DSC 40B - Discussion 03

Problem 1.

Express the followingfunction's rate of growth using Θ-notation, and prove your answer by finding constants which satisfy the definition of Θ-notation. Make sure to show your work by writing out the chain of inequalities to prove each bound.

$$f(n) = \frac{n^2 + 7n - 8}{n - 5}$$

Solution: $f(n) = \Theta(n)$.

We begin with an upper bound.

$$\begin{aligned} f(n) &= \frac{n^2 + 7n - 8}{n - 5} \\ &\leq \frac{n^2 + 7n}{n - 5} \\ &= \frac{n(n + 7)}{n - 5} \end{aligned}$$

Now, $(n + 7)/(n - 5) \leq 2$ for all $n \geq 17$, hence:

$$\leq 2n, \qquad (n \geq 17)$$

Now for a lower bound:

$$\begin{aligned} f(n) &= \frac{n^2 + 7n - 8}{n - 5} \\ &\geq \frac{n^2 - 8}{n - 5} \end{aligned}$$

To get a lower bound, we make this quantity smaller. We can do so by making the denominator bigger by dropping the -5:

$$\begin{aligned} &\geq \frac{n^2 - 8}{n} \\ &\geq \frac{\frac{1}{2}n^2 + \frac{1}{2}n^2 - 8}{n} \end{aligned}$$

Now, $\frac{1}{2}n^2 - 8 \geq 0$ for all $n \geq 4$. Hence:

$$\begin{aligned} &\geq \frac{\frac{1}{2}n^2 + 0}{n}, \qquad (n \geq 4) \\ &= n/2 \end{aligned}$$

So in total, $\frac{n}{2} \leq f(n) \leq 2n$ for all $n \geq 17$.

Problem 2.

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 α occurs when the target is found on iteration α. 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 α?

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

c) What is the running time in case α?

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

Problem 3.

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 $\Omega(1)$.