As mentioned last time in class, when analyzing the running time of the algorithm, we are interested in counting the number of steps the algorithm takes as a function of n (the size of the input).

If n is small, then any algorithm will perform reasonably fast (i.e. brute force is fast). We are interested in how the running time of the algorithm grows as n tends to infinity.

As n gets very large, the multiplicative constants and lower-order terms tend to get dominated by the effects of the input size itself, and thus the highest-order term tends to heavily influence the total number of steps the algorithm takes.

When analyzing algorithms in this fashion, we are studying the **asymptotic efficiency** of the algorithm.

Example: Suppose an algorithm solves a problem using $4n^3 + 2000 \cdot n^2$ steps. For large enough n, the term $4n^3$ will be much larger than $2000 \cdot n^2$ and therefore the lower-order term $2000 \cdot n^2$ will have a negligible effect on the running time on the algorithm.

Because of this, for large enough n, a different algorithm that solves the same problem in $4n^3$ steps will have a "comparable" running time as the algorithm which takes $4n^3 + 2000 \cdot n^2$ steps, so we can ignore the term $2000 \cdot n^2$ when analyzing the running time of the original algorithm.

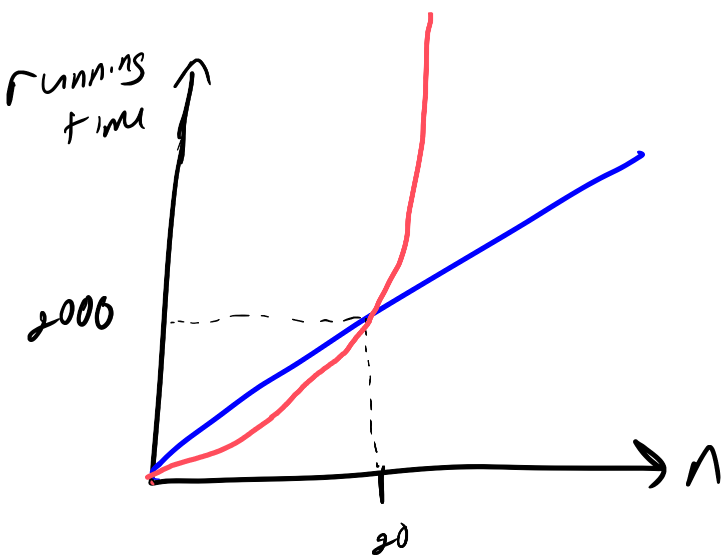
Similarly, for large enough n, $4n^3$ is "comparable" to n^3, and we can ignore the 4. We can then conclude the algorithm essentially takes, in an asymptotic sense, n^3 steps to solve the problem.

When comparing the running times of two different algorithms, we are interested in comparing their asymptotic growth. That is, we are interested in comparing their running times after dropping lower order terms and multiplicative constants.

As an illustration, we can draw the two functions as a graph, and compare the running times for large n.

$f(n) = 100n$ $\qquad$ $g(n) = 5n^2$

Note: $f(20) = g(20) = 2000$



$\log_2 n \Rightarrow 2^{\log_2 n} = n$

O-notation (big-oh)

$$f(n) \in O(g(n)) \iff \underbrace{\exists c > 0,}_{\text{g can be scaled by a constant}} \underbrace{\exists n_0 > 0, \forall n \geq n_0,}_{\text{only for big values of n, where } n_0 \text{ defines what is big.}} \underbrace{f(n) \leq c \cdot g(n)}_{f(n) \text{ is smaller than } g(n)}$$

Intuition: $f(n) \in O(g(n))$ if the asymp growth of f is at most the asymp. growth of g.

$$O(n^2) = \{ n, c, n^2, 15n^2, \tfrac{1}{2}n^2 + 100n, \log n, n \cdot \log n \}$$

$100n \in O(5n^2)$

$$100n \cdot 1 \leq 100n \cdot n = 100n^2 = \underbrace{20}_{c}\underbrace{(5n^2)}_{g}$$

$f(n) \leq c \cdot g(n)$ for $c = 20$ for all $n \geq 1$.
$\uparrow n_0$

$5n^4 + 3n^2 - 7n \in O(n^4)$

$$5n^4 + 3n^2 \underbrace{-7n}_{<0} \leq 5n^4 + 3n^2 + 0 \leq 5n^4 + 3n^4 = \underline{8}\,\underline{n^4}$$

$c \quad g$

$f(n) \leq c\,g(n)$ for $c = 8$ for all $n \geq 1$.

$n! \in O(n^n)$

$$n! = n(n-1)(n-2)(n-3) \cdots 2 \cdot 1$$

$$n^n = n \cdot n \cdot n \cdot n \cdots n \cdot n$$

$c = 1$

$n_0 = 1$.

Ω-notation (omega)

$f(n) \in \Omega(g(n)) \iff \exists c > 0, \exists n_0 > 0, \forall n \geq n_0, f(n) \geq c \cdot g(n)$

Intuition: $f(n) \in \Omega(g(n))$ if the asymp. growth of f is at least the asymp growth of g.

$\Omega(n^2) = \{ n^2, n^3, n!, n^n, n^2 \log n \ldots \}$

Θ-notation (theta)

$f(n) \in \Theta(g(n)) \iff f(n) \in O(g(n))$ and $f(n) \in \Omega(g(n))$

Intuitively, $f(n) \in \Theta(g(n))$ if their asymp growths are the same.

$\Theta(n^2) = \{ n^2, 100n^2, \frac{1}{2}n^2, n^2 - 15n + 6, \ldots \}$

$$5n^4 + 3n^2 - 7n \in \Theta(n^4)$$

We showed O. Need to show Ω.

$$5n^4 + 3n^2 - 7n \geq 5n^4 - 7n = n^4 + \underbrace{4n^4 - 7n}_{>0 \text{ when } n \geq 2} \geq n^4$$

$3n^2$: >0

$5n^4$: n^4, $4n^4$

$n \geq 2$

$f(n) \geq c \cdot g(n)$ for $c = 1$ for all $n \geq 2$.

n_0

o-notation (little-oh)

$f(n) \in o(g(n))$ if $\forall c > 0, \exists n_0 > 0, \forall n \geq n_0, f(n) < c \cdot g(n)$

Intuition, asymp. growth of $f(n)$ is strictly less than that of $g(n)$.

Properties

$f(n) \in o(g(n)) \Rightarrow f(n) \in O(g(n))$

$f(n) \in o(g(n)) \Rightarrow \lim_{n \to \infty} \frac{f(n)}{g(n)} = 0$

Limit Theorem

if $\lim_{n \to \infty} \frac{f(n)}{g(n)}$

a) $0 \Rightarrow f(n) \in o(g(n))$

b) $c \neq 0 \Rightarrow f(n) \in \Theta(g(n))$

c) $\infty \Rightarrow g(n) \in o(f(n))$

$n \in O(n^2)$?

$$\lim_{n\to\infty} \frac{n}{n^2} = \lim_{n\to\infty} \frac{1}{n} = 0 \quad \underline{\text{yes}}$$

$5n^2 \in o(n^2)$?

$$\lim_{n\to\infty} \frac{5n^2}{n^2} = \lim_{n\to\infty} 5 = 5 \quad \underline{\text{no}}$$

$\ln n \in o(n^2)$

$$\lim_{n\to\infty} \frac{\ln n}{n^2} = \lim_{n\to\infty} \frac{(\ln n)'}{(n^2)'} = \lim_{n\to\infty} \frac{\frac{1}{n}}{2n} = \lim_{n\to\infty} \frac{1}{2n^2} = 0$$

$\underline{\text{yes}}$

$\log_b n \in \Theta(\log_c n)$?

$\log_{10} n = \log_2 n \cdot \underline{\log_2 10}$

constant

$$\lim_{n\to\infty} \frac{\log_{10} n}{\log_2 n} = \lim_{n\to\infty} \frac{\cancel{\log_2 n} \cdot \log_2 10}{\cancel{\log_2 n}} = \log_2 10$$

$\underline{\text{yes}}$

$2^n \in o(3^n)$?

$$\lim_{n\to\infty} \frac{2^n}{3^n} = \lim_{n\to\infty} \left(\frac{2}{3}\right)^n = 0 \quad \underline{\text{yes}}$$

```
for(i=0; i<n; i++)
{
    .
}
```

n steps : $\Theta(n)$

```
for(i=0; i<n; i+=2)
{
    ...
}
```

$\frac{n}{2} \Rightarrow \Theta(n)$

```
for(i=1; i<n; i*=2){
    ...
}
```

$\log_2 n \Rightarrow \Theta(\log n)$

```
for(i=1; i≤n; i++){
    for(j=1; j≤n; j+=i)
    {
        ...
    }
}
```

outer loop	inner loop
1	n
2	$\frac{n}{2}$
3	$\frac{n}{3}$
4	$\frac{n}{4}$
⋮	⋮
$\frac{n}{2}$	2
⋮	⋮
n	1

answer is sum this

$$n + \frac{n}{2} + \frac{n}{3} + \frac{n}{4} + \cdots + \frac{n}{n}$$

$$= n\underbrace{\left(1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \cdots + \frac{1}{n}\right)}_{n^{th} \text{ harmonic } \# = \Theta(\ln n)}$$