Universität Karlsruhe
Fakultät für Informatik
ITI Prof. Dr. D. Wagner

Praktikum
WS 2007/08
09. November 2007

Übungsblatt 2

Abgabe: **07. Dezember 2007**

Voraussetzungen:

Die Implementation der statischen Graphstruktur, soll falls nicht anderweitig mit dem Betreuerer abgesprochen, zwei Templateparameter für Knotenkennzeichnung (`NODE_ID`) und Kantengewichte (`EDGE_WEIGHT`) besitzen. Sie können im Weiteren davon ausgehen, dass beide mittels `<<`-operator ausgegeben beziehungsweise mittels `>>`-operator eingelesen werden können. Desweiteren besitzt `NODE_ID` eine Ordnung, d.h. die entsprechenden Vergleichoperatoren ($==$, $!=$, $<$, $\leq$, $>$ und $\geq$) sind verfügbar. Für erste Tests (gerade mit den bereitgestellten Daten) können sie von `NODE_ID=int` und `EDGE_WEIGHT=int` oder `double` ausgehen.
Die Nutzung von implizitem Wissen, wie, dass die `NODE_ID` in den Beispielgraphen natürliche Zahlen sind, die den Wertebereich von 0 bis $n-1$, wobei n die Anzahl der Knoten ist, abdecken, sollte vermieden werden!

Aufgabe 1: Implementationsskizze

Erstellen Sie eine Implementationsskizze für die statische Graphdatenstruktur, die Sie implementieren sollen, die die folgenden Punkte abdeckt:

- grobe Umsetzung von abstrakten Strukturen in C++ Datenstrukturen (dazu gehören auch die entsprechenden internen Verwaltungsstrukturen!)
- Designfragen/-entscheidungen

  Hier sollte man besonderes auf die Unveränderlichkeit des Graphen achten. Welche Erleichterungen werden deswegen möglich, welche Hürden müsste man überwinden, wenn man (anschliessend) die Struktur für den dynamischen Fall anpassen will.
- Adaptionsmechanismen für Algorithmen

  Da die Evaluation Ihrer Datenstruktur primär den Verbrauch von Laufzeit/Speicherplatz bei der Ausführung von einfachen Algorithmen (BFS, DFS, Dijkstra, etc.) berücksichtigen wird, stellt sich die Frage nach Anpassungsmöglichkeiten der Algorithmen an Ihre Struktur. Dies könnte etwa durch Bereitstellung von angepassten Klassen für `node_map` und `edge_map` geschehen. Sammeln Sie Ideen/Vorschläge/Umsetzungsmöglichkeiten für solche Optimierungen.

Aufgabe 2: Implementation

Implementieren Sie die Datenstruktur basierend auf Ihrer Implementationsskizze aus Aufgabe 1.

Datenstrukturen

Adjazenzliste:

Knoten werden in einem Array/einer Liste gespeichert und zu jedem Knoten gibt es eine eigene Liste, in der alle seine ausgehenden Kanten gespeichert sind.

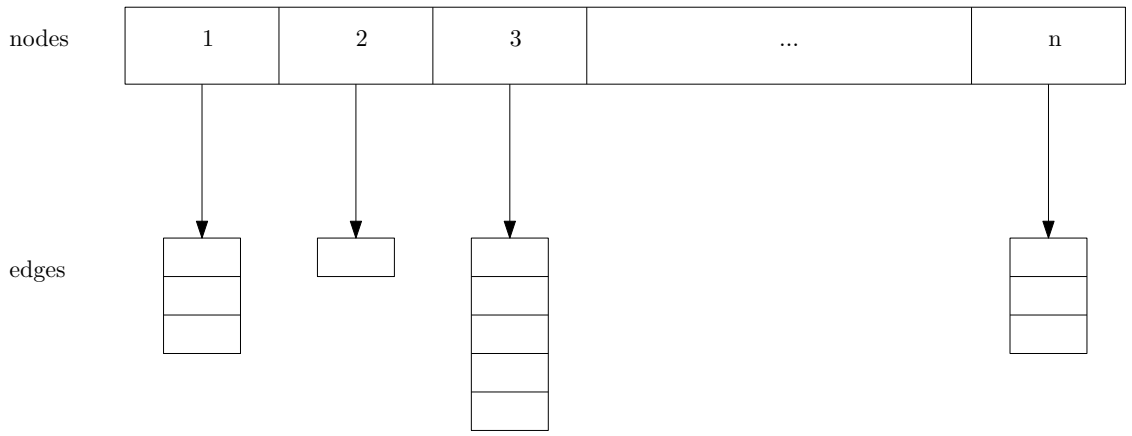

Abbildung 1: Skizze einer Adjazenzliste

Adjazenzarray (Forward Star):

Es gibt sowohl ein Array für Knoten als auch für Kanten. Das Kantenarray ist nach Quellknoten sortiert und jeder Knoten kennt den Index seiner ersten Kante.

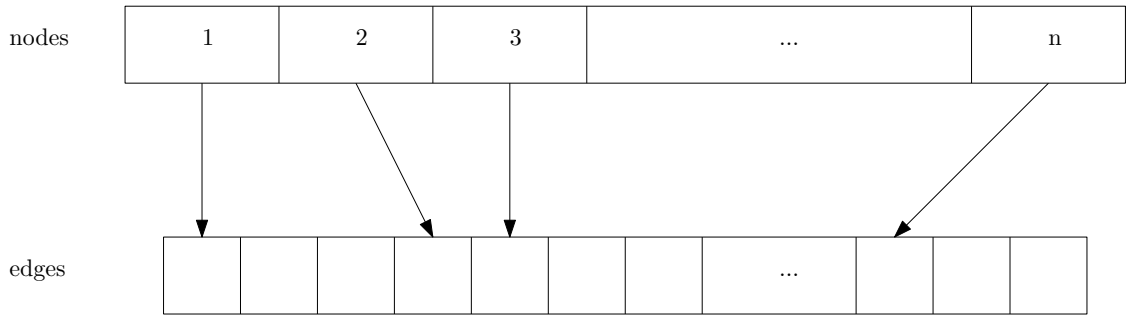

Abbildung 2: Skizze eines Adjazenzarrays