Practice Problem Set 1

This first practice set is based on the material from week 1. There are **no** student-solved problems from the first week

Problem 1:

Using the pseudo-code of *Insertion sort* from class, determine the best-case number of swaps and the worst-case number of swaps when Insertion sort runs on an input array of length n. Repeat for the best-case and worst-case number of comparisons. Using these results, show that the best-case runtime of insertion-sort has the form $T(n) = an + b$ for constants a and b. Do some research to determine the *average-case* runtime of insertion sort.

Problem 2:

Let A be an array of n numbers. Write the pseudo-code for an algorithm that reverses the elements of A. Call the procedure Reverse($A[1,\ldots,n]$). Let $T(n)$ be the worst-case runtime of your algorithm. Find an expression for $T(n)$ and show that this is $O(n)$.

Problem 3:

A sorting algorithm that is similar to Insertion Sort, is **Selection sort** . If you have not seen this algorithm before, I suggest the video

`https://www.youtube.com/watch?v=g-PGLbMth_g`

Let $T(n)$ be the worst-case runtime of Selection sort. Show that $T(n)$ is of the form $an^2 + bn + c$, and that the runtime is $O(n^2)$. Repeat for the best-case runtime. How does the runtime of Selection sort differ from that of Insertion sort?

Problem 4:

Given an input array $A[1,\ldots n]$, write the pseudo-code for an algorithm called RSort($A[1,\ldots,n]$) that sorts the elements of the array A. Your algorithm may *not* use any swaps. You may use comparisons and the Reverse procedure from Problem 2. (Note that you may pass any subarray $A[i,\ldots,j]$ as input to the Reverse procedure).

Problem 5:

You may have already come across another simple sorting algorithm called *Bubble-sort*. Instead of describing the algorithm here, you are asked to do a bit of online research. One great place to start is here:

`https://www.youtube.com/watch?v=lyZQPjUT5B4`

Write the basic pseudo-code for Bubble sort, using comparisons and swaps. Determine the worst-case number of swaps and the worst-case number of comparisons. Repeat for the best-case.

Problem 6:

Suppose we are given an array A of n distinct numbers, such that the second-half of the array is already sorted. Determine the worst-case runtime of Insertion-sort on this array. Determine the big-Oh notation for this runtime.

Problem 7:

Determine the big-Theta notation of the following functions. Prove your result.

- $f(n) = n\log(n) + n\log^2(n) + n^{2.5}$
- $f(n) = n^2\log(n) + n^2$
- $f(n) = n^3 + n^2\log(n^3)$

- $f(n) = \sum_{k=1}^{n}(k+1)$
- $f(n) = n\log_2 n + n\log_4(n)$
- $f(n) = 4^n + (2^n + \log n)(n^2 + 3^n)$
- $f(n) = \log(n^{0.2}) + \log n$
- $f(n) = n^{0.2} + \log(n^8)$
- $f(n) = \sum_{k=1}^{n} kn^2$

Problem 8:

Prove that $f(n) = n^3 + n$ is $\Omega(n^2)$ but *not* $\Omega(n^4)$.

Problem 9:

Prove that $f(n) = n^2 - 3n$ is *not* $O(\log n)$.

Problem 10:

Prove that :

- $f(n) = n^{7/2} + 3n^3 \log n$ is $O(n^4)$.
- $f(n) = n^{7/2} + 3n^3 \log n$ is $O(n^{3.5})$

Explain why it is possible that this function has two different big-oh notations. Which one is "better"?

Problem 11:

Order the following functions by their asymptotic growth (in increasing order):

$$n^n, \quad n\cdot 3^n, \quad 2^n\cdot n^2, \quad 4^n+n, \quad \frac{n^2+1}{n+6}, \quad 6n!, \quad n^2\log n, \quad n(\log n)^2, \quad \sqrt{n^2+\log n}, \quad (\log n^3)$$