Institut for Matematik og Datalogi
Syddansk Universitet

February 19, 2007
JFB

DM508 – Algorithms and Complexity – F07
Lecture 6

Lecture, February 13

We finished with Fibonacci heaps and began on string matching from chapter 32, covering the first two sections.

Lecture, February 15

We will finish string matching from chapter 32 and begin on NP-completeness, from chapter 34 in the textbook and the section by Papadimitriou and Steiglitz from the course notes.

Lecture, February 20

We will continue with NP-Completeness, covering Cook's Theorem and beginning on some reductions.

Problems to be discussed on February 21

1. 32.4-1, 32.4-3, 32.4-4, 32.4-5.
2. 34.1-3, 34.1-5, 34.2-3.
3. Suppose that there is a language L for which there is an algorithm that accepts any string $x \in L$ in polynomial time and rejects any $x \notin L$, but this algorithm runs in super-polynomial (more than polynomial) time if $x \notin L$. Argue that L can be decided in polynomial time.
4. Define an algorithm to show that SATISFIABILITY is in NP.

Assignment due Tuesday, February 27, 10:15

Note that this is part of your exam project, so it must be approved in order for you to take the exam in March. This means that the work must be your group's own work. No part of your work may be copied from another source, and you may not work with others not in your group. You may work in groups of two or three. You may write your solutions in English or Danish, but write very neatly if you do it by hand.

1. A dynamic hash table could be implemented as follows: The hash table initially has size 1. When an insert occurs, if the table is not full, the item is hashed into the table. If the table is full, a new table which is twice as large is created, and all of the current elements are re-hashed into the new table.

   Assume that hashing one item takes constant time. Let T_i be the data structure after operation number i. Let n_i be the number of items in the table after operation number i. (Some operations might be searches, rather than inserts, so it is possible that $n_i \neq i$.) Let s_i be the size of the table after operation number i (always a power of 2). Define a potential function $\Phi(T_i) = 2n_i - s_i$. Note that for all tables of size at least two, the table is always at least one item more than half full. What is the amortized cost of an insertion? Consider both the case where the table is not full just before the insertion, and the case where it is full. (Do the amortized analysis.)

   Suppose n items are inserted into the structure. Give a good worst case bound on the total cost of doing all of these n insertions.

2. A main ingredient in the analysis of Fibonacci heaps is that the degree of a node must small relative to the size of the subtree of which it is a root. Unlike many other efficient data structures, there is no logarithmic bound on the depth of a tree produced by operations on n items. Show this by describing a sequence of Fibonacci heap operations on n items that produces a heap-ordered tree of depth $\Omega(n)$ in a Fibonacci heap.

3. What is the prefix function computed by the KMP algorithm for the string $P = abbabcabba$.