CS 161 **Problem Set 1**
Winter 2020 **Due:** Wednesday, January 15, 2020 at 10am on Gradescope.

Style guide and expectations: Please see the "Homework" part of the "Resources" section on the webpage for guidance on what we are looking for in homework solutions. We will grade according to these standards.

Make sure to look at the "**We are expecting**" blocks below each problem to see what we will be grading for in each problem!

Exercises

Please do the exercises on your own.

---

0. **(1 pt.)** Have you thoroughly read the course policies on the webpage?

   [**We are expecting:** *The answer "yes."*]

1. **(1 pt.)** See the IPython notebook `HW1.ipynb` for Exercise 1. Modify the code to generate a plot that convinces you that $T(x) = O(g(x))$. **Note:** There are instructions for installing Jupyter notebooks in the pre-lecture exercise for Lecture 2.

   [**We are expecting:** *Your choice of c, n_0, the plot that you created after modifying the code in Exercise 1, and a short explanation of why this plot should convince a viewer that $T(x) = O(g(x))$.*]

2. **(3 pt.)** See the IPython notebook `HW1.ipynb` for Exercise 2, parts (a), (b) and (c).

   (a) What is the asymptotic runtime of the function `numOnes(lst)` given in the Python notebook? Give the smallest answer that is correct. (For example, it is true that the runtime is $O(2^n)$, but you can do better).

   [**We are expecting:** *Your answer in the form "The running time of `numOnes(lst)` on a list of size n is $O(\_\_\_)$.", and a short explanation of why this is the case.*]

   (b) Modify the code in `HW1.ipynb` to generate a picture that backs up your claim from Part (a).

   [**We are expecting:** *Your choice of c, n_0, and $g(n)$; the plot that you created after modifying the code in Exercise 2; and a short explanation of why this plot should convince a viewer that the runtime of `numOnes` is what you claimed it was.*]

   (c) How much time do you think it would take to run `numOnes` on an input of size $n = 10^{15}$?

   [**We are expecting:** *Your answer (in whichever unit of time is easiest to interpret) with a short explanation that references the runtime data you generated in part (b). You don't need to do any fancy statistics, just a reasonable back-of-the-envelope calculation.*]

More exercises on next page...

3. **(1 pt.)** Which of the following functions are $O(n^2)$?

   No explanation is required, but you might want to prove your answer to yourself to convince yourself that you are correct.

   (a) $f_1(n) = 5n + 3$

   (b) $f_2(n) = 5n^2 + 3$

   (c) $f_3(n) = 5n^3 + 3$

   (d) $f_4(n) = n \log(n)$

   (e) $f_5(n) = \sin(n) + 5$

   (f) $f_6(n) = 2^n$

   (g) $f_7(n) = 2^{2^{100}}$

   [**We are expecting:** *A list of which functions are $O(n^2)$. No explanation is required and no partial credit will be given.*]

4. **(4 pt.)** Plucky the Pedantic Penguin is trying to prove, from the definition of big-Oh, that $f(n) = O(g(n))$, where $f(n) = 4n$, $g(n) = n^2 - 2n$.

   (a) In the definition of big-Oh, Plucky would really like to take $c = 1$. Give a proof that $f(n) = O(g(n))$ in which "c" is chosen to be 1.

   (b) Plucky has changed their mind: now they don't care what c is, but would like to take $n_0 = 3$. Give a different proof that $f(n) = O(g(n))$ in which "n_0" is chosen to be 3.

   [**We are expecting:** *For both parts, a short but formal proof.*]

5. **(3 pt.)** Prove that 10^n is not $O(2^n)$.

   [**We are expecting:** *A formal proof.*]

More problems on next page...

Problems

You may talk with your fellow CS161-ers about the problems. However:

- Try the problems on your own *before* collaborating.
- Write up your answers yourself, in your own words. You should never share your typed-up solutions with your collaborators.
- If you collaborated, list the names of the students you collaborated with at the beginning of each problem.

---

6. **[Fibonacci] (5 pt.)**

   The Fibonacci sequence $F_0, F_1, ...$, is defined by $F_0 = 0, F_1 = 1$, and $F_n = F_{n-1} + F_{n-2}$ for $n \geq 2$. For example, the first several Fibonacci numbers are $0, 1, 1, 2, 3, 5, 8, \ldots$. Show by induction that

   (a) $F_n = O(2^n)$

   (b) $F_{2n} = \Omega(2^n)$.

   Together, parts (a) and (b) imply that $F_n = 2^{\Theta(n)}$.

   [**We are expecting:** *For each part, a formal proof by induction. Make sure to clearly label your inductive hypothesis, base case, inductive step, and conclusion.*]

More problems on next page...

7. [**Alien Multiplication**] **(10 pt.)** You've been communicating with some aliens, and learning about their arithmetic. Interestingly, they have the same base 10 numerical system, addition, and subtraction, but their system of multiplication is completely different. The symbol they use is $\otimes$, and they've sent you the 1-digit multiplication table that they memorize in grade school. Here are some of the highlights:

$$1 \otimes 1 = 3 \qquad 2 \otimes 3 = 19 \qquad 3 \otimes 5 = 49 \qquad 9 \otimes 9 = 243$$

and so on. After some time, you discover that their multiplication corresponds to the following in human arithmetic:

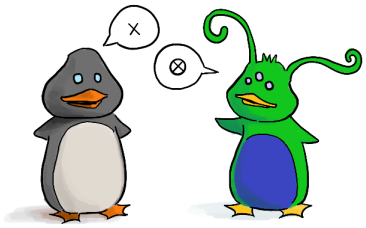
$$x \otimes y = x^2 + xy + y^2.$$

You want to give your alien friends an efficient way to compute the alien product of two n-digit numbers, without referencing or explaining *any* human multiplication to them (even implicitly as a sum). For example, you can't tell them to compute x^2, xy, and y^2 separately since they don't understand what these terms mean.

Find an algorithm to compute the alien product of two n-digit numbers that runs in $O(n^{\log_2 3})$ time. For example, your algorithm might take as input $x = 1234$ and $y = 4321$, and it should return $x \otimes y = 25525911$.

You may assume that n is a power of 2, and that the alien computers can shift digits, add, and subtract just like human computers can. You may also assume that the alien computers have easy access to the $\otimes$-multiplication table you received.

[**We are expecting:** *Pseudocode for your algorithm and a clear English description of what your algorithm is doing and why it is correct. You do not need to prove that your algorithm is correct.*]



More problems on next page...

8. **[Skyline] (13 pt.)** You are handed a scenic black-and-white photo of the skyline of a city. The photo is n-pixels tall and m-pixels wide, and in the photo, buildings appear as black (pixel value 0) and sky background appears as white (pixel value 1). In any column, all the black pixels are below all the white pixels. In this problem, you will design an efficient algorithm that finds the location of a tallest building in the photo.[1]

   The input is an $n \times m$ matrix, where the buildings are represented with 0s, and the sky is represented by 1s. The output is an integer representing the location of a tallest building. For example, for the input 6×5 matrix below, a tallest building has height 5 and is in location 1 (assuming we are 0-indexing). Thus the output is 1.

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & \mathbf{0} & 1 & 1 & 1 \\ 1 & \mathbf{0} & 1 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & 1 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & 1 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}.$$

   (a) **(5 pt.)** Find an algorithm that finds a tallest building in time $O(m \log n)$.
   [**We are expecting:** *Pseudocode, a clear English description of what your algorithm is doing, and a brief justification of the runtime. No proof of correctness is required.*]

   (b) **(5 pt.)** Find an algorithm that finds a tallest building in time $O(m + n)$.
   [**We are expecting:** *Pseudocode, a clear English description of what your algorithm is doing, and a brief justification of the runtime. No proof of correctness is required.*]

   (c) **(3 pt.)** For some values of (n, m) the algorithm from part (a) is more efficient, while for others, the algorithm from part (b) is more efficient. For each of the values of n in terms of m below, determine which of the above algorithms is more efficient (or that they are equally efficient) in terms of big-Oh notation. The case $n = m$ is filled in as an example. If it is helpful, the LaTeXcode for this table is provided at the end of this assignment.

$n = ?$	100	$\sqrt{m}$	$\frac{m}{\log m}$	m	$m \log m$	m^2	2^m
Runtime for (a)				$O(m \log m)$			
Runtime for (b)				$O(m)$			
Which is better?				(b)			

   [**We are expecting:** *For each value of n in terms of m, the best big-Oh runtime you can guarantee for each of your algorithms from (a) and (b), and a conclusion about which is asymptotically more efficient. You do not need to give any formal proofs, but your runtimes should be* ***in the simplest terms possible****.*]

   (d) **(2 BONUS pt.)** Combine your algorithms from parts (a) and (b) to make a better algorithm. That is, for any relationship between m and n, your algorithm should be asymptotically no worse than either your algorithm from (a) or your algorithm from (b), and there should be some settings where it is asymptotically better. You should present your algorithm, its running time, and an example of a function f so that when $n = f(m)$, your algorithm is asymptotically *strictly better* than both $O(n + m)$ and $O(m \log n)$.

   [**We are expecting:** *Nothing, this part is not required. To get the bonus points, you should include pseudocode, the best big-Oh running time for your algorithm that you can come up with, and a function f as described above. All three parts should be accompanied by a clear English description/justification.*]

[1] It could be that there are multiple tallest buildings that all have the same height; in this case, your algorithm should should return any one of them.

Feedback

There's no "correct" answer here, and it is completely anonymous.

9. **(1 pt.)** Please fill out the following poll, which asks about your expectations for the course:

   `https://forms.gle/qWDZLAg3p3JU2xbS6`

   Did you fill out the poll?

   [**We are expecting:** *The answer "yes."*]

LaTeXcode for problem 8(c)

Here is code that generates a table like the one in 8(c). Put your answers between the `&`'s to fill in the table.

```
\begin{center}
\begin{tabular}{c|c|c|c|c|c|c|c}
$n = \ ?$ & 100 & $\sqrt{m}$ & $\frac{m}{\log m}$ & $m$ & $m \log m$ & $m^2$ & $2^m$   \\ \hline
Runtime for (a)  &   &   &   & $O(m \log m)$  &   &   &   \\
Runtime for (b)  &   &   &   & $O(m)$         &   &   &   \\
Which is better? &   &   &   & (b)            &   &   &   \\
\end{tabular}
\end{center}
```

Note: if after filling in the table, it is too wide for the page, you can either use a command like `\footnotesize` before you start the table and `\normalsize` afterwards to put it in a smaller font; or you can edit the code above to break the table up across multiple lines.