Tips on Analyzing the Time Complexity of Algorithms that use Nested Loops

If you have nested loops, and the outer loop iterates $i$ times and the inner loop iterates $j$ times, the statements inside the inner loop will be executed a total of $i \times j$ times. This is because the inner loop will iterate $j$ times for each of the $i$ iterations of the outer loop.

This means that if both the outer and inner loop are dependent on the problem size $n$, the statements in the inner loop will be executed $O(n^2)$ times:

```java
for ( int i = 0; i < n; i++ ) {
    for ( int j = 0; j < n; j++ ) {
        // these statements are executed $O(n^2)$ times
    }
}
for ( int i = 0; i < n / 2; i++ ) {
    for ( int j = 0; j < n / 3; j++ ) {
        // these statements are also executed $O(n^2)$ times
        // since both loops loop $O(n)$ times, and
        // $O(n) \times O(n) = O(n^2)$
    }
}
```

Similarly, if you have triply-nested loops, all of which are dependent on the problem size $n$, the statements in the innermost loop will be executed $O(n^3)$ times:

```java
for ( int i = 0; i < n; i++ ) {
    for ( int j = 0; j < n; j++ ) {
        for ( int k = 0; k < n; k++ ) {
            // these statements are executed $O(n^3)$ times
        }
    }
}
```

However, imagine a case with doubly-nested loops where only the outer loop is dependent on the problem size $n$, and the inner loop always executes a constant number of times, say 3 times:

```java
for ( int i = 0; i < n; i++ ) {
    for ( int j = 0; j < 3; j++ ) {
        // these statements are executed $O(n)$ times
    }
}
```

In this particular case, the inner loop will execute exactly 3 times for each of the $n$ iterations of the outer loop, and so the total number of times the statements in the innermost loop will be executed is $3n$ or $O(n)$ times, not $O(n^2)$ times.
Now, imagine a third case: you have doubly nested loops, and the outer loop is dependent on the problem size \( n \), but the inner loop is dependent on the current value of the index variable of the outer loop:

```java
for ( int i = 0; i < n; i++ ) {
    for ( int j = 0; j < i; j++ ) {
        // these statements are executed \( O(n^2) \) times
    }
}
```

Let's analyze this case iteration-by-iteration:
- On the 1st iteration of the outer loop \( (i = 0) \), the inner loop will iterate 0 times.
- On the 2nd iteration of the outer loop \( (i = 1) \), the inner loop will iterate 1 time.
- On the 3rd iteration of the outer loop \( (i = 2) \), the inner loop will iterate 2 times.
  .
- On the FINAL iteration of the outer loop \( (i = n - 1) \), the inner loop will iterate \( n - 1 \) times.

So, the total number of times the statements in the inner loop will be executed will be equal to the sum of the integers from 1 to \( n - 1 \), which is:

\[
((n - 1)\times n) / 2 = n^2/2 - n/2 = O(n^2) \text{ times}
\]