# Algorithmen und Datenstrukturen 2

ALGO2 · WiSe-2021/22 · tcs.uni-frankfurt.de/algo2 · 2023-10-12 · *a752888*

## Übungen zu Woche 1: Einführung und All Pairs Shortest Paths

Das Übungsblatt enthält alle empfohlenen Lernaktivitäten für die aktuelle Woche.

- **Dienstag 8:00–8:45. (pünktlich!)** Nimm an der Einführungsveranstaltung teil.
- **Dienstag 9:00–9:45.** Bearbeite jetzt die Übungen im Abschnitt **Dienstag**. Sprich mit anderen Studis! Frag das Vorlesungsteam um Hilfe!
- **Heimarbeit bis Mittwoch 17:00.**
  - Schau die Videos an und lies [Erickson, Kapitel 9.1, 9.2, 9.5, 9.6, 9.8]. Mach dir Notizen, wenn du etwas nicht verstanden hast!
  - Bearbeite die 🌱-Aufgabe in Moodle. (Feste Abgabefrist!)
  - Lies den Aufgabentext aller Übungsaufgaben.
  - Bearbeite die Übungsaufgaben im Abschnitt **Donnerstag**. Wenn du für mehr als 30 Minuten stecken bleibst, geh zur nächsten Übungsaufgabe.
- **Donnerstag 8:00–8:15.** Nimm an der Besprechung teil: Fragestunde, erstes Feedback zu 🌱-Aufgabe und ⭐-Aufgabe.
- **Donnerstag 8:15–9:15.** Bearbeite jetzt die Übungen, die du noch nicht lösen konntest. Sprich mit anderen Studis! Frag das Vorlesungsteam um Hilfe!
- **Donnerstag 9:15–9:45.** Nimm am Lösungsspaziergang teil.
- **Heimarbeit bis Freitag 17:00.** Gib deine Lösung zur aktuellen ⭐-Aufgabe in Moodle ab. (Feste Abgabefrist!)

## Dienstag

**Aufgabe 1.1 (Orga).**

a) Lies alle Informationen auf der Webseite des Kurses, auf der Moodle-Seite des Kurses, und auf den Organisationsfolien.

b) Melde dich jetzt mit einem Nickname deiner Wahl auf dem ALGO2 Discord-Server an. Den Link zum Discord-Server findest du in Moodle. (Es handelt sich hierbei **nicht** um den Discord-Server des Lernzentrums!)
Nutze den Discord-Server im ganzen Semester, wenn du andere Studis zum Zusammenarbeiten suchst oder inhaltliche Fragen an das Vorlesungsteam hast.

**Aufgabe 1.2 (Coderunner ausprobieren).** Löse jetzt die Aufgabe „Aufwärmübung: Coderunner ausprobieren“ auf Moodle.

**Aufgabe 1.3 (Telefonnetzwerk).** Gegeben sei ein Diagramm mit Schaltzentren und Verbindungen. Die Verbindungen können durch Kabel, Funk, oder Satellit realisiert werden. Wir modellieren dies, indem wir mit jeder Verbindung eine *Bandbreite* und *Kosten* assoziieren.

a) Gib einen Algorithmus an, der für zwei Schaltzentren $a$ und $b$ einen Verbindungsweg von $a$ nach $b$ ausrechnet, der die kleinsten Kosten verursacht.
b) Die *Bandbreite eines Verbindungsweg* ist die kleinste Bandbreite über alle einzelnen Verbindungen, die auf dem Weg auftauchen. Gib einen Algorithmus an, der für zwei Schaltzentren $a$ und $b$ einen Verbindungsweg von $a$ nach $b$ ausrechnet, der die größtmögliche Bandbreite hat.

**Aufgabe 1.4 (Puzzle der Woche: 99 Polizisten).** In einer Stadt arbeiten 99 Polizisten. Jeder Polizist ist entweder ehrlich oder korrupt, wobei die Mehrheit der Polizisten ehrlich ist. Finde mit weniger als 299 Fragen heraus, welches die korrupten Polizisten sind.

Dabei wissen alle Polizisten, welche der Polizisten ehrlich und welche korrupt sind, aber nur ehrliche Polizisten antworten immer wahrheitsgemäß. Korrupte Polizisten lügen möglicherweise. Aus sicherheitsgründen können nur Fragen folgenden Typs gestellt werden: Du kannst einen Polizisten X fragen, ob Polizist Y korrupt ist. Die Antwort darauf lautet entweder „Y ist korrupt“ oder „Y ist ehrlich“.

## Donnerstag

**Aufgabe 1.5 (APSP-Algorithmen anwenden).** Gegeben sei der gerichtete und gewichtete Graph $G$ aus Abbildung 1. Führe den Algorithmus FloydWarshall [Erickson, Abschnitt 9.8] von Hand Schritt für Schritt auf $G$ aus. Beginne, indem du dir eine $(5 \times 5)$-Matrix über $\mathbb{N} \cup \{\infty\}$ aufmalst, die den Zustand des `dist[.,.]`-Arrays für $r = 0$ enthält (also die Basisfälle). Zeichne danach jeweils den Zustand des Arrays nach den Iterationen $r = 1, 2, \ldots, 5$.

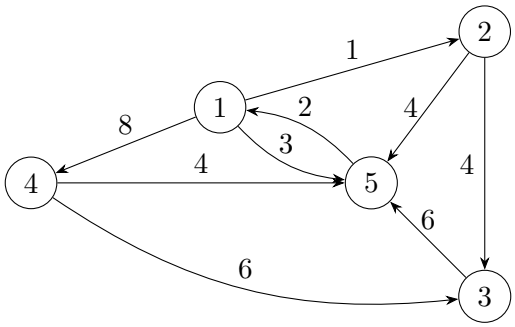


Abbildung 1: Der gerichtete, gewichtete Graph $G$. Die Zahlen an den Kanten definieren die Funktion $w\colon E(G) \to \mathbb{N}$ der Kantengewichte.

**Aufgabe 1.6 (APSP-Algorithmen anpassen).**

a) Beschreibe, wie der Algorithmus LeyzorekAPSP [Erickson, Abschnitt 9.6] modifiziert werden kann, um neben dem Array der Längen kürzester Wege außerdem ein Array von Vorgänger-Pointern zurückzugeben. Die Laufzeit soll weiterhin in $O(V^3 \log V)$ liegen.

b) Beschreibe, wie der Algorithmus FloydWarshall [Erickson, Abschnitt 9.8] modifiziert werden kann, um neben dem Array der Längen kürzester Wege außerdem ein Array von Vorgänger-Pointern zurückzugeben. Die Laufzeit soll weiterhin in $O(V^3)$ liegen.

*Hinweis*: Das gesuchte Array `pred` ist ein zweidimensionales Array, das jedem Knoten-Paar $(u, v)$ einen Knoten `pred[u,v]` zuordnet. Hierbei soll `pred[u,v]` der Vorgänger-Knoten von $v$ auf einem kürzesten Weg von $u$ nach $v$ sein.

**Aufgabe 1.7 (Mortys Labyrinth).** Morty muss einen stabilisierten Plumbus aus dem Klappenspitzen-Labyrinth beschaffen. Er muss dazu mit Ricks interdimensionaler *portal gun* das Labyrinth betreten, durch das Labyrinth zu einem Plumbus vordringen, anschließend den (instabilen) Plumbus durch das Labyrinth zu einem Fleeb bringen, um ihn zu stabilisieren, und schließlich wieder zurück zum ursprünglichen Portal, um nach Hause zurückzukehren. Plumbusse werden durch Fleeb-Saft stabilisiert, der von jedem Fleeb abgesondert wird, sobald er aus seinem Fleeb-Loch entfernt wird. Instabile Plumbusse explodieren, wenn sie von ihrer Aufbewahrungs-Einheit über 137 Flinks weit getragen werden. Zudem stinkt das Klappenspitzen-Labyrinth nach Furz, weshalb Morty so wenig Zeit wie möglich dort verbringen will.

Rick hat Morty für diese Aufgabe eine detaillierte Karte des Labyrinths gegeben, die aus einem gerichteten Graphen $G = (V, E)$ mit nicht-negativen Kantengewichten besteht (diese geben die Abstände in Flinks an). Zusätzlich hat er ihm die Menge $P \subseteq V$ der Plumbus-Aufbewahrungseinheiten und die Menge $F \subseteq V$ der Fleeb-Löcher mitgeteilt.

Morty hat nun die Aufgabe, einen Startknoten $s \in V$, eine Plumbus-Aufbewahrungseinheit $p \in P$ und ein Fleeb-Loch $f \in F$ zu finden, sodass die Länge eines kürzesten Wegs von $p$ nach $f$ höchstens 137 Flinks beträgt und die Länge eines kürzesten Wegs, der von $s$ zunächst über $p$ und dann über $f$ schließlich zurück zu $s$ führt, so kurz wie möglich ist.

Beschreibe einen Algorithmus, um Mortys Problem zu lösen, und analysiere seine Laufzeit. Du darfst annehmen, dass eine Lösung existiert.