EX1: Algorithmic Analysis I

Due date: Friday April 15, 2022 at 11:59 pm PDT
Latest turn-in date: Monday April 18, 2022 at 11:59 pm PDT

Instructions:
High-level collaboration is allowed, but exercises are to be completed and submitted individually. Submit your responses to the “EX1: Algorithmic Analysis” on Gradescope here: https://www.gradescope.com/courses/379339/assignments/1938580/.
Make sure to log in to your Gradescope account using your UW email to access our course.

Runtime Bound Review:

| $O$: Upper bound on how quickly function grows. | Informally: “Less than or equal to”. |
| $\Omega$: Lower bound on how quickly function grows. | Informally: “Greater than or equal to”. |
| $\Theta$: Tight bound on how quickly function grows. | Informally: “Equal to”. |

1. Asymptotic Analysis: Visually

For each of the following plots, list all possible $\Omega$, $\Theta$, and $O$ bounds from the following choices, not just the tight bounds.

$$ n^2, 1, n, log(n), \frac{1}{n} $$

You do not need to show your work; just list the bounds. If a particular bound doesn’t exist for a given plot, briefly explain why. Assume that the plotted functions continue to follow the same trend shown in the plots as $n$ increases. Each provided bound must either be a constant or a simple polynomial.
2. **Asymptotic Analysis: Conceptual**

The following two statements are true. Explain why each of them are true in 1-2 sentences.

- If a function is in $\Theta(\log N)$, then it could also be in $O(N)$.

- If a function is in $\Omega(N^2)$, then it could also be in $O(N^2 \log N)$.

The following statement is false. Provide a counterexample function that shows why this statement is false.

- If a function is in $\Omega(N^2)$, then it is never in $O(N^2)$.

Congratulations! You got the job offer at Hash Tea. Today marks your first day on the job and your employers help catch you up to speed by giving you a function to complete \( n \) boba orders. Analyze the code snippet below and answer the following questions.

**Note:** the method call to `blend()` is known to take time \( \Theta(\log k) \), where \( k \) is the size of `blend`'s input. The method `blend` outputs `true` or `false` depending on if blending the sugar and `milkTea` made the delicious foam correctly.

```java
boolean makeBobaOrder(int n) {
    int sugar = 2 * n;
    int milkTea = n;
    if (n <= 0) {
        return false;
    }
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < 8; j++) {
            if (!blend(sugar * milkTea)) {
                return false;
            }
        }
    }
    return true;
}
```

For the following questions, express your answer in terms of \( n \) (the input size).

In what case will the `makeBobaOrder()` function be a constant time operation?

What is the worst-case upper bound (\( O \)) of the `makeBobaOrder()` function? *(Note: multiple solutions exist for this question, please provide one)*

What is the worst-case tight bound (\( \Theta \)) of the `makeBobaOrder()` function? *(Hint: In algorithmic analysis, logarithms are in base 2. Recall, \( \log_2(3^2) = 2\log_2(3) \))