CSOR W4246–Summer, 2021
Eleni Drinea

Homework 1 (120 points)

Out: Friday, May 7, 2021
Due: 11:59pm, Friday, May 14, 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, even if you don't give an entirely formal proof.
    - 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 your assignment as a **pdf** file on Gradescope. Other file formats will not be graded, and will automatically receive a score of 0.
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. (20 points) Consider the following recursive algorithm. On input a list of distinct numbers, the algorithm runs in three phases. In the first phase, the first $\lceil 2/3 \rceil$ elements of the list are sorted recursively; when the list has size 2, return the list if ordered, otherwise swap. In the second phase, the last $\lceil 2/3 \rceil$ elements are sorted recursively. Finally, in the third phase, the first $\lceil 2/3 \rceil$ elements are sorted recursively again.

   Give pseudocode for this algorithm, prove its correctness and derive a recurrence for its running time. Use the recurrence to bound its asymptotic running time. Would you use this algorithm in your next application to sort?

2. (30 points) In this problem, we will analyze an algorithm that finds an item *close enough* to the median item of a set $S = \{a_1, \ldots, a_n\}$ of n distinct numbers. Specifically, the algorithm finds an item a_i such that at least $n/4$ items are smaller than a_i and at least $n/4$ items are greater than a_i.

Algorithm 1

Randomized Approximate Median(S)

```
1: Select an item $a_i \in S$ uniformly at random
2: $rank = 1$
3: for $j = 1$ to $n$ do
4:     if $a_j < a_i$ then $rank = rank + 1$
5:     end if
6: end for
7: if $n/4 \leq rank \leq 3n/4$ then return $a_i$
8: else return error
9: end if
```

(a) What is the running time of this algorithm?

(b) What kind of randomized algorithm is `Randomized Approximate Median` and why? What is the **success probability** of this algorithm, that is, the probability that it will return an element (and not `error`)?

(c) How can you improve the success probability of the algorithm to over 99%? What is the running time of the new algorithm?

3. (30 points) Suppose that a sequence of items passes by one at a time. We want to maintain a sample of one item with the property that it is uniformly distributed over all the items that we have seen so far. Moreover we do not know the total number of items in advance and we cannot store more than one item at any time.

(a) Consider the following algorithm. When the first item appears, we store it. When the k-th item appears, we replace the stored item with probability $1/k$. Show that this algorithm solves the problem.

(b) Now suppose that when the k-th item appears, we replace the stored item with probability $1/2$. What is the distribution of the stored item in this case?

4. (40 points) The Fibonacci numbers are defined by the recurrence

$$F_n = \begin{cases} 0, & \text{if } n = 0 \\ 1, & \text{if } n = 1 \\ F_{n-1} + F_{n-2}, & \text{if } n \geq 2 \end{cases}$$

(a) (6 points) Show that $F_n \geq 2^{n/2}$, $n \geq 6$.

(b) Assume that the cost of adding, subtracting, or multiplying two integers is $O(1)$, independent of the size of the integers.

i) (6 points) Give an algorithm that computes F_n based on the recursive definition above. Develop a recurrence for the running time of your algorithm and give an asymptotic lower bound for it.

ii) (8 points) Give a non-recursive algorithm that asymptotically performs fewer additions than the recursive algorithm. Discuss the running time of the new algorithm.

iii) (20 points) Given an algorithm to compute F_n in $O(\log n)$ time using only integer additions and multiplications.

(*Hint: Observe that*

$$\begin{bmatrix} F_2 \\ F_1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} F_1 \\ F_0 \end{bmatrix}.$$

Can you build on this to compute F_n?)

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.