# DSC 40B - Discussion 02

**Problem 1.**

**a)** Let $f(n) = 12log_2(3^{n^2-2n} + 2^{logn} - 10n^2 - log_3 n)$. Which of the following asymptotic bounds on $f$ is true?

> **Solution:** $\Theta(n^2)$
> This question is meant to develop some intuition for finding asymptotic bounds on tricky functions, so we won't write up the formal proof (although it is possible, just difficult algebraically.) We want to start off by finding a simpler expression which is approximately equal to the expression inside the log for large values of n:
>
> $3^{n^2-2n} + 2^{logn} - 10n^2 - log_3 n \approx 3^{n^2-2n}$ as n grows to infinity, since $3^{n^2-2n}$ is the highest order term. Furthermore, since $n^2$ grows much faster than $-2n$, $3^{n^2-2n} \approx 3^{n^2}$. So, we have $f(n) \approx 12log_2(3^{n^2})$.
>
> Applying log rules, we can bring down the exponent to get $12log_2(3^{n^2}) = n^2 \cdot 12log_2(3)$. Since $12log_2(3)$ is a constant with respect to $n$, we now know that this function is approximately $\theta(n^2)$.

**b)** What is the best case time complexity of the following function?

```
def foo(arr):
''' arr is a sorted array of size n'''
        i = 0
        j = len(arr) - 1

        while i < j:
                current_sum = arr[i] + arr[j]

                if current_sum == 5:
                        return sum(arr)
                elif current_sum < 5:
                        i += 1
                else:
                        j -= 1

        return False
```

> **Solution:** $\Theta(n)$
> In the best case, we find $current\_sum == 5$ in the first iteration.

**Problem 2.**

State the growth of the function below using $\Theta$ notation in the simplest terms possible, and prove your answer by finding constants that satisfy the definition of $\Theta$ notation.

$$f(n) = \frac{n^2 + 2n - 5}{n - 10}$$

**Solution:** We will show that $f(n) = \Theta(n)$.

Recall that $f(n) = \Theta(g(n))$ if there exist positive constants $c_1, c_2$, and $N$ such that

$$c_1 g(n) \leq f(n) \leq c_2 g(n) \quad \text{for all } n \geq N.$$

In this case, we will prove the inequality for $g(n) = n$. There are infinitely many choices of $c_1$, $c_2$, and $N$ that will satisfy the inequality. We will demonstrate such a proof:

Upper Bound: $f(n) \leq c_2 n$

$$\begin{aligned} f(n) &= \frac{n^2 + 2n - 5}{n - 10} \\ &\leq \frac{n^2 + 2n}{n - 10} \quad (\text{since } -5 \leq 0) \\ &= \frac{n(n+2)}{n - 10} \end{aligned}$$

For $n \geq 22$, $n - 10 \geq \frac{n}{2}$ (since $n - 10 \geq 12$ and $n/2 = 11$), so:

$$\begin{aligned} &\leq \frac{n(n+2)}{n/2} = 2(n+2) \\ &\leq 2n + 4 \\ &\leq 3n \quad \text{for all } n \geq 4 \end{aligned}$$

Therefore, we can choose $c_2 = 3$ and $N \geq 22$ for the upper bound.

Lower Bound: $f(n) \geq c_1 n$

$$\begin{aligned} f(n) &= \frac{n^2 + 2n - 5}{n - 10} \\ &\geq \frac{n^2 + 2n - 5}{n} \quad (\text{since } n - 10 \leq n) \\ &= n + 2 - \frac{5}{n} \\ &\geq n + 2 - 1 \quad \text{for } n \geq 5 \quad \left(\text{since } \frac{5}{n} \leq 1\right) \\ &= n + 1 \\ &\geq n \quad \text{for all } n \geq 5 \end{aligned}$$

Therefore, we can choose $c_1 = 1$ and $N \geq 5$ for the lower bound.

Conclusion:

Combining both bounds, we have for $n \geq 22$:

$$n \leq f(n) \leq 3n$$

Hence, $f(n) = \Theta(n)$ with constants $c_1 = 1$, $c_2 = 3$, and $N = 22$.

**Problem 3.**

Consider the algorithm below.

```
def bogosearch(numbers, target):
    """search by randomly guessing. `numbers` is an array of n numbers"""
    n = len(numbers)

    while True:
        # randomly choose a number between 0 and n-1 in constant time
        guess = np.random.randint(n)
        if numbers[guess] == target:
            return guess
```

We will set up the analysis of the expected time complexity of this algorithm.

**a)** What are the cases? How many are there?

> **Solution:** Case $\alpha$ occurs when the target is found on iteration $\alpha$. In principle, the algorithm can run forever, although this is very unlikely (it happens with probability zero). As such, there are infinitely-many cases.

**b)** What is the probability of case $\alpha$?

> **Solution:** $P(\alpha)$ is the probability of guessing wrong $\alpha - 1$ times and guessing right on the $\alpha$th time:
>
> $$(1 - 1/n)^{\alpha-1} \cdot (1/n)$$

**c)** What is the running time in case $\alpha$?

> **Solution:** We perform $\alpha$ iterations in case $\alpha$, each taking constant time, $c$. The total work in case $\alpha$ is therefore $c\alpha$.

**Problem 4.**

Provide a tight theoretical lower bound for the problems given below. Provide justification for your answer.

**a)** Given an array of n numbers, find the sum of the numbers in the array.

> **Solution:** Adding all the elements in the array requires you to visit all the elements at least once. Therefore, the lower bound is $\Omega(n)$.

**b)** Given a sorted array of $n \geq 2$ numbers, find the second largest number in the array.

> **Solution:** As the array is sorted, we just need to check the second last element in the array to get the second largest number in the array. This takes constant time. Therefore, the lower bound is $\Omega(1)$.