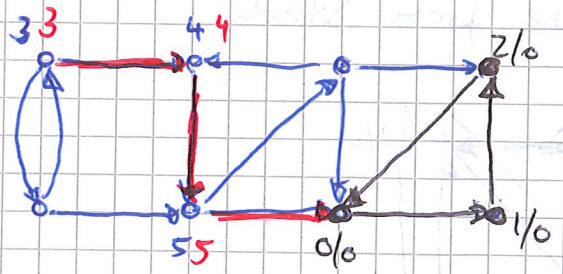
Stack
0
1
2

0 on Stack → lowlink[2] = min(lowlink[2], lowlink[0]) = 0
→ v_2 → 0

2 on Stack → lowlink[1] = min(lowlink[1], lowlink[2]) = 0
→ v_1 → 0

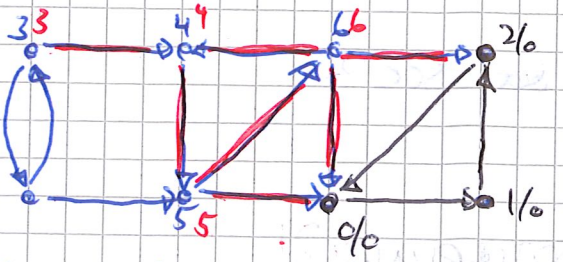
1 on Stack → lowlink[0] = min(lowlink[0], lowlink[1]) = 0
→ v_0 → 0

- Wir haben eine SCC gefunden, weil wir alle Nachbarn von 0 besucht haben (und 1 / 2) und lowlink[0] = id[0] = 0 → entferne alle Knoten der Komponente von Stack!



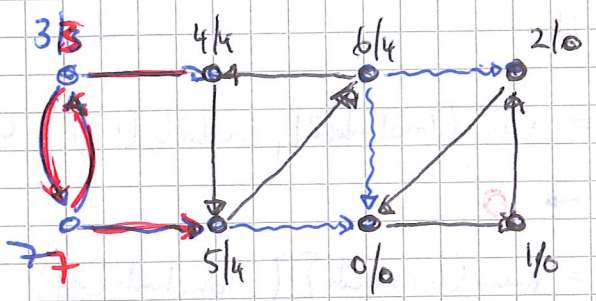
Stack
3
4
5

- Wenn wir die Kante (5,0) besuchen wollen und backtracken weil 0 schon besucht wurde, wird lowlink[5] nicht aktualisiert, da Knoten v_0 nicht auf dem Stack liegt!



Stack
3
4
5
6

→ lowlinks: 6/4, 5/4, 4/4 → remove (6,5,4) vom Stack...



Stack
3
7

Wenn wir jetzt zu v_3 zurückkommen, können wir nichts mit dem lowlink von v_4 machen weil vom Stack gestrichen... aber lowlink[3] = id[3] = 3

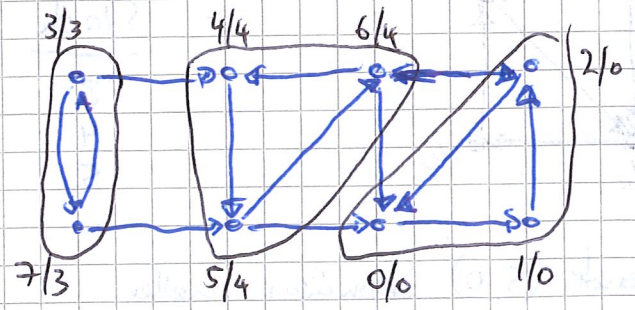
→ Start eines SCC ?!

→ noch nicht alle ~~besuchten~~ Nachbarn besucht

→ update lowlink[7] nicht mit v_5

→ update lowlink[7] mit v_3

→ Scc found... remove from Stack



→ Was passiert mit lowlinks von 6 und 5, wenn es eine Kante (4,3) geben würde?

→ Was passiert mit der SCC?

→ Stack?

→ TOPOLOGISCHE SORTIERUNG?!