Institut for Matematik og Datalogi
Syddansk Universitet

October 8, 2008
JFB

Introduction to Computer Science E08 – Week 8

Lecture, October 6

We covered sections 12.1, 12.2 and 12.4 in chapter 12.

Lecture, November 3

We will finish chapter 12.

Lecture, November 5

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

Lecture, November 10

We will cover security from the last sections on operating systems and networks and introduce cryptography. There are notes on cryptology below. In addition, there are two links for information on cryptology and PGP on the course's homepage.

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 $gcd(e, (p-1)(q-1)) = 1$.

Note that Jacob Allerelli's notes on modular arithmetic are available through the course home page.

Discussion section: week 46

The first exercise involves programming. It should be done before you come to discussion section. (You may use Java or Maple or some other language.)

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

1. Hashing: Write a program to compute the probability of at least one collision when hashing is used with m records and n buckets. (See the calculation on page 468 of your textbook and generalize it.) Assume that the the hash function spreads data out essentially randomly. Use your program to answer problem 7 on page 469 and problem 57 on page 478. How did you use your program?

2. Hashing: Question 6 on page 469.

3. Sequential files: Question 3 on page 469 and Problem 54 on page 478.

4. Merging: Question 1 on page 469.

5. 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 $(A \cup B) \cup C$ from A, B and C Note that $(A \cup B) \cup C$ can be done by first computing $A \cup B$ and computing the union of this with C. Instead of giving this solution, process the three files simultaneously, as you do with two files.

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 9 on page 479–480.

10. Question 2 on page 469 (it has been mentioned in a previous lecture and is in the sorting simulator you used).

Assignment due 12:15, November 24

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. If you use a program to compute something, please include the code or a good description of what it does.

1. Assume sets of numbers are represented by arrays sorted on element value. For example, the set $\{4, 7, 13, 9, 2\}$ is represented by an array of length 5 containing $[2, 4, 7, 9, 13]$. Write a program in Java or Maple for constructing $A \cap (B \cup C)$. Use an algorithm similar to that in Figure 9.15. As in problem 5 above, process the three arrays simultaneously (you should not first calculate $B \cup C$ and then intersect with A). Test your algorithm. Turn in both the program code (commented) and your test results.

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.