Institut for Matematik og Datalogi February 27, 2012
Syddansk Universitet JFB

DM508 – Algorithms and Complexity – 2012
Lecture 7

Lecture, February 20

We finished with NP-Completeness, showing that SUBSET SUM is NP-Complete and considering options when faced with an HP-hard problem. We began on amortized analysis from Chapter 17 of the textbook.

Lecture, February 24

We will finish amortized analysis from chapter 17 and begin on Fibonacci heaps from chapter 19 in the textbook.

Lecture, February 27

We will finish Fibonacci heaps and begin on string matching from chapter 32.

Problems to be discussed on March 7

Do problems:

1. 19.2-1
2. 19.3-1
3. 19.4.1
4. 19-1
5. 19-3a
6. 19-2.

Assignment due Monday, March 5, 8:15

Note that this is part of your exam project, so it must be approved in order for you to take the exam in March, and you may not work with or get help from others not in your group (though you may talk with Magnus Find or myself). 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. Submit the assignment via Blackboard's "SDU Assignment" as one PDF file. Note that you should not turn in a paper copy. Turn in one assignment per group.

The assignments will be graded by 9:00 on March 9. You may pick them up in my office at that time. If you do not get this one approved, but have not used any of your retries, you may have until 8:15 Monday, March 12, to turn in a retry.

1. Consider creating a table with the hash values of all strings of length at most n over an alphabet, Σ, with at most 2^{16} different symbols. Suppose your hash function h works as follows: for $r \geq 2$, $h(x_1x_2...x_r) = f(x_r, h(x_1x_2...x_{r-1}))$, and $h(x_1) = f(x_1, 0)$, where $x_1, x_2, ..., x_r \in \Sigma$. Suppose f is a function which takes two inputs, one of length 16 bits (a symbol from the alphabet) and the second of length 128 bits and produces an output of length 128 bits.

   (a) Design an algorithm for computing hash values for all strings over Σ of length at most n which only $\sum_{i=1}^{n} |\Sigma|^i$ calls to the function f in the worst case.

   (b) Use standard analysis techniques to show that your algorithm uses only $\sum_{i=1}^{n} |\Sigma|^i$ calls to f.

   (c) Consider using the accounting method to analyze your algorithm. When your algorithm hashes a string of length one, charge $\sum_{j=0}^{n-1} |\Sigma|^j$ as the amortized cost. Then, charge amortized cost zero for the other calls to f. Argue that this gives the same result. (Note: You need to explain how you move credits around to make this work. When you hash a string of length r, let its credit initially be $\sum_{j=0}^{n-r} |\Sigma|^j$.)

2. UNICODE is a coding scheme for representing symbols, such as digits, letters, punctuation marks, etc. There are variations, but assume that each character we use here is represented using 16 bits. Consider a counter which, instead of being binary, is kept in UNICODE form, base 16 (use the digits 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F), and assume that m digits are stored. Thus, the number one hundred forty-one (8D in base 16, hexadecimal) is stored in an array, C, of length m, where $C[0] = 68$ (0000000001000100 in binary, the UNICODE code for D), $C[1] = 56$ (0000000000111000), and $C[2] = C[3] = ... = C[m-1] = 48$ (UNICODE for 0). Give an algorithm for incrementing this type of counter and analyze it using the potential function method. You may assume that you can increment a UNICODE value, check a UNICODE value, or store a UNICODE value in one time unit.