Lehrstuhl für Informatik 3
Rheinisch-Westfälische Technische Hochschule
Aachen
Univ.-Prof. Dr.-Ing. M. Nagl

Aachen, 30. 06. 2006

Nagl, Becker, Ranger, Wörzberger

Übungen zur Vorlesung „Grundgebiete der Informatik 2: Algorithmen und Programmiertechniken“

— Blatt 11 —

25. Aufgabe *Adjazenzmatrix*: *(9 Punkte)*

In dieser Aufgabe können zusätzlich noch 6 Zusatzpunkte erreicht werden.

Wenn gerichtete Graphen mit relativ vielen Kanten gespeichert werden sollen und die maximale Anzahl von Knoten eines Graphen bekannt ist, dann lassen sich als Datenstruktur gut Adjazenzmatrizen einsetzen. Für einen Graphen mit n Knoten benötigt man eine $n \times n$-Matrix, wobei der Eintrag (i, j) angibt, ob die Kante von Knoten i nach Knoten j existiert.

(a) Implementieren Sie ein zur unten angegebenen Schnittstelle passendes einfaches Datentypmodul, das einen Graphen in Adjazenzmatrixdarstellung darstellt und Prozeduren zum Initialisieren des Graphen und zum Einfügen bzw. Löschen von Kanten anbietet. Die Anzahl der Knoten kann als konstant vorausgesetzt werden. *(5 Punkte)*

(b) Schreiben Sie zwei Methoden, die einen Knoten maximalen Auslauf- bzw. Einlaufgrades finden. *(4 Punkte)*

Freiwillige Zusatzaufgabe: Erweitern Sie die Klasse so, daß zu jeder Kante eine *Länge* gespeichert wird und schreiben Sie eine Funktion zur Berechnung der Länge eines kürzesten Weges zwischen zwei Knoten.
Tip: entsprechende Verfahren finden Sie in der Literatur, z.B. in den Büchern zu Algorithmen und Datenstrukturen von Knuth, Sedgewick und Ottmann/Widmayer. *(6 Punkte)*

```
#include <limits.h>
typedef unsigned int NodeIdT;
typedef unsigned int DistanceT;

class AdjacencyGraph {
public:
  static const NodeIdT MAX_NODES = 10;        // maximale Knotenanzahl
  static const DistanceT INFINITE = UINT_MAX; // falls keine Kante da

  AdjacencyGraph();
  void makeNull();
  void insertEdge( NodeIdT from, NodeIdT to );
  void deleteEdge( NodeIdT from, NodeIdT to );
  void maxIn ( NodeIdT &node, int &degree );
  void maxOut( NodeIdT &node, int &degree );
```

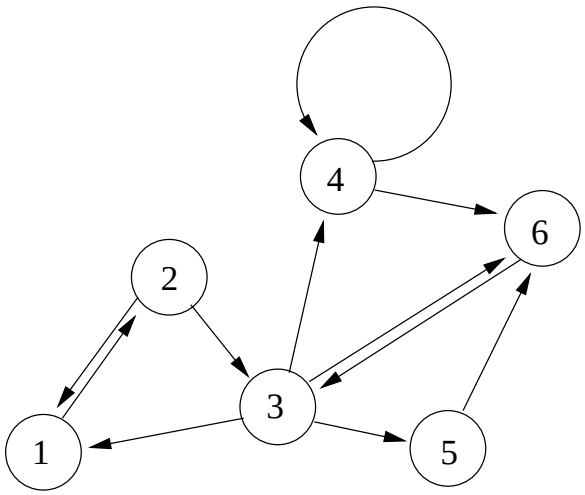
```
  // Zusatzaufgabe:
  void insertEdge( NodeIdT from, NodeIdT to, DistanceT length );
  DistanceT distance( NodeIdT from, NodeIdT to );   // INFINITE falls kein Weg
private:
  DistanceT adjacencyMatrix[MAX_NODES][MAX_NODES]; // die Adjazenzmatrix
};
```

26. Aufgabe *Graphdarstellung*: *(11 Punkte)*

(a) Wählen Sie eine knotenorientierte Realisierung von gerichteten Graphen aus der Vorlesung und erweitern Sie diese um die Information über die in jeden Knoten *einlaufenden* Kanten. Geben Sie C++-Typen an, die die von Ihnen entworfene Datenstruktur implementieren. *(3 Punkte)*

(b) Implementieren Sie auf Ihrer Datenstruktur Operationen zum Einfügen und Löschen von Knoten und Kanten. Welche der Operationen lassen sich durch die Erweiterung effizienter gestalten, welche werden ineffizenter und welche bleiben davon unbeeinflußt? *(6 Punkte)*

(c) Testen Sie Ihre Datenstruktur und die Operationen, indem Sie den unten angegebenen Graphen zunächst erzeugen, dann den Knoten `3` löschen und anschließend alle Knoten und Kanten des Graphen ausgeben. *(2 Punkte)*



Abgabe: 07. 07. 2006 vor der Vorlesung.