1A Lent Algorithms

Supervision 1: Complexity and Sorting Algorithms

Hayk Saribekyan

February 12, 2018

1 Introduction

In this supervision we will discuss Chapter 2 from the lecture notes. That includes the notion of computational complexity and its analysis and sorting algorithms.

In real life you will almost never need to implement a sorting algorithm from scratch as all modern programming languages have very efficient implementations. However, the knowing how each algorithm works greatly helps in writing efficient programs because not all sorting algorithms perform equally well on all data sets. Additionally, most sorting algorithms wonderfully demonstrate few algorithm design options and algorithm analysis.

2 Supervision

For this supervision, work on the problems up until 3.13 from the examples sheet for lecture 1-10. Previous years' lecture notes have exercises within the text, which may help you understand the material better in the beginning.

3 Bonus

- Problem 3.2: make a guess and then implement all three algorithms and compare their runtime. Remember that the input should be random. Any surprises?
- Problem 3.10i: can you show that you cannot do better than $n-1$ comparisons?
- Problem 3.10i asks you to find minimum of a given sequence. That can be done using $n-1$ comparisons. If you want to find both the min and max of a sequence, you can do it in $2(n-1)$ comparisons using the same algorithm. Can you find the min and the max of a sequence using $\approx 3n/2$ comparisons?

3.1 From online judges

Below is a list of problems you may want to try on online judges. Remember, the point of these is learning algorithms and not coding. Coding is just for testing your ideas. If you have convinced yourself that you know and can implement the algorithm, but you are having trouble getting started with coding, move on. Later on (e.g. during the break) when you have better coding skills or more time, you can come back and try them.

- Bubble sort: `http://codeforces.com/contest/53/problem/D`

  This is a very good starting point for online judges. See if you can implement bubble sort and get Pass on CodeForces.

- Modified version of stable sort: `http://www.spoj.com/problems/ADAUSORT/`.

  Problem 3-11 asks you to implement quick sort. How can you change the input data (*not the algorithm*) to solve this problem?

- Another sorting problem: `http://www.spoj.com/problems/AMR10G/`.

  Since we are learning sorting algorithms, do not use standard sorting functions provided by the programming language you are using. Implement yourself.

  Notice, in this problem $N \leq 20000$ and there are $T \leq 30$ tests. If you use an $O(N^2)$ algorithm then your implementation will need approximately $T \cdot N^2 = 1.2 \cdot 10^{10}$ steps, which is too many. So you need to implement an $O(N \log N)$ sorting algorithm.

  Notice that each problem has a Time Limit - the duration your implementation is allowed to run for. Generally, think that $0.5 \cdot 10^9$ *steps take 1 second. So the above bubble-sort algorithm would take 24 sections, many times more than the time limit.*