Homework 01: AVL trees, algorithm design and analysis

Due date: January 24, 2018 at 11:30 pm

Instructions:

Submit a typed or neatly handwritten scan of your responses on Canvas in PDF format.

Note: you will need to submit a separate PDF per each section.

1. AVL tree rotations

Rather then submitting anything on Canvas, you will complete this exercise using an online tool located here:

`https://grinch.cs.washington.edu/cse373/avl`

The online tool will automatically update your grade on Canvas once you complete the exercise.

2. Simplifying expressions

Submit your answers here: `https://canvas.uw.edu/courses/1124150/assignments/4067297`

(a) Simplify the following summation to produce a closed form. Show your work, clearly stating when you apply each summation identity.

$$\sum_{i=0}^{n-1} \left(\sum_{j=0}^{i-1} j + \sum_{j=0}^{n^2-1} 5i \right)$$

(b) Convert the following recurrence into a summation by applying the unfolding technique discussed in lecture. Then, simplify your summation to find a closed form. You may assume that the initial input n is always > 7.

$$E(n) = \begin{cases} 4 & \text{When } n \leq 7 \\ E(n-1) + n & \text{Otherwise} \end{cases}$$

3. Asymptotic analysis: mathematically

Submit your answers here: `https://canvas.uw.edu/courses/1124150/assignments/4067300`

(a) Show that $6n + n\log(n) \in \Omega(10\log(n))$ is true by finding a c and n_0 that satisfies the definition of "dominates" and big-Ω. Please show your work.

(b) Show that $\log_3(n) \in \mathcal{O}(\log_5(n))$ by finding a c and n_0 that satisfies the definition of "dominated by" and big-$\mathcal{O}$. Please show your work. As a hint, you will need to use the change-of-base logarithm identity somewhere.

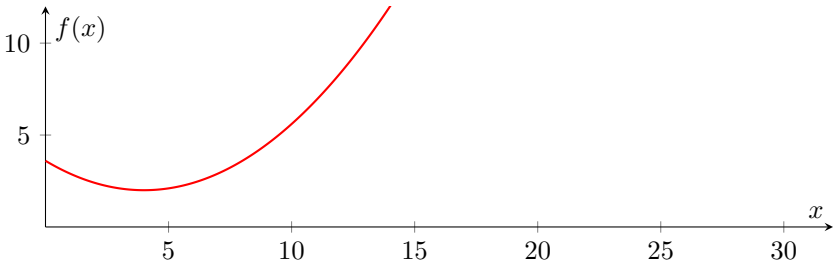
4. Asymptotic analysis: visually

Submit your answers here: `https://canvas.uw.edu/courses/1124150/assignments/4067303`

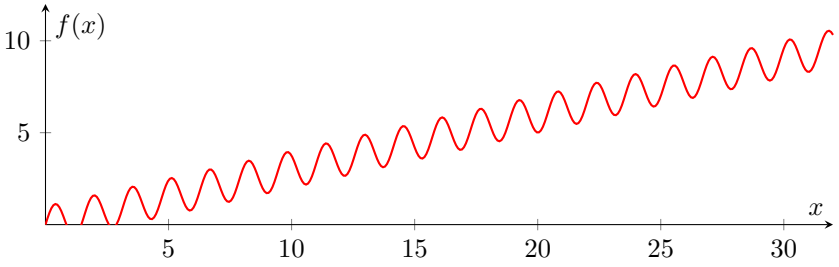
For each of the following plots, provide a tight big-$\mathcal{O}$ bound, a tight big-Ω bound, and a big-Θ bound. You do not need to show your work; just list the bounds. If a particular bound doesn't exist for a given plot, briefly explain why. Each provided bound should either be a constant or a simple polynomial (e.g. $\mathcal{O}(1)$, $\mathcal{O}(n)$, $\mathcal{O}(n^2)$...).

Assume that the plotted functions continue to follow the same trend shown in the plots as x increases.

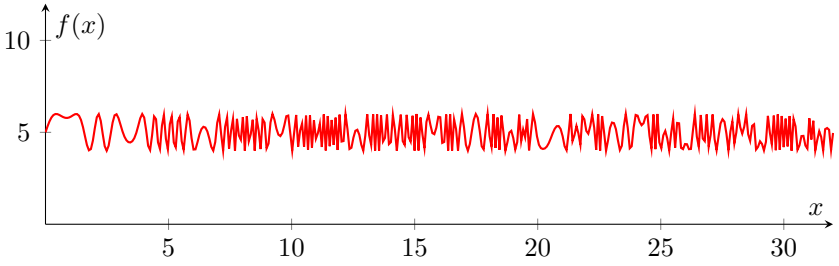
(a)



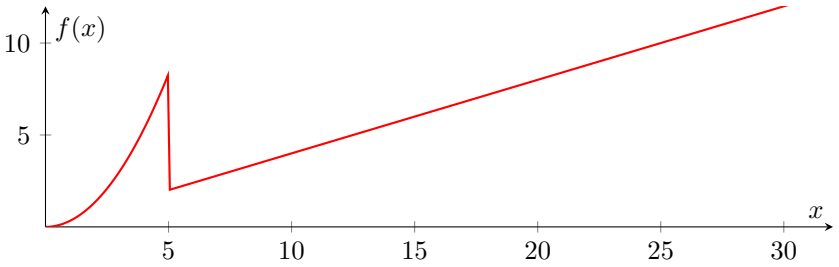
(b)



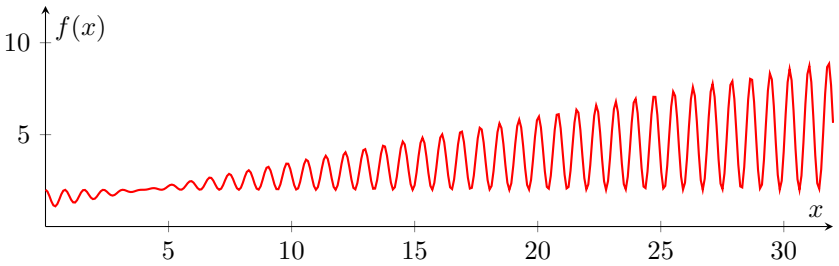
(c)



(d)



(e)



5. Modeling code

Submit your answers here: `https://canvas.uw.edu/courses/1124150/assignments/4067308`

(a) Construct a mathematical function $T_1(n)$ modeling the approximate *worst-case runtime* of the `mystery1` method. Your answer should be written as a summation. You do not need to find the closed form of this summation.

```
public static int mystery1(int n) {
    int out = 0;
    for (int i = 0; i < n; i++) {
        if (i % 5 == 0) {
            for (int j = 0; j < i; j++) {
                out += 2;
            }
        }
    }
    return out;
}
```

(b) Construct a mathematical function $T_2(n)$ modeling the approximate *worst-case runtime* of the `mystery2` method. Your answer should be a recurrence. You do not need to find the closed form of this recurrence.

```
public static int mystery2(int n) {
    if (n == 0) {
        return 3;
    } else {
        return mystery2(n / 2) + n;
    }
}
```

6. Algorithm design: merge

Submit your answers here: `https://canvas.uw.edu/courses/1124150/assignments/4067312`

Suppose we are given two sorted arrays containing comparable elements (such as integers or strings). Our goal is to design an algorithm that returns a new array containing all items from both arrays in sorted order. The input arrays should remain unmodified. The algorithm should throw an exception given invalid input.

For example, suppose we receive as input the arrays `[-5, 0, 0, 2]` and `[-1, 2, 3]`. The output should be the array `[-5, -1, 0, 0, 2, 2, 3]`.

(a) Write an **English description** or **high-level pseudocode** describing an algorithm to perform this task. Note: do **NOT** submit Java code. We want to see a high-level description of the algorithm, not a low-level one.

Please see the following link for more details on what an acceptable response to this question should look like.
`https://courses.cs.washington.edu/courses/cse373/18wi/resources/explaining-algorithms.html`

(b) List at least four distinct kinds of inputs you would try passing into your `merge` algorithm to test it. For each input, also list the expected outcome (assuming the `merge` algorithm was implemented correctly). Be sure to think about different edge cases.

The following link may be a good source of inspiration when coming up with test inputs:
`https://courses.cs.washington.edu/courses/cse373/18wi/resources/tips-on-testing-code.html`

(c) Provide a tight big-Θ bound of the worst-case runtime of your algorithm. Write your answer in terms of n and m, where n is the length of the first input array and m is the length of the second. Briefly justify your answer.