Homework 1

Due: Thursday, January 23, before lecture via Gradescope.

Suggested reading: Chapter 0.3 and Chapter 2 of the book. Section 2.2 contains the Master Theorem covered in class.

Practice problems (don't turn in)

Problem (Big-O order, part (a) and (b) are independent)

(a) For the following list of functions, cluster them into groups of functions of the same order (i.e., f and g are in the same group if and only if $f = O(g)$ AND $g = O(f)$), and then rank the groups in increasing order. You do not have to justify your answer.

- $\log_2(n)$
- $\log(n^2)$
- $\sqrt{n}$
- n
- $2n + 5$
- $n\log(n) + 2019$
- $n^{2.5}$
- 2^n
- $n\log^2(n)$

(b) *The geometric series of ratio* $a \neq 1$ is the sum

$$S(n) = 1 + a + a^2 + \cdots + a^n.$$

Prove that

$$S(n) = \frac{a^{n+1} - 1}{a - 1}.$$

Deduce that $S(n) = O(a^n)$ for $a > 1$ and $S(n) = O(1)$ for $a < 1$. (*Hint: you may prove the equality by induction or just directly by checking that both sides are equal after multiplication by* $a - 1$.)

Problem (Problems from the book) [DPV], Chapter 2, exercises:
2.4 (Choosing between three algorithms).
2.5 (Solving recurrences).
2.19 (k-way merge, we discussed this problem in class).

Problem (Order Statistics in a union of two lists) Describe an algorithm that takes as input two sorted lists of length n and m and an integer k and outputs the k^{th} smallest element in their union. You can assume both lists contain integers and all entries are different.

See next page for those problems you need to submit.

INSTRUCTIONS.

- Your solution must be in plain English. On every problem, you have to explain your design and why it is correct in your own words, and you have to analyze and justify the running time.
- Unless otherwise stated, fastest (and correct) algorithms will receive more credit. If we ask for a specific running time, a correct solution achieving it will worth full credit, even if a faster solution exists.
- **Do not use pseudocode**. Your answer will receive a zero, even if the pseudocode is correct.

Problem 1 (2.16 in DPV: finding x in an infinite array)

You are given an infinite array $A[.]$ in which the first n entries contain different integers in sorted order and the rest are filled with ∞. You are not given the value of n. Describe an algorithm that takes as an input an integer x and finds a position in the array containing x, if such position exists, in $O(\log(n))$ time.

Problem 2 (2.17 in DPV: fixed point)

Given a sorted array of n distint integers $A = \{a_1, a_2, \ldots, a_n\}$, you want to find out whether there is an index i for which $a_i = i$. Give a divide and conquer algorithm to solve this problems that runs in time $O(\log(n))$.

Problem 3 (2.20 in DPV: ordering bounded input)

Design an algorithm that takes as input an array A with n integers and sort it in time $O(n + M)$ where $M = \max_i A[i] - \min_i A[i]$. Conclude that, for small values of M, sorting this way yields a linear time algorithm.

Problem 4 (Almost ordered)

You are given an unsorted list of numbers. You are told that the index of every number is at distance *at most* 100 from the index it will correspond to it if the list is sorted. Use this information to design an algorithm that takes as input a list with this property and outputs the list sorted. Explain the correctness of your algorithm and explain its running time. (*Hint: Think how to find the minimal entry in constant time!*)