Graphentheorie v1.0

Theorie (½ EH)

- Was sind Graphen?
- Woraus bestehen Graphen?
- Was unterscheidet gerichtete von ungerichtete Graphen?
- Was unterscheidet azyklische von zyklische Graphen?
- Wie kann man Graphen speichern?

Programmierung (7½ EH)

Teilt die kommenden Aufgaben sinnvoll im Team auf.

Grundlagen (½ EH)

- Ladet das **Projekt `Graphtheory`** aus dem *Moodle*-Kurs herunter und öffnet es.
- Erstellt **keine weiteren Klassen**-Dateien, da sie beim automatischen Test ignoriert werden.
- **Innere Klassen** (z.B. anonyme oder verschachtelte) sind aber weiterhin erlaubt.
- Programmiert und testet: **`Graph#read(File)`**
    - Liest eine als `File` übergebene Adjazenzmatrizen wie bspw. *Flussproblem.csv* oder *Suchproblem.csv* ein und speichert sie.
    - Schreibt mindestens einen Unit-Test.

Suchprobleme (3 EH)

Szenario

Gegeben ist der Graph eines **Straßennetzes**: Die **Knoten** stehen dabei für die Kreuzungen und die **Kanten** für die Straßen dazwischen. Die **Kantengewichte** beschreiben die **Länge der Straßen** in Kilometern. Ziel ist es einen Algorithmus zu programmieren, welcher den **kürzesten Weg** zwischen zwei beliebigen Kreuzungen findet, um damit ein sinnvolles **Nachverkehrsangebot** aufbauen zu können.

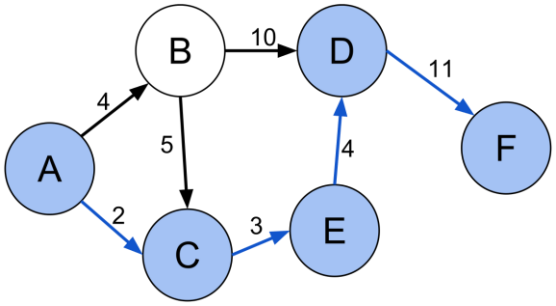

Abbildung 1: https://en.wikipedia.org/wiki/Shortest_path_problem

Aufgabe

- Lest die Datei *Suchproblem.csv* ein.
- Programmiert und testet: **`Graph#determineShortestPath(int, int)`**
    - Bestimmt den kürzesten Weg zwischen den beiden als Index übergebenen Knoten.
    - Gibt diesen Weg als Knoten-Indizes in einem `Path`-Objekt zurück (siehe unten).
    - Schreibt mindestens einen Unit-Test, behandelt auch ungültige Parameter.
- Programmiert und testet: **`Path#getNodeIds()`**
    - Gibt den gespeicherten Pfad als `Array` von geordneten Knoten-Indizes zurück.

    - Schreibt mindestens einen Unit-Test.
- Programmiert und testet: **`Path#computeDistance()`**
    - Berechnet die Länge des gespeicherten Pfades und gibt diese zurück.
    - Schreibt mindestens einen Unit-Test.
- Programmiert und testet: **`Graph#determineShortestPath(int, int, int...)`**
    - Bestimmt den kürzesten Weg zwischen den beiden als Index übergebenen Knoten mit vordefinieren Zwischenstationen als letztem Parameter (sog. `varargs`).
    - Schreibt mindestens einen Unit-Test, behandelt auch ungültige Parameter.

Flussprobleme (2 EH)

Szenario

Gegeben ist der Graph eines **Straßennetz**: Die **Knoten** stehen dabei für die Kreuzungen und die **Kanten** für die Straßen dazwischen. Die **Kantengewichte** beschreiben die **Kapazität an Fahrzeugen**, die die jeweilige Straße in einer bestimmten Zeit passieren können. Ziel ist es einen Algorithmus zu programmieren, welcher die **maximal mögliche Menge an Fahrzeugen** (sog. *Fluss*) zwischen zwei beliebigen Kreuzungen findet, um damit **Engpässe im Straßennetz** zu finden.

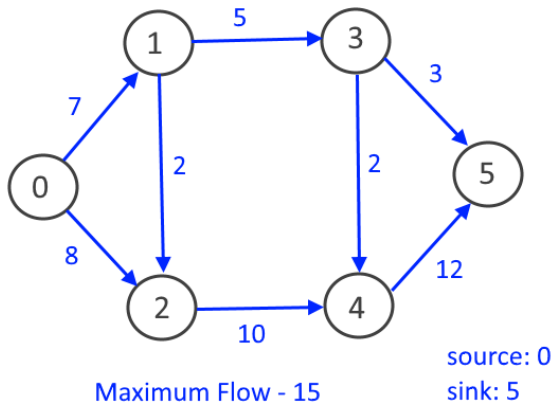

Abbildung 2: https://algorithms.tutorialhorizon.com/max-flow-problem-introduction/max-flow-graph/

Aufgabe

- Lest die Datei *Flussproblem.csv* ein.
- Programmiert und testet: **`Graph#determineMaximumFlow(int, int)`**
    - Bestimmt den maximalen Fluss zwischen den beiden als Index übergebenen Knoten.
    - Schreibt mindestens einen Unit-Test, behandelt auch ungültige Parameter.
- Programmiert und testet: **`Graph#determineBottlenecks(int, int)`**
    - Bestimmt alle Kanten im Graph, bei denen der maximale Fluss zwischen den beiden als Index übergebenen Knoten dazu führt, dass ihre Kapazität erschöpft ist.
    - Schreibt mindestens einen Unit-Test, behandelt auch ungültige Parameter.
- Programmiert und testet: **`Edge#getFirstNodeId()`** und **`Edge#getSecondNodeId()`**
    - Schreibt mindestens einen Unit-Test.
- Überlegt gemeinsam, wie man den zweitgrößten Fluss zwischen zwei beliebigen Kreuzungen finden kann.

Lineare Programmierung (2 EH)

Theorie

- Was ist lineare Programmierung (LP)?
- Sucht und erklärt eine einfache Problemstellung, die man mit LP lösen kann.

- Versucht, den Simplex Algorithmus zu verstehen.
- Was ist die GMPL?

GLPK

- Installiert GLPK.
- Sucht eine einfache GMPL-Beschreibung eines LP und löst sie mit GLPK.
- Schreibt ein Java-Programm, das die Adjazenzmatrix *Flussproblem.csv* in eine entsprechende GMPL-Beschreibung zur Bestimmung des maximalen Flusses übersetzt.
- Bestimmt mit GLPK und eurer generierten GMPL-Beschreibung den maximalen Fluss für die Adjazenzmatrix *Flussproblem.csv*.

Beurteilungskriterien

Wenn ihr das Thema abschließen wollt, ladet bitte euer Projekt als **Abgabe im *Moodle*** hoch und meldet euch zur **mündlichen Prüfung** an, welche normalerweise in der nächsten Unterrichtseinheit stattfindet.

In der Prüfung muss jeder von euch in der Lage sein, die **Theoriefragen** zu beantworten und euren **Code inkl. Tests** zu erklären. Wer aus eurem Team auf die Fragen antworten muss, entscheidet der Lehrer.

Sollte die mündliche Prüfung **nicht bestanden** werden, kann sie beliebig oft, frühestens aber eine Woche später wiederholt werden. Ihr dürft mit der **nächsten Aufgabe** erst dann starten, wenn das aktuelle Thema abgeschlossen wurde.

Die **Bewertung** bekommt ihr für die Programmierung gemeinsam als Gruppe. Eure Abgabe wird dafür mit einem **Plagiatscheck** überprüft und mit speziellen **Unit-Tests** getestet. Im Falle eines erkannten **Plagiats** werden die Punkte der kopierten Aufgabe(n) unter den betroffenen Gruppen aufgeteilt.