Algorithm Engineering (2013, Q3)
Project 2 – Choose a Topic

In this project you should do a throughout experimentation and evaluation of one of the algorithmic problems listed below. Through experimental evaluation of the performance of your solution, try to optimize your structure. Your report should document the steps of improvements, comparing the steps against each other.

The project should be done in groups of 2-3 persons. The work should be documented in a report. This report will be part of the final grade. Part I of the report should be the content of the report from Project 1, with possible revisions. Part II of the report should cover the second project.

Deadline: Wednesday April 3, 2013 (the latest).
Project alternatives

For each of the possible project topics listed below is given a reference. It is not expected that the content of these papers necessarily is implemented – the references are primarily provided as a starting point for finding relevant literature.

1. **QuickSort in Java**
   Try to improve Java’s build in QuickSort in Java7. Since the Java runtime system is influencing the runtime, the outcome of experiments might be harder to explain. The following is a recent paper on the problem:
   [http://knowledgecenter.siam.org/0238-000024/0238-000024/1]

2. **Matrix Multiplication**
   Develop an efficient matrix multiplication algorithm, e.g. using ideas from:
   [http://dx.doi.org/10.1145/2071379.2071383]

3. **RadixSort**
   Consider sorting 32-bit integers using RadixSort. RadixSort is theoretically fast, but cache performance is an important issue in practice. A new variant of RadixSort is presented in:
   [http://dx.doi.org/10.1007/978-3-642-23397-5_16]

4. **Rank-Select Data Structures**
   A rank-select data structure supports the following two operations for a static vector of length n containing 0-1 values: Rank(i) returns the number of 1s up to position i in the vector, and Select(r) returns the position of the r’th 1 in the vector. Such data structures are fundamental to more complex data structures. The goal is to develop space efficient data structures for this problem, i.e. O(n) bits (not words), with constant query time. A previous experimental study is described in:
   [http://personales.dcc.uchile.cl/~gnavarro/ps/wea05.pdf]

5. **Priority Queues (Insert & DeleteMin)**
   Binary heaps are known to have poor cache performance. The following paper describes an alternative priority queue approach reducing the number of cache faults:
   [http://doi.acm.org/10.1145/351827.384249]

6. **Something completely different?**
   If you have a good idea for a completely different project you can also do that – but you need to get the project idea approved before starting.