Algorithms & Complexity 1/25/2017 – 1/27/2017

0145-344-001

Note Taker: Jai Punjwani

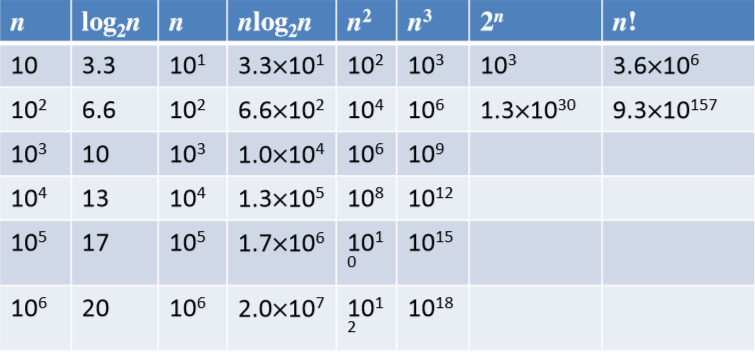
Topic: Algorithm Analysis and Big-OH

PowerPoint: <http://home.adelphi.edu/~siegfried/cs344/344l2.pdf>

NOTES

Analyzing algorithms are about **runtime vs. space (memory)**. Algorithm analysis is all about scalability – we care about how our programs perform as the data they work with gets bigger and bigger.

Below is a useful table showing the growth of several important functions in mathematics:



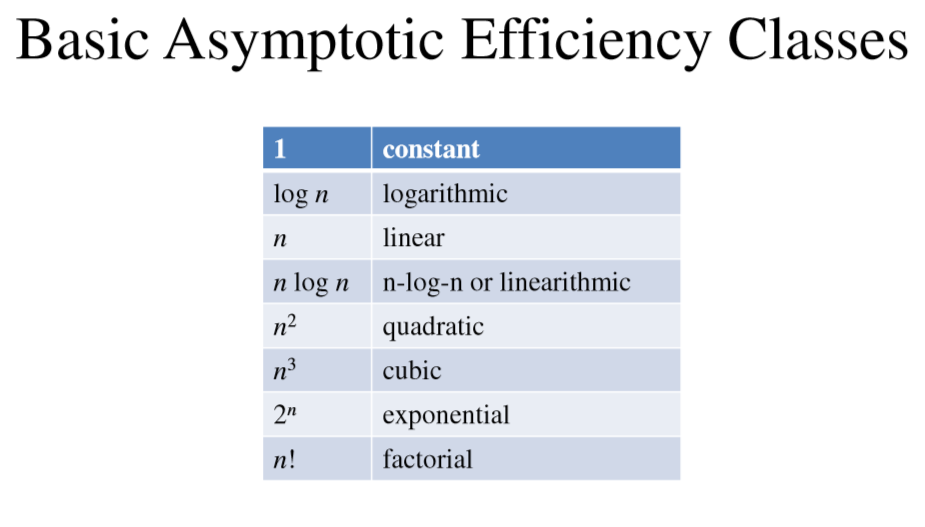
Our input size, **n**, increases as we go down our first column.

As we go down each of the columns, from left to right, we see that the functions on the right increase significantly quicker than the functions on the left.

**Asymptotic Analysis**

1. Lower bound: Big ***Ω*** (Omega):
2. (Upper Bound = Lower Bound): Big ***Θ*** (Theta)
3. **Upper** **bound:** Big ***Ο*** (O)

**Generally, the most important analysis is Big-O** **because we care about how bad things can get with large amounts of data.**



Examples:

for (int i =0; i< n; i++) {

// some simple operation

}

This has order **O**(n) because our program is directly dependent on the size of n.

for (int i =0; i< N; i++) {

for (int i =0; i < N; i++) {

// some simple operation

}

}

This has order **O**(n2) because our inner and outer for-loops run n times each (n x n). Note that in real-life algorithms, **O**(n2) is usually considered bad.