Prof. Dr. Frank Deinzer
Hochschule für angewandte Wissenschaften
Würzburg-Schweinfurt

Übungsblatt 1

Grundlagen von Graphen

Aufgabe 1

Grundlegende Graphenfunktionalität

a) Entscheiden Sie sich in Ihrer Kleingruppe für eine Programmiersprache Ihrer Wahl.

b) Suchen Sie nach einer geeigneten Graphen-Bibliothek für Ihre Programmiersprache. Zwei Möglichkeiten:

- C++: Boost Graph Library (BGL), `http://www.boost.org/libs/graph/`
- Java: JGraphT, `http://www.jgrapht.org/`

Achten Sie darauf, dass die Graphenbibliothek sowohl *gerichtete* als auch *ungerichtete* Graphen bereitstellt.

c) Realisieren Sie das Einlesen von Graphen in dem hier beschriebenen Dateiformat. Damit wollen wir sicherstellen, dass für zukünftige Übungsaufgaben bereitgestellte Graphen von allen Gruppen eingelesen werden können.

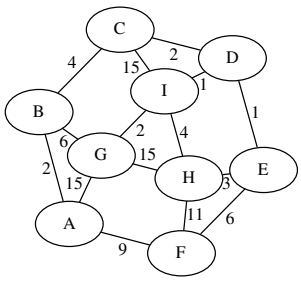
```
knoten NAME-1 [DATEN]
knoten NAME-2 [DATEN]
      ⋮
knoten NAME-n [DATEN]
# Kommentarzeilen beginnen mit #

# Leerzeilen werden ignoriert
kante VON-NAME NACH-NAME [GEWICHTE]
kante VON-NAME NACH-NAME [GEWICHTE]
      ⋮
kante VON-NAME NACH-NAME [GEWICHTE]
```

Die Angabe `GEWICHTE` und `DATEN` ist optional. Lesen Sie den Graphen aus Aufgabe 3 a) ein.

d) Suchen Sie nach einer Visualisierungsmöglichkeit für Ihre Graphen. Zwei Möglichkeiten:

- C++/BGL: Diese Bibliothek unterstützt direkt das Schreiben von Graphen im GraphViz-Format, `http://www.graphviz.org/`.
- Java/JGraphT: Diese Bibltiothek unterstützt eine Vielzahl an Exportformaten, z. B. *Microsoft Visio* oder auch die *DOT language*, die wieder von GraphViz gelesen und dargestellt werden kann.

Lesen Sie den Graphen aus Aufgabe 3 a) ein und visualisieren Sie ihn. Ihr Ergebnis könnte z. B. so aussehen:



Aufgabe 2

Grundlegende Graphenalgorithmen

a) Schlagen Sie in geeigneter Literatur das *Eulerkreisproblem* nach. Implementieren Sie einen Test, der prüft, ob ein gegebener Graph einen *Eulerkreis* oder einen *Eulerpfad* aufweist. Sie müssen keinen konkreten Kreis oder Pfad suchen, sondern nur eine Aussage treffen, ob der Graph einen Eulerkreis bzw. einen Eulerpfad haben muss.

Testen Sie Ihre Implementierung an den beiden Graphen `Euler1.txt` und `Euler2.txt`. Was stellen diese beiden Graphen dar? Was sagen die Ergebnisse zum Eulerkreis und -pfad aus?

b) Wie wird durch Ihre Graphenbibliothek technisch eine Tiefensuche (depth-first search, DFS) bereitgestellt? Implementieren Sie die Möglichkeit, den Ablauf der Tiefensuche darzustellen, z. B. einfach dadurch, dass alle besuchten Knoten eines Graphen in der Reihenfolge des Besuchs ausgegeben werden. Testen Sie dies an einigen Graphen.

c) Realisieren Sie einen Test, der prüft ob ein Graph einen Kreis enthält. Ziehen Sie geeignete Literatur heran, um ein geeignetes Verfahren auszuwählen. Testen Sie Ihre Lösung an einigen Beispielen.

Aufgabe 3

Fortgeschrittene Graphenalgorithmen

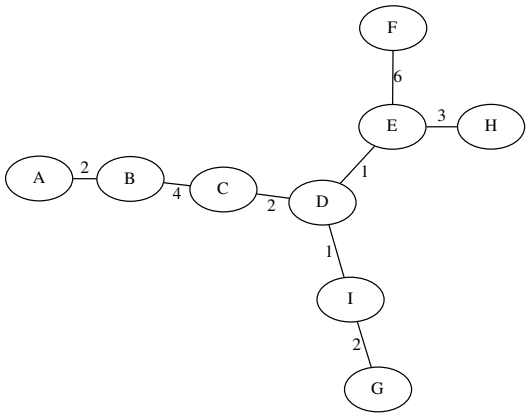
a) Schlagen Sie den Algorithmus von Dijkstra nach, der das Problem der kürzesten Pfade in Graphen löst.

Implementieren Sie diesen Algorithmus mit Hilfe Ihrer Graphenbibliothek. Verzichten Sie aber darauf, eine fertige Dijkstra-Implementierung zu verwenden, die Ihre Graphenbibliothek wahrscheinlich bietet. Machen Sie sich dabei ganz bewusst Gedanken, wie eine elegente Implementierung mit möglichst wenig Programmcode aussehen kann. Gegebenenfalls macht es Sinn, sich unterschiedliche Darstellungen des Algorithmus in der Literatur anzusehen.

Berechnen Sie mehrere kürzeste Wege für den Graphen `Dijkstra.txt` zwischen jeweils zwei Knoten, z. B. von (A) nach (D) und von (H) nach (B).

b) Informieren Sie sich darüber, was in der Graphentheorie als Spannbaum, vor allem als minimaler Spannbaum (minimum spanning tree) bezeichnet wird. Suchen Sie einen Algorithmus, der einen minimalen Spannbaum berechnen kann und implementieren diesen.

Testen Sie Ihren Algorithmus unter anderem an dem Graphen `Dijkstra.txt` und visualisieren Sie Ihr Ergebnis. Ein minimaler Spannbaum ist zum Beispiel:



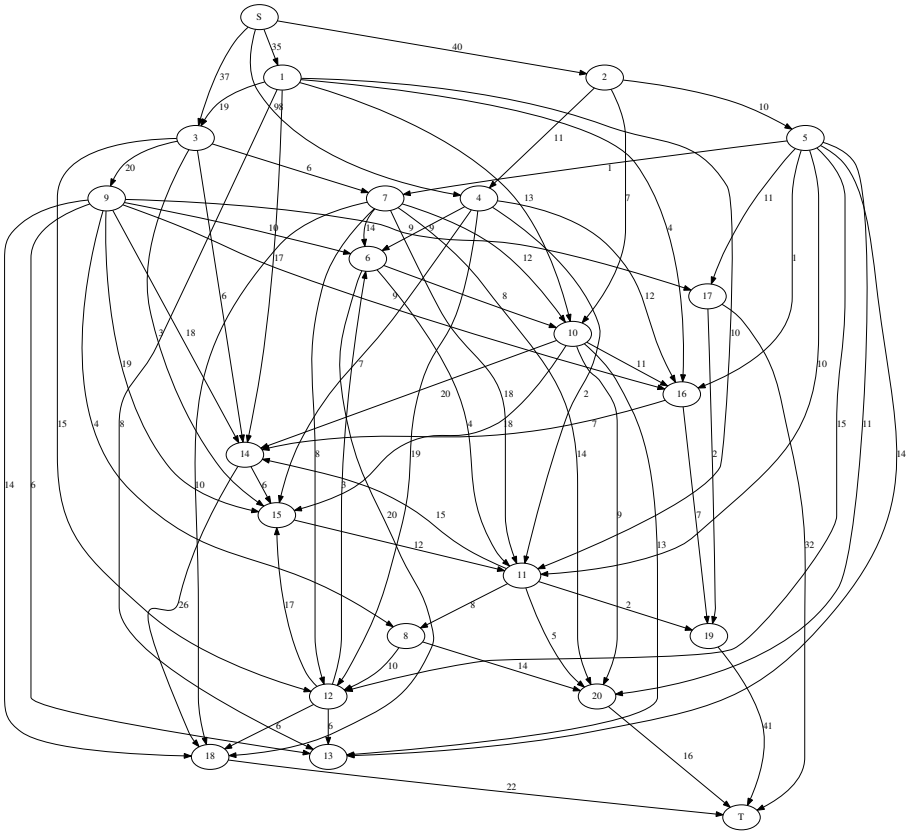
Aufgabe 4

Graphenalgorithmen zum maximalen Fluss

a) Schlagen Sie den Algorithmus von Ford-Fulkerson und den Algorithmus von Edmonds-Karp zur Berechnung des maximalen Flusses in einem Graphen nach.

Implementieren Sie einen dieser beiden Algorithmem mit Hilfe Ihrer Graphenbibliothek.

Testen Sie Ihre Implementierung an einigen Graphen; u. a. an den Graphen `Fluss.txt` und `Fluss2.txt`.



Hinweis: Alle folgenden Aufgaben lassen sich am Ende mit Hilfe des maximalen Flusses lösen.

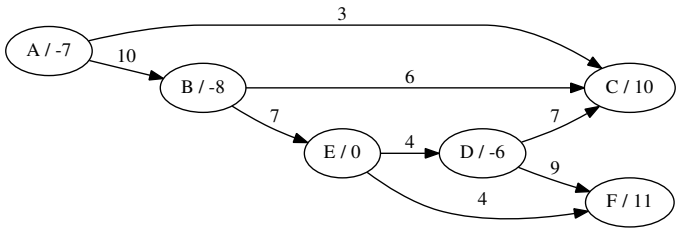
b) Zwei Pfade innerhalb eines Graphen sind kantendisjunkt, falls sie keine gemeinsame Kanten besitzen. Wie findet man in einem gerichteten Graphen die maximale Anzahl kantendisjunkter Pfade von einem Start- zu einem Zielknoten?

Sie möchten für ein komplexes Computernetzwerk prüfen, wie viele Verbindungen zwischen einem Start- und einem Zielknoten ausfallen dürfen, bevor mangels Routing-Möglchkeiten keine Datenverbindung mehr möglich ist. Wie lösen Sie dieses Problem?

Testen Sie Ihre Lösung am Graphen `Fluss2.txt` und geben sowohl die Anzahl kantendisjunkter Pfade an als auch die konkreten gefundenen Pfade.

c) Was versteht man unter dem Problem *Zirkulation mit Anforderungen* (engl. circulation with demands)? Was sind praktische Fragestellungen, die damit gelöst werden können? Wie löst man dieses Problem mit Hilfe des maximalen Flusses?

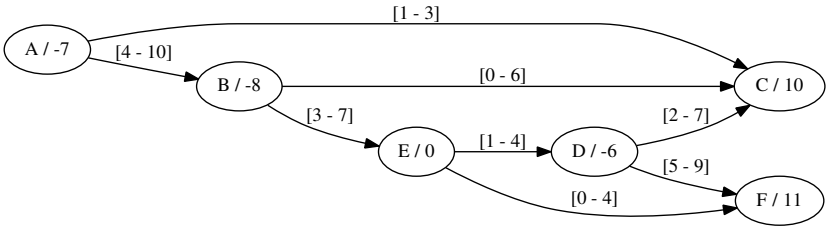
Existiert für den Graphen `CirculationDemands.txt` eine Lösung im Sinn einer Zirkulation?



Positive Knotenwerte geben Anforderungen d.h den Bedarf der Knoten an. Negative Werte dagegen geben das Angebot an. Knoten mit einem Wert von Null dienen nur als Umschlagplatz.

d) Was versteht man unter dem Problem *Zirkulation mit Anforderungen und unterer Grenze*? Wie löst man dieses?

Existiert für den Graphen `CirculationDemandsLowerBounds.txt` eine Lösung im Sinn dieser Art von Zirkulation?



Aufgabe 5

Praktische Aufgabenstellungen

a) *Umfragedesign*

Sie sollen folgende praktische Aufgabe lösen: Entwerfen Sie eine Umfrage, bei der Kunden zu den von ihnen gekauften Produkten befragt werden. Dabei sollen n_1 Kunden zu n_2 Produkten befragt werden. Jeder Kunde i kann nur dann zu einem Produkt j befragt werden, wenn er es auch erworben hat. Um die Akzetanz bei den befragten Kunden sicherzustellen, sollen jedem Kunden i nur zwischen c_i und c_i' Fragen gestellt werden. Um eine ausreichende Aussagekraft sicherzustellen, sollen zu einem Produkt j zwischen p_j und p_j' Kunden befragt werden.

Wie prüft man, ob eine Umfrage mit diesen Randbedingungen realisierbar ist? Wie erstellt man konkret die Umfrage?

b) *Projektplanung mit Voraussetzungen*

Gegeben ist eine Menge P von Projekten. Jedes Projekt $v \in P$ hat einen Ertrag von p_v. Mit manchen Projekten verdient man Geld ($p_v > 0$, z. B. Webseite für Kunden erstellen) und manche Projekte kosten Geld ($p_v < 0$, z. B. Kosten für Lizenzen von Software zur Webseitenerstellung). Zudem gibt es Abhängigkeiten (v, w) zwischen den Projekten: Ein Projekt v kann erst dann durchgeführt werden, wenn Projekt w umgesetzt wurde.

Wie findet man die optimale Menge von Projekten, die am Ende in Summe einen maximalen Ertrag liefern und dabei alle Abhängigkeiten erfüllen?