Übung 10: Breiten- und Tiefensuche

Aufgabe 1: Breitensuche

Die Adjazenzlisten der Knoten in den folgenden Graphen seien ***jeweils aufsteigend alphabetisch*** bzw. numerisch sortiert, die Nachbarknoten werden also immer in ***aufsteigender*** Reihenfolge besucht.

a) Auf dem ***gerichteten*** Graphen der Abbildung 1 wird eine Breitensuche mit der **3** als ***Startknoten*** durchgeführt. Geben Sie an, in welcher Reihenfolge die Knoten „schwarz“ eingefärbt werden! Geben Sie ferner die sich ergebenden Werte von ***d*** und ***π*** für jeden Knoten an!
Hinweis: Der *d* Wert entspricht der Entfernung zum Startknoten, der *π*-Wert dem Vorgängerknoten im Kürzeste-Wege-Baum.

b) Auf dem ***ungerichteten*** Graphen der Abbildung 2 wird eine Breitensuche mit ***u*** als **Startknoten** durchgeführt. Geben Sie an, in welcher Reihenfolge die Knoten „schwarz“ eingefärbt werden! Geben Sie ferner die sich ergebenden Werte von ***d*** und ***π*** für jeden Knoten an!

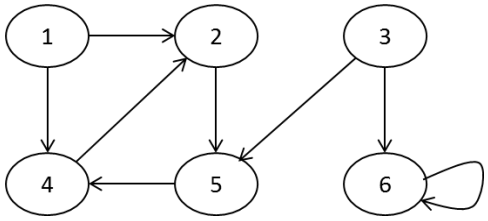


Abbildung 1

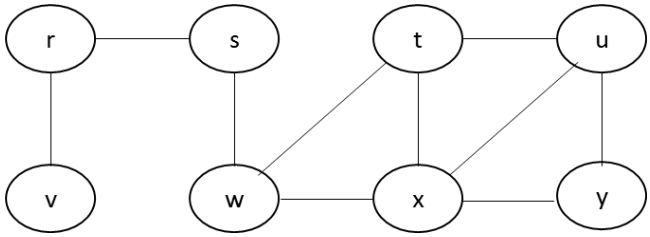
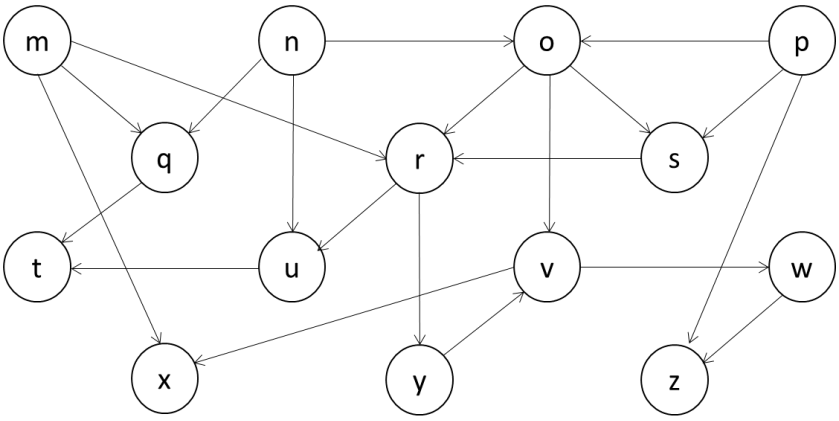


Abbildung 2

c) Sie haben mit der Breitensuche (Startknoten *s*) für 2 Knoten *u* und *v* die Werte *u.d* und *v.d*. berechnet. Was können Sie über die Entfernung zwischen *u* und *v* sagen?

Aufgabe 2: Tiefensuche



Führen Sie eine Tiefensuche auf dem abgebildeten gerichteten Graphen durch. Verwenden Sie dazu den Algorithmus der Vorlesung. Gehen Sie davon aus, dass der Algorithmen Knoten stets in ***alphabetisch aufsteigender*** Reihenfolge besucht. Ferner seien die Adjazenzlisten jedes Knoten ***alphabetisch sortiert***. Es wird also mit dem Knoten *m* begonnen.

a) Geben Sie die *Discovery Time* ***d*** und *Finish Time* ***f*** für jeden Knoten an!
b) Geben Sie die Reihenfolge an, in der die Knoten schwarz gefärbt werden!
c) Markieren Sie alle Kanten, die bei der Tiefensuche verwendet werden, also alle Kanten die über ***v.π*** gespeichert werden! Diese ergeben den sogenannten „Tiefensuchenwald“.
d) Geben Sie eine topologische Sortierung der Knoten an!

Aufgabe 3: Iterative Tiefensuche in Java

Implementieren Sie die Tiefensuche in einem gerichteten Graphen mit Hilfe eines Stacks und ***ohne Rekursion.*** Verwenden Sie dazu die 2 Dateien aus der Community und implementieren Sie in der Klasse `DepthFirstSearch.java` die folgende Methode, die ausgehend von einem **Startknoten** `u` eine Tiefensuche startet.

```
public void iterativeDFS(Graph G, int u)
```

Hinweise:

- Schauen Sie sich kurz den vorgegebenen rekursiven Code der Methode `dfs(GraphG, int u)` an. Das Verhalten der neuen Methode soll identisch zu dieser Methode sein.
- Gehen Sie davon aus, dass vom Startknoten `u` alle anderen Knoten des Graphen erreichbar sind.
- Färben Sie Knoten analog zum rekursiven Algorithmus der Vorlesung weiß, grau und schwarz ein.
- Testen Sie mit der main-Methode. Der getestete Graph entspricht dem rechts abgebildeten Graphen. Der Startknoten ist der Knoten 0.
- *Tipp:* Element nicht sofort vom Stack nehmen sondern erst mal anschauen („`peek`“)!