Institut for Matematik og Datalogi
Syddansk Universitet

October 10, 2011
JFB

Introduction to Computer Science
E11 – Lecture 9

Lecture, September 27, 8:15–10, U26

Kim Skak Larsen lectured on chapter 9 in the textbook. Section 9.5 will be covered later.

Lecture, October 4, 8:15–10, U26

We will finish chapter 5 in the textbook.

Lecture, October 6, 12:15–14, U26

Rolf Fagerberg will lecture on section 9.5 in the textbook (merging and hashing).

Discussion section (and study group before that): October 11, U103 (not terminal room)

For the problems below Group 1 should prepare to present Problem 6 on page 227, Problem 3 on page 236, Problem 23 on page 247, Problems 46 and 49 on page 250. Group 2 should prepare to present Problem 7 on page 227, Problem 4 on page 236, problem 55 on page 250, and the second part of the square root algorithm problem. Group 3 should prepare to present Problem 5 on page 227, Problem 2 on page 236, problem 49 on page 250, and the first part of the square root algorithm problem.

1. Page 227: Problems 5, 6, 7.

2. Page 236: Problems 2, 3, 4.

3. Design an algorithm for finding the square root of a positive integer, rounded down to the nearest integer. Thus, given input N, a positive integer, you should find a positive integer m such that $m^2 \leq N$, but $(m+1)^2 > N$. Use the binary search idea.

   (a) Express the algorithm in pseudocode.

   (b) Find a fundamental operation and use big theta notation to express how long your algorithm takes. Express this as a function of the positive integer N which is input.

4. Page 247: Problem 23.

5. Page 250: Problems 55, 46, 48, 49. For problem 55, you can discuss preconditions, postconditions, loop invariants, and posssible improvements.

Assignment due 12:30, October 14

Late assignments will not be accepted. Working together is not allowed. (You may write this either in English or Danish.) Explain your answers thoroughly.

Write your solution to this assignment in LaTeX. Submit a PDF file through the Blackboard system and turn in an identical paper copy, to me in my office. I will be in my office from 12:10 through 12:30. Write your full name and your section number clearly on the first page of your assignment.

1. Page 247: Problem 17. Express the algorithm in pseudocode. Your algorithm should be general enough to solve general problems like this, with arbitrary numbers of bits in the numbers (though the sum will always have one more bit than the other two numbers, which you may assume have the same number of bits). Your algorithm should also report correctly that there is no solution when there is none. In addition, give an example where there is no solution.

2. Suppose that company, OurFavoriteCompany, has a database listing N_i customers at the beginning of day $i \geq 1$, and this database is stored as a list, `Cust`. Suppose that initially the database is sorted according to customer number, so the ith record in `Cust` has a lower customer

number than the $i+1$st record in `Cust`. Suppose that OurFavoriteCompany adds m_i new customers on day i, starting with day 1. Assume that there is always enough empty space at the end of the list to hold records for these new customers.

Suppose when new customers come, records for them are initially added to the end of the list, and during the night the list is sorted. Suppose $m_i < \log_2(N_1)$ for all $i \geq 1$, and suppose that on day i you have the value N_i available.

Do all of one of the following two problem groups:

- Write pseudocode for an algorithm to search for a customer in the database during day i (i.e. before the list is sorted at night). Do not use Sequential Search for the entire list; do something more efficient. Note that the new customers are not added all at once, so you need to have a variable, c saying how many new customers from that day are currently in the list. How many comparisons does your algorithm do in the worst case? (Use $\Theta()$ notation.) Why might it be an advantage to always maintain the database in sorted order?
- Write pseudocode for an algorithm to sort the list at night. Your algorithm should do $\Theta(N_{i+1} \cdot m_i)$ comparisons. Why does it take less than $(N_{i+1}^2 - N_{i+1})/2$ in the worst case. Explain why it takes this amount of time. (It is also possible to do it in $\Theta(N_i + m_i^2)$ comparisons or less, so it would be nice if you commented on this, but it's not necessary.) Why might it be an advantage to always maintain the database in sorted order?