CSOR W4231–Spring, 2021
Eleni Drinea

Homework 1 (130 points)

Out: Friday, January 22, 2021
Due: 11:59pm, Friday, February 5, 2021

Homework Instructions.

1. For all algorithms that you are asked to "give" or "design", you should

   - Describe your algorithm clearly in English.
   - Give pseudocode.
   - Argue correctness.
   - Give the best upper bound that you can for the running time.

   You are also encouraged to analyze the space required by your algorithm but we will not remove marks if you don't, unless the problem explicitly asks you to analyze space complexity.

2. You should submit this assignment as a **pdf** file on courseworks. Other file formats will not be graded, and will automatically receive a score of 0. You may also be asked to submit this assignment on Gradescope; if so, there will related announcements on courseworks, piazza and in class by January 29.

3. I recommend you type your solutions using LaTeX. For every assignment, you will earn 5 extra credit points if you type your solutions using LaTeX or other software that prints equations and algorithms neatly. If you do not type your solutions, make sure that your hand-writing is very clear and that your scan is high quality.

4. You should write up the solutions **entirely on your own**. Collaboration is limited to discussion of ideas only. You should adhere to the department's academic honesty policy (see the course syllabus). Similarity between your solutions and solutions of your classmates or solutions posted online will result in receiving a 0 in this assignment, and possibly further disciplinary actions. There will be no exception to this policy and it may be applied retroactively if we have reasons to re-evaluate this homework.

Homework Problems

1. (15 points) Consider the problem of computing $N! = 1 \cdot 2 \cdot 3 \cdots N$.

   (a) If N is an n-bit integer, how many bits long is $N!$ in $\Theta()$ notation?

   (b) Give an algorithm to compute $N!$ and analyze its running time. Assume that the time to multiply two n-bit integers is $O(n^2)$.

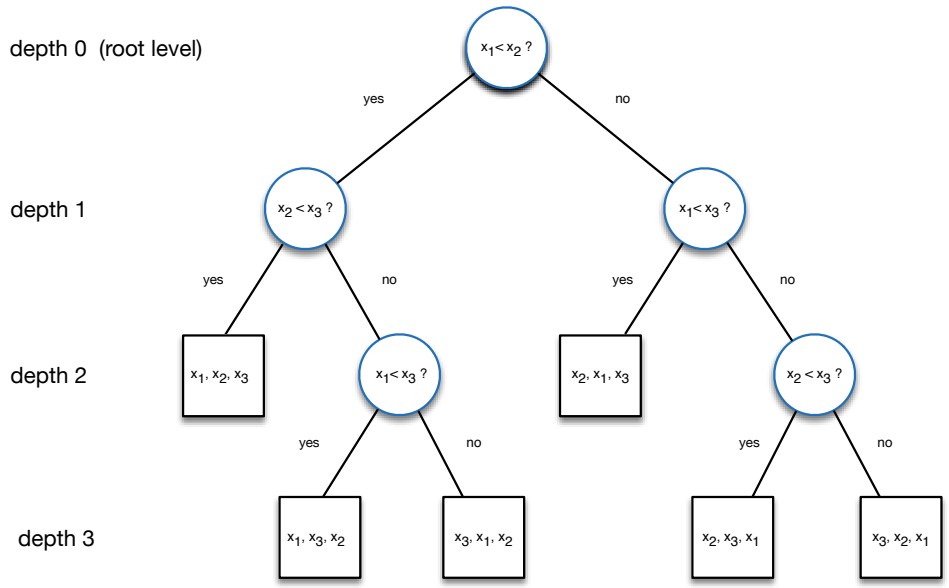

Figure 1: A tree corresponding to a comparison-based sorting algorithm on input x_1, x_2, x_3.

2. (35 points) On sorting

   (a) (15 points) *A lower bound for comparison-based sorting*

   You may view a comparison-based sorting algorithm as a binary tree. For example, the tree in Figure 1 sorts 3 integers x_1, x_2, x_3: it first compares x_1 to x_2; if $x_1 < x_2$, it compares x_2 to x_3, else is compares x_1 to x_3, etc. Each leaf corresponds to a possible permutation of the 3 numbers. For example, if $x_1 < x_2 < x_3$, we get the leaf labelled x_1, x_2, x_3.

   Suppose you want to sort n integers.

   i. (2 points) Argue that every possible permutation of the n numbers must appear as a leaf in the tree corresponding to a sorting algorithm.

   ii. (2 points) *Fill in the missing number in the following sentence:*
   Thus the tree has at least ... leaves.

   iii. (3 points) The worst-case time complexity of the sorting algorithm is given by the maximum number of comparisons performed by the algorithm. What does the latter quantity correspond to in the tree?

   iv. (3 points) Consider a binary tree of depth d. Give with a proof an upper bound on the number of leaves of the tree.

   v. (5 points) Derive a lower bound for the worst-case time complexity of a comparison-based sorting algorithm.

   (b) (17 points) Give an algorithm that sorts any array of n integers $x_1, \ldots, x_n$ in $O(n + D)$ time, where $D = \max_i x_i - \min_i x_i$.

   (c) (3 points) Suppose D is small (that is, $O(n)$). Does the algorithm you proposed in part (b) contradict the lower bound you derived in part (a)?

3. (25 points) In the table below, indicate the relationship between functions f and g for each pair (f, g) by writing "yes" or "no" in each box. For example, if $f = O(g)$ then write "yes" in the first box. Here $\log^b x = (\log_2 x)^b$.

f	g	O	o	Ω	ω	Θ
$\log^2 n$	$6 \log n$					
$\sqrt{\log n}$	$(\log \log n)^3$					
$4n \log n$	$n \log (4n)$					
$n^{3/5}$	$\sqrt{n} \log n$					
$5\sqrt{n} + \log n$	$2\sqrt{n}$					
$n^5 4^n$	5^n					
$\sqrt{n} 2^n$	$2^{n/2+\log n}$					
$n \log (2n)$	$\frac{n^2}{\log n}$					
$n!$	2^n					
$\log n!$	$(\log n)^{\log n}$					

4. (25 points) The Hadamard matrices $H_0, H_1, H_2, \ldots$ are defined as follows:

   - H_0 is the 1×1 matrix $\begin{bmatrix} 1 \end{bmatrix}$
   - For $k > 0$, H_k is the $2^k \times 2^k$ matrix

$$H_k = \begin{bmatrix} H_{k-1} & H_{k-1} \\ H_{k-1} & -H_{k-1} \end{bmatrix}$$

   Let ν be a column vector of length $n = 2^k$. Can you compute the matrix-vector product $H_k \nu$ faster than the straightforward algorithm that requires $O(n^2)$ time? (Assume that all the numbers involved are small enough that basic arithmetic operations like addition and multiplication take unit time.)

5. (30 points) Consider a complete binary tree T on $n = 2^d - 1$ nodes, for some integer $d > 1$. Each node v in T is labeled with a real number x_v; all x_v are distinct. A node v in T is a *local minimum* if its label x_v is less than the label x_u for every node u joined to v by an edge.

   You are given such a complete binary tree T but the labeling is only specified in the following *implicit* way: for each node v, you can determine x_v by *probing* the node v. Give an efficient algorithm to find a local minimum of T. (You may assume that each probe takes constant time.)

RECOMMENDED EXERCISES (do NOT submit, they will not be graded)

1. Give tight asymptotic bounds for the following recurrences.

   - $T(n) = 4T(n/2) + n^3 - 1$.
   - $T(n) = 8T(n/2) + n^2$.
   - $T(n) = 6T(n/3) + n$.
   - $T(n) = T(\sqrt{n}) + 1$.

2. Show that, if λ is a positive real number, then $f(n) = 1 + \lambda + \lambda^2 + \ldots + \lambda^n$ is

   (a) $\Theta(1)$ if $\lambda < 1$.

   (b) $\Theta(n)$ if $\lambda = 1$.

   (c) $\Theta(\lambda^n)$ if $\lambda > 1$.

   Therefore, in big-Θ notation, the sum of a geometric series is simply the first term if the series is strictly decreasing, the last term if the series is strictly increasing, or the number of terms if the series is unchanging.