Simple examples on analysis of algorithms

Q. Determine how many times the output statement is displayed in each of the following program segments in terms of n. Present your answer using “big oh” notation.

(a) for (int i = 0; i < n; i++)
    for (int j = n - 1; j >= i; j--)
        cout << i << " " << j << endl;

Method of attack: Start from the innermost loop and proceed inside out. It is clear that the inner loop index \( j \) varies from \( n - 1 \) downto \( i \), so the number of times this loop is executed is equal to \( (n - 1) - i + 1 = (n - i) \). Next, the outer loop index \( i \) varies from 0 to \( n - 1 \). Therefore, the number of times the output statement is displayed is given by:

\[
\sum_{i=0}^{n-1} (n-i) = n + (n-1) + \cdots + 1 = \frac{n(n+1)}{2} = \frac{n^2}{2} + \frac{n}{2}
\]

that is \( O(n^2) \).

Here is another way of looking at the same thing:

<table>
<thead>
<tr>
<th>When the outer loop index is equal to</th>
<th>The inner loop index assumes values</th>
<th>Number of times the inner loop executes is equal to</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>( n-1 )</td>
<td>( n )</td>
</tr>
<tr>
<td></td>
<td>( n-2 )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( \vdots )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>( n-1 )</td>
<td>( n-1 )</td>
</tr>
<tr>
<td></td>
<td>( n-2 )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( \vdots )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
</tr>
<tr>
<td>( n-2 )</td>
<td>( n-1 )</td>
<td>2</td>
</tr>
<tr>
<td>( n-1 )</td>
<td>( n-1 )</td>
<td>1</td>
</tr>
</tbody>
</table>
It follows that the number of times the output statement is displayed is given by

\[ n + (n - 1) + \ldots + 2 + 1 = \frac{n(n+1)}{2} = \frac{n^2}{2} + \frac{n}{2} \]

that is \( O(n^2) \).

(b) 

```cpp
for (int i = 0; i < n; i++)
    for (int j = 0; j < 2; j++)
        cout << i << "  " << j << endl;
```

Note that the outer loop index \( i \) varies from 0 through \( n-1 \). Further, for each \( i \), the inner loop is executed twice. Therefore, the number of times the output statement is displayed is given by \( 2n \) that is \( O(n) \).

(c) 

```cpp
for (int i = 1; i < n; i++)
    for (int j = 0; j < i; j++)
        if (j % i == 0)
            cout << i << "  " << j << endl;
```

Since \( j \) is always smaller than \( i \), it is clear that \( j \% i \) is equal to \( j \). Therefore, \( (j \% i == 0) \) will be true exactly when \( j = 0 \). Now \( j \) is equal to 0 exactly once for each iteration of the outer loop. Therefore, "cout" statement will be executed \( n - 1 \) times that is \( O(n) \).

(d) 

```cpp
for (int i = 1; i < n; i = i*2)
    cout << i << "  " << n << endl;
```

As far as count in “big Oh” notation is concerned, there is no loss of generality in assuming that \( n = 2^k \), where \( k \geq 1 \).

The loop index \( i \) gets doubled after each iteration. The termination condition (i.e., \( i < n \)) is such that index \( i \) will assume values in the following sequence: \( 1, 2, 2^2, \ldots, 2^{k-1} \). Therefore, the number of times the output statement will be displayed is equal to \( k = \log_2 n \) that is \( O(\log_2 n) \).