CS 341, Spring 2020
Semih Salihoglu

# Assignment 1 Part 2 (due Sunday, May 31, midnight EST)

**Instructions:**

- Hand in your assignment using Crowdmark. Detailed instructions are on the course website.
- Give complete legible solutions to all questions.
- Your answers will be marked for clarity as well as correctness.
- For any algorithm you present, you should justify its correctness (if it is not obvious) and analyze the complexity.

1. [*15 marks*] A group of hackers from an enemy organization has attempted to install a virus to $n$ of your company's computers. Your software engineers have designed a test, called `TEST-EACH-OTHER`, that takes two computers $c_A$ and $c_B$, where each input computer tests the other and outputs whether the other one is infected with the virus $(+)$ or not infected with the virus $(-)$. If a computer is actually $-$ than it will always output a correct result. Unfortunately, if it is $+$, its reply is unrelated to the real state of the other computer and hence cannot be trusted. In other words, a computer $c_A$ that is infected with the virus can be "dishonest" and output the correct or the incorrect state of $c_B$.

   The following table summarizes the four possible outcomes of running `TEST-EACH-OTHER` on two computers $c_A$ and $c_B$, and what we can conclude from it. Please review the table to ensure that these are indeed the possible outcomes.

   | $c_A$**'s output** | $c_B$**'s output** | **Conclusion** |
   |---|---|---|
   | $c_B$ is $-$ | $c_A$ is $-$ | either both $-$ or both $+$ |
   | $c_B$ is $-$ | $c_A$ is $+$ | at least one is $+$ |
   | $c_B$ is $+$ | $c_A$ is $-$ | at least one is $+$ |
   | $c_B$ is $+$ | $c_A$ is $+$ | at least one is $+$ |

   Luckily your security experts have told you that more than $n/2$ computers were not infected (so they are $-$). Your goal is to identify all the $+$ and $-$ computers. Below, running one instance of `TEST-EACH-OTHER` constitutes one test.

   (a) [*12 marks*] Describe an algorithm to find a single $-$ phone by performing $O(n)$ tests. [Hint: Think of how you can use $O(n)$ tests to reduce the problem size by a constant factor.]

   We first partition all computers in to two equally sized groups or two groups with size difference of 1 if $n$ is odd. Let $A$ and $B$ be the two groups. Then run `TEST-EACH-OTHER` using the $i$-th computer in $A$ $a_i$ to test the $i$-th computer in $B$ $b_i$. And using $b_i$ to test

$a_i$. If the outputs are both $-$, put them into a new set $D$. And we pick the first element of each pair and do the same process above, until there is less than 2 elements left. We assume `TEST-EACH-OTHER`$(c_A, c_B)$ is equivalent to use $c_A$ to test $c_B$.

**Algorithm 1:** FindUninfectedCmptr($c[1,\ldots,n]$)

**if** $size(c) \leq 2$ **then**
  **return** $c[1]$
$A = c[1,\ldots,\lfloor n/2 \rfloor]$
$B = c[\lfloor n/2 \rfloor + 1,\ldots,n]$
$D = \emptyset$
**for** $c_a \in A$ *and* $c_b \in B$ **do**
  **if** *TEST-EACH-OTHER*$(c_a, c_b) == -$ *and* *TEST-EACH-OTHER*$(c_b, c_a) == -$ **then**
    $D = D \cap \{c_a\}$
**if** $|D|$ *is even and* $|c|$ *is odd* **then**
  $D = D \cap \{c[n]\}$
**return** FindUninfectedCmptr($D$)

Every time with results are not $--$, we remove one uninfected computer and one infected or two infected computers. So at least $\frac{1}{2}$ of the computers in the new set are uninfected. Since there are more that $n/2$ computers are not infected. There is at least one piar where both computers are $-$. $D$ is not empty if $n$ is even. If $n$ is odd, $D$ is even and there are two more $-$ computers than $+$ computers, the number of $-$ computers is more than the number of $+$ computers after adding any computer with either $-$ or $+$. If $n$ is odd, $D$ is even and the number of $+$ computers and the number of $-$ computers is equal, then the single computer is $-$, since ther are more than $n/2$ computers were not infected. Every time we reduce the set to at least $\frac{1}{2}$ of its original size, we will eventually have a set with size at most 4. Hence, the program always terminates and return the correct output.

**Analysis:** Every time we left with at most $\lfloor \frac{n}{2} \rfloor + 1$ of the computers and call `TEST-EACH-OTHER` $2n$ times. Let $T(n)$ be the number of tests of $n$ computers.

$$\begin{aligned}
T(n) &\leq 2n + (2 \times \frac{n}{2} + 1) + (2 \times \frac{n}{4} + 1) + \cdots + 2 + 1 \\
&= 2(n + \frac{n}{2} + \frac{n}{4} + \cdots + \frac{n}{\frac{n}{2}}) + \log_2 \frac{n}{2} + 1 \\
&= 2n \frac{1 - (\frac{1}{2})^{\log_2 \frac{n}{2} + 2}}{1 - \frac{1}{2}} + \log_2 n - \log_2 2 + 1 \\
&= 4n(1 - (\frac{1}{2})^{\log_2 n + 1}) + \log_2 n \\
&= 4n - 2 + \log_2 n \\
&\in O(n)
\end{aligned}$$

(b) [*3 marks*] Using part (a), show how to identify the condition of each computer by performing $O(n)$ tests.

We first use the algorithm from (a) to find a single computer $c$ that is not infected and call `TEST-EACH-OTHER` using $c$ to determine whether other computers are infected. Since finding $c$ costs $O(n)$ tests and using $c$ to determine the other computers' status uses $n - 1$ tests, so the overall runtime is $O(n)$.

2. [*12 marks*] Consider the recurrence:

$$T(n) = 2T(\lfloor n/9 \rfloor) + \sqrt{n} \qquad \text{if } n \geq 9$$

$$T(n) = 5 \qquad \text{if } n < 9$$

Prove $T(n) = O(\sqrt{n})$ by induction (i.e., guess-and-check or substitution method). Show what your $c$ and $n_0$ are in your big-oh bound. Note that depending on the choice of your $n_0$, you might have to cover multiple base cases in your inductive proof.

Assume T$(n) \leq 5\sqrt{n}$
**Base Case:** Assume $k < 9$, $T(k) = 5 \leq 5\sqrt{k}$
**IH:** Assume $T(k) \leq 5\sqrt{n}$. for all $k \leq n-1$
**IS:**

$$\begin{aligned} T(n) &= 2T(\lfloor n/9 \rfloor) + \sqrt{n} \\ &= 2(5\sqrt{\lfloor n/9 \rfloor}) + \sqrt{n} \\ &\leq \frac{10}{\sqrt{9}}\sqrt{n} + \sqrt{n} \\ &\leq 5\sqrt{n} \end{aligned}$$

Let $c = 6$, $n_0 = 1$.
Clearly, $5\sqrt{n} \leq 6\sqrt{n}$ for all $n \geq n_0$. Hence, $T(n) \leq 5\sqrt{n} \in O(\sqrt{n})$

3. [*16 marks*] Give tight asymptotic ($\Theta$) bounds for the solution to the following recurrences by using the recursion-tree method or the induction method (your choice). You may assume that $n$ is a power of 10 in (a), or a power of 3 in (b). Show your work.

(a) [*8 marks*]

$$T(n) = \begin{cases} 2\,T(n/10) + \sqrt{n} & \text{if } n > 1 \\ 7 & \text{if } n \le 1 \end{cases}$$

$$\begin{aligned} T(n) &= \sqrt{n} + 2\sqrt{\frac{n}{10}} + 4\sqrt{\frac{n}{100}} + \cdots + 7 \times 2^{\log_{10} n} \\ &= \sqrt{n}(\frac{2}{\sqrt{10}} + (\frac{2}{\sqrt{10}})^2 + \cdots + (\frac{2}{\sqrt{10}})^{\log_{10} n - 1}) + 7 \times 2^{\log_{10} n} \\ &= \sqrt{n} \times \Theta(1) + 7n^{\log_{10} 2} \\ &\in \Theta(n^{\frac{1}{2}}) \end{aligned}$$

(b) [*8 marks*]

$$T(n) = \begin{cases} 10\,T(n/3) + n^2 & \text{if } n > 1 \\ 1 & \text{if } n \le 1 \end{cases}$$

$$\begin{aligned} T(n) &= n^2 + 10 \times (\frac{n}{3})^2 + 100 \times (\frac{n}{9})^2 + \cdots + 1 \times 10^{\log_3 n} \\ &= n^2(1 + \frac{10}{9} + (\frac{10}{9})^2 + \cdots + (\frac{10}{9})^{\log_3 n - 1}) + 10^{\log_3 n} \\ &= n^2 \times 9((\frac{10}{9})^{\log_3 n} - 1) + n^{\log_3 10} \\ &= 9n^{\log_3 \frac{10}{9} + 2} - 9n^2 + n^{\log_3 10} \\ &= 10n^{\log_3 10} - 9n^2 \\ &\in \Theta(n^{\log_3 10}) \end{aligned}$$

4. [*6 marks*]

(a) Solve part (a) of the previous question by the master method.
$a = 2, b = 10, c = 1/2$. $2 < 10^{1/2}$ by master method $T(n) \in \Theta(\sqrt{n})$

(b) Solve part (b) of the previous question by the master method.
$a = 10, b = 3, c = 2$. $10 > 3^2$ by master method $T(n) \in \Theta(n^{\log_3 10})$