CS4102 - ALGORITHMS - SPRING 2020

DUE THURSDAY, JANUARY 30, 2019 AT 11P VIA COLLAB

H1(YL4DF)

Collaboration Policy: You are encouraged to collaborate with up to 4 other students, but all work submitted must be your own *independently* written solution. List the computing ids of all of your collaborators in the `collabs` command at the top of the tex file. Do not share written notes, documents (including Google docs, Overleaf docs, discussion notes, PDFs), or code. Do not seek published or online solutions for any assignments. If you use any published or online resources (which may not include solutions) when completing this assignment, be sure to cite them. Do not submit a solution that you are unable to explain orally to a member of the course staff. Any solutions that share similar text/code will be considered in breach of this policy. Please refer to the syllabus for a complete description of the collaboration policy.

Collaborators: mw5ew

Sources: Cormen, et al, Introduction to Algorithms

PROBLEM 1 *Asymptotics*

Prove or disprove each of the following statements. For 1-3, let $f(n)$ and $g(n)$ be arbitrary asymptotially positive functions.

1. $f(n) \in O(g(n))$ implies $g(n) \in \Omega(f(n))$.

   Proof. Direct Prove. Since $f(n) \in O(g(n))$, then $\exists c, n_0 > 0$ s.t. $\forall n > n_0$, $f(n) \leq c \cdot g(n)$. This implies $\exists k = \frac{1}{c}, n_0 > 0$ s.t. $\forall n > n_0$, $g(n) \geq k \cdot f(n)$. Thus, $g(n) \in \Omega(f(n))$. □

2. $f(n) \notin \Omega(g(n))$ implies $\log(f(n)) \notin \Omega(\log(g(n)))$.

   Proof. Disprove. Let $f(n) = n$ and $g(n) = n^2$. Since $\forall c, n_0 > 0$ s.t. $\exists n > n_0 = c$, $f(n) < c \cdot g(n)$, then $f(n) \notin \Omega(g(n))$. $\log f(n) = \log n$ and $\log g(n) = \log(n^2) = 2 \log n$. Then $\exists c = 2, n_0 = 1$ s.t. $\forall n > n_0$, $\log f(n) \geq c \cdot \log g(n)$. Therefore, $\log(f(n)) \in \Omega(\log(g(n)))$ □

3. $f(n) \in \Theta(f(n/6))$.

   Proof. Disprove. We picked one specific function $f(n) = e^n$. Suppose $e^n \in \Theta(e^{\frac{n}{6}})$. Then $\exists c, n_0 > 0$ s.t. $\forall n > n_0$, $e^n \leq c \cdot e^{\frac{n}{6}}$. This implies $c \geq e^{\frac{5n}{6}}$, which contradicts the fact c is constant. Thus, $f(n) \notin \Theta(f(n/6))$. □

4. $n(\log n)^2 \in O\left(n^{1.5}(\log n)\right)$.

   Proof. Direct Prove. Let $c = 1$ and $n = 1$. We know $\log n = 0$ and $\sqrt{n} = 1$ when $n = 1$. Then $\forall n_0 > 1$, we have $\log n < \sqrt{n}$. It follows that $n(\log n)^2 \leq 1 \cdot n^{1.5}(\log n)$. Thus, $n(\log n)^2 \in O\left(n^{1.5}(\log n)\right)$. □

PROBLEM 2 *Iterated Functions*

When solving recurrence relations, we typically need to determine the depth of the recursion. For instance, consider the BetterAttendance algorithm discussed in Lecture 2. The depth of our recursion in this case was $\lceil \log_2 n \rceil$ for a class of n students, i.e. the number of times you can repeatedly divide n by 2 before reaching a value ≤ 1.

Definition 1 *Define the iterated function* $f_c^*(n) = \min\{i \geq 0 | f^{(i)}(n) \leq c\}$ *where* f *is a monotonically increasing function over the reals,* c *is a given constant, and* $f^{(i)}(n) = f(f^{(i-1)}(n))$*, e.g.* $f^{(3)}(n) = f(f(f(n)))$.

Intuitively, $f_c^*(n)$ gives the count of how many times f must be repeatedly applied to n before reaching a value $\leq c$. Note that since the value is a count, it will always be integral. We could describe the depth of the recursion for BetterAttendance using this iterated functions notation with $f(n) = \frac{n}{2}$ and $c = 1$, giving $f_1^*(n) = \lceil \log_2 n \rceil$.

Fill in the table below. For each of the given functions $f(n)$ and constants c, give as tight a bound as possible on $f_c^*(n)$. The first row is done for you.

$f(n)$	c	$f_c^*(n)$
$n/2$	1	$\lceil \log_2 n \rceil$
$n/2$	2	$\lceil (\log_2 n) - 1 \rceil$
$n/3$	1	$\lceil \log_3 n \rceil$
$3n/4$	1	$\lceil \log_{4/3} n \rceil$
$n-1$	0	$\lceil n \rceil$
$\sqrt{n}$	1	∞
$n^{1/3}$	2	$\lceil \log_3 (\log_2 n) \rceil$

PROBLEM 3 *Solving Recurrences*

Prove a (as tight as possible) O (big-Oh) asymptotic bound on the following recurrences. You may use any base cases you'd like, but you may not use the Master Theorem for this homework.

1. $T(n) = 4T(\frac{n}{2}) + n^2$

   Proof. Let base case be $T(1) = 1$. Then $T(n) = 4T(\frac{n}{2}) + n^2 = 4(4T(\frac{n}{4}) + \frac{n^2}{4}) + n^2 = 2n^2 + 16T(\frac{n}{4}) = 3n^2 + 64T(\frac{n}{8}) = ... = d \cdot n^2 + 4^{\log_2 n} \cdot T(1)$ where d is the depth of recurrence. The depth is $\log_2 n$, so we have $T(n) = (\log_2 n)n^2 + 4^{\log_2 n} = (\log_2 n)n^2 + n^2 = O((\log_2 n)n^2)$.
   Now I am going to use proof by induction to prove $T(n) \leq (\log_2 n)n^2 + n^2$.
   Base Case: $T(1) = (\log_2 1) \cdot 1^2 + 1^2 = 1$.
   Hypothesis: $\forall n \leq x_0$, $T(n) \leq (\log_2 n)n^2 + n^2$.
   Inductive Step:

$$
\begin{aligned}
T(x_0+1) &= 4T(\frac{x_0+1}{2}) + (x_0+1)^2 && (1)\\
&\leq 4((\log_2 \frac{x_0+1}{2})(\frac{x_0+1}{2})^2 + (\frac{x_0+1}{2})^2) + (x_0+1)^2 && (2)\\
&\leq (\log_2 \frac{x_0+1}{2})(x_0+1)^2 + (x_0+1)^2 + (x_0+1)^2 && (3)\\
&\leq (\log_2(x_0+1) - 1)(x_0+1)^2 + (x_0+1)^2 + (x_0+1)^2 && (4)\\
&\leq \log_2(x_0+1)(x_0+1)^2 + (x_0+1)^2 && (5)
\end{aligned}
$$

   Thus, $T(n) \leq (\log_2 n)n^2 + n^2 = O((\log_2 n)n^2)$. □

2. $T(n) = 3T(\frac{n}{3}) + 3n$

   Proof. Let base case be $T(1) = 1$. Then $T(n) = 3T(\frac{n}{3}) + 3n = 3(3T(\frac{n}{9}) + n) + 3n = 6n + 9T(\frac{n}{9}) = ... = d \cdot 3n + 3^{\log_3 n} \cdot T(1)$ where d is the depth of recurrence. The depth is $\log_3 n$, so we have $T(n) = (\log_3 n)3n + 3^{\log_3 n} = (\log_3 n)3n + n = O((\log_3 n)n)$. Now I am going to use proof by induction to prove $T(n) \leq (\log_3 n)3n + n$.
   Base Case: $T(1) = (\log_3 1) \cdot 3 + 1 = 1$.
   Hypothesis: $\forall n \leq x_0$, $T(n) \leq (\log_3 n)3n + n$.
   Inductive Step:

$$
\begin{aligned}
T(x_0+1) &= 3T(\frac{x_0+1}{3}) + 3(x_0+1) && (6)\\
&\leq 3((\log_3 \frac{x_0+1}{3})(x_0+1) + (\frac{x_0+1}{3})) + 3(x_0+1) && (7)\\
&\leq 3(\log_3 \frac{x_0+1}{3})(x_0+1) + (x_0+1) + 3(x_0+1) && (8)\\
&\leq 3(\log_3(x_0+1) - 1)(x_0+1) + (x_0+1) + 3(x_0+1) && (9)\\
&\leq (\log_3(x_0+1))3(x_0+1) + (x_0+1) && (10)
\end{aligned}
$$

   Thus, $T(n) \leq (\log_3 n)3n + n = O((\log_3 n)n)$ □

3. $T(n) = 2T(\frac{n}{3}) + T(\frac{n}{6}) + n$

   Proof. Let base case be $T(1) = 1$. We use the recursion tree to solve the recurrences. The depth is determined by 3 so the depth is $\log_3 n$. The cost at the first level is n, the second

level is $\frac{5n}{6}$, and the third level is $\frac{25n}{36}$. Then we have $T(n) = n + \frac{5n}{6} + \frac{25n}{36} + ... + \frac{5}{6}n^{(\log_3 n)-1}$. By applying the property of geometric series, we can simplify

$$T(n) = \frac{n(1 - \frac{5}{6}^{log_3 n})}{1 - \frac{5}{6}} = 6n(1 - (\frac{5}{6})^{log_3 n}) \leq 6n = O(n) \tag{11}$$

Base Case: $T(1) = 1 \leq 6$.
Hypothesis: $\forall n \leq x_0$, $T(n) \leq 6n$.
Inductive Step:

$$T(x_0 + 1) = 2T(\frac{x_0 + 1}{3}) + T(\frac{x_0 + 1}{6}) + (x_0 + 1) \tag{12}$$
$$\leq 2 \cdot 6 \cdot (\frac{x_0 + 1}{3}) + 6 \cdot (\frac{x_0 + 1}{6}) + (x_0 + 1) \tag{13}$$
$$\leq 4(x_0 + 1) + (x_0 + 1) + (x_0 + 1) \tag{14}$$
$$\leq 6(x_0 + 1) \tag{15}$$

Thus, $T(n) \leq 6n = O(n)$. □

PROBLEM 4 *Where is Batman when you need him?*

As the newly-hired commissioner of the Gotham City Police Department, James Gordon's first act is to immediately fire all of the dirty cops, stamping out Gotham's widespread police corruption. To do this, Commissioner Gordon must first figure out which officers are honest and which are dirty. There are n officers in the department. The majority ($> n/2$) of the officers are honest, and every officer knows whether or not each other officer is dirty. He will identify the dirty cops by asking the officers, in pairs, to indicate whether the other is dirty. Honest officers will always answer truthfully, dirty cops may answer arbitrarily. Thus the following responses are possible:

Officer A	Officer B	Implication
"B is honest"	"A is honest"	Either both are honest or both are dirty
"B is honest"	"A is dirty"	At least one is dirty
"B is dirty"	"A is honest"	At least one is dirty
"B is dirty"	"A is dirty"	At least one is dirty

1. A group of n officers is uncorrupted if more than half are honest. Suppose we start with an uncorrupted group of n officers. Describe a method that uses only $\lfloor n/2 \rfloor$ pair-wise tests between officers to find a smaller uncorrupted group of at most $\lceil n/2 \rceil$ officers. Prove that your method satisfies each of the three requirements.

   Method:
   Step 0. Case 1: If n is even, there is nothing to do at this step. Case 2: If n is odd, remove one random officer.
   Step 1. Pair up all officers and let them ask each other to indicate whether the other is dirty.
   Step 2. Case 1: If one or more officers indicate the other is dirty, we remove both officers from the group. Case 2: If both officers indicate the other is honest, we keep one of the officers(no matter which one).
   Step 3. If the original group is even, then there is nothing to do at this moment. If the original group is odd, then we add back the officer which was removed in Step 0.

   Proof. Requirement 1 ("Only $\lfloor n/2 \rfloor$ pair-wise tests"): When n is even, there are $\frac{n}{2}$ tests. When n is odd, there are $\frac{n-1}{2}$ tests. Thus, we use only $\lfloor n/2 \rfloor$ pair-wise tests.
   Requirement 2 ("Smaller group consisting of at most $\lceil n/2 \rceil$ officers"): If Case 1 of Step 2

happens, we remove both officers. If Case 2 of Step 2 happens, we remove one of them. This implies that we at least remove one half officers from the group, so we have a smaller group consisting of at most $\lceil n/2 \rceil$ officers. If Step 3 is necessary, the smaller group has $\frac{n+1}{2} = \lceil n/2 \rceil$ officers and the requirement is still satisfied.
Requirement 3 ("The smaller group is uncorrupted"): We analyze case by case. In Case 1, we probably remove both dirty officers or remove one honest and one dirty officer. Since our original group is uncorrupted, the smaller group still has more honest officers than dirty officers. In Case 2, we either remove one honest officer or one dirty officer. Since our original group is uncorrupted, the pair of honest officers is more than or equal to the pair of dirty officers. Removing one honest officer or one dirty officer will always keep the number of honest officers more than or equal to the number of dirty officers. At this point, if the number of honest officers is equal to the number of dirty officers, this implies that the officer we removed in Step 0 is honest. As the result, the Step 3 will always add one honest officer back to get the majority of honesty. Thus, overall, the smaller group is uncorrupted.
□

2. Using this approach, devise an algorithm that identifies which officers are honest and which are dirty using only $\Theta(n)$ pairwise tests. Prove the correctness of your algorithm, and prove that only $\Theta(n)$ tests are used.

   Based on my method above, I could divide the group in half and make $n/2$ tests every time until we have only 1 officer left. The cost for each level is $n/2$, so the equation for $T(n)$ is

$$T(n) = T(\frac{n}{2}) + \frac{n}{2} \tag{16}$$

   Now we would like to prove $T(n) \in \Theta(n)$.

   Proof. Proof by Induction. Our goal is to prove $T(n) = n$.
   Base Case: $T(0) = 0$.
   Hypothesis: $\forall n \leq x_0$, $T(n) = n$.
   Inductive Step:

$$T(x_0 + 1) = T(\frac{x_0 + 1}{2}) + \frac{x_0 + 1}{2} \tag{17}$$

$$= \frac{x_0 + 1}{2} + \frac{x_0 + 1}{2} \tag{18}$$

$$= x_0 + 1 \tag{19}$$

   Therefore, $T(n) = n \in \Theta(n)$. □

3. Prove that a conspiracy of $\lfloor n/2 \rfloor + 1$ dirty officers (who may share a plan) can foil *any* attempt to find a honest officer. I.e., not only will method above not work, but that there is no way *at all* for Commissioner Gordon to identify even one honest officer if there is not an honest majority.

   Proof. Consider this case: there are two dirty officers and one honest officer at the n level. The two dirty officers will indicate others to be dirty. As the result, the answers we get from each one will be the same, i.e. every one in the group is dirty. Thus, there is no algorithm to determine the honest officer in the group of three because all responses are the same. To expand the case to a more general situation, we can conclude that when dirty officers are more than honest officers, we cannot identify honest officers in any way. □

PROBLEM 5 (EXTRA CREDIT) *Karatsuba Example*

Illustrate the Karatsuba algorithm on 20204102×17526624. Use 2-digit multiplication as your base case.

Proof.

$x = 20204102$ and $y = 17526624$. Let $a_0 = 2020$, $a_1 = 4102$, $b_0 = 1752$, $b_1 = 6624$.
$a_0 = 2020$, $a_1 = 4102$, $b_0 = 1752$, $b_1 = 6624$.
$z_0 = 2020 * 1752$
$z_2 = 4102 * 6624$
$z_1 = (2020 + 4102) * (1752 + 6624) - z_0 - z_2 = 6122 * 8376 - z_0 - z_2$

To calculate z_0, $c_0 = 20$, $c_1 = 20$, $d_0 = 17$, $d_1 = 52$
$t_0 = 20 * 17 = 340$
$t_2 = 20 * 52 = 1040$
$t_1 = (20 + 20) * (17 + 52) - t_0 - t_2 = 2760 - 340 - 1040 = 1380$
$z_0 = t_0 * 100^2 + t_1 * 100 + t_2 = 3400000 + 138000 + 1040 = 3539040$

To calculate z_2, $e_0 = 41$, $e_1 = 02$, $f_0 = 66$, $f_1 = 24$
$s_0 = 41 * 66 = 2706$
$s_2 = 2 * 24 = 48$
$s_1 = (41 + 2) * (66 + 24) - 2706 - 48 = 1116$
$z_2 = 2706 * 100^2 + 1116 * 100 + 48 = 27060000 + 111600 + 48 = 27171648$

To calculate $6122 * 8376$, $g_0 = 61$, $g_1 = 22$, $h_0 = 83$, $h_1 = 76$
$r_0 = 61 * 83 = 5063$
$r_2 = 22 * 76 = 1672$
$r_1 = (61 + 22) * (83 + 76) - 5063 - 1672 = 83 * 159 - 5063 - 1672$

To calculate $83 * 159$, $p_0 = 8$, $p_1 = 3$, $q_0 = 15$, $q_1 = 9$
$u_0 = 8 * 15 = 120$
$u_2 = 3 * 9 = 27$
$u_1 = (8 + 3) * (15 + 9) - 120 - 27 = 264 - 120 - 27 = 117$
$83 * 159 = 120 * 10^2 + 117 * 10 + 27 = 12000 + 1170 + 27 = 13197$

Then we have $r_1 = 13197 - 5063 - 1672 = 6462$ and $6122 * 8376 = 5063 * 100^2 + 6462 * 100 + 1672 = 51277872$

Now, $z_0 = 2020 * 1752 = 3539040$
$z_2 = 4102 * 6624 = 27171648$
$z_1 = (2020 + 4102) * (1752 + 6624) - z_0 - z_2 = 6122 * 8376 - z_0 - z_2 = 51277872 - 3539040 - 27171648 = 20567184$
Finally, $20204102 * 17526624 = 3539040 * 10000^2 + 20567184 * 10000 + 27171648 = 353904000000000 + 205671840000 + 27171648 = 354109699011648$

□