Karlsruher Institut für Technologie

Institut für Theoretische Informatik
T.T.-Prof. Dr. Thomas Bläsius
https://scale.iti.kit.edu

Übungsblatt 07

Algorithmen I — Sommersemester 2023

Abgabe im ILIAS bis 16.06.2023, 18:00 Uhr

Die Abgabe erfolgt als *eine* PDF-Datei über das Übungsmodul in der Gruppe deines Tutoriums im ILIAS.
Beachte bitte die Hinweise zum Bearbeiten auf der Webseite.

Aufgabe 1 - Erbsenzähler (5+3 Punkte)

Gegeben sei folgender Pseudocode.

```
1: EXPLORE(G = (V, E): Graph, v ∈ V, D: [ℕ]) : ℕ
2:     count := 0
3:     for u ∈ N(v) do
4:         if D[u] > D[v] then
5:             count += 1 + EXPLORE(G, u, D)
6:         end
7:     end
8:     return count
```

Sei G ein Graph, $s \in V(G)$ ein Knoten und D ein Array, das für jeden Knoten $v \in V(G)$ die Distanz von s nach v in $D[v]$ speichert.

a) Wir betrachten nun den Aufruf $\textsc{Explore}(G, s, D)$. Beschreibe, was die Ausgabe des Algorithmus ist. (2 Punkte)

b) Gib eine unendliche Menge von Graphen an, die für jedes $n \in \mathbb{N}$ einen Graphen mit n Knoten enthält, auf dem der Algorithmus $\Theta(n)$ Zeit benötigt. (1 Punkt)

c) Zeige, dass die Laufzeit des Algorithmus in $2^{\Omega(n)}$ ist.

Hinweis 1: Eine Erklärung für die Notation $2^{\Omega(n)}$ findest du in der ersten Vorlesung auf Folie 17.
Hinweis 2: Um eine asymptotische untere Schranke zu zeigen genügt es nicht einen einzelnen Graphen anzugeben. Versuche stattdessen eine Konstruktion zu finden, die für beliebig große n funktioniert. (2 Punkte)

*) Beschreibe einen Algorithmus, der dasselbe Ergebnis wie EXPLORE in Linearzeit (d.h. $O(n+m)$ für n Knoten und m Kanten) berechnet. (3 Punkte)

Aufgabe 2 - Zugverbiberung (7 Punkte)

Habt ihr wirklich geglaubt, dass die Welt mit nur einem einzigen Damm erobert werden kann?! Dr. Meta expandiert. Die Biber sollen im ganzen Land Dämme bauen. Doch Vorsicht ist geboten! Wenn schlagartig überall Dämme entstehen, könnte jemand Verdacht schöpfen. Um das Projekt also langsam und mit diversen Verzögerungen aufzuziehen, entscheidet sich Dr. Meta dafür, die Biber mit der Bahn zu den Baustellen zu schicken.

Im Gegensatz zu euch haben die Biber nicht viel aus der Vorlesung mitgenommen und brauchen auf ihren Reisen jetzt eure Hilfe, um ihre Routen aus einzelnen Zugverbindungen herzuleiten.

Eine Zugverbindung ist durch ein Tupel $C = (Z, B_{\text{ab}}, B_{\text{an}}, \tau_{\text{ab}}, \tau_{\text{an}})$ mit $\tau_{\text{ab}} < \tau_{\text{an}}$ repräsentiert, was bedeutet, dass der Zug Z zum Zeitpunkt τ_{ab} am Bahnhof B_{ab} losfährt und ohne Zwischenstopp zum Zeitpunkt τ_{an} am Bahnhof B_{an} ankommt. Sei $\mathcal{C}$ eine Menge solcher Verbindungen und sei $\mathcal{B}$ die Menge aller Bahnhöfe. Eine Routenanfrage besteht aus einem Startbahnhof $B_s \in \mathcal{B}$, einem Zielbahnhof $B_t \in \mathcal{B}$, sowie einem Startzeitpunkt τ_s. Ziel ist es, eine Route von B_s nach B_t zu finden, die nicht vor τ_s startet und möglichst früh endet. Die Biber wollen auf jeden Fall noch am *gleichen* Tag ankommen, d.h., wenn sie zum Beispiel um 16 Uhr an einem Bahnhof ankommen, können sie nicht eine Nacht warten, um in einen Zug umzusteigen, der um 8 Uhr losfährt.

a) Beschreibe, wie das Problem als kürzeste Wege Problem in einem Graphen modelliert werden kann. Wende dann deine Modellierung auf die folgende Probleminstanz an und zeichne den resultierenden Graphen: (4 Punkte)

$$\mathcal{B} = \{\text{Potsdamm},\ \text{Biberlin},\ \text{Rostock}\}$$

$$\begin{aligned}\mathcal{C} = \{&(Z_0, \text{Potsdamm}, \text{Biberlin}, &&(08\text{ Uhr}), (10\text{ Uhr}))\\ &(Z_1, \text{Potsdamm}, \text{Rostock}, &&(12\text{ Uhr}), (16\text{ Uhr}))\\ &(Z_2, \text{Potsdamm}, \text{Biberlin}, &&(17\text{ Uhr}), (19\text{ Uhr}))\\ &(Z_3, \text{Biberlin}, \text{Potsdamm}, &&(10\text{ Uhr}), (12\text{ Uhr}))\\ &(Z_4, \text{Biberlin}, \text{Rostock}, &&(10\text{ Uhr}), (13\text{ Uhr}))\\ &(Z_5, \text{Rostock}, \text{Potsdamm}, &&(13\text{ Uhr}), (17\text{ Uhr}))\}\end{aligned}$$

Hinweis: Du darfst annehmen, dass mit Uhrzeiten gerechnet werden kann wie mit ganzen Zahlen.

b) Sei nun für jeden Bahnhof noch eine Umstiegszeit gegeben und für je zwei Bahnhöfe B_1 und B_2 die Dauer, in der man von B_1 nach B_2 in einem Fluss schwimmen kann. Beschreibe, wie deine Modellierung erweitert werden kann, sodass die Umstiegszeiten eingehalten werden und die Flusswege zwischen Bahnhöfen beachtet werden.

(3 Punkte)

Aufgabe 3 - Dijkstra und fiese Kanten (2 Punkte)

In dieser Aufgabe wollen wir untersuchen, was passiert, wenn Dijkstras Algorithmus auf einem gerichteten, gewichteten Graphen mit (unter anderem) negativen Kantengewichten aufgerufen wird. Wir beschränken uns hierbei auf Instanzen mit ganzzahligen Kantengewichten aus $[-1, \infty)$ und ohne negative Kreise (d.h. die Summe der Kantengewichte in jedem Kreis ist nicht negativ). Gib einen Graphen G und zwei Knoten s und t an, sodass Dijkstras Algorithmus aus der Vorlesung auf (G, s) nicht den kürzesten s-t-Weg findet und begründe kurz weshalb. (2 Punkte)

Aufgabe 4 - Das Orakel von Dijkstra (6 Punkte)

Zu einem Graphen $G = (V, E)$ und einem Knoten $s \in V$ bezeichnen wir mit dem *Kürzeste-Wege-Baum* von s einen Baum $T = (V, E')$ mit $E' \subseteq E$ bei dem jeder Pfad von s nach $t \in V$ in T auch einem kürzesten Pfad von s nach t in G entspricht. Entscheide für jede der folgenden Aussagen, ob sie wahr oder falsch ist und begründe deine Antwort. *Hinweis:* Für Graphattribute gelten, wenn nicht anders spezifiziert, diese Konventionen:

- ein "Graph" ist ungewichtet und ungerichtet
- ein "gerichteter Graph" ist ungewichtet
- ein "gewichteter Graph" ist ungerichtet und hat keine negativen Gewichte.

a) Wenn in einem gewichteten Graphen alle Kanten paarweise verschiedene Gewichte haben, dann ist der Kürzeste-Wege Baum von jedem Startknoten s aus eindeutig. (1 Punkt)

b) Wenn in einem positiv-gewichteten, gerichteten Graphen die Kantenrichtungen entfernt werden (indem man jede Kante als ungerichtet auffasst), dann ändern sich die Distanzen (bezüglich kürzester Wege) nicht. (1 Punkt)

c) Wenn in einem gewichteten Graphen alle Gewichte um $k > 0$ vergrößert werden, dann vergrößern sich die Distanzen zwischen Knotenpaaren um Vielfache von k. (1 Punkt)

d) Wenn in einem gewichteten Graphen alle Gewichte mit $k > 0$ multipliziert werden, ändern sich die Distanzen zwischen Knotenpaaren um einen Faktor k. (1 Punkt)

e) In einem gewichteten Graphen findet Dijkstras Algorithmus unter allen kürzesten Wegen zwischen zwei Knoten immer denjenigen mit den wenigsten Kanten. (1 Punkt)

f) Enthält ein zusammenhängender, (potenziell negativ-) gewichteter Graph keine negativen Kreise, dann gibt es zwischen jedem Knotenpaar einen *einfachen* kürzesten Pfad. Zur Erinnerung: ein Pfad ist einfach, wenn er keine Kreise enthält. (1 Punkt)