Testcode für beide Aufgaben inklusive Beispieldaten finden sich auf der Kursseite.
Achtung: Beachten Sie die geforderten Klassennamen/Schnittstellen.

1 Tarjan's SCC Algorithmus (7.5 Punkte)

Implementieren Sie den in der Vorlesung vorgestellten Algorithmus zum Auffinden der Starkzusammenhangskomponenten als Visitor, analog zu dem in der letzten Übung zu implementierenden Visitor.
Geben Sie als Resultat eine Liste von Listen von Knoten (Integers) zurück, die die Komponenten darstellen.

Tipps:

- Der `elsif`-Fall im Pseudocode bedeuted, dass eine Nicht-Baumkante benutzt wird.
- Weil wir `VertexPropertyMap` benutzen, kann man den Fall, dass für `low` noch nichts in der Map steht, als $+\infty$ interpretieren. Man spart Initialisierung, muss aber eventuell Fallunterscheidungen treffen.
- Außerdem können Sie `low` dazu benutzen, den Test durchzuführen, ob ein Knoten noch in `S` enthalten ist. Dazu löschen Sie den entsprechenden Eintrag für einen Knoten in `low`, wenn Sie ihn vom Stack `S` entfernen; er wird dann ohnehin nicht mehr gebraucht.

2 Dijkstra's SSSP Algorithmus (7.5 Punkte)

Implementieren Sie den Single Source Shortest Path Algorithmus von Dijkstra für ein Paar von Knoten (source und target). Benutzen sie dazu als `EdgeInfo` ein `Integer`.
Brechen Sie ab, wenn der Zielknoten erreicht ist und geben Sie den besten Pfad als Liste von `Edge`s zurück, nicht als Liste von Knoten, wie in der Vorlesung beschrieben.
Implementieren Sie Q mit einer `PriorityQueue`, und nehmen Sie den Nachteil für `lower_key` beim Anpassen der Distanz in Kauf.

Abgabetermin ist der 17. Januar

Bitte nur in gewohnter Form an steffen@dfki.de