GSB

Algorithm Engineering (2014, Q3)
Project 2 – Choose Two Topics

In this project you should do a throughout experimentation and evaluation of **two** 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. Parts II and III of the report should cover the topics of the second project.

Deadline: Tuesday March 18, 2014.

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:
   > Sebastian Wild, Markus Nebel, Raphael Reitzig, Ulrich Laube. *Engineering Java 7's Dual Pivot Quicksort Using MaLiJAn*. Annual Meeting on Algorithm Engineering and Experiments (ALENEX). Pages 55-69, 2013.
   > [http://knowledgecenter.siam.org/0238-000024/0238-000024/1]
   > See also: Multi-Pivot Quicksort: Theory and Experiments, ALENEX 2014 [doi.org/10.1137/1.9781611973198.6].

2. **Matrix Multiplication**
   Develop an efficient matrix multiplication algorithm, e.g. using ideas from the below paper. The project could also cover the role of efficient matrix transposition.
   > Matteo Frigo, Charles E. Leiserson, Harald Prokop, Sridhar Ramachandran. *Cache-Oblivious Algorithms*. ACM Transactions on Algorithms, 8(1), Article No. 4, 2012.
   > [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:
   > Jan Wassenberg, Peter Sanders. *Engineering a Multi-core Radix Sort*. Euro-Par. Lecture Notes in Computer Science, volume 6853, 160-169, Springer, 2011.
   > [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:

   Rodrigo González, Szymon Grabowski, Veli Mäkinen, and Gonzalo Navarro. *Practical Implementation of Rank and Select Queries*. *Poster Proceedings Volume of WEA'05*, pages 27-38 (poster). [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:

   Peter Sanders. *Fast Priority Queues for Cached Memory*. ACM Journal of Experimental Algorithmics 5:7, 2000. [http://doi.acm.org/10.1145/351827.384249]

6. **Are Fenwick Trees Sensitive to Cache Associativity?**

   Make a throughout experimental evaluation of Fenwick Trees. Are some parts of the address space more sensitive to cache faults due to cache associativity? Is the same true for binary heaps?

7. **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.