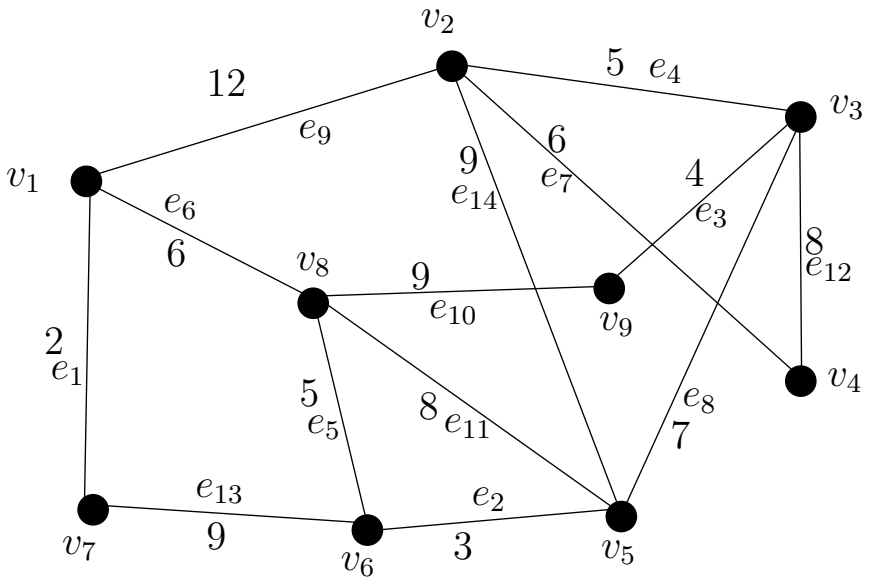
Abteilung Algorithmik
Institut für Betriebssysteme und Rechnerverbund **SoSe 11**
TU Braunschweig

Prof. Dr. Sándor P. Fekete
Christiane Schmidt

Netzwerkalgorithmen
Übung 1 vom 18.04.2011

Abgabe der Lösungen bis Montag, den 09.05.11, bis 13:00 Uhr in der Abteilung *Algorithmik* (IZ 262).
Bitte die Blätter vorne deutlich mit eigenem Namen und Gruppennummer versehen!

Aufgabe 1 (Algorithmus von Kruskal):



Bestimme mit Hilfe des Algorithmus von Kruskal einen minimalen aufspannenden Baum. Gib dazu die Kanten in der Reihenfolge an in der sie in den Baum aufgenommen werden und zeichne die gefundene Gesamtlösung. Kommen in einem Schritt mehrere Kanten in Frage, wähle die mit dem kleinsten Kantenindex.
Damit der Algorithmus von Kruskal eine Laufzeit von $O(m \log n)$ erreicht, kann man die in der Vorlesung vorgestellte Datenstruktur verwenden. Gib jeweils nach dem Einfügen einer Kante den Zustand der Datenstruktur an.
(Hinweis: Kommen beim Einfügen einer Kante in die Datenstruktur zwei Möglichkeiten in Frage, wähle die Kante, so dass sie vom Knoten mit dem kleineren Index zum Knoten mit dem größeren Index verläuft.)

(15 Punkte)

Aufgabe 2 (Divide and Conquer):
Professor Axtimwalde schlägt den folgenden Divide-and-Conquer-Algorithmus zur Bestimmung eines minimalen aufspannenden Baumes vor:
Für einen gegebenen zusammenhängenden Graphen $G = (V, E)$ wird die Knotenmenge V so in zwei Mengen V_1 und V_2 aufgespalten, dass sich ihre Kardinalitäten um höchstens eins unterscheiden. Sei E_1 die Menge der Kanten, die nur zu Knoten aus V_1 inzident sind und E_2 die Menge der Kanten, die nur zu Knoten aus V_2 inzident sind. Rekursiv wird nun das Minimum-Spanning-Tree-Problem für jeden der beiden Teilgraphen $G_1 = (V_1, E_1)$ und $G_2 = (V_2, E_2)$ gelöst. Schließlich wird eine Kante von E mit minimalem Gewicht gewählt, für die ein Endknoten in V_1 und der andere in V_2 liegt, um die beiden resultierenden Bäume zu verbinden.
Funktioniert das immer? Zeige entweder, dass der Algorithmus immer einen minimalen aufspannenden Baum für G bestimmt, oder gib ein Beispiel an, in dem er es nicht schafft. (Welche Laufzeit hätte der Algorithmus?)

(15 Punkte)

Einschub - Gerichtete Graphen: Für einen gerichteten Graphen $D = (V, A)$ ist $G = (V, E)$ mit $E = \{\{v, w\} | (v, w) \in A\}$ der zu D gehörende ungerichtete Graph. Ein gerichteter Graph is zusammenhängend, wenn der ungerichtete Graph zusammenhängned ist.
Ein gerichteter Graph ist ein *Branching*, wenn der zugrundeliegende ungerichtete Graph ein Wald ist, und jeder Knoten nur eine eingehende Kante hat. Eine *Arboreszenz* ist ein zusammenhängendes Branching. Der eindeutige Knoten in einer Arboreszenz mit Eingrad 0 ist die *Wurzel*. Ein gerichteter Graph ist *azyklisch*, wenn er keinen gerichteten Kreis enthält.

Aufgabe 3 (Gerichtete, azyklische Graphen): Zeige: In einem gerichteten, azyklischen Graphen können die Knoten so numeriert werden, dass für alle Kanten (i, j) $i < j$ gilt.

(15 Punkte)

Aufgabe 4 (Kürzeste r-Arboreszenzen): Gegeben sei ein gerichteter Graph $D = (V, A)$, ein Knoten r, und eine Längenfunktion $\ell : A \to \mathbb{Q}_+$.
Wir suchen eine kürzeste r-Arboreszenz (also eine in r verwurzelte Arborezenz).
Der Greedy-Algorithmus für dieses Problem lautet folgendermaßen:
Starte bei r und erweitere iterativ eine r-Arboreszenz der Teilmenge U von V durch die kürzeste Kante, die aus U herausgeht.
Liefert dieser Algorithmus eine kürzeste r-Arboreszenz? Beweise Deine Behauptung.

(15 Punkte)