Institut for Matematik og Datalogi
Syddansk Universitet

September 19, 2006
JFB

Introduction to Computer Science
E06 – Lecture 6

Lecture, September 19

During the first hour, I lectured on algorithms from chapter 5, covering parts of sections 5.4 and 5.6, using sequential search as the example. During the second hour, Peter Kornerup lectured on networks from chapter 4.

Lecture, September 22

We will finish chapter 5 on algorithms and begin on sections 3.5 and 4.5 on security. It is possible that we will also begin on section 11.6 on cryptography.

Lecture, September 26

Rolf Fagerberg will lecture about file structures from section 9.5.

Discussion section: week 39, starting in the Terminal Room

Try to do the last problem on your own before coming to discussion section. If you are not successful, there might be time to try to combine your ideas in your group.

Discussion in groups (only two, or possibly three, people per group, since you will sit at a computer):

The goal of this lab is to help you to gain some understanding of the fact that most problems have more than one algorithmic solution and that these solutions can differ greatly as to how practical they are. You will experiment with three different sorting algorithms and compare them. Review sections

5.4, 5.5, and 5.6 in the textbook (including problem 6 on page 239) before coming to the lab.

To start up the programs, you should find the home page for this course:
http://www.imada.sdu.dk/Courses/DM501/index.html.
Near the bottom of that page, there are links to a sorting simulator. Click on the first of those two links. Note that the program sorts bars of different lengths, rather than numbers. It is easy to think of the bars as numbers, and it is easiest to see what is happening with the bars. Following the second link instead of the first, you can find program code for these sorting algorithms.

1. Start using the xSortLab Applet from the first link. Choose the algorithm **Insertion Sort** instead of **Bubble Sort**. Click on **Go** and watch the algorithm execute. *What do the pink and green rectangles mean? How many comparisons are done and how many copies? What is a copy?*

   Now use the Sorting Demo from the second link. Choose **Insertion Sort**, a **length** of 10, a **Special-case array**, and **Random**. To start it, click on **Start Demo** and then on **OK** in the applet. (Note that you can make it run faster by decreasing the **Delay in msec**.) *How many comparisons are done and how many exchanges? What is an exchange?*

   After it has stopped, try starting it again on sorted values. Killing the window which is running the applet will allow you to start again, and you simply choose **Sorted** instead of **Random**. *Now how many comparisons are done and how many exchanges?* Try again with reverse sorted data. *Now how many comparisons are done and how many exchanges? When does the algorithm do best and when worst. Why?*

2. Go back to using the xSortLab Applet and choose **Selection Sort**. (See problem 6 on page 239 of your textbook. The algorithm here finds the largest entry remaining, instead of the smallest.) Try running Selection Sort (note you get random data). *What does the algorithm do? How does it work?* Write down the current values for **Comparisons** and **Copies**. If the number of bars is n, the number of comparisons should be
$$\sum_{i=2}^{n}(i-1) = \frac{1}{2}n^2 - \frac{1}{2}n.$$

Why is this the number of comparisons? How many comparisons should there theoretically be in this case, where $n = 16$? How does this compare with practice? What is the maximum number of copies there could be?

3. In the following, when the program is just running for awhile, try going on to something else, so you don't just waste time. In the xSortLab Applet, change the **Visual Sort** to **Timed Sort**. Run both Insertion and Selection Sort with 1000, 10,000, and 100,000 entries. Write down in each case the **# of Comparisons**, **# of Copies**, and **Approximate Compute Time**. *How do these compare with the predicted values? What values would you expect if the number of bars was 1,000,000?*

4. Switch back to **Visual Sort** and Change the algorithm to **QuickSort**. Try running QuickSort. *What does the algorithm do? How does it work?* It might help to also try the other simulator which has program code. If the bar chosen to partition on (the pivot element) ends up in the middle of the current range every time, then you are left with two problems (the left and right) half as big as the original, and it is not too hard to show that the number of comparisons is $\Theta(n \log n)$. This does not happen every time, and a complete average case analysis is beyond the scope of this course. It is not so hard to show that if the pivot element is always within the middle 15/16 of the elements, then you get $\Theta(n \log n)$ time anyway. Intuitively, it seems likely that this will happen most of the time, so that is why one generally gets this behavior.

   Switch back to **Timed Sort**. Try running it three times with 1000 random bars, writing down the **# of Comparisons**, **# of Copies**, and **Approximate Compute Time**. *Why didn't you get the same answer every time?*

   Try repeating this with 10,000 bars and 100,000 bars. *Write down in each case the* **# of Comparisons**, **# of Copies**, *and* **Approximate Compute Time**. *What do you conclude about Quick Sort's running time? Is $\Theta(n \log n)$ believable?* (Note that Θ notation is discussed on pages 252–253 of the textbook. Informally, it means in this case that for large n, the running time is close to some constant times $n \log n$.) *Try to make an estimate as to how long Quick Sort would take with 1,000,000 bars. How does Quick Sort compare with Selection Sort?*

5. Using the Sorting Demo from the second link, try running Quicksort using random data, sorted data, and reverse sorted data. *How does this compare with Insertion Sort? Explain your results.*

6. (The next problems are independent of the previous ones.) Do problem 6 on pages 258 of the textbook. What precondition and loop invariant should hold?

7. Do problem 50 on page 263.

8. Do all the questions on page 196 of the textbook.

9. Consider the following problem (mentioned in lecture): There are three politicians, A, B, and C. You know that one of them always tells the truth, one of them always lies, and one of them sometimes tells the truth and sometimes lies. You are allowed to ask these three politicians any three true/false questions you like, and you may choose which politician is asked which question. How would you determine how to order the politicians by how often they tell the truth? This problem is quite difficult. Try your problem solving abilities, but do not be disappointed if you fail.

Assignment due 8:15, October 3

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.) Remember to show your work. When the question is "how many?" and specific values are given, give integer answers, not just formulas. Do all four problems.

1. How many comparisons would Insertion Sort do in the worst case if there were 450 numbers to be sorted?

2. Suppose there were only 7 numbers to be sorted. Create a best case and a worst case list in terms of the number of comparisons needed by Insertion Sort. How many comparisons are needed in each case?

3. What is the maximum number of entries that must be interrogated when applying binary search to a list of 450 entries? What about a list of 4,500,000 entries?

4. Do problem 53 on page 264. Show either that the program segment works correctly on all cases or give some case where it does not work correctly.