Institut for Matematik og Datalogi
Syddansk Universitet

September 27, 2005
JFB

Introduction to Computer Science
E05 – Lecture 8

Lecture, September 26

Rolf Fagerberg lectured on file structures from section 9.5.

Lecture, September 29

Kim Skak Larsen will lecture on databases from chapter 9.

Lecture, October 3

We will talk about security from sections 3.5 and 4.5. Then, we will begin on encryption from section 11.6 and discuss exponentiation, an efficiency concern with RSA and many other public key cryptosystems. There are some notes on cryptography, from PGP, using a link on the course's homepage.

Lecture, November 15

We will begin on the theory of computation from chapter 11.

Supplementary notes on RSA

The textbook leaves out many important details regarding the implementation of RSA. For example, it gives the incorrect impression that in computing $m^e \pmod{n}$ that one would first compute m^e and then reduce modulo n. This is not what occurs in practice since it is infeasible for the large numbers used. The intermediate result would have about $e \log(m)$ bits, which would usually be more than 2^{500} bits (either m^e or c^d would be extremely long)! Thus, one computes this using intermediate computations and reducing modulo n after each step. This works because of the following:

Lemma. For all nonnegative integers a, b and any integer $n > 1$,
$a \cdot b \pmod{n} = (a \pmod{n})(b \pmod{n}) \pmod{n}$.

Note that this can be proven using the fact that $a = x \pmod{n}$ if and only if $0 \leq a < n$ and there is an integer k such that $a = x + k \cdot n$.

The powers can be computed efficiently using the following algorithm:

```
function power(a,exp,n)

# Compute a^exp (mod n) for nonnegative exp

   if exp = 0 then return(1)
   else if (exp is odd) then
           return((a*power(a,exp-1,n)) mod n)
        else
           c <- (power(a,exp/2,n))
           return((c * c) mod n)
```

The values e and d are *multiplicative inverses* of each other modulo $(p-1)(q-1)$ (i.e. $e \cdot d \pmod{(p-1)(q-1)} = 1$). They can be computed by using the Extended Euclidean Algorithm, which computes greatest common divisors.

Def. $\gcd(a, b)$ = greatest common divisor of a and b = largest $d \in \mathbb{Z}$ (the integers) such that $d|a$ and $d|b$

If $\gcd(a, b) = 1$, then a and b are *relatively prime.*

Thm. $a, b \in \mathbb{N}$ (nonnegative integers). There exist $s, t \in \mathbb{Z}$ such that $sa + tb = \gcd(a, b)$.

Claim: The integers $d = gcd(a, b)$, s and t can be found efficiently, using the Extended Euclidean Algorithm.

For RSA, the value e is chosen so that $\gcd(e, (p-1)(q-1)) = 1$. To find d, we also need a value k such that $e \cdot d = 1 + k(p-1)(q-1)$. Thus, we can compute d by solving for s in the equation $se + t(p-1)(q-1) = 1$. This can be done using the Extended Euclidean algorithm since $gcd(e, (p-1)(q-1)) = 1$.

Discussion section: week 40

Discuss the following problems from the textbook in groups of three or four.

1. Sequential files: Question 3 on page 390 and Problem 51 on page 399.
2. Hashing: Questions 6 and 7 on page 391.
3. Merging: Problem 55 on page 399.
4. Assume sets of numbers are represented by sequential files sorted on element value. For example, the set $\{4, 7, 13, 9, 2\}$ is represented by a sequential file containing $2, 4, 7, 9, 13$.

   Describe algorithms for constructing $A \cup B$ and $A - B$ from A and B (finding $A \cup B$ is Problem 48 on page 399; $A \cap B$ is given as an assignment below).
5. Problems 7, 8, 12, 22, and 35 on pages 395–399.
6. Assume the database relations A and B each are stored as sequential files of tuples, ordered according to attribute X (which is an attribute of both relations).

   Sketch (details not necessary) an algorithm based on merging for executing the statement

   $$C \leftarrow \text{JOIN } A \text{ and } B \text{ where } A.X = B.X$$
7. Assume again that the database relations A and B each are stored as sequential files, but now no longer ordered on the X attribute.

   Describe an algorithm based on nested loops for executing the statement

   $$C \leftarrow \text{JOIN } A \text{ and } B \text{ where } A.X = B.X$$

   How many comparisons between tuples are performed (as a function of $|A|$ and $|B|$, the numbers of tuples in each relations)?

   Describe how to speed up the algorithm by first using hashing on each relation.
8. Explain how a poorly chosen hash function can result in a hash storage system becoming little more than a sequential file.

9. Discuss questions 3, 6, 7, and 8 on page 400.

10. Question 2 on page 390.

Assignment due 8:15, October 10

Late assignments will not be accepted. Working together is not allowed. (You may write this either in English or Danish, but write clearly if you do it by hand.) Show your work where it is relevant.

1. From problem 4 from the discussion section (above), give an algorithm for constructing $A \cap B$. Do this formally as in Figure 9.14.

2. If a hash file is partitioned into 12 buckets, what is the probability of at least two of three arbitrary records hashing to the same sections? Assume that the hash function is such that a randomly chosen record is equally likely to hash to any of the sections. How many records must be stored in the file until it is more likely for collisions to occur than not? Assume again that there are 12 bins.

3. Problem 10 on pages 396.