Homework 2

Due Date: September 16, 2019 at 11:59PM on Gradescope.
Instructor: Jeremiah Blocki

Homework Guideline Reminders

- Assignments must be typed. Submit one pdf file to Gradescope by 11:59PM, or else late penalties will apply. The pdf file can include hand-drawn images of figures.
- Each question needs to start with the resources and collaborator (RC) statement. You will not be penalized for using resources or having collaborators if your answers are expressed in your own words. If you consulted no resources outside of course material or had no collaborators, you must state so. A question without a complete RC statement will not be graded.

Question 1: Recurrences (28 points)

Find upper and lower bounds for the following functions (big O and Ω). You may use the Master Theorem without proof (when applicable), but you should explain which case is being used. If you solve the recurrence using another technique you will need to prove that your solution is correct. You may assume that $1 \leq T(n) \leq 381^{2020}$ whenever $1 \leq n \leq 100$.

(a) T(n) = 8T(n/5) + n^5 (3 points)

(b) T(n) = 5T($n^{1/10}$) + 2$\log_2 n$ (3 points)

(c) T(n) = 5T(n/2) + $n^{log_7 22} \cdot \log(n^2)$ (3 points)

(d) T(n) = $n + \sum_{i=1}^{k}$ T($a_i n$) where $\sum_{i=1}^{k} a_i = 1 - \epsilon$ for some constants $\epsilon > 0$, $k \geq 1$ and $a_i > 0$ for each $i \leq k$. (5 points)

(e) T(n) = $n + \sum_{i=1}^{k}$ T($a_i n$) where $\sum_{i=1}^{k} a_i = 1$ and $a_i > 0$ for each $i \leq k$. You may assume that $k \geq 2$ is a constant. (5 points)

(f) T(n) = 36T(n/9) + $10n \cdot n^{\log_3 112}$ (3 points)

(g) T(n) = 9T(n/3) + n^2 (3 points)

(h) T(n) = T(n-1) + $2020n^{2020}$ (3 points)

Question 2: Linear Time Selection (22 points)

Consider the algorithm `FindKthLargest` (defined below) which takes as input an (unsorted) array $x_1, \ldots, x_n$ and an integer k and outputs the kth smallest item in the array. Formally, if $y_1, \ldots, y_n = \texttt{Sort}(x_1, \ldots, x_n)$ then the algorithm should return y_k.

(a) The algorithm `FindKthLargest` is parameterized by an integer $t \geq 1$. When $t = 2$ we have $(2t+1) = 5$ and we get the "Median of Fives" algorithm discussed in class. Write down a recurrence describing the running time of this algorithm. Use this recurrence to upper/lower bound the running time.

(b) The algorithm `FindKthLargest` is still correct when we set $t = 1$. Write down a recurrence describing the running time of this algorithm. Use this recurrence to upper/lower bound the running time.

(c) Write down the general recurrence describing the running time of `FindKthLargest` when $t > 2$. What is the running time of the algorithm when t is a constant greater than 2?

(d) Use induction to prove that the algorithm `FindKthLargest` (described below) is correct.

Algorithm 1 `FindKthLargest` $(\{x_1, \ldots, x_n\}, k)$

```
1: If $n \leq (2t+1)$ then sort the list $\{x_1, \ldots, x_n\}$ to obtain $y_1 \leq y_2 \leq ... \leq y_n$ and return $y_k$.
2: Partition $x_1, \ldots, x_n$ into $n/(2t+1)$ groups
       $G_1 = \{x_1, \ldots, x_{2t+1}\}, G_2 = \{x_{2t+2}, \ldots, x_{4t+2}\}, \ldots$
   and compute the median $y_i$ of each group $G_i$ by setting
       $y_i \leftarrow \texttt{FindKthLargest}(G_i, t+1)$.
3: Recursively compute the median $y_{med}$ of the list $y_1, \ldots, y_{n/(2t+1)}$ by running
       $y_{med} \leftarrow \texttt{FindKthLargest}\left(\{y_1, \ldots, y_{n/(2t+1)}\}, k = n/(4t+2)\right)$
4: Split $x_1, \ldots, x_n$ into lists
       $L = \{x_i : x_i < y_{med}\}$,
       $M = \{x_i : x_i = y_{med}\}$,
       $R = \{x_i : x_i > y_{med}\}$.
5: If $k \leq |L|$ then return $\texttt{FindKthLargest}(L, k)$; else if $k \leq |L| + |M|$ return $y_{med}$; otherwise
   return $\texttt{FindKthLargest}(R, k - |L| - |M|)$.
```

Question 3: Divide and Conquer (25 points)

After invading Zone 51, invaders have gathered n items with distinct positive weights and stacked them in a pile. A pair of items in the pile is said to be "γ-unbalanced" for $\gamma \geq 1$ if the weight of the upper item times γ exceeds (is strictly greater than) the weight of the lower item. Moving the pile in a vehicle is very risky unless stacked in a particular order of weights from top to bottom such that there is no γ-unbalanced pairs. Such an ordering is called γ-ordering. Invaders want to know how far the pile is from γ-ordering. Invaders are in a time constraint as they are being chased by security personnel; given a specific γ, they need an efficient algorithm to compute the number of γ-unbalanced pairs in the pile. Note that invaders do not need the actual γ-ordering, once they know the number of γ-unbalanced pairs, they can do the γ-ordering by themselves.

(a) Help the invaders by providing such a divide and conquer algorithm to find the number of γ-unbalanced pairs. Prove the correctness of your algorithm.

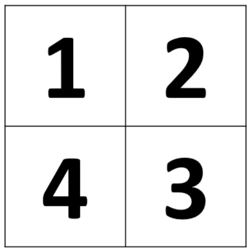
(b) Convince the invaders that your algorithm is efficient by showing time complexity analysis. State and explain the recurrence relation describing the running time of your algorithm. Solve the recurrence and provide the asymptotic running time.

Question 4: Divide and Conquer (25 points)

Jane was given a square matrix of size $n \times n$ with entries 0's and 1s, the goal was to come up with an efficient divide and conquer algorithm to find the length of the longest sequence of 1's in the matrix. The sequence can be horizontal or vertical, in other words, it can be contiguous sequence of 1's in a row or in a column. Note that a combination of both horizontal and vertical is NOT valid, for example, L shaped figures are not to be considered.

Jane devised the following algorithm to solve the problem:

1. If matrix is size 1×1, return value of the single element.
2. Divide matrix into 4 quadrants each having side of length $\frac{n}{2}$ as shown below. Recursively find the longest sequences in each quadrant: S_1, S_2, S_3, S_4.



3. In the combine step, compare each pair of adjacent quadrants. For each pair, check the length of all possible sequences of 1s that go across the two quadrants. For instance, if we are looking at quadrants 1 and 2, for each row, count the number of consecutive 1s on the right side of the row in quadrant 1 and on the left side of the row in quadrant

2, and sum them to obtain the longest sequence of 1s in that particular row across the two quadrants. Choose the row with the maximum length sequence of 1s found. We will end up with the lengths of the longest sequences of 1s between each pair of adjacent quadrants: $S_{12}, S_{23}, S_{34}, S_{41}$.

4. Return maximum of all values: longest sequences in each quadrant and calculated lengths of sequences that go across quadrants. That is,

$$\max(S_1, S_2, S_3, S_4, S_{12}, S_{23}, S_{34}, S_{41})$$

For this problem,

(a) State and explain the recurrence relation describing the running time of Jane's algorithm. Solve the recurrence and provide the asymptotic running time.

(b) Unfortunately, Professor Blocki is not happy with the run-time of her algorithm. He believes she can do better. Can you help Jane by tweaking her algorithm to give a divide and conquer solution with a better runtime? Clearly specify the divide step, conquer step, and what values are returned during each iteration. You must use divide and conquer even if there are other simpler solutions.

(c) State and explain the recurrence relation describing the running time of your modification of Jane's algorithm. Solve the recurrence and provide the asymptotic running time.