Exercise class 3

XYZ

Groups 2, 4

Presenter:

Jürgen Mattheis
(juergmatth@gmail.com)

Lecture by:
Prof. Dr. Schindelhauer

Supervisor for the exercises:
Sneha Mohanty

June 2, 2023

University of Freiburg, Chair of Computer Networks and Telematics

Table of Contents

Exercise 1

Aufgabe 1 I

Reachability

Solution 1.1

Algorithm 1 BREADTH-FIRST SEARCH

Eingabe: (Di-) Graph $G = (V, E)$, start node $s \in V$ to which applies: $\forall v \in V \setminus s : \exists\ s, v$-path

```
Q ← new.Queue()
counter ← 1
Q.enqueue(s); mark node s
s.count ← counter; counter ← counter +1
As long as not Q.empty()
    u ← Q.dequeue()
    BFS-Visit(u)
    for all adjavent nodes v of u
        if v not marked then
            Q.enqueue(v); mark node v
            v.count ← counter; counter ← counter +1
```

Aufgabe 1 II

Reachability

Solution 1.1

Algorithm 2 BREADTH-FIRST SEARCH

Eingabe: (Di-) Graph $G = (V, E)$, start node $s \in V$ to which applies: $\forall v \in V \setminus s : \exists\ s, v$-path, target node t

```
Q ← new.Queue()
counter ← 1
Q.enqueue(s); mark node s
s.count ← counter; counter ← counter +1
As long as not Q.empty()
    u ← Q.dequeue()
    if u is t then
        return TRUE
    for all adjavent nodes v of u
        if v not marked then
            Q.enqueue(v); mark node v
            v.count ← counter; counter ← counter +1
return FALSE
```

Aufgabe 1 III

Reachability

Solution 1.2

Algorithm 3 DEPTH-FIRST SEARCH

Eingabe: (Di-) Graph $G = (V, E)$ with nodes V unmarked, start node $s \in V$ to which applies: $\forall v \in V \setminus s : \exists\ s, v$-path

```
𝒮 ← new.Stack()
counter ← 1
𝒮.push(s); mark s
s.start ← counter; counter ← counter +1
DFS-Visit(s)
As long as not 𝒮.empty()
    u ← 𝒮.pop(); 𝒮.push(u)
    if a not marked adjacent node v of u exists then
        𝒮.push(v); mark v
        DFS-Visit(s)
    v.start ← counter; counter ← counter +1
else
    u ← 𝒮.pop()
    u.end ← counter; counter ← counter +1
return FALSE
```

Aufgabe 1 IV

Solution 1.2

Algorithm 4 Depth-First Search

Eingabe: (Di-) Graph $G = (V, E)$ with nodes V unmarked, start node $s \in V$ to which applies: $\forall v \in V \setminus s : \exists\ s, v$-path, target node t

```
𝒮 ← new.Stack()
counter ← 1
𝒮.push(s); mark s
s.start ← counter; counter ← counter +1
if u is t then
    return TRUE
As long as not 𝒮.empty()
    u ← 𝒮.pop(); 𝒮.push(u)
    if a not marked adjacent node v of u exists then
        𝒮.push(v); mark v
        if u is t then
            return TRUE
        v.start ← counter; counter ← counter +1
    else
        u ← 𝒮.pop()
        u.end ← counter; counter ← counter +1
return FALSE
```

Exercise 2

Aufgabe 2 I

Reachability in infinite graphs

Exercise 2.1

$$V(G) := \{v_i | i \in \mathbb{N}_0\}$$
$$E(G) := \{(v_2, v_0)\} \cup \{(v_1, v_i) | i \in \mathbb{N}_0 \setminus \{1, 0\}\}$$

Aufgabe 2 II

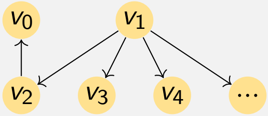
Reachability in infinite graphs

Solution 2.1

Algorithm 5 BREADTH-FIRST SEARCH

Eingabe: (Di-) Graph $G = (V, E)$, start node $s \in V$ to which applies: $\forall v \in V \setminus s : \exists\ s, v$-path, target node t

```
Q ← new.Queue()
counter ← 1
Q.enqueue(s); mark node s
s.count ← counter; counter ← counter +1
As long as not Q.empty()
    u ← Q.dequeue()
    if u is t then
        return TRUE
    for all adjavent nodes v of u
        if v not marked then
            Q.enqueue(v); mark node v
            v.count ← counter; counter ← counter +1
return FALSE
```



- *If the following rule gets applied: If there's a choice between several adjacent nodes the node with the lowest index gets taken.*

- *$BFS(G, v_1, v_0)$*
  - *starte bei v1*
  - *visit v_2*
  - *visit v_3*
  - *visit ...*
  - *doesn't terminate*
- *$DFS(G, v_1, v_0)$*
  - *visit v_2*
  - *visit v_0*
  - *terminates*

Aufgabe 2 III

Reachability in infinite graphs

Exercise 2.2

$$V(F) := \{v_i | i \in \mathbb{N}_0 \backslash \{0, 1\}\}$$
$$E(F) := \{(v_i, v_{i \cdot 2}) | i \in \mathbb{N}_0 \backslash \{0, 1\}\} \cup \{(v_i, v_{i+1}) | i \in \mathbb{N}_0 \backslash \{0, 1\}\}$$

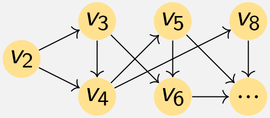
Aufgabe 2 IV

Solution 2.2

Algorithm 6 DEPTH-FIRST SEARCH

Eingabe: (Di-) Graph $G = (V, E)$ with nodes V unmarked, start node $s \in V$ to which applies: $\forall v \in V \setminus s : \exists\ s, v$-path, target node t

```
𝒮 ← new.Stack()
counter ← 1
𝒮.push(s); mark s
s.start ← counter; counter ← counter +1
if u is t then
    return TRUE
As long as not 𝒮.empty()
    u ← 𝒮.pop(); 𝒮.push(u)
    if a not marked adjacent node v of u exists then
        𝒮.push(v); mark v
        if u is t then
            return TRUE
        v.start ← counter; counter ← counter +1
    else
        u ← 𝒮.pop()
        u.end ← counter; counter ← counter +1
return FALSE
```



- *If the following rule gets applied: If there's a choice between several adjacent nodes the node with the highest index gets taken.*

- $BFS(G, v_2, v_3)$
  - *start at v_2*
  - *visit v_4*
  - *visit v_3*
  - *terminates*
- $DFS(G, v_2, v_3)$
  - *start at v_2*
  - *visit v_4*
  - *visit v_8*
  - *visit ...*
  - *doesn't terminate*

Exercise 3

Aufgabe 3 I

Control Flow Graphs

Solution 3.2

```
void example() {
  int var;
  if (0) {
    var = 42;
  }
  var = 24;
}
```

Figure 1: *Control flow graph*

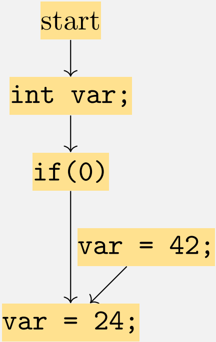


Figure 2: *Assignment of code lines to the CFG*

Aufgabe 3 II

Control Flow Graphs

Solution 3.3

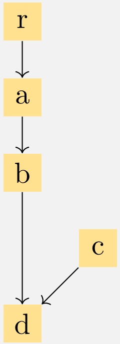


Figure 3: *Control flow graph*

- S_v: *Alle Knoten die in* $G - \{v\}$ *von* r *aus erreicht werden können:*
  - $S_r = \{\}$
  - $S_a = \{r\}$
  - $S_b = \{r, a\}$
  - $S_c = \{r, a, b, d\}$
  - $S_d = \{r, a, b\}$
- $D_v : V - \{v\} - S_v$:
  - $D_r = \{r, a, b, c, d\} - \{r\} - \{\} = \{a, b, c, d\}$
  - $D_a = \{r, a, b, c, d\} - \{a\} - \{r\} = \{b, c, d\}$
  - $D_b = \{r, a, b, c, d\} - \{b\} - \{r, a\} = \{c, d\}$
  - $D_c = \{r, a, b, c, d\} - \{c\} - \{r, a, b, d\} = \{\}$
  - $D_d = \{r, a, b, c, d\} - \{d\} - \{r, a, b\} = \{c\}$

Aufgabe 3 III

Control Flow Graphs

Solution 3.3

- *S_v: Alle Knoten die in $G - \{v\}$ von r aus erreicht werden können:*
  - $S_r = \{\}$
  - $S_a = \{r\}$
  - $S_b = \{r, a\}$
  - $S_c = \{r, a, b, d\}$
  - $S_d = \{r, a, b\}$
- *$D_v : V - \{v\} - S_v$:*
  - $D_r = \{r, a, b, c, d\} - \{r\} - \{\} = \{a, b, c, d\}$
  - $D_a = \{r, a, b, c, d\} - \{a\} - \{r\} = \{b, c, d\}$
  - $D_b = \{r, a, b, c, d\} - \{b\} - \{r, a\} = \{c, d\}$
  - $D_c = \{r, a, b, c, d\} - \{c\} - \{r, a, b, d\} = \{\}$
  - $D_d = \{r, a, b, c, d\} - \{d\} - \{r, a, b\} = \{c\}$

	r	a	b	c	d
r		1	1	1	1
a			1	1	1
b				1	1
c					
d				1	

Aufgabe 3 IV

Control Flow Graphs

Solution 3.3

- *S_v: Alle Knoten die in $G - \{v\}$ von r aus erreicht werden können:*
  - $S_r = \{\}$
  - $S_a = \{r\}$
  - $S_b = \{r, a\}$
  - $S_c = \{r, a, b, d\}$
  - $S_d = \{r, a, b\}$
- *$D_v : V - \{v\} - S_v$:*
  - $D_r = \{r, a, b, c, d\} - \{r\} - \{\} = \{a, b, c, d\}$
  - $D_a = \{r, a, b, c, d\} - \{a\} - \{r\} = \{b, c, d\}$
  - $D_b = \{r, a, b, c, d\} - \{b\} - \{r, a\} = \{c, d\}$
  - $D_c = \{r, a, b, c, d\} - \{c\} - \{r, a, b, d\} = \{\}$
  - $D_d = \{r, a, b, c, d\} - \{d\} - \{r, a, b\} = \{c\}$

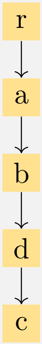


Figure 4: *Dominance tree*

Aufgabe 3 V

Control Flow Graphs

Solution 3.1

- $v \operatorname{dom} w$ *iff*
  - *there exists a path from r to w in G* and
  - *every path in G from r to w contains the node v*
- $v \operatorname{idom} w$ *iff*
  - $(v \operatorname{dom} w)$ *and*
  - *there's no node u to which applies:* $(v \operatorname{dom} u)$ *and* $(u \operatorname{dom} w)$

Appendix

Literature

Online

Download White background with orange geometric for free. Vecteezy. URL: https://www.vecteezy.com/vector-art/11171111-white-background-with-orange-geometric (visited on 04/25/2023).