Time Complexity Analysis - 2

1. Code Snippet 1:

```
for(int i = 0; i < n; i++) {
  for(int j = 0; j * j < n; j++) {
    cout << “PhysicsWallah”;
  }
}
```

Time Complexity: O(n * sqrt(n))
Explanation: The outer loop runs n times, and the inner loop runs until j^2 < n, which means it runs approximately sqrt(n) times.

2. Code Snippet 2:

```
int c = 0;
for(int i = 0; i < n; i++) {
  for(int j = 1; j < n; j *= 2) {
    c++;
  }
}
```

Time Complexity: O(n * log n)
Explanation: The outer loop runs n times, and the inner loop is logarithmic because j is multiplied by 2 in each iteration.

3. Code Snippet 3:

```
int c = 0;
for(int i = 0; i < n; i++) {
  for(int j = 1; j * j < n; j *= 2) {
    c++;
  }
}
```

Time Complexity: O(n * log log n)
Explanation: The outer loop runs n times, and the inner loop is a logarithmic loop with a condition based on j^2, leading to log log n iterations.

4. Code Snippet 4:

```
int c = 0;
for(int i = n; i > 0; i /= 2) {
  for(int j = 0; j < i; j++) {
    c++;
  }
}
```

Time Complexity: O(n)
Explanation: The outer loop divides i by 2, running log n times. The inner loop runs i times in each iteration, forming a sum that simplifies to O(n).

5. Code Snippet 5:

```
int c = 0;
for(int i = 1; i < n; i *= 2) {
  for(int j = n; j > i; j--) {
    c++;
  }
}
```

Time Complexity: O(n)
Explanation: The outer loop runs log n times as i is doubled. The inner loop runs n - i times, but the overall complexity still simplifies to O(n).