1.4.6 Give the order of growth (as a function of N) of the running times of each of the following code fragments:

a.
```
int sum = 0;
for (int n = N; n > 0; n /= 2)
  for(int i = 0; i < n; i++)
     sum++;
```

b.
```
int sum = 0;
for (int i = 1 i < N; i *= 2)
    for (int j = 0; j < i; j++)
        sum++;
```

c.
```
int sum = 0;
for (int i = 1 i < N; i *= 2)
   for (int j = 0; j < N; j++)
       sum++;
```

Soln

1. $O(N)$. For part a, notice how the inner loop dominates because a very large value of N, say inf, the inner loop runtime would be $O(N)$. The outter loop executes that n = N, N/2, N/4, N/8, . . . , 1. By the first case, it is N which is the largest it will ever be, so it is $O(N)$ order of growth.

2. $O(N)$.Note that the times the inner loop is executed are the same as in the previous question (only in reverse order). So here, the last case would result in $O(N)$ growth.

3. $NlogN$. Pay close attention to the 2nd loop. It is $j < N$ not $j < i$. This means that for every run, $O(n)$ complexity for inner loop. However, the outter loop is executed $logN$ times since the number of indices seen is halved each time. Therefore, the complexity is $O(nlogn)$