Design and Analysis of Algorithms

Lecture - 4

Success is always inevitable with Hard Work and Perseverance

N. Ravitha Rajalakshmi

Learning Objective

- How to express running time using Asymptotic Notations?
- Different Notations
- Problems with Asymptotic Notations

Story So far - Finding the efficient algorithm

$$T(n) = ?$$

$$T(n) = c_{op} \cdot C(n)$$

$$T(n) \approx C(n)$$

Need for Asymptotic Notation

- Compare algorithm A1 which has T(n) as $3n^2 + 8n + 6$ and algorithm A2 which has T(n) as $8n^3 + 9$?

  Difficult. We can approximate it to one of the known functions

- Compare algorithm A1 which has T(n) as $n!$ and algorithm A2 which has T(n) as n^2 ?

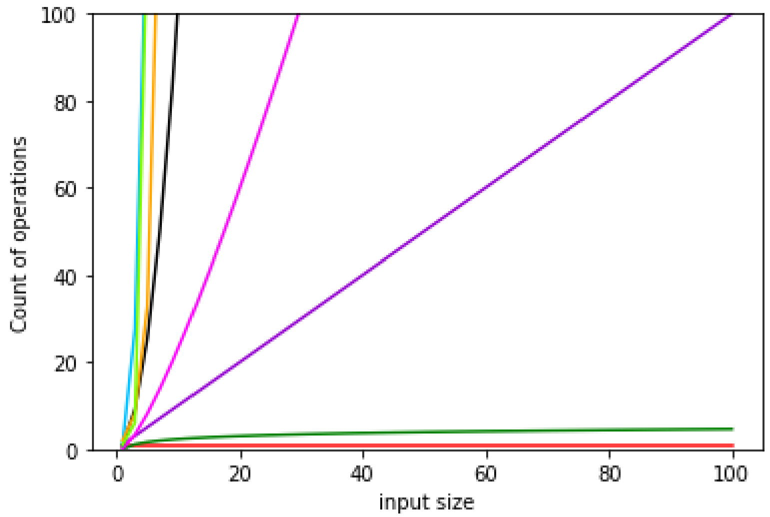
  Easier. As the order of growth is already known

T(n) ~= g(n)

- Approximate these with ordinary functions whose order of growth is already known

$$1 < \log n < n < n\,log n < n^2 < n^3 < 2^n < n!$$

T(n) = ? (g(n))



Larger inputs

Asymptotic Notations

- To rank and compare order of growth
- O (Big Oh), θ (Big theta), Ω (Big omega)

$f, g : \mathbb{N} \rightarrow \mathbb{R}^+$
c – positive constant
N- non negative integer

Definition O (Big Oh) Notation

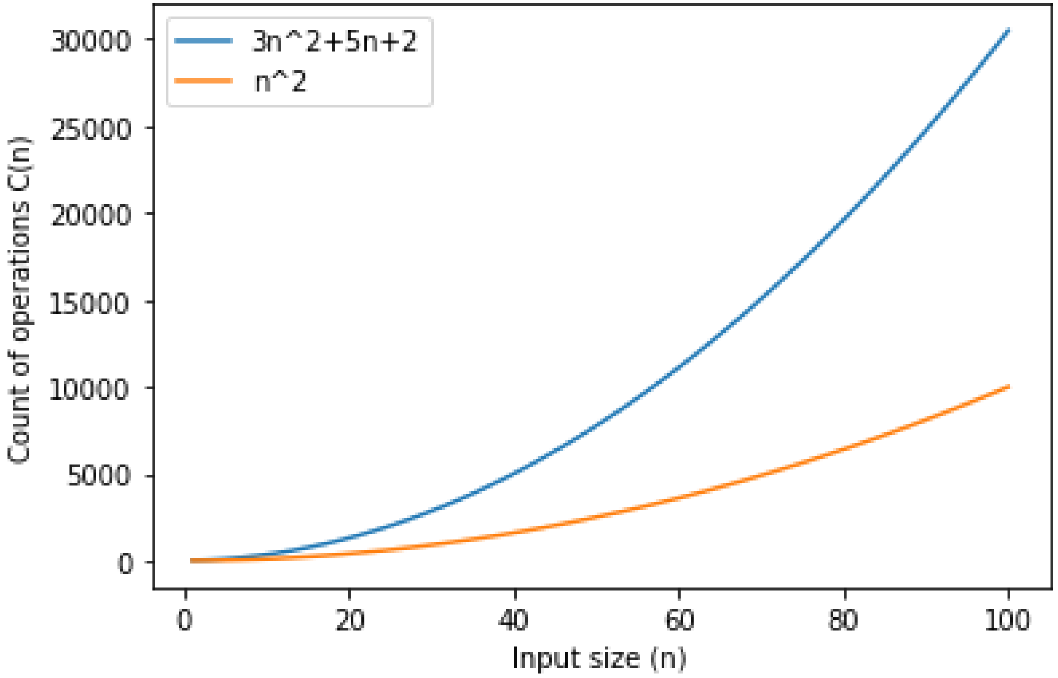
$f(n) = O(g(n))$ (f is Big-O of g) or $f \precsim g$ if there exist constants N and c so that for all $n \geq N$, $f(n) \leq c \cdot g(n)$.

Larger inputs

f is bounded by g

Asymptotic Notation - Big Oh

- Actual Runtime f(n) = $3n^2 + 5n + 2$
- Can we represent f (n) using a function g(n) = n^2



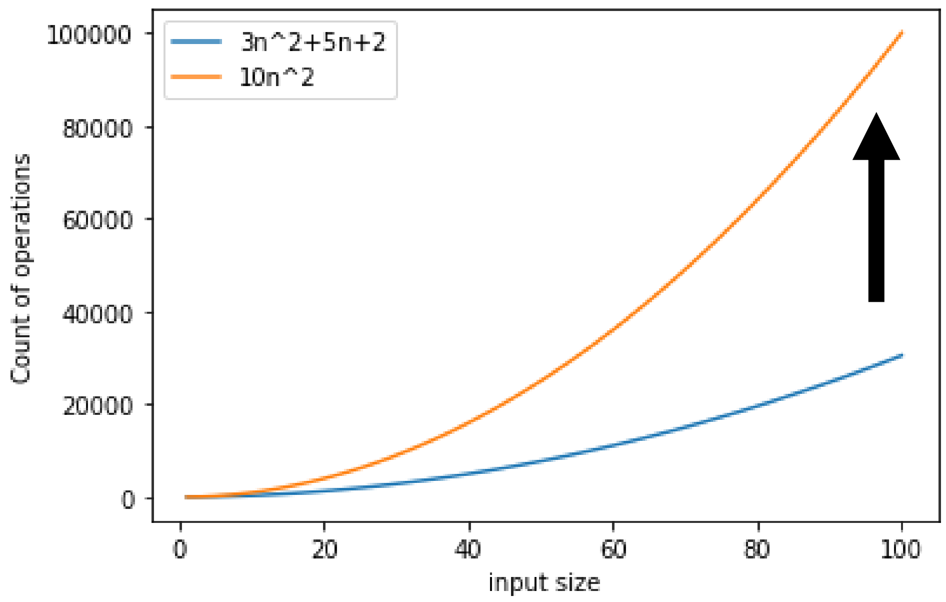
- $3n^2 + 5n + 2$= **$O(n^2)$**

- What will be the value of N and c?

$\mathrm{N} = 1 \;\; \mathrm{c} = 2 \quad 3n^2 + 5n + 2 \;= 10 \quad n^2 = 1$

Asymptotic Notation - Big Oh

- For c = 10 and N= 1?



Any function whose order of growth is greater than T(n)

Common Rules

- Multiplicative constants can be omitted $C \cdot f \leq f$

$$7n^3 = O(n^3) \qquad \frac{n^2}{3} = O(n^2)$$

- Out of two polynomials, the one with larger degree grows faster

  $n^a < n^b, 0 < a < b$

$$n = O(n^2) \qquad \sqrt{n} = O(n)$$

- Any polynomial is slower than any exponential: $n^a < b^n \;\; (a > 0, b > 1)$

$$n^5 = 0\left(\sqrt{2}^n\right)$$

Common Rules

- Any polylogarithm grows slower than any polynomial $(\log n)^a < n^b$ *(a,b >0)*

$$(\log n)^3 = O(\sqrt{n}) \quad , \qquad n \log n = O(n^2)$$

- Smaller terms can be omitted. if $f < g$ then $f + g < g$

$$n^2 + n = O(n^2)\,, \quad 2^n + n^9 = 0(2^n)$$

Big Ω Notation

$f, g : \mathbb{N} \rightarrow \mathbb{R}^+$
c – positive constant
N- non negative integer

Definition (Big Omega) Notation

$f(n) = \Omega(g(n))$ (f is Big-Omega of g) or $f \succeq g$ if there exist constants N and c so that for all $n \geq N$, $f(n) \geq c \cdot g(n)$.

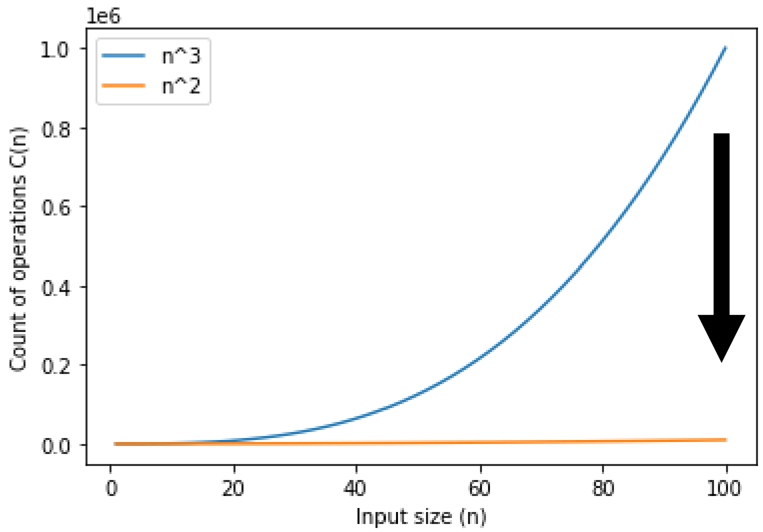
Larger inputs

f grows no slower than g

Big Ω Notation

- $f(n) = n^3$
- $g(n) = n^2$

$f(n) = \Omega(n^2)$



Any function whose order of growth is less than T(n)

Big θ Notation

$f, g : \mathbb{N} \rightarrow \mathbb{R}^{+}$
c1, c2 – positive constant
N- non negative integer

Definition (Big theta) Notation

f (n) = θ(g(n)) (f is Big-theta of g) or f ~g if **f =** $\boldsymbol{O(g)}$ and **f =** $\boldsymbol{\Omega(g)}$. The graph is bounded by two constants for all n≥ N

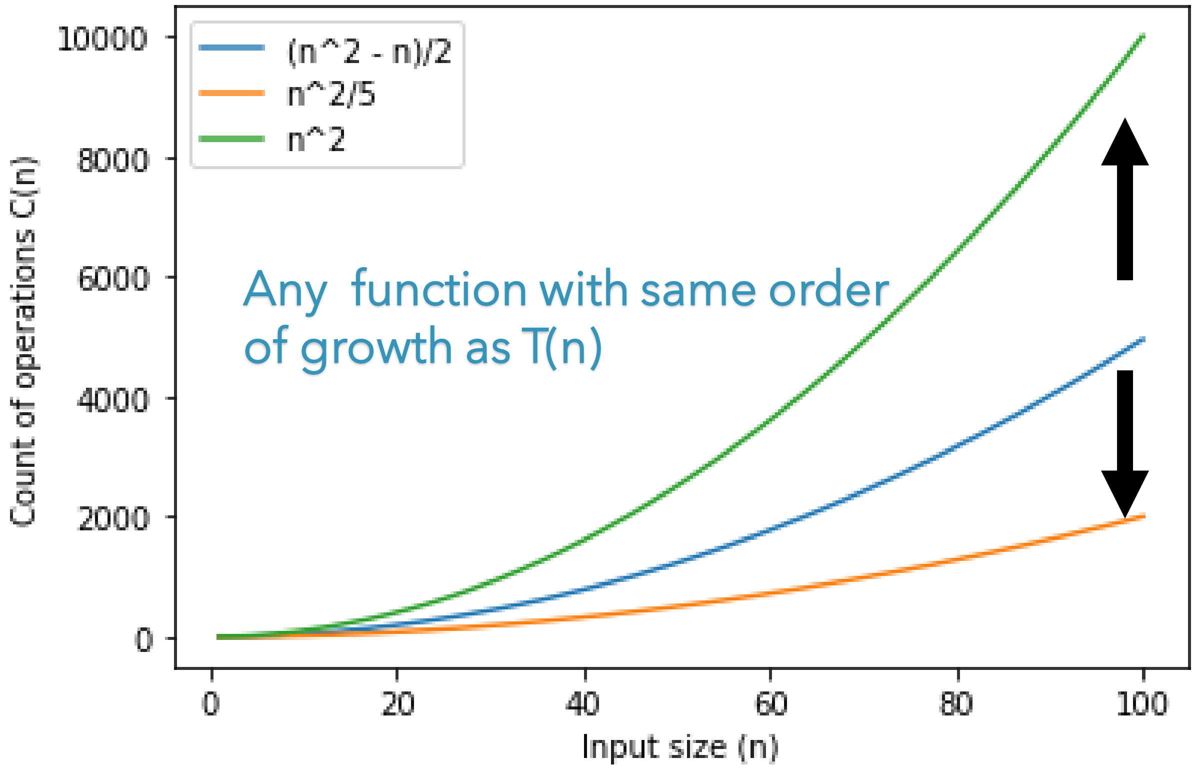
$$c1.g(n) \leq f(n) \leq c2.g(n)$$

f grows at the same rate as g

Big θ Notation

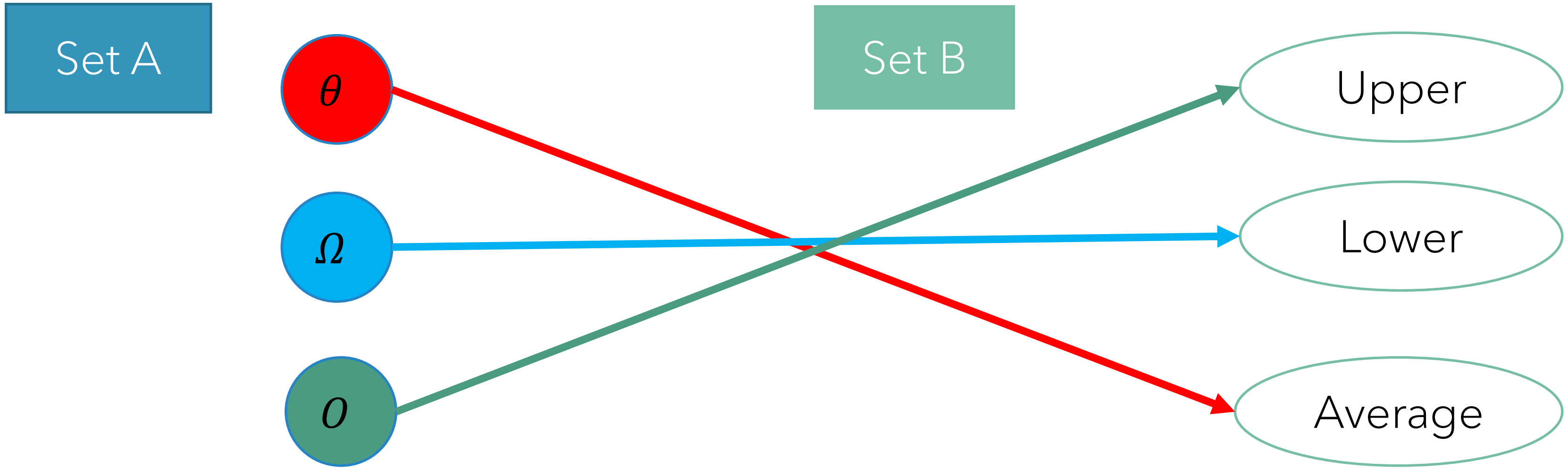
- $f(n) = \frac{n^2 - n}{2}$
- $g(n) = n^2$

$f(n) = \theta(n^2)$



Pause & Think

1. Match the sets A (Notations) and B (Bounds)



Other Notations

- little oh (o) Notation and little omega (ω)
- Not asymptotically tight

$f(n) = o\big(g(n)\big)$ $f(n) < g(n)$ order of growth of f(n) is strictly lower

$f(n) = \omega\big(g(n)\big)$ $f(n) > g(n)$ order of growth of f(n) is strictly greater

- Big Notations – exact order of growth

Using limits for comparison

- L 'Hôpital's rule

$$\lim_{n\to\alpha} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{f'(n)}{g'(n)}$$

$$\mathbf{f(n) = {}^{1}/_{2}\, n(n-1)}$$
$$\mathbf{g(n) = n^2}$$

$$\lim_{n\to\infty} \left[\frac{{}^{1}/_{2}n(n-1)}{n^2}\right] = {}^{1}/_{2} \lim_{n\to\infty} \frac{\mathrm{n}-1}{n}$$

$= ½$

$$f(n) = \theta(g(n))$$

Notation	Comparison	Limit Definition
Little o Notation $f(n) \in o(g(n))$	$f(n) < g(n)$	$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$
Big O Notation $f(n) \in O(g(n))$	$f(n) \leq g(n)$	$\lim_{n\to\infty} \frac{f(n)}{g(n)} < \infty$
Big theta Notation $f(n) \in \theta(g(n))$	$f(n) = g(n)$	$\lim_{n\to\infty} \frac{f(n)}{g(n)} \in \mathbb{R} > 0$
Big Omega Notation $f(n) \in \Omega(g(n))$	$f(n) \geq g(n)$	$\lim_{n\to\infty} \frac{f(n)}{g(n)} > 0$
Little omega Notation $f(n) \in \omega(g(n))$	$f(n) > g(n)$	$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$

Pause & Think

- Arrange the functions based on their order of growth from lowest to the highest

$(n-2)!$
$5\log(n+100)^{10}$
2^{2n}
$\sqrt[3]{n}$
3^n
$\ln^2 n$
$0.001n^4 + 3n^3 + 1$

$5\log(n+100)^{10}$
$\sqrt[3]{n}$
$\ln^2 n$
$0.001n^4 + 3n^3 + 1$
2^{2n}
3^n
$(n-2)!$

Guidelines for Asymptotic Analysis of Non Recursive Algorithm

- Loops
  - Number of iterations * amount of time taken by basic operation

```
for (int i = 0; i<=n; i++)
        m = m+ 2
```

T(n) = C(n) = O(n)

- Nested Loops
  - Product of size of all the loops

```
for(int i=0; i<=n; i++)
        for (int j=0; j<=n; j=j*2)
                x = x*2
```

T(n) = C(n) = O(n log n)

- Consecutive Statements
  - Add the count of operations

```
for (int i=0; i<=n; i++)
        arr[i] = 0;
for (int j=0;j<=n;j++)
        for(int k=0; k<=n;k++)
            if(arr[j] ==arr[k])
                count+=1;
```

n

n^2

T(n) = C1(n) + C2(n)
= n + n^2 = O(n^2)

- If then statements
  - Use the complexity of if (or) else part (whichever is larger)

Issues with Asymptotic Notations

- Though it produces clean representation
- Still a lot of practically useful information is lost

$$\left.\begin{array}{l} 3n^2 + 5 \\ \\ 9n^2 + 8n + 10 \end{array}\right\} O(n^2)$$

Summary

- Various Notations used for comparing the functions
- How Asymptotic Notation simplify the representation of running time
- Issues with Asymptotic Notations

Thank You
Happy Learning

Success is always inevitable with Hard Work and Perseverance