Charles Harmon

Algorithms and Data Structures 1 (Assignment 1)

1.

a.

```
a) # n > 0

for i in range(n*n): #O(n²)
    # Statements
    for j in range(i, n): # Partially runs hence it becomes a constant
        # Statements
# Statements
```

This would be O(n²) because the outer loop dominates the inner loop. This is because the inner_loop stops at i == n, because having for example range(5,5) is not possible hence it stops if i == n. Even if i == n stop running the outer loop will keep on running until n * n is full filled. Since the outer loop dominates over the inner loop, we will take the time complexity of the outer loop making it O(n²).

b.

```
b) # n, k, i, j > 0

for i in range(n): #O(n)
  for j in range(n): #O(n)
    k = n
    # Statements
    while k > 1: #O(log_2(n))
     k = k / 2
     # Statements
```

This would be $O(n^2 * log_2(n))$ because the outer for loop is O(n) then there is a nested for loop hence making it O(n^2) because we do O(n) * O(n) then the while loop has a complexity of $O(log_2(n))$. Since the while loop is inside the for loop, we have to multiply it making it $O(n^2 * log_2(n))$.

c.

```
c) # m, n > 0, n < m < n²

i = 1
while i < n: #O(log_10(n))
    j = 1
    while j < n: #O(n)
        # Statements
        j += 1
    k = m
    while k > m: #Doesn't run hence ignored
        # Statements
        k -= 1
    i *= 10
i = 1
while i < n: #O(n)
    # Statements
    i += 10
```

The overall complexity of the code is $O(log_{10}(n) * (n) + n)$, the first while loop is $O(log_{10}(n))$because at the "i" will be multiplied by ten hence the log and since it runs up until n it is $O(log_{10}(n))$. The second while loop is inside the first while loop hence we multiply the time complexity with $O(log_{10}(n))$ .The second while loop is O(n) because it j starts with 1 and always adds 1 hence it will go for the full n hence it is O(n). Since k = m, there will never be a case where k > m hence that condition will never be met which means that the while loop will never run. The last while loop is O(n) however since the first loop has a worse complexity, we will use the first while loop time complexity as the worse case scenario hence the time complexity is $O(log_{10}(n) * (n))$

d.

```
d) # a, b, c > 0

if a < b and b < c:
    for i in range(a):  #O(a)
        # Statements
    if c < a:
        for j in range(c):  #Doesn't run hence ignored
            # Statements
    else:
        for k in range(b):  #O(b)
            # Statements
elif a > b and b > c:
    for i in range(c, b):  #O(b)
        # Statements
else:
    for i in range(a, a + 5):  #O(5)
        # Statements
```

The time complexity would be O(b). The if statement has two notable time complexities the for i-loop and the for k-loop. The k-loop would run more times because of the the if statement condition. The condition states that a < b hence the k-loop will have a higher range causing it to run more times. The elif statement also has a O(b) time complexity however it won't exactly be O(b) because it needs to take into account "c" however it can still be considered as O(b). Lastly the else statement has a time complexity of O(5) since it will always run 5 times no matter what because of the a + 5. This means due to the if statement condition a < b, the worse time complexity it can run is O(b).

Charles Harmon

2.

a.

$$T(1) = 1$$
$$T(n) = 100T(n/10) + n^2$$

$$T(n) = 100T\left(\tfrac{n}{10}\right) + n^2$$
$$T(1) = 1$$

Build Solution

$$T(n) = 100T\left(\tfrac{n}{10}\right) + n^2$$
$$= 100\left[100T\left(\tfrac{n}{10^2}\right) + \left(\tfrac{n}{10}\right)^2\right] + n^2$$
$$= 100^2 T\left(\tfrac{n}{10^2}\right) + n^2 + n^2$$
$$= 100^2 T\left(\tfrac{n}{10^2}\right) + n^2 + n^2$$
$$= 100^2\left[100T\left(\tfrac{n}{10^3}\right) + \left(\tfrac{n}{10^2}\right)\right] + n^2 + n^2$$
$$= 100^3 T\left(\tfrac{n}{10^3}\right) + \underbrace{n^2 + n^2 + n^2}_{3 \cdot n^2}$$

Expand Solution

$$T\left(\tfrac{n}{10}\right) = 100T\left(\tfrac{n}{10^2}\right) + \left(\tfrac{n}{10}\right)^2$$

$$T\left(\tfrac{n}{10^2}\right) = 100T\left(\tfrac{n}{10^3}\right) + \left(\tfrac{n}{10^2}\right)^2$$

Pattern: $100^i T\left(\tfrac{n}{10^i}\right) + i(n^2)$

$$\tfrac{n}{10^i} = 1, \quad n = 10^i, \quad \log_{10} n = i$$

$$100^{\log_{10} n} T(1) + \log_{10} n \times n^2$$
$$n^2 + \log_{10} n \times n^2 = O(n^2 \times \log n)$$

b.

Guessing and proofing:

Guess $= n^2 + \log_{10} n \times n^2$ (base case)

Proofing (Induction):

$$100T\left(\frac{n}{10}\right) + n^2 = 100\left[\left(\frac{n}{10}\right)^2 + \log_{10}\left(\frac{n}{10}\right) \cdot \left(\frac{n}{10}\right)^2\right] + n^2$$

$$= n^2 + \left[100 \, \frac{\log_{10}(n) - 1}{100}\right] + n^2$$

$$= n^2 + (\log_{10}(n) - 1)n^2 + n^2$$

$$= n^2 + n^2 \log_{10} n$$

$$n^2 + n^2 \log_{10} n = n^2 + \log_{10} n \times n^2$$

3.

a.

$$T(n) = 8T(n/2) + n^3$$

$$T(n) = 8T\left(\frac{n}{2}\right) + n^3$$

$$\log_2 8 = 3$$

$$3 = 3$$

$f(n)$ has the same growth rate as $n^{\log_2 8}$

Hence case 2:

$\therefore$ Time complexity $= \Theta(n^3 \log n)$

b.

$$T(n) = T(n/2) + n * \log n$$

$T(n) = 1T\left(\frac{n}{2}\right) + n^1 \cdot \log n$

$\log_2 1 = 0$

$1 > 0$

$n^{\log_b a} = n^{\log_2 1} = n^0$

$f(n)$ grows

$f(n) = \Omega(n^{0+\varepsilon}) = \varepsilon = 1$, $\underline{\varepsilon > 0}$

$f(n)$ grows atleast as fast as $n^{0+\varepsilon}$ if constant is 1

Additional Proof: $af\left(\frac{n}{b}\right)^{\varepsilon} = 1\left(\frac{n}{2}\right)^1 = \frac{n}{2^1} = \frac{n}{2} = f_n \cdot \frac{1}{2} \longrightarrow \underline{c = \frac{1}{2} < 1}$

Hence Case 3:

$\therefore$ Time complexity $= \Theta(n^1 \log n)$

c.

$T(n) = 3T(n/3) + \log n$
$T(n) = 3T(\frac{n}{3}) + \log n$ $\log_3 3 = 1$, $f(n) = \log n$ $f(n)$ grows slower than $n^{\log_3 3}$ Hence Case 1: $\therefore$ Time Complexity $= \Theta(n^{\log_3 3})$ $\Theta(n)$