CS 3313
Foundations of Computing:

Lab 9 – Asymptotic Notation

Time Complexity Background

In programming, we want to minimize the time it takes for our algorithms to run

By reducing the number of operations we need to compute, we see dramatic decreases in run-time

- This is *essential!* We want things ASAP!

For example, consider the following:

n	n^2	n^3
1	1	1
10	100	1,000
100	10,000	1,000,000
1,000	1,000,000	1,000,000,000

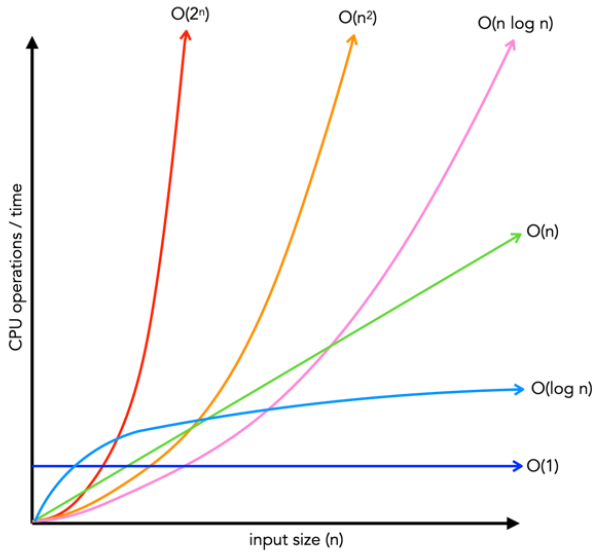
Asymptotic Notation

When we compare programs, we look at them *asymptotically*

This is because we are concerned with the *growth in time* of our functions as our value *n* (the number of elements we have) increases

We saw an example of this on the previous slide, where a higher power resulted in a higher growth rate

- We can demonstrate this visually as well



Note: $O(1) < O(\log n) < O(n) < O(n \log n) < O(n^2) < O(n^3) < O(2^n)$

Big-O Notation

Big-O Notation helps us describe how long an algorithm takes by setting an *upper bound*

For example, if we take two functions, *f(n)* and *g(n)*, we can say:

- *f(n) = O(g(n))* if and only if there exists constants $c > 0$ and $n_0 \geq 1$ s.t.

$$f(n) \leq c * g(n) \text{ for all } n \geq n_0$$

True or False:

- $5n + 3 = O(n)$?
- $n^2 + 5n + 3 = O(n)$?
- $n^2 + 5n + 3 = O(n^2)$?
- $n^2 + 5n + 3 = O(n^3)$?
- $3^n = O(3^{n+1})$?

Big-O Notation

Now that we have defined what Big-O means, how can we show that this holds true as n increases?

The trick to this is through the induction proof technique

Big-O Notation: Induction Proof

Let's do the following example from the previous slide:

- $n^2 + 5n + 3 = O(n^2)$
- $n^2 + 5n + 3 \leq cn^2$ Now, we pick values for c and n_0
- $n^2 + 5n + 3 \leq 9n^2 \; \forall \; n \geq n_0 = 1$
- Base case: $n = 1$ → $(1)^2 + 5(1) + 3 \leq 9(1)^2$
  → $9 \leq 9$
- Induction Hypothesis: Assume that $k^2 + 5k + 3 \leq 9k^2$ holds for a value $n=k$
- Induction: Demonstrate that the equality holds for $k + 1$:

$(k + 1)^2 + 5(k + 1) + 3 \leq 9(k + 1)^2$

$(k^2+2k + 1) + (5k + 5) + 3 \leq 9(k^2 + 2k + 1)$

$k^2 + 7k + 9 \leq 9k^2 + 18k + 9$

$0 \leq 8k^2 + 11k$

Therefore, this is true, as we know $k \geq n_0 = 1$

Exercise 1:

$L_1 = \{ww^r \mid w \in \{\mathrm{a}, \mathrm{b}\}^*\}$

- Give a 1-tape TM solution to solve this problem in $O(n^2)$ time
- Can we improve the time by having a 2-tape TM? If so, give a 2-tape TM solution and describe its time complexity

Big-Omega and Big-Theta (Big-Ω and Big-θ)

Big-Ω denotes the following relationship between functions *f(n)* and *g(n)*:

- $f(n) = \Omega(g(n))$ if and only if there exists constants $c > 0$ and $n_0 \geq 1$ s.t.

$$f(n) \geq c * g(n) \text{ for all } n \geq n_0$$

- For example, $3n^2 = \Omega(n)$

Big-θ denotes the following relationship between functions *f(n)* and *g(n)*:

- $f(n) = \theta(g(n))$ if and only if there exists constants $c_1, c_2 > 0$ and $n_0 \geq 1$ s.t.

$$c_1 * g(n) \leq f(n) \leq c_2 * g(n) \text{ for all } n \geq n_0$$

- For example, $3n = \theta(n)$

Transformations Between Notations

Here, we see that these relationships are connected

- For example, if $f(n) = O(g(n))$, then $g(n) = \Omega(f(n))$
  - Why is this?

- Additionally, if $f(n) = O(g(n))$ and simultaneously $f(n) = \Omega(g(n))$, then $f(n) = \theta(g(n))$
  - Why is this?

Exercise 2:

Prove or disprove the following:

- $\frac{n(n+1)}{2} = \Omega(n^2)$
- $\frac{n(n+1)}{2} = \theta(n^2)$
- $2^{2n} = O(2^n)$