Datenstrukturen und Algorithmen

Cedric Aebi 17-103-235

Nicolas Müller 17-122-094

Datenstrukturen und Algorithmen
Übung 11, Frühling 2018

10. Mai 2017

Abgabe: Diese Übung muss zu Beginn der Übungsstunde bis spätestens um 16 Uhr 15 am 17. Mai abgegeben werden. Die Abgabe der DA Übungen erfolgt immer in schriftlicher Form auf Papier. Programme müssen zusammen mit der von ihnen erzeugten Ausgabe abgegeben werden. Drucken Sie wenn möglich platzsparend 2 Seiten auf eine A4-Seite aus. Falls Sie mehrere Blätter abgeben heften Sie diese bitte zusammen (Büroklammer, Bostitch, Mäppchen). *Der gesammte Sourcecode muss ausserdem elektronisch über Ilias abgegeben werden.*

Die Übung sollte vorzugsweise in Zweiergruppen bearbeitet werden, kann aber auch einzeln abgegeben werden. Vergessen Sie nicht, Ihren Namen und Ihre Matrikelnummer auf Ihrer Abgabe zu vermerken. Jede Übungsserie gibt 10 Punkte. Im Durchschnitt müssen Sie 7 von 10 Punkten erreichen, um die Testatbedingungen zu erfüllen.

Theoretische Aufgaben

1. Das Quadrat eines gerichteten Graphen $G = (V, E)$ ist der Graph $G^2 = (V, E^2)$, in dem zwei Knoten u und w genau dann durch eine Kante verbunden sind, wenn für ein $v \in V$ sowohl $(u, v) \in E$ als auch $(v, w) \in E$ gilt. Das heisst, in G^2 gibt es genau dann eine Kante zwischen u und w, wenn es in G einen Pfad zwischen u und w gibt, der aus genau zwei Kanten besteht. Geben Sie jeweils einen effizienten Algorithmus an (Pseudocode), der G^2

   (a) aus der Adjazenzlisten-Darstellung

   (b) aus der Adjazenzmatrix-Darstellung

   berechnet. Analysieren Sie die Laufzeiten Ihrer Algorithmen. **2 Punkte**

2. Ermitteln Sie für jeden Knoten die Werte für d (Distanz in Anzahl Kanten) und π (Vorgänger im Breitensuchbaum), die sich durch das Ausführen einer Breitensuche auf dem Graphen in Abbildung 1 unten ergeben. Verwenden Sie Knoten b als Startknoten. Stellen Sie den Breitensuchbaum graphisch dar. **1 Punkt**

3. Zeigen Sie, dass bei einer Breitensuche der Wert $d[u]$ eines Knotens unabhängig von der Reihenfolge ist, in der die Knoten in den Adjazenzlisten gegeben sind. **1 Punkt**

4. Zeigen Sie anhand des Graphen in Abbildung 1, dass der durch BFS erzeugte Breitensuchbaum von der Reihenfolge innerhalb der Adjazenzlisten abhängen kann. **1 Punkt**

5. Zeigen Sie, wie die Tiefensuche auf dem Graphen in Abbildung 2 arbeitet. Bestimmen Sie für jeden Knoten die Entdeckungszeit $d[u]$, die Endzeit $f[u]$, den Vorgänger $\pi[u]$, sowie die Klassifikation jeder Kante. Stellen Sie den Tiefensuchwald graphisch dar. Nehmen Sie an, dass die Knoten in der Hauptschleife in aufsteigender Folge der Indizes bearbeitet werden, und dass die Knoten in aufsteigender Folge der Indizes in den Adjazenzlisten gespeichert sind. **2 Punkte**

6. Geben Sie Gegenbeispiele zu folgenden Vermutungen an:

   (a) Falls ein gerichteter Graph G einen Pfad von u nach v enthält und bei der Tiefensuche auf G $u.d < v.d$ gilt, dann ist v im erzeugten Tiefensuchwald ein Nachfahre von u.

   (b) Falls ein gerichteter Graph G einen Pfad von u nach v enthält, dann muss jede Tiefensuche zu $v.d \leq u.f$ führen.

   2 Punkte

7. Erklären Sie, wie es passieren kann, dass ein Knoten u eines gerichteten Graphen in einem Tiefensuchbaum landet, der allein u enthält, obwohl u sowohl eintretende als auch austretende Kanten hat. **1 Punkt**

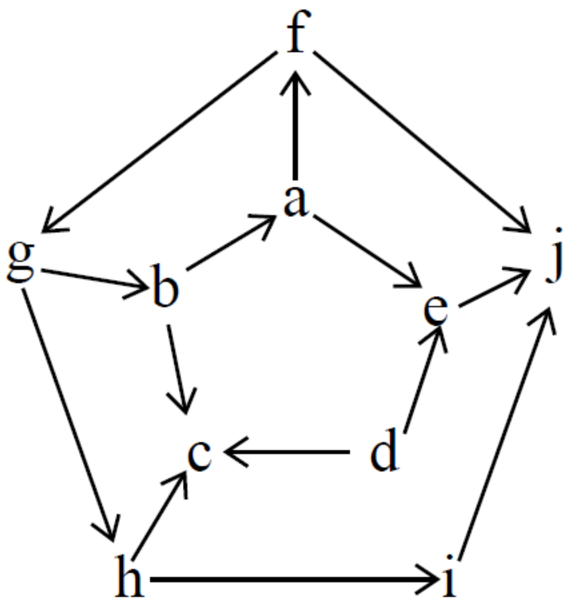


Abbildung 1: Graph für Aufgabe 2 und 4.

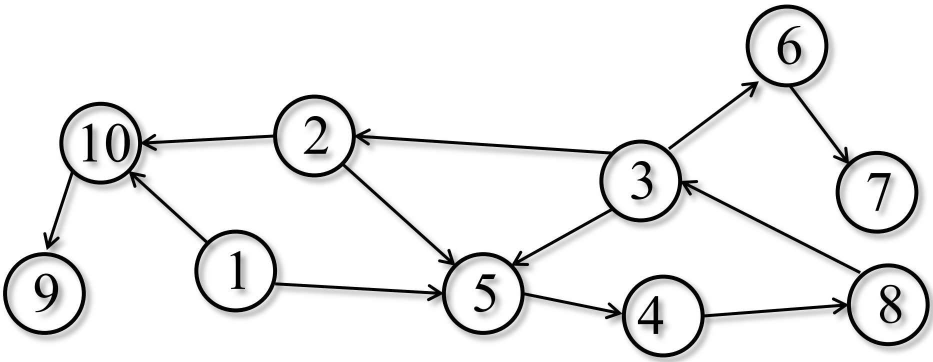


Abbildung 2: Graph für Aufgabe 5.

1. Das Quadrat eines gerichteten Graphen $G = (V, E)$ ist der Graph $G^2 = (V, E^2)$, in dem zwei Knoten u und w genau dann durch eine Kante verbunden sind, wenn für ein $v \in V$ sowohl $(u, v) \in E$ als auch $(v, w) \in E$ gilt. Das heisst, in G^2 gibt es genau dann eine Kante zwischen u und w, wenn es in G einen Pfad zwischen u und w gibt, der aus genau zwei Kanten besteht. Geben Sie jeweils einen effizienten Algorithmus an (Pseudocode), der G^2

   (a) aus der Adjazenzlisten-Darstellung

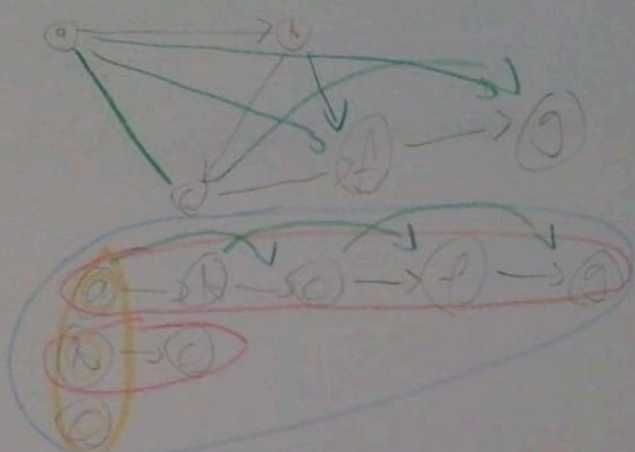
   (b) aus der Adjazenzmatrix-Darstellung

   berechnet. Analysieren Sie die Laufzeiten Ihrer Algorithmen. **2 Punkte**

a)

1 DA 11

Adjazenzlisten:



Pseudo-code

```
for each Vertex u ∈ (G.V)                          // if sublist is bigger
  if u.size > 2                                     than 2.
    for each Vertex v ∈ G.Adj[u]
      if (u.next.next != null && !E(u,u.next.next)) // if there is an
                                                    // "Overnext" vertex
        new Edge (u, u.next.next)                   / and they are not
                                                    // connected.
```

Laufzeit $= O(V \cdot E)$

b)

Adjazenzmatrix:

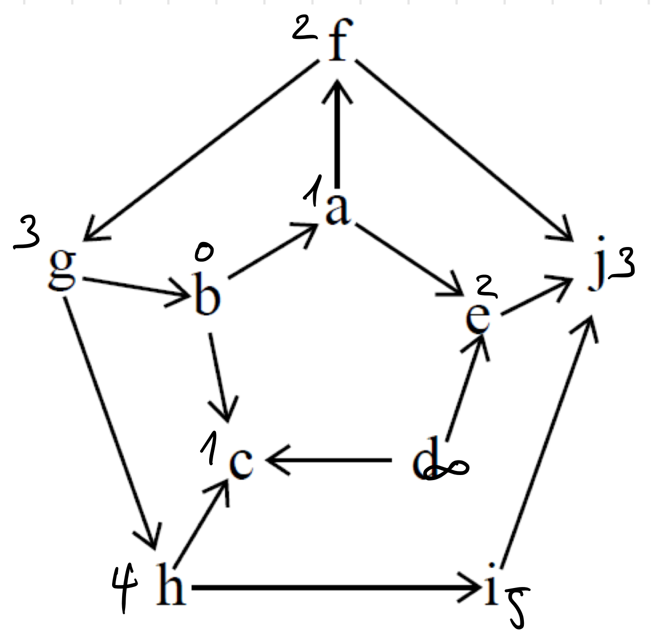
\	a	b	c	f	g
a	0	1	1	1	1
b	0	0	1	1	1
c	0	0	0	1	1
f	0	0	0	0	1
g	0	0	0	0	0

```
V[5,5]   (x y)
for each row x
  for each column y
    if V[x,y] == 1
      copyRow(V,y,x)
```

```
copyRow(Matrix V, row 1, row
  for each column x
    if (V[x,row1] == 1)
      V[x,row2] = 1
```

Laufzeit $= O(V^3)$

2. Ermitteln Sie für jeden Knoten die Werte für d (Distanz in Anzahl Kanten) und π (Vorgänger im Breitensuchbaum), die sich durch das Ausführen einer Breitensuche auf dem Graphen in Abbildung 1 unten ergeben. Verwenden Sie Knoten b als Startknoten. Stellen Sie den Breitensuchbaum graphisch dar. **1 Punkt**



b, a, c, e, f, j, g, h, i

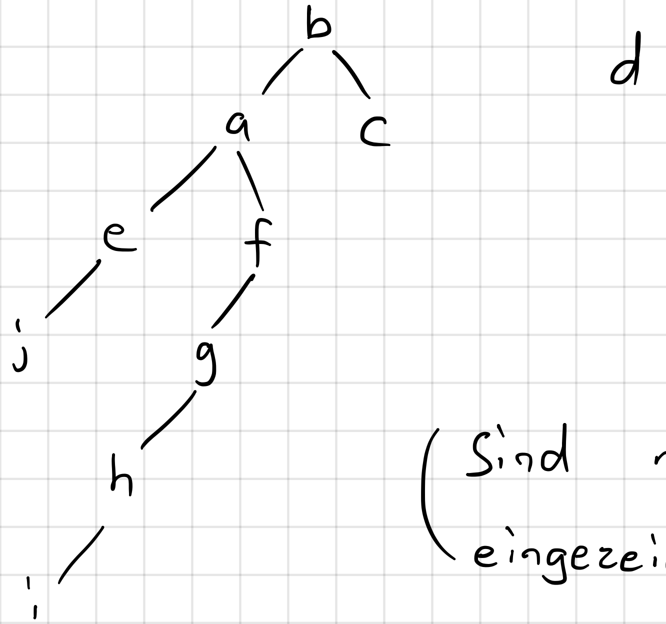
Abbildung 1: Graph für Aufgabe 2 und 4.

$b.d = 0$ $a.d = 1$ $c.d = 1$ $e.d = 2$ $f.d = 2$ $j.d = 3$ $g.d = 3$ $h.d = 4$

$i.d = 5$ $d.d = \infty$

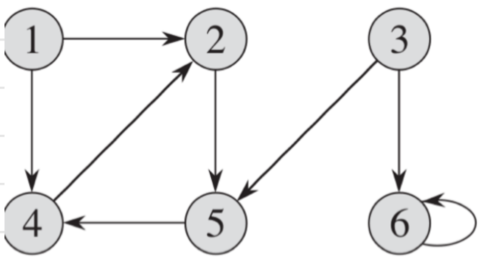
$b.\pi = NIL$ $a.\pi = b$ $c.\pi = b$ $e.\pi = a$ $f.\pi = a$ $g.\pi = f$

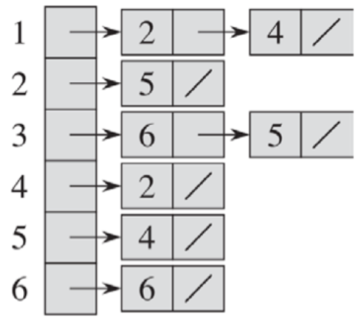
$h.\pi = g$ $i.\pi = h$ $j.\pi = e$ $d.\pi = NIL$



(Sind nur T-Kanten eingezeichnet)

3. Zeigen Sie, dass bei einer Breitensuche der Wert $d[u]$ eines Knotens unabhängig von der Reihenfolge ist, in der die Knoten in den Adjazenzlisten gegeben sind. **1 Punkt**





	1	2	3	4	5	6
1	0	1	0	1	0	0
2	0	0	0	0	1	0
3	0	0	0	0	1	1
4	0	1	0	0	0	0
5	0	0	0	1	0	0
6	0	0	0	0	0	1

Sieht man sich folgendes Beispiel eines gerichteten Graphen an. d[u] ist der kürzeste Abstand von einem gegebenen Startknoten zu einem verbundenen Nachbarn. Ob man nun von 1 aus zuerst 4 entdeckt oder zuerst 2 spielt keine Rolle. d[u] ist 1 so oder so. d[u] muss laut Definition der kürzeste Abstand sein.

4. Zeigen Sie anhand des Graphen in Abbildung 1, dass der durch BFS erzeugte Breitensuchbaum von der Reihenfolge innerhalb der Adjazenzlisten abhängen kann. **1 Punkt**

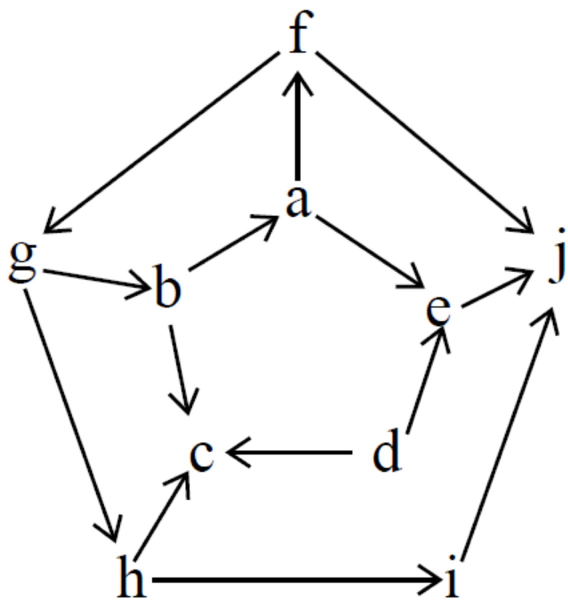


Abbildung 1: Graph für Aufgabe 2 und 4.

Vergleiche $FIFO_1$ mit $FIFO_2$. Es kommt drauf an, ob man von a zuerst f oder e entdeckt (Reihenfolge innerhalb Ajazentliste). Somit kann j entweder Kind von e, oder aber f sein im Baum.

$FIFO_1$ = b, a, c, e, f, j, g, h, i

$FIFO_2$ = b, a, c, f, e, j, g, h, i

5. Zeigen Sie, wie die Tiefensuche auf dem Graphen in Abbildung 2 arbeitet. Bestimmen Sie für jeden Knoten die Entdeckungszeit $d[u]$, die Endzeit $f[u]$, den Vorgänger $\pi[u]$, sowie die Klassifikation jeder Kante. Stellen Sie den Tiefensuchwald graphisch dar. Nehmen Sie an, dass die Knoten in der Hauptschleife in aufsteigender Folge der Indizes bearbeitet werden, und dass die Knoten in aufsteigender Folge der Indizes in den Adjazenzlisten gespeichert sind. **2 Punkte**

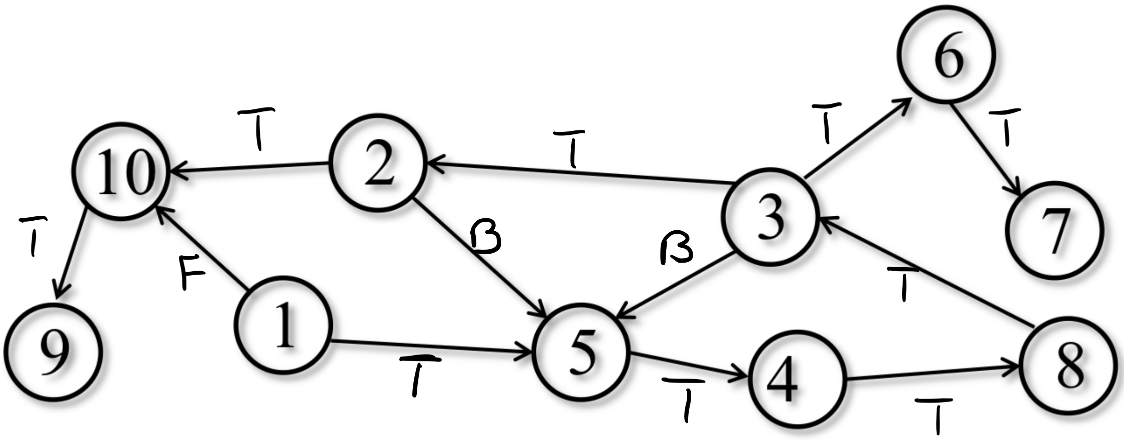
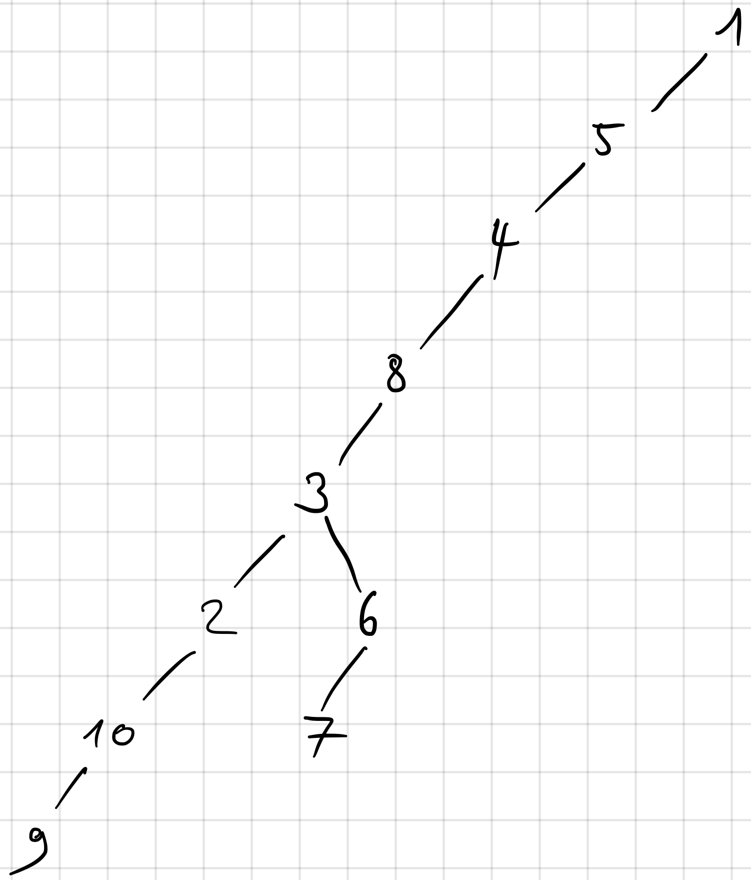


Abbildung 2: Graph für Aufgabe 5.

Start bei 1.

$d[1] = 1$	$f[1] = 20$	$\pi[1] = NIL$
$d[2] = 6$	$f[2] = 11$	$\pi[2] = 3$
$d[3] = 5$	$f[3] = 16$	$\pi[3] = 8$
$d[4] = 3$	$f[4] = 18$	$\pi[4] = 5$
$d[5] = 2$	$f[5] = 19$	$\pi[5] = 1$
$d[6] = 12$	$f[6] = 15$	$\pi[6] = 3$
$d[7] = 13$	$f[7] = 14$	$\pi[7] = 6$
$d[8] = 4$	$f[8] = 17$	$\pi[8] = 4$
$d[9] = 8$	$f[9] = 9$	$\pi[9] = 10$
$d[10] = 7$	$f[10] = 10$	$\pi[10] = 2$

Wald:

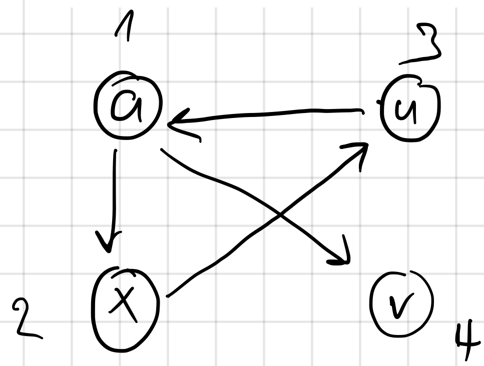


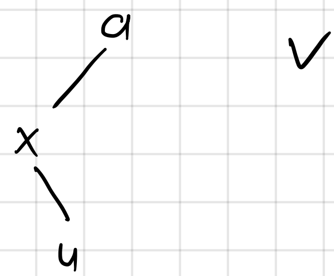
6. Geben Sie Gegenbeispiele zu folgenden Vermutungen an:

   (a) Falls ein gerichteter Graph G einen Pfad von u nach v enthält und bei der Tiefensuche auf G $u.d < v.d$ gilt, dann ist v im erzeugten Tiefensuchwald ein Nachfahre von u.

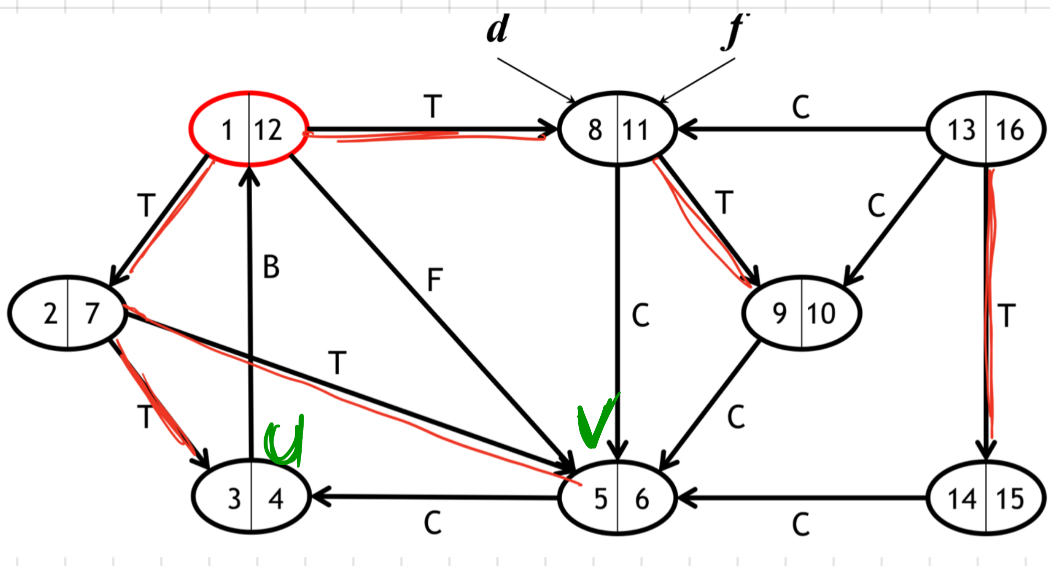
   (b) Falls ein gerichteter Graph G einen Pfad von u nach v enthält, dann muss jede Tiefensuche zu $v.d \leq u.f$ führen.

a)

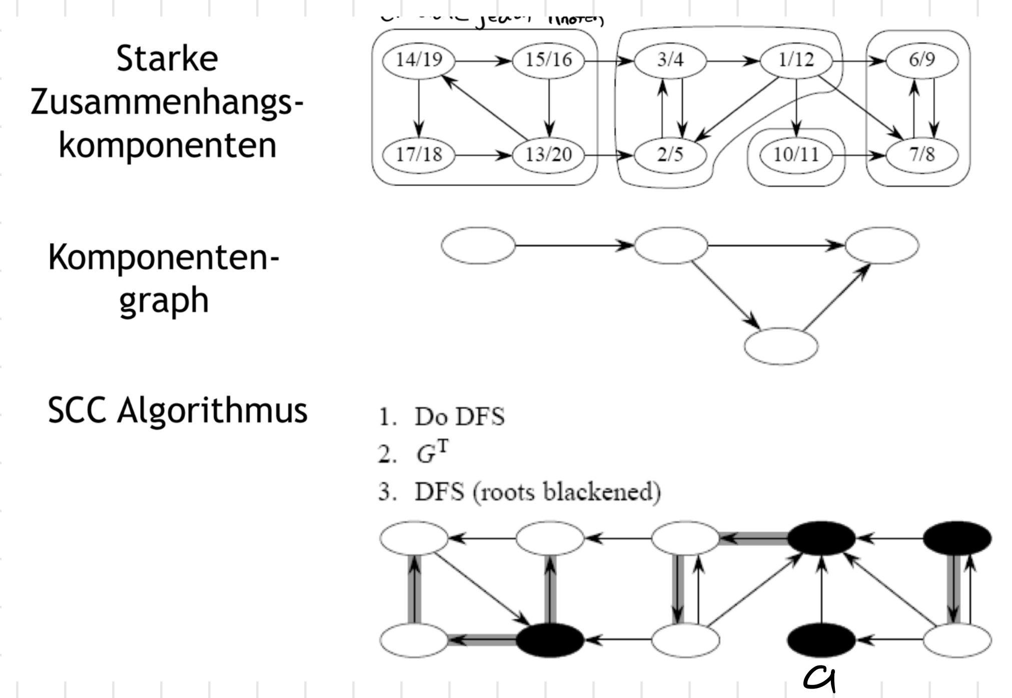




b)



7. Erklären Sie, wie es passieren kann, dass ein Knoten u eines gerichteten Graphen in einem Tiefensuchbaum landet, der allein u enthält, obwohl u sowohl eintretende als auch austretende Kanten hat. **1 Punkt**



Beispiel für SCC Algorithmus. a steht alleine in einem Baum.