Algorithmische Mathematik I

Wintersemester 19/20
Prof. Dr. J. Gedicke
Johannes Rentrop und Jannik Schürg

UNIVERSITÄT BONN

Programmieraufgabenblatt 7. Abgabedatum: **02.12.2019–06.12.2019**

Programmieraufgabe 1. (Graphen-Klasse)

Machen Sie sich mit der in der Vorlesung gezeigten Graphen-Klasse vertraut. Auf der Vorlesungs-Website finden Sie Links zum Herunterladen der benötigten Dateien. Ihr Tutor oder Ihre Tutorin kann Ihnen insbesondere zeigen, wie sie Objekte einer Klasse instanzieren und deren zugehörige Variablen und Funktionen aufrufen. Bei Programmieraufgaben, die die Graphen-Klasse verwenden, muss in C++ programmiert werden.

a) Auf der Vorlesungs-Website finden Sie eine Datei, die einen Graphen in einem geeigneten Format enthält. Schreiben Sie ein Programm, das den Graphen aus der Datei einliest und auf dem Bildschirm ausgibt. Nutzen Sie dafür die eingebauten Funktionen der Graphen-Klasse.

b) Schreiben Sie eine Funktion, die alle benachbarten Knoten eines Eingabeknotens auf dem Bildschirm ausgibt.

c) Schreiben Sie eine Funktion, die die Breitensuche (BFS) implementiert, um eine Liste aller von einem gegebenen Eingabeknoten aus erreichbaren Knoten zurückzugeben (inklusive des Eingabeknotens). Nutzen Sie dafür die Datenstrukturen `std::vector` und `std::queue` für die Liste an besuchten Knoten bzw. für die Warteschlange.

d) Nutzen Sie die Funktion aus Teil c), um alle Zusammenhangskomponenten des Graphen zu finden, und geben Sie die zugehörigen Knotenlisten aus.

Hinweise: Übergeben Sie den Graphen als Referenz an Ihre Funktionen, z.B.:

```
void eine_funktion(const Graph & graph) { }
```

Um über alle Nachbarn eines Knotens zu iterieren, können Sie das Konstrukt

```
for (auto neighbor: graph.get_node(some_node_id).adjacent_nodes()) { }
```

verwenden. Vergessen Sie nicht, den Header `"graph.h"` einzubinden, und beim Kompilieren die Datei `graph.cpp` mit zu übergeben, z.B:

```
g++ -o myprogram myprogramm.cpp graph.cpp
```

(10 Punkte)

Programmieraufgabe 2. (Pfade im Dreieck)

Abbildung 1 zeigt ein Zahlendreieck. Wir betrachten Pfade durch das Dreieck mit folgenden Eigenschaften:

i) Der Pfad beginnt in der ersten Reihe und endet in der letzten.

ii) Es darf in jedem Schritt nur die Zahl direkt links oder direkt rechts unter der vorigen Zahl verwendet werden.

Die *Länge* eines Pfades sei die Summe der besuchten Zahlen. Implementieren Sie einen Algorithmus mit folgenden Ein- und Ausgaben:

Eingabe: Ein Zahlendreieck dessen Zahlen aus $0, \ldots, 99$ sind. Eingabe-Dateien für Dreiecke mit 4, 15, und 10,000 Zeilen finden Sie auf der Vorlesungs-Webseite.

Ausgabe: Die Länge eines kürzesten Pfades.

Wie ist die Laufzeit ihres Algorithmus in $\mathcal{O}$-Notation, wenn n die Anzahl aller Zahlen ist? Sie können einen Algorithmus aus der Vorlesung verwenden, müssen dies aber nicht.

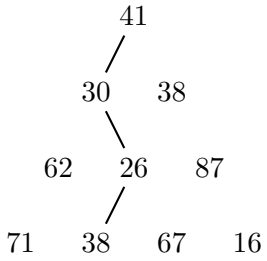

Abbildung 1: Pfad durch ein Dreieck. Die Länge beträgt 135.

(10 Punkte)