Institut for Matematik og Datalogi — September 26, 2011
Syddansk Universitet — JFB

Introduction to Computer Science E11 – Lecture 8

Lecture, September 22, 12:15–14, U26

We will covered section 5.3 and up through sequential search in section 5.4 of chapter 5 in the textbook. For sequential search we considered both an analysis showing $\Theta(n)$ comparisons and a proof of correctness, with a proof by induction showing that the loop invariant held.

Lecture, September 27, 8:15–10, U26

Kim Skak Larsen will lecture on chapter 9 in the textbook.

Lecture, October 4, 8:15–10, U26

We will cover most of chapter 5 in the textbook.

Discussion section: October 5, Terminal Room

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 different sorting algorithms and compare them. Review sections 5.4, 5.5, and 5.6 in the textbook (including problem 6 on page 250) before coming to the lab.

To start up the programs, you should find the home page for this course: http://www.imada.sdu.dk/∼joan/intro/index.html.

In the “Miscellaneous” section on that page, the fifth item is a sorting simulator, written by Jacob Aae Mikkelsen. Download the file to your own directory. Then start it up by typing

```
java -jar sorting.jar &
```

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.

1. Begin with the algorithm **Insertion Sort**. Click on **Start sorting** and watch the algorithm execute. *What do the red and green rectangles mean? How many comparisons are done and how many copies? What is a copy?*

2. Repeat the experiment above. *Do you get the same results?* Try with **Decreasingly selected elements** and **Increasingly selected elements**. *What are the minimum and maximum number of comparisons possible with 8 elements? With* n *elements?* Note that after you understand what is happening, you can increase the speed, so that it takes less time to run.

3. Repeat both of the above two steps with Selection sort, Quick sort, Merge sort, and Randomized Quick sort. *Which appears to be fastest?*

4. *How do Selection sort, Quick sort, Merge sort, and Randomized Quick sort work?*

5. For Selection sort,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?*

6. Now change from **Visualization of algorithms** to *Timetaking of the algorithms* and use the **Settings** menu to choose other array sizes (numbers of elements to sort). Try all the algorithms with 100, 1000, 10,000, 100,000, and 500,000 (for the slower algorithms, you might not want

to try them three times for 500,000) elements and random data. For each algorithm, run it three times and compute the average **Number of comparisons**. *What does the "E" mean in these numbers? How do these compare with the predicted values (for Insertion sort and Selection sort)? What values would you expect if the number of bars was 1,000,000?* Run the program and save the results for the values 100, 200, 300, 400, 500 for Insertion sort and Quick sort, for your next lab.

7. Discuss which algorithm you would use in which situations (number of elements, almost sorted vs. random data, for example).