1. (8 pt.) In this problem, we’ll utilize two different applications of the definition of big-O to prove that $f(n) = O(g(n))$, where $f(n) = 4n$, $g(n) = 2n^2 - 2n$.
   
   (a) In the definition of big-O, suppose we have to use $c = 1$. Give a proof that $f(n) = O(g(n))$ in which “$c$” is chosen to be 1.
   
   (b) Now instead, suppose we don’t care what $c$ is, but would like to take $n_0 = 2$. Give a different proof that $f(n) = O(g(n))$ in which “$n_0$” is chosen to be 2.
   
   [We are expecting: For both parts, a short but formal proof.]

2. (5 pt.) Prove that $8^n$ is not $O(2^n)$.
   
   [We are expecting: A formal proof.]

3. (6 pt.) Your friend gives you an array $A$ of $n$ integers such that the last element is strictly larger than the first element, i.e. $A[n-1] > A[0]$. Your friend claims that every element is at least as large as the following element, i.e. $A[i] \geq A[i+1]$ for all $i \in \{0, \ldots, n-2\}$.
   
   Prove that your friend’s claim is false.
   
   [We are expecting: A formal proof (using proof by contradiction and proof by induction). In your proof by induction, make sure to clearly label your inductive hypothesis, base case, inductive step, and conclusion.]
4. **Skyline** (20 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 and analyze efficient algorithms 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 \( 6 \times 5 \) matrix below, a tallest building has height 5 and is in location 1 (assuming we are 0-indexing). Thus the output is 1.

\[
\begin{bmatrix}
1 & 1 & 1 & 1 & 1 \\
1 & 0 & 1 & 1 & 1 \\
1 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
\]

(a) (8 pt.) Find an algorithm that finds a tallest building in time \( O(m \log n) \). You may assume that your input is a \( n \times m \) matrix \( A \), and you may access an element in the \( i \)-th row and \( j \)-th column in constant time (via the syntax \( A[i,j] \)).

**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) (8 pt.) Find an algorithm that finds a tallest building in time \( O(m + n) \). You may assume that your input is a \( n \times m \) matrix \( A \), and you may access an element in the \( i \)-th row and \( j \)-th column in constant time (via the syntax \( A[i,j] \)).

**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) (4 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 algorithm runtimes is more efficient (or that they are equally efficient) in terms of big-O notation. The case \( n = m \) is filled in as an example (in blue).

If it is helpful, some tips for formatting the table in \LaTeX{} are provided at the end of this assignment and in the provided \LaTeX{} template.

<table>
<thead>
<tr>
<th>( n = ? )</th>
<th>100</th>
<th>( \sqrt{m} )</th>
<th>( \frac{m}{\log m} )</th>
<th>( m \log m )</th>
<th>( m^2 )</th>
<th>( 2^m )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Runtime for (a)</td>
<td></td>
<td></td>
<td></td>
<td>( O(m \log m) )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Runtime for (b)</td>
<td></td>
<td></td>
<td></td>
<td>( O(m) )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Which is better?</td>
<td></td>
<td></td>
<td></td>
<td>( (b) )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**We are expecting:** For each value of \( n \) in terms of \( m \), the best big-O 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.

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

There’s no “correct” answer here, and it is completely anonymous.

5. (1 pt.) Please fill out the following poll, which asks about why you’re here and how the lectures have been so far:

https://forms.gle/2FhPHiy3S9b2WKeLA

Did you fill out the survey?
[We are expecting: The answer “yes.”]

\LaTeX tips for formatting tables (for problem 4c)

Note: if after filling in the table provided in the Homework 1 Template, 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 provided to break the table up across multiple lines.

More bonus problems on next page...
Bonus Problem

This bonus question is entirely optional and is simply extra practice, so please feel free to completely ignore this section if you’ve got other things on your plate. And remember, these bonus points do not directly boost a specific assignment score. Instead, they may be applied to your overall grade after the curves are determined — please see the course website home page for more details.

6. (2 BONUS pt.) [Fibonacci]

The Fibonacci sequence $F_0, F_1, \ldots$, 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) $F_{2^n} = \Omega(2^n)$.

Together, parts (a) and (b) imply that $F_n = 2^{\Theta(n)}$. (You do not need to prove this part).

[HINT: We want you to combine what we’ve learned about induction proofs with what we’ve learned about Big-O proofs. It will help to structure each of your overall proofs as an induction proof within a Big-O proof.]

[We are expecting: For each of the two parts, a formal proof that involves induction. Make sure to clearly label your inductive hypothesis, base case, inductive step, and conclusion.]