Algorithmen und Datenstrukturen 2

ALGO2 · WiSe-2023/24 · tcs.uni-frankfurt.de/algo2/ · 2024-02-05 · 7ae310c

Übungen zu Woche 1: Einführung und All Pairs Shortest Paths

🎂 **1.1 Kennenlernen I.** Einführung.

🎂 **1.2 Kennenlernen II.** Wie heißt die Person neben dir? Was studiert sie? Was hat sie in den Ferien gemacht?

🎂 **1.3 Kennenlernen III.** Aufgaben sind meistens mit folgenden Emojis markiert:

- 🐣 Eigenständig **vor** dem nächsten Plenum bearbeiten (ca. 30 Minuten pro Aufgabe)
- 🎂 Wird vorraussichtlich im Plenum bearbeitet und besprochen
- 🔑 Eine mittelschwere Aufgabe (ca. 1 Stunde)
- 🔭 Eine besonders schwere Aufgabe (mehrere Stunden)
- 🌈 Eine optionale Spaßaufgabe zum Knobeln

🎂🔑 **1.4 Telefonnetzwerk.** Wir betrachten die Schaltzentren eines Telefonnetzwerks sowie die Verbindungen zwischen diesen Schaltzentren. Die Verbindungen können zum Beispiel durch Kabel, Funk, oder Satellit realisiert werden. Wir assoziieren daher mit jeder Verbindung eine *Bandbreite* und *Kosten*.

a) Welches Konzept ist geeignet, um die Eingabe abstrakt zu modellieren?

b) Gib einen Algorithmus an, der für zwei beliebige Schaltzentren a und b einen Verbindungsweg von a nach b ausrechnet, der die kleinsten Kosten verursacht.

c) 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.

🎂🐣 **1.5 Mortys Labyrinth I.** 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.

Zeichne Mortys Problem in geeigneter Weise auf und beschreibe die Anforderungen anhand der Zeichnung.

1.6 Organisiert euch.

a) Wo findest du die Videos, wo die Übungsblätter, wo die alten Klausuren? Was passiert eigentlich im Plenum und was im Tutorium? Gibt es Bonuspunkte und wann ist die Klausur? All diese Informationen findest du auf der Webseite des Kurses oder auf der Moodle-Seite des Kurses. Welche Fragen zum Kurs hast du jetzt noch?

b) Finde Mitstreiter:innen! Lernen macht mehr Spaß in der Gruppe. Wir empfehlen, dass ihr euch in Gruppen von 3-4 Personen organisiert. Trefft euch zum Anschauen der Videos und zum Lösen der Übungsaufgaben. Ihr könnt euch gegenseitig helfen, wenn ihr bei den Übungsaufgaben nicht weiterkommt. Und nach den Anstrengungen gemeinsam Tee trinken.

c) Das ist der vermutlich schwierigste Kurs im B.Sc. Informatik, rechne also damit, dass er viel Zeit in Anspruch nehmen wird. Erstelle für jede Woche im Semester einen detaillierten Wochenplan für dich und deine Lerngruppe. Entscheide, an welchen Angeboten von ALGO2 du wann teilnehmen möchtest. Zum Beispiel:

- **Inhaltlicher Input:** Falls ihr die Videos schauen wollt, plant ca. 2 Stunden mit Pausen, hitzigen Diskussionen, im Buch Nachlesen ein. Am Besten ein paar Tage vor dem Plenum. Nicht erst in der Nacht davor! Falls ihr ein Konzept nicht versteht, schaut mal in diesem Internet nach, oder ob es auf Youtube ein passendes Video dazu gibt.
- **Vorbereitende Übungsaufgaben:** Vor dem Plenum solltet ihr alle Übungsaufgaben und Teilaufgaben bearbeiten, die mit 🐣 markiert sind. Diese Aufgaben sind so gewählt, dass ihr sie auf Grundlage des inhaltlichen Inputs in eurer Lerngruppe eigenständig bearbeiten könnt. Wir erwarten nicht, dass ihr immer eine perfekte Antwort findet, aber falls ihr bei diesen Aufgaben regelmäßig gar keine Ahnung habt, müsst ihr unbedingt herausfinden, woran das liegt! Fehlen euch zum Beispiel Grundlagen aus einer vorherigen Vorlesung oder aus der Schule? Diese müsst ihr unbedingt nacharbeiten.
- **Plenum Dienstag und Donnerstag 8-10 Uhr:** Wir besprechen im Plenum kurz die Lösungen zu den 🐣-Aufgaben. Danach geht es im Plenum um die mittelschweren bis schweren Lernziele und Übungsaufgaben. Die meiste Zeit werdet ihr in Gruppen unter Anleitung eines Dozenten an diesen Aufgaben arbeiten.
- **Nachbereitung und Projekte:** Ihr solltet euch detaillierte Notizen machen und die Übungsaufgaben fertig bearbeiten, am Besten schriftlich. Ihr habt auch die Möglichkeit, Übungsaufgaben eurer Wahl zu einem Projekt zu machen—hierzu reicht ihr eure möglichst fleißig ausgearbeitete Lösung bei eurem Tutor oder eurer Tutorin ein—er oder sie gibt euch dann detailliertes Feedback. Reicht lieber eine Lösung ein, an der ihr lange gearbeitet habt, als viele Lösungen, die ihr nur halbherzig bearbeitet habt. Falls das Feedback nicht nützlich war, überlegt euch, woran das gelegen haben könnte und sprecht das unbedingt im Tutorium an.
- **Tutorium:** Das Ziel des Tutoriums ist es, dir und deiner Lerngruppe möglichst individuelles und nützliches **Feedback** zu geben. Die meiste Zeit arbeitet ihr im Tutorium in Kleingruppen von 3-4 Personen zusammen an denjenigen Übungs- oder Klausuraufgaben, die ihr noch lösen möchtet (ihr sucht euch aus welche!). Das kann eine Aufgabe sein, von der ihr die Lösung noch nicht kennt. Oder auch eine, von der ihr die Lösung zwar schon gesehen habt (z.B. im Plenum), aber noch nicht ganz sicher damit seid. Ihr könnt auch während des Tutoriums eure Lösung schriftlich ausarbeiten. Schriftliche Ausarbeitungen könnt ihr eurem Tutor zeigen und ihr erhaltet dann mündliches und schriftliches Feedback darauf.
- **Lösungsspaziergänge:** Die Lösungsspaziergänge enthalten hübsch aufgearbeitete Lösungsideen zum Wohlfühlen. Die Lösungsspaziergänge sind nicht Teil des Tutoriums, sondern werden in den zwei Recap-Wochen anstatt des Plenums von Tutor:innen präsentiert.

🐣 **1.7 Coderunner ausprobieren.** Löse „Aufwärmübung: Coderunner ausprobieren“ auf Moodle.

🐣 **1.8 APSP-Algorithmen Motivation.** Löst nicht schon Dijsktra's Algorithmus das APSP-Problem? Warum brauchen wir besondere APSP-Algorithmen, wenn wir doch kürzeste Wege schon ausrechnen können? In welchen Anwendungen will man überhaupt jemals APSP ausrechnen, anstatt nur normale kürzeste Wege zwischen zwei Knoten? Inwiefern wäre es sinnvoll, umgekehrt die neuen APSP-Algorithmen zu nutzen, um damit einen kürzesten Weg zwischen zwei Knoten auszurechnen?

🐣 **1.9 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 (5×5)-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 $r = 1, 2, \ldots, 5$.

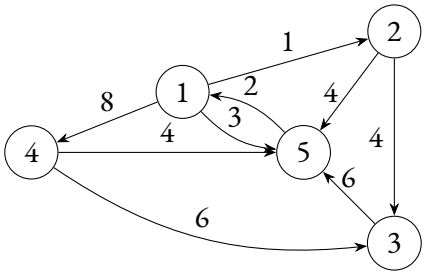

Abbildung 1: Der gerichtete, gewichtete Graph G. Die Zahlen an den Kanten definieren die Funktion $w\colon E(G) \to \mathbb{N}$ der Kantengewichte.

1.10 Backtracking in APSP-Algorithmen.

a) 🐣 Betrachte LeyzorekAPSP [Erickson, Abschnitt 9.6] und FloydWarshall [Erickson, Abschnitt 9.8]. Welche Daten werden von diesen Algorithmen aufrecht erhalten? Was ist das Ergebnis am Ende der Berechnung? Welche Gemeinsamkeiten haben die beiden Algorithmen? Welche Unterschiede? Warum benutzen wir die Einträge ∞?

b) 🎂 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.

c) 🎂 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 zu b) und c): 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.

🎂 **1.11 Mortys Labyrinth II.** Beschreibe einen Algorithmus, um das Problem aus Mortys Labyrinth I zu lösen, und analysiere seine Laufzeit. Du darfst annehmen, dass eine Lösung existiert.

🌈 **1.12 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“.

🖋 Projekte

Bearbeite mit deiner Lerngruppe jede Woche mindestens ein *Projekt*, stelle es im Tutorium vor und gib es bei deinem Tutor ab. Du erhälst dadurch mündliches und/oder schriftliches Feedback vom Tutor und von anderen Studis.

Wann und mit wem soll ich das Projekt bearbeiten? Du entscheidest dich jede Woche spätestens im Tutorium für dein Projekt der Woche. Such dir in deinem Tutorium Gleichgesinnte aus, die dasselbe oder ein ähnliches Projekt bearbeiten möchten. Bearbeitet euer Projekt in einer Gruppe von maximal 4 Personen. Die Zusammenstellung der Gruppen kann jede Woche neu sein. Ihr fangt direkt im Tutorium an, an dem Projekt zu arbeiten. Ihr schließt das Projekt im Laufe der folgenden Tage ab und gebt die schriftliche Ausarbeitung spätestens eine Woche später bei eurem Tutor ab.

Welches Projekt soll ich bearbeiten? Es ist völlig dir überlassen. *Echt jetzt?* Du bist Expertin bzw. Experte für dich und weißt am Besten, was du noch nicht kannst und was dir Spaß macht. Das Projekt sollte etwas mit ALGO2 zu tun haben und weder zu einfach noch zu schwierig für dich sein. Siehst du den genauen Schritt-für-Schritt Ablauf schon vor deinem inneren Auge und bist sicher, dass du das auch wirklich problemlos umsetzen könntest, dann ist das Projekt zu einfach. Wenn du auch nach 15 Minuten Überlegen gar keine Ahnung hast, wie du überhaupt anfängst und was du tun müsstest, dann ist das Projekt zu schwierig. Wir sind gespannt, welche Projekte ihr euch überlegt!

Konkrete Projektideen für die aktuelle Woche

- Wähle eine oder mehrere Aufgaben vom aktuellen Übungsblatt aus und bearbeite sie schriftlich.
- Implementiere einen oder mehrere APSP-Algorithmen deiner Wahl in einer Programmiersprache deiner Wahl. Um zu prüfen, ob deine Implementierung korrekt ist, steht dir in Moodle ein Coderunner-Modul zur Verfügung, welches Python, C++, C und Java unterstützt. Lass deine Implementierung auf Eingabegraphen aus der echten Welt laufen. Bei *Network Repository* (https://networkrepository.com/) gibt es tausende von Graphen, etwa Straßennetzwerke oder Gehirnbahnen. Wie groß darf der Graph sein, damit deine APSP-Berechnung noch in vernünftiger Zeit fertig wird?

Formale Anforderungen an die Projekte

- Nutzt dieses LaTeX-Template für eure schriftlichen Ausarbeitungen: https://github.com/goethe-tcs/note-template. Auch ein Link zu Overleaf findet sich dort, damit ihr direkt im Browser und ohne großes Vorwissen loslegen könnt. Wichtig ist, dass auf der Ausarbeitung eure Namen und eure `@stud.uni-frankfurt.de` E-Mail-Adresse stehen. Matrikelnummern sollen dort nicht stehen.
- Ganz wichtig ist, dass ihr am Anfang die Fragestellung wiederholt und erklärt. Was war die Übungsaufgabe oder das Problem, die oder das ihr gelöst habt? Reflektiert in eurer Ausarbeitung außerdem kurz, wie schwierig ihr die Aufgabe fandet, wie lange ihr gebraucht habt und was ihr gelernt habt.
- Schreibt eure Lösung mit so vielen zusätzlichen Erklärungen, dass ihr sie problemlos nachvollziehen könnt, wenn ihr in zehn Jahren wieder drauf schauen würdet. Aber haltet euch dennoch kurz und entfernt alle Textteile, die nicht zur Lösung oder zum Verständnis beitragen.
- Wechselt jedesmal die Rollen in der Lerngruppe: Einer schreibt den ersten Textentwurf, eine redigiert—das heißt, sie liest das Geschriebene ganz kritisch und gibt Feedback—, eine editiert und korrigiert den Text. Jeder in der Lerngruppe sollte im Laufe des Semesters mal geschrieben haben, jeder sollte mal gegengelesen haben, jeder sollte mal editiert haben.
- Die Abgabe erfolgt dann im Tutorium als ausgedrucktes DIN-A4 Blatt oder in Absprache bei eurem Tutor per E-Mail als PDF-Datei.
- Feedback erhälst du auf deine schriftliche Ausarbeitung von deinem Tutor im Tutorium selbst.
- Jedes Projekt darf nur bei einem Tutor abgegeben werden!