Readings

- Lecture Notes Chapter 6: Analyzing Runtime of Code Snippets
- Lecture Notes Chapter 7: Divide & Conquer and Recurrence Relations

Problems: Recurrences

Problem 1

Problem 1 a

\[
T(n) = \begin{cases} 
T(n-1) + n & n \geq 1 \\
1 & \text{otherwise}
\end{cases}
\]

Problem 1 b

Assume \( n \) is a power of 2.

\[
T(n) = \begin{cases} 
2T(\frac{n}{2}) + n^2 & n > 1 \\
1 & \text{otherwise}
\end{cases}
\]

Code Snippets

We can apply our knowledge of Big-O and summations to find the run time of a snippet of our code. Besides recursion, nested iteration is where our code’s efficiency will be bottlenecked. We should consider the loop as a summation, and use our knowledge to simplify it from there. Try starting from the innermost loop with fixed bounds and working outwards.

Problems (Code Snippets)

Problem 2

Problem 2 a

Provide a running time analysis of the following loop:

```java
for (int i = 4; i < n; i = i*i)
    for (int j = 2; j < Math.sqrt(i); j = j+j)
        System.out.println("*" );
```

Problem 2 b

Provide a running time analysis of the following loop. That is, find both Big-O and Big-Ω:

```java
for (int i = 0; i < n; i++)
    for (int j = i; j <= n; j++)
        for (int k = i; k <= j; k++)
            sum++;`
```
Additional Practice Problems

Problem 3
Assume $n$ is a power of 3.

$$T(n) = \begin{cases} \frac{2}{3} + n & n > 1 \\ 1 & \text{otherwise} \end{cases}$$

Problem 4
You are given the following algorithm for Bubble-Sort:

```
Algorithm 1 Bubble Sort
  function Bubble-Sort(A, n)
    for i ← 0 to n - 2 do
      for j ← 0 to n - i - 2 do
          swap(A[j], A[j + 1])
        end if
      end for
    end for
  end function
```

Given some sequence $(a_1, a_2, \ldots, a_n)$ in $A$, we say an inversion has occurred if $a_j < a_i$ for some $i < j$. At each iteration, Bubble-Sort checks the array $A$ for an inversion and performs a swap if it finds one. How many swaps does Bubble-Sort perform in the worst-case and in the average-case?