Technische Universität Darmstadt
Fachgebiet Theoretische Informatik

Prof. Johannes Buchmann
Nabil Alkeilani Alkadri
Nina Bindel
Patrick Struck

Algorithmen und Datenstrukturen

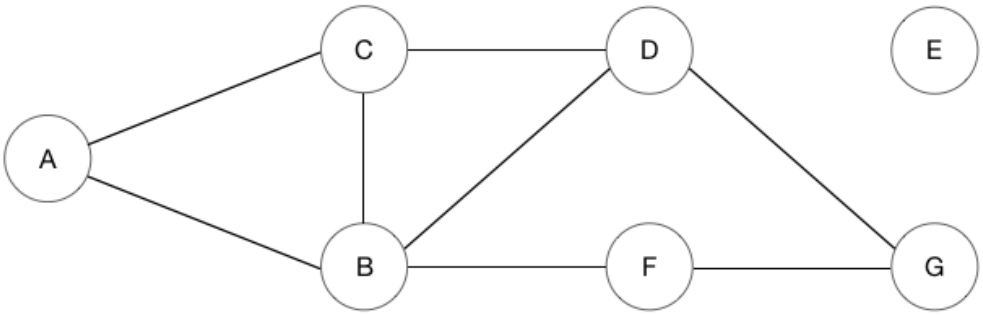
SoSe 2018

TECHNISCHE UNIVERSITÄT DARMSTADT

8. Lösungsblatt — 04.06.2018
v1.0

P1 Breitensuche

Führen Sie auf dem folgenden Graphen mit ungewichteten Kanten eine Breitensuche aus. Tragen Sie hierfür die Werte von d[v], π[v] (für alle Knoten v), Q und u für jeden Iterationsschritt in die folgende Tabelle ein. Benutzen Sie A als Startknoten und gehen Sie davon aus, dass die Adjazenzlisten alphabetisch sortiert sind.



d[A]	d[B]	d[C]	d[D]	d[E]	d[F]	d[G]	π[A]	π[B]	π[C]	π[D]	π[E]	π[F]	π[G]	Q	u
0	∞	∞	∞	∞	∞	∞	nil	nil	nil	nil	nil	nil	nil	{A}	-

Lösung.

d[A]	d[B]	d[C]	d[D]	d[E]	d[F]	d[G]	π[A]	π[B]	π[C]	π[D]	π[E]	π[F]	π[G]	Q	u
0	∞	∞	∞	∞	∞	∞	nil	nil	nil	nil	nil	nil	nil	{A}	-
0	1	1	∞	∞	∞	∞	nil	A	A	nil	nil	nil	nil	{B, C}	A
0	1	1	2	∞	2	∞	nil	A	A	B	nil	B	nil	{C, D, F}	B
0	1	1	2	∞	2	∞	nil	A	A	B	nil	B	nil	{D, F}	C
0	1	1	2	∞	2	3	nil	A	A	B	nil	B	D	{F, G}	D
0	1	1	2	∞	2	3	nil	A	A	B	nil	B	D	{G}	F
0	1	1	2	∞	2	3	nil	A	A	B	nil	B	D	$\emptyset$	G

P2 Breitensuchbaum

(a) Erstellen Sie von dem Graphen $G = (V, E)$ aus P1 den Vorgängerbaum, der aus der Breitensuche resultiert.

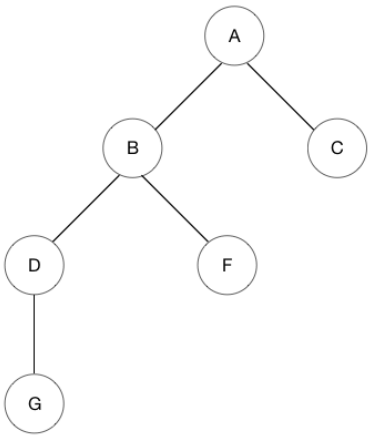
Hinweis: Es gilt $G_\pi = (V_\pi, E_\pi)$ mit $V_\pi = \{v \in V : v.\pi \neq nil\} \cup \{s\}$ und $E_\pi = \{(v.\pi, v) : v \in V_\pi - \{s\}\}$.

(b) Der Vorgängerbaum entspricht einem Breitensuchbaum, da V_π alle erreichbaren Knoten von s enthält und für jeden Knoten $v \in V_\pi$ der eindeutige Pfad von s nach v in G_π dem kürzesten Pfad von s nach v in G entspricht.

Gilt dies auch für Vorgängerbäume in gewichteten Graphen, falls die Adjazenzlisten nach Kantenlänge sortiert sind? Beweisen oder widerlegen Sie.

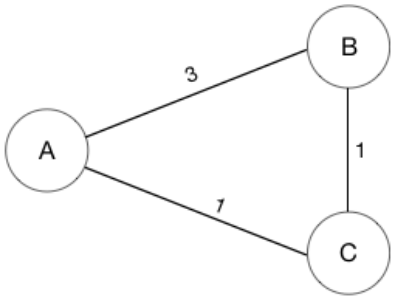
Lösung.

(a)

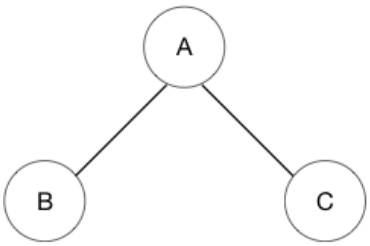


(b) Bei Graphen mit gewichteten Kanten gilt dies nicht, da bei der Breitensuche zunächst alle Nachbarn eines Knoten besucht werden, auch wenn ein Pfad über mehrere Knoten zu einem direkten Nachbarn evtl. kürzer ist.

Gegenbeispiel: Wenn man von folgendem Graphen eine Breitensuche von A aus durchführt, ist der Vorgänger von B im Baum A, obwohl der Pfad von A nach B über C im Graphen kürzer wäre.



(a) Graph



(b) Vorgängerbaum

P3 Breitensuche - Zyklen

Schreiben Sie einen Algorithmus in Pseudocode, der für einen **ungerichteten** Graphen ausgibt, ob dieser zyklisch (true) oder azyklisch (false) ist. Orientieren Sie sich an dem BFS Algorithmus aus der Vorlesung. Sie dürfen davon ausgehen, dass von jedem Knoten alle Knoten über einen Pfad erreichbar sind.

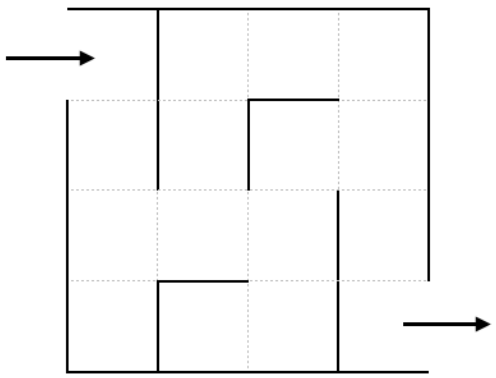
Lösung.

Algorithm 1 Pseudocode zur Erkennung von Zyklen in ungerichteten Graphen

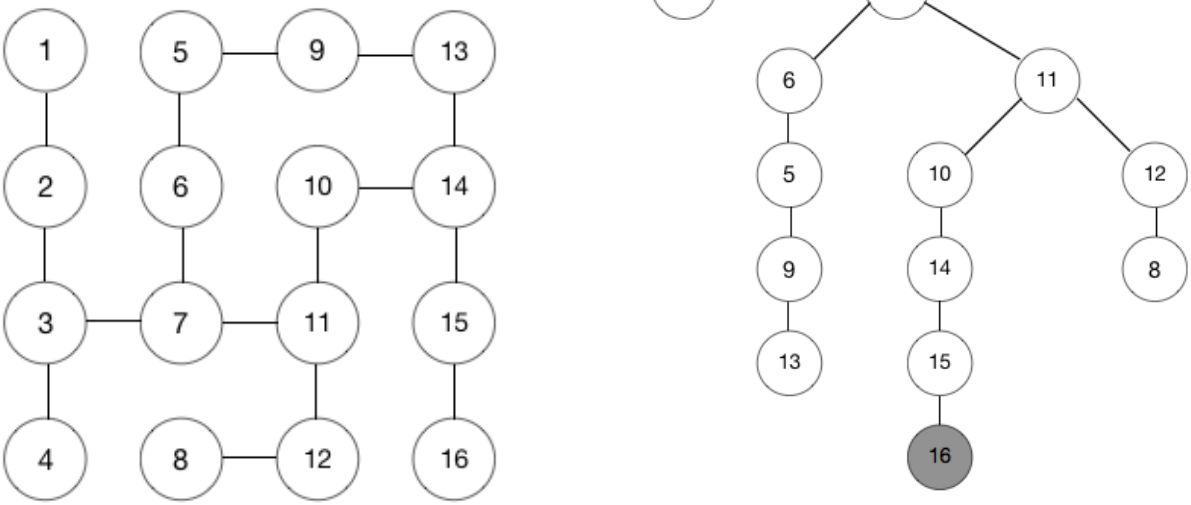
```
1:  procedure BFS-CYCLE(G, s)                                   ▷ s ist beliebig
2:      for each vertex u ∈ V[G] − {s} do
3:          color[u] ← WHITE
4:          π[u] ← NIL
5:      end for
6:      color[s] ← GRAY
7:      π[s] ← NIL
8:      Q ← ∅
9:      ENQUEUE(Q, s)
10:     while Q ≠ ∅ do
11:         u ← DEQUEUE(Q)
12:         for each v ∈ Adj[u] do
13:             if color[v] ≠ WHITE and π[u] ≠ v then       ▷ zweite Bed. notwendig, da Graph ungerichtet
14:                 return true
15:             end if
16:             color[v] ← GRAY
17:             π[v] ← u
18:             ENQUEUE(Q, v)
19:         end for
20:         color[u] ← BLACK
21:     end while
22:     return false
23: end procedure
```

H1 Labyrinth

Gegeben sei folgendes Labyrinth. Finden Sie mithilfe von Breitensuche den kürzesten Weg vom Start zum Ziel. Es genügt, wenn Sie als Lösung den Graphen und den Breitensuchbaum angeben.



Lösung. Um den kürzesten Weg im Labyrinth zu finden, wird zunächst ein Graph erstellt, wobei jedes Feld einem Knoten und jeder Durchgang einer Kante entspricht (Abbildung 1(a)). Dann wird Breitensuche mit dem Startknoten 1 ausgeführt bis der Knoten 16 gefunden wurde (Abbildung 1(b)).



(a) Graph

(b) Breitensuchbaum

Abbildung 1: Lösung H1

H2 Durchmesser

Wir definieren den Durchmesser eines Baumes als den maximalen Abstand zwischen zwei Knoten, wobei der Abstand als kürzester Pfad definiert ist. Beschreiben Sie einen Algorithmus, der den Durchmesser eines Baumes zurückgibt und geben Sie die Komplexität des Algorithmus an.

Hinweis: Es gibt einen Algorithmus, der die selbe Komplexität wie Breitensuche hat.

Lösung. Sei der gegebene Baum $T = (V, E)$ mit Wurzel r. Zunächst wird $BFS(T, r)$ aufgerufen (es kann hier auch jeder andere Knoten als Startknoten verwendet werden). Sei v der letzte Knoten, der bei diesem Aufruf besucht wurde, dann wird anschließend $BFS(T, v)$ durchgeführt. Der Durchmesser ist nun $d[w]$, wobei w der zuletzt besuchte Knoten von der zweiten Breitensuche ist. Da in dem Algorithmus (neben Anweisungen mit konstanter Laufzeit) lediglich zweimal Breitensuche aufgerufen wird, hat dieser die selbe Komplexität wie BFS.