Algorithmen und Datenstrukturen

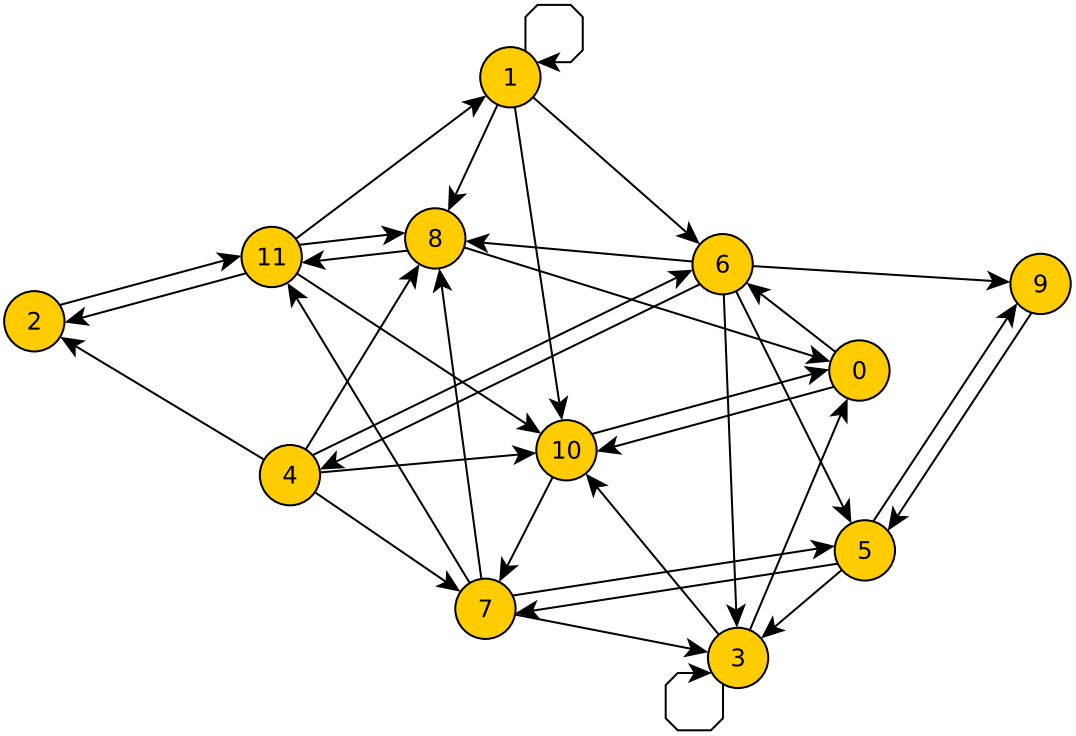
Dr. M. Lüthi, Dr. G. Röger
Frühjahrssemester 2018

Universität Basel
Fachbereich Informatik

Übungsblatt 8

Abgabe: 4. Mai 2018

Aufgabe 8.1 (Tiefen- und Breitensuche, 1.5+0.5+1.5+0.5 Punkte)
Betrachten Sie folgenden gerichteten Graphen:



Wenn bei den folgenden Aufgaben ein Knoten mehrere Nachfolger hat, die Sie besuchen können, dann wählen Sie jeweils den Knoten mit dem kleinsten Schlüssel.

(a) Welche Knoten besucht die *Tiefensuche* von Knoten 0 aus und in welcher Reihenfolge? Geben Sie auch den induzierten Suchbaum an (Liste der Kanten oder gezeichnet).

(b) Ist der Graph azyklisch? Falls nein, welchen Zykel findet der auf dieser Suche basierende Algorithmus zur *Zykelerkennung*?

(c) Welche Knoten besucht die *Breitensuche* von Knoten 0 aus und in welcher Reihenfolge? Geben Sie auch den induzierten Suchbaum an (Liste der Kanten oder gezeichnet).

(d) Welchen kürzesten Pfad von Knoten 0 zu Knoten 1 findet der auf dieser Breitensuche basierende Algorithmus?

a) 0-6-3-10-7-5-9-8-11-1-2-4
b) 0-6-5-9
c) 0-6-10-3-4-5-8-9-7-2-11-1
d) 0-6-8-11-1

Aufgabe 8.2 (Union-Find, 1.5 + 1.5 + 2 + 1 Punkte)

Sie möchten mit Hilfe einer Union-Find-Datenstruktur die Äquivalenzklassen der feinsten Äquivalenzrelation über den Objekten 0, ..., 9 bestimmen, die folgende Äquivalenzen enthält:

$$8 \sim 7, 0 \sim 9, 7 \sim 5, 7 \sim 1, 3 \sim 7, 0 \sim 6$$

Hierzu rufen Sie für $v \sim w$ jeweils `union(v, w)` Ihrer Union-Find-Datenstruktur auf.

(a) Geben Sie für `QuickUnion` jeweils den Wald von Bäumen an, den `parent` nach jedem Aufruf repräsentiert. Welche Äquivalenzklassen haben Sie am Ende berechnet?

(b) Geben Sie für `RankedQuickUnion` jeweils den Wald von Bäumen an, den `parent` nach jedem Aufruf repräsentiert.

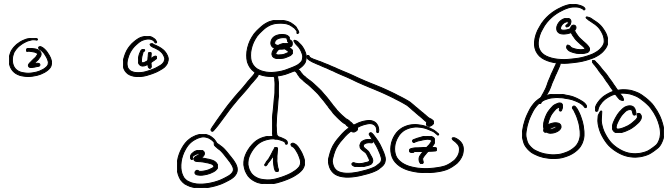
(c) Vervollständigen Sie die Implementierung der Klassen `QuickFind.java`, `QuickUnion.java` und `RankedQuickUnionWithPathCompression.java`. Stellen Sie sicher, dass Ihr Code die Testfälle und den Stylecheck besteht.

(d) In Datei `input.txt` finden Sie in der ersten Zeile die Anzahl der Objekte und danach eine Reihe von union- und find-Aufrufen. Die `main`-Methode parst eine solche Eingabe von der Standardeingabe (mit `gradlew run < input.txt`). Updaten Sie in Ihrer Implementierung der Union-Find-Datenstrukturen jeweils den Zähler `noAccess`, so dass er die Anzahl *aller* Arrayzugriffe erfasst. Wie viele Zugriffe benötigen die Implementierungen jeweils für die Eingabedatei `input`?

Die Übungsblätter dürfen in Gruppen von zwei Studierenden bearbeitet werden. Bitte schreiben Sie beide Namen auf Ihre Lösung.

a)

i	0	1	2	3	4	5	6	7	8	9
Node	9	8	2	8	4	8	9	8	8	9



Äquivalenzklasse: {2} {4} {8,3,1,5,7} {9,6,0}

b) Rang: 0 0 1 1

Wald ist gleich wie bei A

d) QuickFind: 55822
QuickUnion: 45519
RankedQuick...: 26877