**Abgabe 5 - Angewandte Mathematik - 02.06.2019**

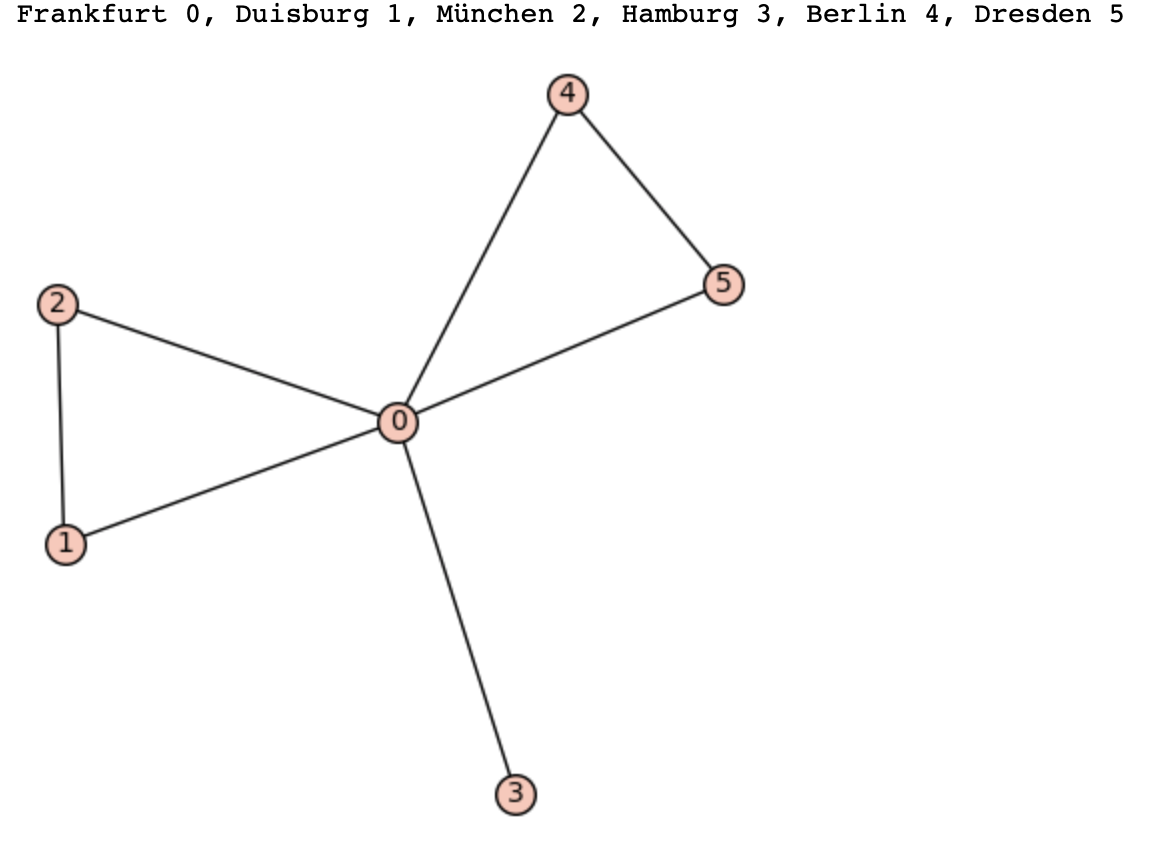
**Beschreibung der Aufgabenstellung:**

Abgabe 5 behandelt die Netzplantechnik. Genauer wird eine Fluglinie eingeführt, welche mehrere deutsche Städte verbinden soll. Diese Vernetzung soll in Form einer Adjazenzmatrix abgebildet werden.

Diese Matrix bildet die Grundlage. Darauf aufbauend soll eine Funktion implementiert werden, welche die Anzahl der Umstiege ermittelt.

Zum Schluss soll unabhängig der Matrix, ein Algorithmus implementiert werden, welcher die maximalen Umstiege zu 2 gegebenen Destinationen wiedergibt.

**Gegeben:**



**Herangehensweise und Probleme Aufgabe 1:**

Die jeweiligen Verbindungen wurden bereits gegeben. Diese wurden einem Graphen übermittelt und geplottet.

**Herangehensweise und Probleme Aufgabe 2:**

Hierbei haben wir zwei Lösungswege. Zum einen haben wir die Adjazenzmatrix auf einem Blatt Papier bestimmt und somit eine feste Lösung für unsere gegebenen Punkte aus der Aufgabe 1 bestimmt. Da Aufgabe 3 & 4 jedoch mit beliebigen Netzen funktionieren soll, haben wir uns zusätzlich überlegt, dass wir eine Liste mit Pairs <Stadt-A, Stadt-B> erstellen. Diese repräsentiert alle Städte, die nebeneinander liegen. Aus dieser Liste erzeugt dann unsere Funktion *generateMatrix* eine Matrix, welche jedes Pair auswertet und diese dann in eine Adjazenzmatrix überträgt.

**Herangehensweise und Probleme Aufgabe 3:**

Problem: Wir müssen uns eine geeignete Datenstruktur überlegen mit der wir die Städte sinnvoll und so einfach wie möglich verwalten können und darüber hinaus eine Zuordnung von Stadt zu Nachbarstadt haben.

Lösung: Für uns stand ganz am Anfang schon fest, dass diese Aufgabe sehr einfach mit Hilfe von Hashmaps lösbar ist, um genau zu sein eine Hashmap<integer,List[integer]>. Der Integer Key repräsentiert dabei die aktuelle Stadt. Diesem Integer Wert wird eine Liste mit Integern als Value übergeben; in dieser Liste stehen alle zu erreichbaren Städte. Die „direkte Nachbar“-Hashmap (Städte, die ohne einen Umstieg erreichbar sind) füllen wir mit der Matrix, welche in Aufgabe 2 beschrieben wurde.

Um nun die „Nachbarn vom Nachbarn“ (Städte, welche mit einmal umsteigen erreichbar sind) heraus zu finden iterieren wir von jeder Stadt einmal über jeden direkten Nachbarn und schauen welche Nachbarn dieser hat. Diese Einträge speichern wir dann in der Hashmap für die „Nachbarn vom Nachbarn“, um nun die Lösung für den „Nachbarn vom Nachbarn vom Nachbarn“ zu bekommen. Haben wir nun nicht die direkte Nachbar-Hashmap als Ausgangslage, sondern die „Nachbar vom Nachbar“-Hashmap, iterieren wir wieder wie zuvor von jeder Stadt einmal über jeden direkten Nachbarn und schauen welche Nachbarn diese hat. Diese speichern wir wieder in eine neue Hashmap und erhalten somit die Hashmap „Nachbar vom Nachbar von Nachbar“ (Städte, welche mit zweimaligen umsteigen erreichbar sind).

**Herangehensweise und Probleme Aufgabe 4:**

Bei dieser Aufgabe wird die Anzahl der maximalen Umstiege zu 2 Stationen gesucht. Hierbei wichtig ist, dass die Funktion nicht loopt, bedeutet dass jeder Weg nur einmal benutzt wird. Die Idee ist eine rekursive Methode, die sich mit den Nachfolgern des Startpunktes aufruft, solange bis das Ziel erreicht wurde. Der Startpunkt wird bei jedem Durchlauf durch die Nachfolger ersetzt. Um sicherzugehen, dass nicht geloopt wird, wird der Funktion bei jedem Aufruf eine Liste mitgegeben, die die Vorgänger beinhaltet. Durch einen Check, ob der Nachfolger nicht in dieser Liste ist, wird das loopen verhindert.

Jedesmal wenn ein Durchlauf auf den Nachfolger trifft, wird die Anzahl der Umstiege gespeichert. Dieser Wert wird überschrieben sobald ein Durchlauf einen höheren Wert liefert. (counter = Anzahl Umsteige, max = maximale Umstiege).

Sind alle Durchläufe beendet, gibt die Funktion des max Wert aus.