Universität Leipzig
Institut für Informatik
Abteilung Datenbanken
Prof. Dr. E. Rahm
T. Böhme, Dr. P. Richter

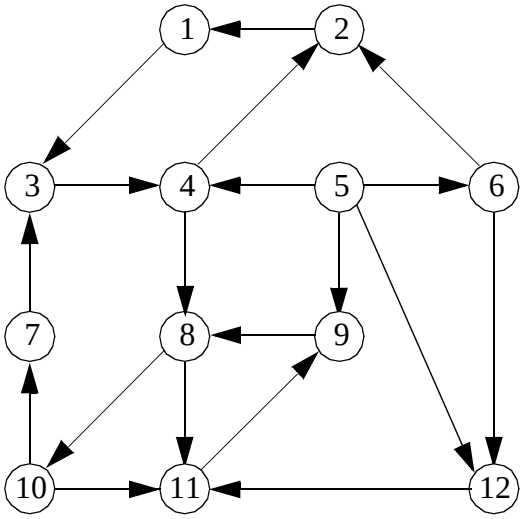
Ausgabe: 10.06.2002
Abgabe: 24.06.2002

Algorithmen und Datenstrukturen 2
SS 2002 - Übungsblatt 5

Hinweis: Alle *Listings* müssen in der ersten Zeile nach der Klassendefinition den *Namen des Programmautors* als Kommentar enthalten. Ohne diesen Namen werden die Listings **nicht gewertet**. Ebenfalls führt das Fehlen eines geforderten Ausdrucks der Programmausführung zur Nichtbewertung der Aufgabe.

1. Aufgabe (Graphen-Theorie) (6 Punkte)

Gegeben sei der folgende gerichtete Graph



Bestimmen Sie zu nachfolgend vorgegebenen Knotenfolgen, ob es sich um eine *Kantenfolge*, einen *Kantenzug*, einen *Weg* oder einen *Zyklus* handelt (geben Sie jeweils alle zutreffenden Eigenschaften an).

a) 4, 8, 11, 9, 8, 10

b) 11, 9, 8, 4, 2

c) 7, 3, 4, 8, 11, 9, 8, 10, 7

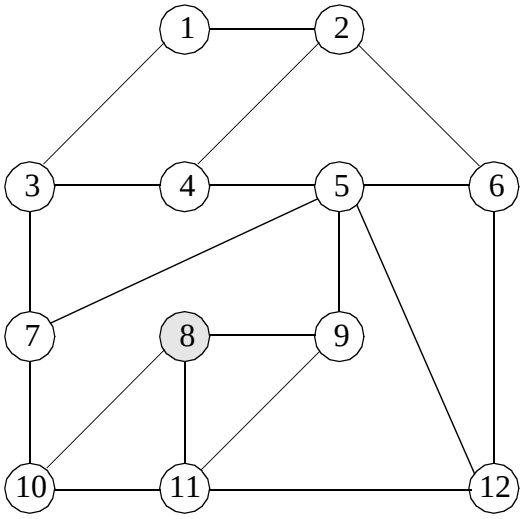
d) 3, 4, 2, 1, 3, 4, 8

e) 5, 6, 12, 11, 9

f) 4, 8, 10, 7, 3, 4

2. Aufgabe (Breitensuche) (4 Punkte)

Gegeben sei der nachfolgende ungerichtete Graph



a) Führen Sie auf diesem Graph eine Breitensuche beginnend beim Knoten 8 durch. Bearbeiten Sie dabei die von einem Knoten ausgehenden Kanten in der Reihenfolge der Nummerierung der Zielknoten (von klein nach groß).
Geben Sie die resultierende Traversierungsreihenfolge der Knoten sowie deren Distanz zum Startknoten an.

b) Geben Sie den Spannbaum an, der sich aus der in Teilaufgabe a durchgeführten Breitensuche ergibt.

3. Aufgabe (Topologische Sortierung) (6 Punkte)

Ein Softwareprojekt besteht aus 12 Klassen (c1, c2, ..., c12), die folgende Abhängigkeiten aufweisen:

Klasse	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10	c11
wird von Klassen benötigt	c3, c4	c3, c6	c6	c7	c1, c4, c7	c9	c10	c9	c11	c6, c8, c12	c12

a) Erstellen Sie für die 12 Klassen einen Abhängigkeitsgraphen (einen Graph, der mittels gerichteter Kanten die Abhängigkeiten zwischen den Klassen enthält).

b) Wenden Sie auf dem Graphen die Tiefensuche an und bestimmen Sie die *in*- und *out*-Zeitpunkte. Bestimmen Sie damit eine mögliche Reihenfolge, in der alle Klassen des Projektes übersetzt werden können (topologische Sortierung).

4. Aufgabe (Dijkstra-Algorithmus) (12 Punkte)

Implementieren Sie in Java die Bestimmung des kürzesten Weges zwischen 2 Knoten eines Graphen mittels Dijkstra-Algorithmus. Auf der Web-Übungsseite haben wir Ihnen dafür die Klasse *Graph* bereitgestellt, die einen Graphen implementiert. Die darin als Rumpf vorgegebene Methode *public void printShortestRoute (Object startKey, Object endKey)* ist von Ihnen zu

implementieren. Sie soll den kürzesten Weg zwischen den beiden Knoten, die durch die Knotenbezeichner *startKey* und *endKey* definiert sind, ausgeben. Neben dem Gesamtgewicht für diesen Weg soll auch die Knotenfolge ausgegeben werden. Die Knotenfolge kann durch minimale Erweiterung des vorgegebenen Dijkstra-Algorithmus gespeichert werden. Hierfür wird, wie bei der Breitensuche, in einer pred-Struktur der (optimale) Vorgänger eines Knotens gespeichert. Der Eintrag darin erfolgt jedes Mal, wenn sich für einen Knoten der Distanzwert ändert (d.h. wenn D[u]=D[v]+g((v,u)) dann pred[u] = v).

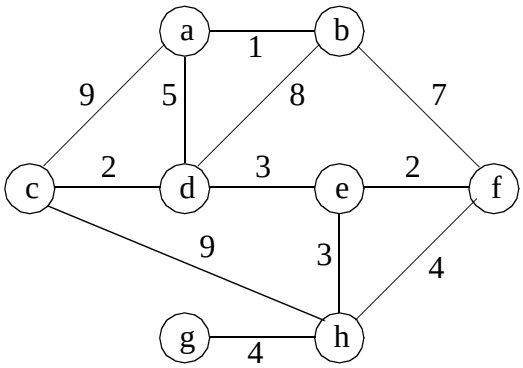
Überprüfen Sie die Implementierung mit dem ebenfalls bereitgestellten Programm *Ueb5Test*, welches einen Graphen mit Knoten (Städten) und Kanten initialisiert und für drei Städtepaare die kürzeste Verbindung ermittelt. Für das Übersetzen und Ausführen des Programms benötigen Sie die auf der Web-Übungsseite bereitgestellten Dateien *Vertex.java*, *Edge.java*, *Graph.java*, *GraphException.java* und *Ueb5Test.java*.

Abgabe:

1. dokumentiertes Listing Ihrer Implementierung
2. Ausgabe eines Programmlaufs von *Ueb5Test*

5. Aufgabe (Minimale Spannbäume) - optional (4 Punkte)

Gegeben sei der folgende gewichtete Graph



Bestimmen Sie für diesen Graphen einen minimalen Spannbaum mit Hilfe des Kruskal-Algorithmus analog zum Beispiel auf Folie 3-37.