Implementation of Directed Multigraphs in Java

Jaanus Pöial

University of Tartu

Estonia

jaanus@cs.ut.ee

Supported by EITSA
(The Estonian Information Technology Foundation)

Motivation

- Teaching algorithms and data structures
  - beginners need some framework to experiment with graphs and graph algorithms
  - it is possible to give students more meaningful assignments than just implementation of basic operations on graphs
  - comparision of algorithmic vs OOP approach (Pascal vs Java)
- General purpose reference implementation

Basic structure

- The graph is represented by an object that contains list of vertices
- Each vertex is represented by an object that contains list of outgoing edges
- Each edge is represented by an object that contains (pointer to) its end-vertex

Implementation

- Classes
  - `Graph` (ca 30 public methods)
  - `Vertex` (16 public methods)
  - `Edge` (14 public methods)
  - `Matrix` (3 public methods)
    - `AdjMatrix` (4 public methods)
    - `DistMatrix` (4 public methods)
  - `GraphSource` (main for debugging)

Conclusion

- Each object (graph, vertex, edge) has exactly one representative, all updates (insert, delete) made to the graph are "local" (only one of the lists is changed)
- Graph is scalable and dynamic (we do not use static containers)
- This representation is universal (we can implement all operations, but not all operations are efficient, e.g. iterator over incoming edges)

Availability

`http://www.cs.ut.ee/~jaanus/Graphs/GraphSource.java`