CS 3510: Homework 1

Due on Thursday, May 27, 11:59pm via Gradescope.
Late submission with no penalty until Friday, May 28, 11:59am.

Professor Brito

CS 3510 Staff

Instructions.

For the graded problems, you are allow to use the following theorems and algorithm from classes as black-boxes. In doing so, you do not need to explain your algorithm. Clearly state what is your input and how you use the output. You can report their runtime as presented in class with no further explanation.

- Master Theorem, QuickSort, MergeSort and the Merging subroutine, FastSelect (median of medians).

Suggested reading.

We are covering topics from Chapter 2. You do not need to cover topics with haven't cover in lectures.

Suggested Problems.

All problems are from Chapter 2. The numeration corresponds to the 2006 edition of the book. You do not need to turn this problem in.

- 2.5 (to use the MAster Theorem directly. Some parts of this problem are more general recurrence we won't see in the class but you are welcome to explore.)
- 2.14 (deleting duplicates).
- 2.15 (implementing in place).
- 2.17 (fixed point).
- 2.19 (k-merge).
- 2.20 (sorting a bounded array).
- 2.22 (k^{th} element of the union of two sorted lists).
- 2.23 (majority element).

Practice problem.

Problem 2.16 (finding x in an infinite array.)

Problem 1

You are given a list of n bits $\{x_1, x_2, \ldots, x_n\}$ with each $x_i \in \{0, 1\}$. You have to output either:

- A natural number K such that $x_K = 1$, or
- 0 if all bits are equal to zero.

The only operation you are allow to access the inputs is a function $I(i, j)$ defined as:

$$I(i,j) = \begin{cases} 1, & \text{if some bit in } x_i, x_{i+1}, \ldots, x_j \text{ has value 1;} \\ 0, & \text{if all bits } x_i, x_{i+1}, \ldots, x_j \text{ have value 0.} \end{cases}$$

The function $I(\cdot, \cdot)$ runs in constant time.

Design a divide and conquer algorithm to solve this problem. Describe your algorithm with words (no pseudocode) and justify its correctness. Analyze and justify its running time. Make sure to include the recurrence relation.

Problem 2

You are given an array $A = [a_1, \ldots, a_n]$ of n positive numbers that represent the price of a particular stock on n days. You want to buy the stock at one day and sell on a later day. Your goal is to determine what would have been the best pair of days for buying/selling. You simply have to output the difference in price. You can assume n is a power of 2. Here is an example: $A = [10, 15, 6, 3, 7, 12, 2, 9]$. Then the optimal decision would be to purchase on day 4 for \$3 and sell on day 6 for \$12, so your algorithm should output \$9.

(a) Suppose you want to buy in the first $n/2$ days and you want to sell in the last $n/2$ days. Give an $O(n)$ time algorithm for finding the best pair of days for buying/selling under this restriction. Explain your algorithm in words and analyse its running time. You can use any approach you prefer, in particular, you do not have to use divide and conquer.

(b) Give a divide and conquer algorithm with running time $O(n \log n)$ for finding the best pair of days for buying/selling. (There are no restrictions on the days, except that you need to buy at an earlier date than you sell.) Explain your algorithm in words and analyze your algorithm, including stating and solving the relevant recurrence. *Hint: Your algorithm should use your solution to part (a).*