Fachhochschule Köln
Cologne University of Applied Sciences

Prof. Dr. Heiner Klocke
Faculty of Computer Science and
Engineering Science

Algorithms - Practical Training

WS 2010/2011

Subject: Experimental analysis of path-finding and spanning tree algorithms

A. Formal remarks

The practical training teams consist of 2 or 3 students. Please register online via the link on my homepage. Changing to a different team is possible.

1. Dates

There will be 5 fixed dates:

1. Explanation and discussion of the organisation chart (8.11.2010)
2. **First milestone**. Present and discuss your test-environment for the experimental algorithms-analysis. Apply the Power Test to analyse an $O(n^2)$-algorithm, e.g. BubbleSort.
3. **Second milestone**. Explain the significance of priority queues if used by path-finding algorithms as Kruskal or Dijkstra. Present the actual state of the programs you have developed in your team.
4. **Third milestone**. Present the actual state of your programs especially the test-environment user interface for the **path-finding-** and **tree-spanning-algorithms**. Keep in mind the relevance of priority queues.
5. Final presentation of the project and release of the documentation.

2. Documentation within Algorithms-Wiki

Please put the relevant information about the progress of your project into the Algorithms-Wiki as well as the software:

http://www.software-quality.fh-koeln.de/algorithmik/index.php/Gruppen_WS1011

3. Evaluation

The results of your work, your algorithm-competence as well as the style of presentation will be evaluated. The highest possible score is 36 pts. These score points are added to the points you reach in the examination at the end of the semester.

4. Presentation and documentation

Please prepare a presentation (up to 15 minutes) as screen-slide formatted as pdf. Store the presentation-slides in the Wiki two days before the presentation-date. Use the filename **„<teamname>_MS_<number of milestone>**“.

4.1 Presentation

1. Explain the mission of the milestone
2. Present your ideas, methods, approaches and solutions.
3. Show the runtime-version of the programs. Do not show the source code.
4. Discuss your work and results
5. Define changes and improvements that you will do until the next practical date.
6. Review the documentation

4.2 Documentation

Please send the documentation as a PDF to alex.maier@fh-koeln.de and heinrich.klocke@fh-koeln.de with the subject **„<teamname>_MS_<number of mildestone>“.** Structure of the documentation:

1. **Frontpage**. A sample is shown on my homepage.
2. **Contents**
3. Detailed **problem definition**
4. Documentation of the milestones
5. **Literature**: books, papers and internet sources

B. Project definition

1. Subject

Path-Finding- and **Tree-Spanning-Algorithms** using **Priority-Queues** and analysis of their runtime.

Each team can choose between a Path-Finding and a Tree-Spanning application.

2. Learning targets:

- How can the asymptotical runtime of algorithms be diagnosed by using the power test?
- Why can priority-queues speed up algorithms?
- What kind of practical problems can be solved with graph algorithms?

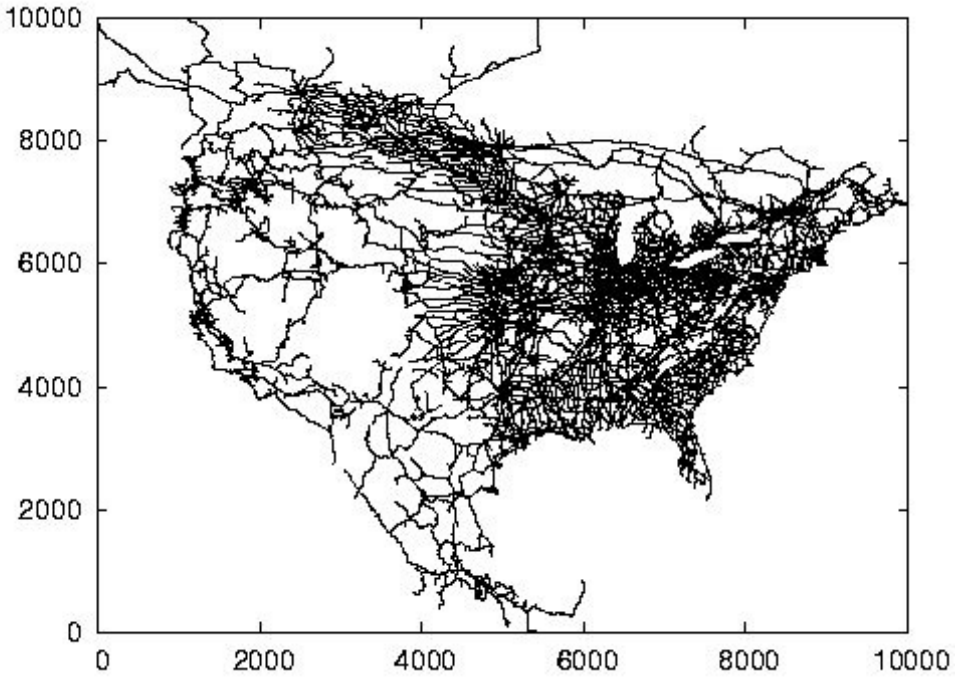
3. Experimental analysis by the power test

Develop a test-environment to measure and display the polynomial runtime of algorithms using the power test (see lecture chap 1.2.2). Try the first experiments with an $O(n^2)$-algorithms, e.g. BubbleSort. Set up a precise test plan (select and produce test data), execute the runtime-tests (measure correct) and record the results. Display the (n, t)-pairs and the (log n, log t)-pairs in a coordinate system; n = size of data, t = measured runtime for n data. For that you can use the graphical Java-library **Ptplot 5.8**. (http://ptolemy.eecs.berkeley.edu/java/ptplot/). Don't forget to compute the coefficient of correlation to check the quality of linear relation between the log-points.

Choose the path-finding project or the tree-spanning project, not both!

4. Path-Finding project

- Develop a program for computing the fastest path between two locations. Choose therefore a graph-algorithm from the lecture, which you think the best for this problem. Justify the decision. If you program in Java then you can use the Java priority queue class to support the algorithm by this data structure.
- Try to develop a graphical user interface for entering the start and end point of the search. Also show on the screen how the algorithm works and what he is doing to find the best path.
- First test your path-finding application with small acyclic graphs. After that do the test with the USA road network data from ***http://www.cs.fsu.edu/~lifeifei/SpatialDataset.htm***. Run and analyse the algorithm with this dataset. Implement appropriate displays to enable observing the algorithm at work. Show the runtime analysis results once the algorithm stops.



Pic 1: A graphical view of the USA road network generated from the normalized dataset ***http://www.cs.fsu.edu/~lifeifei/SpatialDataset.htm***

5. Tree-Spanning project

- Write a program to compute a network with minimal costs, called spanning tree with minimal costs. Select an algorithm from the lecture that you think is the best to solve this problem. Use a priority queue to manage the data needed by the tree-spanning algorithm.
- Display the minimal spanning tree in an appropriate graphical style. Furthermore show on the screen how the algorithm operates the path-finding problem.
- Give relevant information about the priority queue e.g. the number of nodes, the minimum node etc.
- First test your path-finding application with small acyclic graphs. After that do the test with the USA road network data from ***http://www.cs.fsu.edu/~lifeifei/SpatialDataset.htm***. Run and analyse the algorithm with this dataset. Implement appropriate displays to enable observing the algorithm at work. Show the runtime analysis results once the algorithm stops.

Literature

1. **Thomas Cormen**, Charles E. Leiserson, Ronald E. Rivest, Clifford Stein. Introduction to Algorithms. The MIT Press. 2009 3rd edition. ISBN-13: 978-0262533058 (Hardcover empfehlenswert, da Buch sehr dick und schwer).
2. **Sara Baase**, Allen Van Gelder. Computer Algorithms. 3rd Edition Addison Wesley 2000, ISBN: 978-0201612448
3. **Michael T. Goodrich**, Roberto Tamassia. Algorithm Design. Wiley 2002. ISBN 0471383651