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.

Problem 2:

Show that the best-case runtime of insertion sort is $T(n) = an + b$ for constants a and b, and use this result to deduce that the best-case runtime is $O(n)$. Do some research to determine the *average-case* runtime of insertion sort.

Problem 3:

Let A be an array of n numbers. Write the pseudo-code for an algorithm that reverses the elements of A between indices i and j. Call the procedure $\text{Reverse}(A, i, j)$. Let $T(n)$ be the worst-case runtime of your algorithm when run on A between indices 1 and n. Find an expression for $T(n)$ and show that this is $O(n)$.

Problem 4:

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 5:

Given an input array $A[1, \ldots n]$, write the pseudo-code for an algorithm called $\text{RSort}(A, i, j)$ that sorts the elements of the array A between indices i and j. Your algorithm may use comparisons and the Reverse procedure from Problem 3. It may not perform any direct swaps.

Problem 6:

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 (the simple version, not the optimal version), using comparisons and swaps. Determine the worst-case number of swaps and the worst-case number of comparisons. Repeat for the best-case. Justify that the worst-case runtime is $O(n^2)$ and the best-case runtime is $O(n)$.

Problem 7:

An optimal version of Bubble sort is such that the inner for loop iterates over fewer and fewer elements. Write the pseudo-code for a version of Bubble-sort that performs fewer comparisons in the worst-case. Nevertheless, justify why this new version is still $O(n^2)$ in the worst-case.

Problem 8:

Consider the two sorting algorithms below, which each take as input array $A[]$ indexed from s to f.

```
SwapSort1(A, s, f)
    swapped = true
    while (swapped)
        swapped = false
        for i = s to f-2
            if A[i] > A[i+2]
                Swap A[i] and A[i+2]
                swapped = true
```

```
SwapSort2(A, s, f)
    swapped = true
    while (swapped)
        swapped = false
        for i = s to f-2
            if A[i] > A[i+2]
                Swap A[i] and A[i+2]
                swapped = true
    swapped = true
    while (swapped)
        swapped = false
        for i = s to f-1
            if A[i] > A[i+1]
                Swap A[i] and A[i+1]
                swapped = true
```

- Execute SwapSort1 on array $A = [7, 6, 5, 4, 3, 2, 1]$ indexed from $s = 1$ to $f = 7$.
- Which of the above two algorithms is *correct?*. Justify your answer.
- Justify that the worst-case runtime of SwapSort2 is of the form $T(n) = an^2 + bn + c$ for constants a, b, c.

Problem 9:

For each of the following statements, determine if they are true or false, and justify your answer:

- Suppose algorithm A runs in time $O(n^2)$. Does it also run in time $O(n^3)$?
- Suppose algorithm A runs in time $O(n^2)$. Does it also run in time $O(n)$?
- Suppose algorithm A runs in time $O(n^2)$. Does it also run in time $\Theta(n^2)$?
- Suppose algorithm A runs in time $\Omega(n^2)$. Does it also run in time $\Omega(n)$?
- Suppose algorithm A runs in time $\Omega(n^2)$. Does it also run in time $\Omega(n^3)$?

Problem 9:

Let $f(n) = n^2 + \log n + n$.
Determine which of the below are valid for the function $f(n)$, (there may be more than one).

$$O(n^2), O(n^3), O(n), \Theta(n), \Theta(n^2), \Omega(n^2), \Omega(\log n), \Omega(n)$$

Problem 10:

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

- $f(n) = \log(n^2) + \log^2(n) + \sqrt{n}$
- $f(n) = n^2 \log(n) + n(\log n)^2$
- $f(n) = n^3 + n^2 \log(n)$
- $f(n) = \sum_{k=1}^{n} (2k + 1)$

Problem 11: Determine the big-Theta notation of the following functions. Prove your result.

- $f(n) = \log_2 n + \log_3(n)$
- $f(n) = (2^n + n \cdot 2^n)(n^2 + 3^n)$
- $f(n) = \log(n^{0.2}) + \log(n^2)$
- $f(n) = n^{0.2} + \log(n^8)$
- $f(n) = \sum_{k=1}^{n} kn$

Problem 12:

- Prove that $f(n) = n^2 + n$ is $\Omega(n^2)$ and $\Omega(n)$. Which bound is tighter?
- Prove that $f(n) = n^2 - 3n$ is $O(n^3)$ and $O(2^n)$ and $O(n^2)$. Which bound is tighter?

Problem 13:
Order the following functions by their asymptotic growth (in increasing order):

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