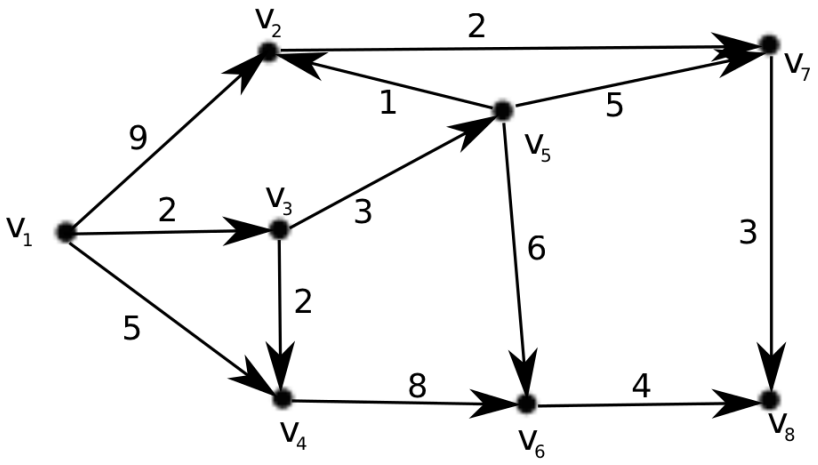
Abteilung Algorithmik
Institut für Betriebssysteme und Rechnerverbund **SoSe 14**
TU Braunschweig

Dr. Christiane Schmidt
Florian Maurer
Arne Schmidt

Netzwerkalgorithmen
Übung 3 vom 28.05.2014

Abgabe der Lösungen bis Mittwoch, den 25.06.14, bis 13:00 Uhr in der Abteilung *Algorithmik*.
Bitte die Blätter vorne deutlich mit eigenem Namen und Gruppennummer versehen!

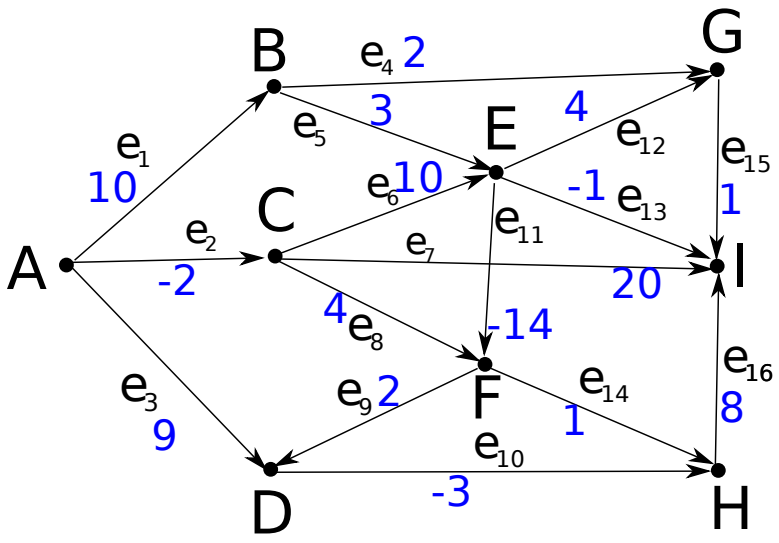
Aufgabe 1 (Algorithmus von Dijkstra):



Bestimme mit Hilfe des Algorithmus von Dijkstra einen kürzesten Weg von v_1 nach v_8. (Hinweis: Kommen während einer Iteration mehrere Knoten in Frage, wähle den mit dem kleinsten Index.) Gib jeweils an, wenn sich Label und Vorgänger ändern.

(10 Punkte)

Aufgabe 2 (Algorithmus von Moore, Bellman und Ford):



Bestimme mit dem Moore-Bellman-Ford-Algorithmus einen kürzesten Pfad von A nach I. Gib jeweils an, wenn sich Längenwerte oder Vorgänger ändern.

(10 Punkte)

Aufgabe 3 (Fibonacci Heaps):

Bereite einen Kurzvortrag zum Thema "Fibonacci Heaps" vor - d.h. Du solltest in 5-15 Minuten in der Lage sein, Deinen Kommilitonen zu erläutern, wie diese Datenstruktur aussieht, warum man sie benutzen möchte, und wie die Operationen Insert, Union und DeleteMin aussehen. Siehe z.B. Kapitel 19 im Buch "Algorithmen - eine Einführung" von Cormen, Leiserson, Rivest und Stein.

Erläuterung der Punktevergabe für diese Aufgabe - analog zu Blatt 1: Wir nennen das mal Vertrauensaufgabe - Ihr tragt Euch zu Beginn der kleinen Übung in eine Liste ein, womit Ihr sagt "Ja, ich habe diese Aufgabe bearbeitet, möchte also gerne die 10 Punkte, d.h. auch, wenn ich ausgewählt werde, kann ich problemlos den Kurzvortrag halten." - Florian und Arne wählen dann zufällig jemanden aus der Liste, und wenn der-/diejenige dann keinen Kurzvortrag hat, gibt es logischerweise keine Punkte, und ein Nächster wird ausgewählt - alles klar?

(10 Punkte)

Aufgabe 4 (Kürzeste Wege in Graphen mit beliebigen Gewichten):

Die Algorithmen von Dijkstra und Moore-Bellman-Ford für nichtnegative bzw. konservative Kantengewichte nutzen Lemma 3.3. Zeige, dass für allgemeine Graphen das Minimum nicht bestimmt ist, d.h., zeige, dass es Graphen mit beliebigen reellen Kantengewichten gibt, für

die es zwischen zwei Knoten s und t keinen kürzesten Weg gibt.

(5 Punkte)

Aufgabe 5 (Dijkstra und MSTs):
Gegeben sei ein ungerichteter Graph G mit Kantengewichten $c : E(G) \mapsto \mathbb{R}$ (nicht paarweise verschieden).
Zeige oder widerlege: Der mit dem Dijkstra-Algorithmus berechnete Kürzeste-Wege-Baum ist auch ein minimaler aufspannender Baum.

(10 Punkte)

Aufgabe 6 (Kürzeste Wege zwischen allen Knoten):
Betrachte das folgende Problem:
ALLE KÜRZESTE WEGE
Gegeben: Gerichteter Graph G, konservative Gewichte $c : E(G) \to \mathbb{R}$.
Gesucht: kürzeste Wege für alle $s, t \in V(G)$, d.h.
l_{st}: Länge des kürzesten $s - t-$Pfades
p_{st}: Vorgänger von t in einem kürzesten $s - t-$Pfad.

(a) Welche Laufzeit liefert Lösung des Problems mittles wiederholter Anwendung von Moore-Bellman-Ford?

(b) Betrachte den Algorithmus von Floyd und Warshall, Algorithmus 1, Seite 3. Welche Laufzeit hat der Algorithmus?

(c) Beweise die Korrektheit des Algorithmus von Floyd und Warshall. Hinweis: Zeige die folgende Behauptung:
Nach Durchlaufen der äußeren Schleife mit $j = 1, \ldots, j_0$ enthält die Variable l_{ik} die Länge eines kürzesten $i - k-$Pfades, der nur die Zwischenknoten $1, \ldots, j_0$ benutzt; (p_{ik}, k) ist die letzte Kante eines solchen Pfades.

(2+3+10 Punkte)

Algorithm 1: Floyd, Warshall (1962)

Input : Digraph G mit konservativen Gewichten, $c : E(G) \to \mathbb{R}$
Output: Für jedes Knotenpaar $i, j \in V(G)$ die Angaben: l_{ij}: Länge des kürzesten $i-j-$Pfades, p_{ij}: Vorgänger von j in einem kürzesten Pfad.

Betrachte $V(G) = \{1, \ldots, n\}$

1.

$l_{ij} := c((i,j))$ für alle $(i,j) \in E(G)$

$l_{ij} := \infty$ für alle $(i,j) \in V(G)^2 \backslash E(G), i \neq j$

$l_{ii} := 0$ für alle $i \in V(G)$

$p_{ij} := i$ für alle $(i,j) \in E(G)$

2.

for $j = 1$ **to** n **do**
  for $i = 1$ **to** n **do**
    if $i \neq j$ **then**
      for $k = 1$ **to** n **do**
        if $k \neq j$ **then**
          if $l_{ik} > l_{ij} + l_{jk}$ **then**
            $l_{ik} := l_{ij} + l_{jk}$;
            $p_{ik} := p_{jk}$