Philipps-Universität Marburg
Fachbereich Mathematik und Informatik
Arbeitsgruppe Datenbanksysteme

Sommersemester 2021
20. 06. 2021

Prof. Dr. Bernhard Seeger
Nikolaus Glombiewski, M.Sc.
Amir El-Shaikh, M.Sc.

Übungen zur Vorlesung
Algorithmen und Datenstrukturen

Abgabe: 27. 06. 2021,
bis **spätestens** 23:59 Uhr
über die ILIAS Plattform

Übung 10

Aufgabe 10.1: Anwendung von Graphen (3+2) (5 Punkte)

In dieser Aufgabe sollen Sie n Gefährten, die zu einer Reise aufbrechen in einer Reihe aufstellen. Sie bekommen dazu eine Liste von m Aussagen wie „i mag nicht j“. Ein Gefährte i der einen anderen Gefährten j nicht mag, soll nicht hinter diesem aufgestellt werden, da er sonst Dinge nach j werfen könnte.

a) Geben Sie einen Algorithmus an, der eine entsprechend geordnete Reihe von Gefährten mit einem erwarteten Aufwand von $\mathcal{O}(m + n)$ erzeugt. Der Input des Algorithmus ist eine Liste von n einzigartigen Gefährten und eine Liste von m einzigartigen Aussagen. Output ist eine Liste von Gefährten in der gesuchten Reihenfolge. Sie dürfen für Ihre Lösung davon ausgehen, dass eine solche Reihenfolge existiert. Argumentieren Sie was bei Ihrer Lösung passiert, wenn eine Reihenfolge nicht existiert.

b) Erzeugen Sie aus der folgenden Liste von Aussagen zunächst einen gerichteten Graphen und erzeugen Sie anschließend eine entsprechende Reihe von Gefährten:

- Frodo mag nicht Boromir
- Frodo mag nicht Legolas
- Boromir mag nicht Aragorn
- Boromir mag nicht Pippin
- Sam mag nicht Boromir
- Aragorn mag nicht Pippin
- Pippin mag nicht Legolas
- Pippin mag nicht Gandalf
- Gimli mag nicht Aragorn

Aufgabe 10.2: Implementierung von Graphen (2+4+2+2+5*) (15 Punkte)

Im Ilias finden Sie ein Interface `Graph<T>`, welches einen Graphen repräsentiert, und eine Implementierung dieses Interfaces `AdjacencyListGraph<T>`, welche einen Graphen durch Adjazenzlisten realisiert. Die Knoten und Adjazenzlisten dieser Implementierung werden durch die ebenfalls zur Verfügung gestellte Klasse `AdjacencyListNode<T>` umgesetzt.

a) *Nennen* Sie zwei Datenstrukturen, welche in der Klasse AdjacencyListNode<T> verwendet werden könnten, um ausgehend von einem Knoten v_1 folgendes Problem in sublineare

Laufzeit zu lösen: Ist ein gegebener Knoten v_2 ein direkter Nachfolger von v_1?

b) Implementieren Sie das Interface `Graph<T>` durch eine Klasse `AdjacencyMatrixGraph<T>`, welche einen Graphen mit Hilfe einer Adjazenzmatrix verwaltet. Die Matrix soll mit einer Größe von 10x10 Feldern initialisiert werden und wachsen, sobald diese Kapazität überschritten wird.

c) Erweitern Sie Ihre Klasse `AdjacencyMatrixGraph<T>` um eine Methode `convert()`, welche den Graphen in Adjazenzlistenrepräsentation überführt, d.h. ein Objekt der Klasse `AdjacencyListGraph<T>` zurückliefert. Geben Sie die Laufzeit Ihrer Methode an.

d) Implementieren Sie in der Klasse `AdjacencyListGraph<T>` eine Methode `undirected()`, welche aus dem Graphen einen neuen Graphen erstellt, dessen Kanten ungerichtet sind.

e) Ein Graph heißt *bipartit*, wenn die Knoten des Graphen mit genau 2 Farben so gefärbt werden können, dass Knoten, welche über eine Kante verbunden sind, unterschiedlich Farben besitzen. Entwickeln Sie einen Algorithmus, welcher auf einem ungerichteten Graphen überprüft, ob dieser bipartit ist. Beschreiben Sie Ihren Algorithmus zunächst textuell. Implementieren Sie Ihren Algorithmus anschließend in der Klasse `AdjacencyListGraph<T>` durch eine Methode `undirectedGraphIsBipartit()`. Hierbei soll der Graph zuerst in eine ungerichtete Repräsentation überführt werden, auf welcher danach die Überprüfung stattfindet.

*** Aufgabe 10.2 e) ist für Lehramtsstudierende optional.**