DSC 40B - Discussion 01

Problem 1.

What is the time complexity of the following functions? State your answer using Θ notation.

a)
```
def foo(n):
    for i in range(n**2 - 2*n + 100):
        j = 0
        while j < n:
            j += 1
```

Solution: $\Theta(n^3)$

b)
```
def foo(n):
    while n > 1:
        n /= 10
        print(n)
```

Solution: $\Theta(\log n)$

c)
```
def foo(n):
    for i in range(n):
        for j in range(i**2): # <-- notice the bound!
            print(i + j)
```

Solution: $\Theta(n^3)$

Problem 2.

Consider the code below:

```
def foo(n):
    i = 1
    while i * i < n:
        i += 1
    return i
```

a) What does `foo(n)` compute, roughly speaking?

Solution: It computes, approximately, $\sqrt{n}$. Of course, `foo` always returns an integer, so the result of the function is usually not exactly correct.
More precisely, `foo` returns the largest integer greater than or equal to $\sqrt{n}$. For example, $\sqrt{5}$ is between 2 and 3; `foo(5)` returns 3.

b) What is the asymptotic time complexity of `foo`?

Solution: $\Theta(\sqrt{n})$

Problem 3.

Let $f(n) = \sum_{p=0}^{n} 3^p$. What is f in Θ notation?

Solution:

General form of a geometric sum $\displaystyle\sum_{p=0}^{n} x^p = \frac{1 - x^{n+1}}{1 - x}$.

Substituting our equation yields $\displaystyle\sum_{p=0}^{n} 3^p = \frac{1 - 3^{n+1}}{1 - 3}$.

Therefore, $\boxed{f(n) = \Theta(3^n)}$ after throwing out the constants.

Problem 4.

Consider the code below where heights is an array of n elements:

```
for i in range(n):
        for j in range(2*i):
                height = heights[i] + heights[j]
```

What is the time complexity of the code?

Solution:

We can see that outer iteration runs n times, for inner iteration:
On outer iter 1, inner body runs 0 times
On outer iter 2, inner body runs 2 times
On outer iter 3, inner body runs 4 times
Hence,
On outer iter α, inner body runs $(2\alpha - 2)$ times

$$\sum_{\alpha=1}^{n} f(\alpha) = \sum_{\alpha=1}^{n} 2 * \alpha - 2$$

$$\sum_{\alpha=1}^{n} 2 * \alpha - 2 = 0 + 2 + 4 + ... + (2n - 4) + (2n - 2)$$

This is an arithmetic series and we know the formula for sum of an arithmetic series is:

$$Sum = \frac{n}{2} * (a_1 + a_n)$$

Where n is the number of terms, a_1 is the first term in the series and a_n is the last term. Therefore sum of the series:

$$Sum = \frac{n}{2} * (0 + 2n - 2) = n(n - 1) = n^2 - n$$

Therefore the time complexity can be given as $\theta(n^2)$.