Carson Sager
11 April 2024

Assignment 5 - Time Complexity Analysis

1. What is the time complexity of the following code:

```
int a = 0; // 2
for (i = 0; i < N; i++) { // N
     for (j = N; j > i; j--) { //N -> 1 iterations
          a = a + i + j; // 3
     }
}
```

The i loop iterates through N times. At its worst case, the j loop iterates through N times until its last loop which is only 1 iteration for i = N-1. This would make the inner loop iterate (N/2)(N+1) total times (N + [N-1] + [N-2]+...+2+1). Thus, the time complexity of this case would be* ***O(N^2)***.

2. What is the time complexity of the following code:

```
int a = 0, i = N; //4
while (i > 0) //floor(log2N)+1
     a += i; //2
     i /= 2; //2
}
```

Since the while loop iterates floor($\log_2 N$) + 1 times, the log term dominates, thus causing the time complexity of the code to be ***O(log(N))***.

3. Two loops in a row:

```
for (i = 0; i < A; i++) { // A iterations
     sequence of statements
}
for (j = 0; j < B; j++) { // B iterations
     sequence of statements
}
```

Since these loops are not nested in any way, the time complexity will simply be the addition of end indices of the two loops. Thus the complexity will be ***O(A + B)***.

How would the complexity change if the second loop went to A instead of B?

In this case, the complexity would still be addition, O(A + A). This, however, results in O(2A) which is O(A). Thus our complexity would now be ***O(A)****.*

4. What is the time complexity of the following code?

```
int n = 1000; //2 operations
System.out.println("Your input is: " + n); //constant operations
```

Since both lines only take a constant amount of operations, the time complexity can be reduced to ***O(1)*** *regardless of input size.*