Big-O Notation

How does runtime scale with respect to inputs?

We can use the 7 models to describe this scaling:

you don't have to use 'n'! Any variable.

- LINEAR: $O(n)$
  e.g. iterating through each element in an array.
- Quadratic: $O(n^2)$
  e.g. printing pairs of an array.

$\boxed{\text{RULES}}$

1. Different steps get added.

If you have two steps in your algorithm, then you add the order of these.
e.g. $O(a)$ and $O(b) \Rightarrow O(a+b)$

2. You drop constants

Imagine if you had two steps which were $O(n)$: $n+n = 2n$, but the running order ignores constants. Therefore, $\cancel{O(2n)} \Rightarrow O(n)$

3. Different inputs $\Rightarrow$ Different variables.

4. Drop non-dominate terms (use the highest degree term)

Imagine the following algorithm:

```
function (array) {
    { O(n) }
    { O(n²) }
}
```

We do not do $O(n+n^2)$:

$O(n^2) \leq O(n+n^2) \leq O(n^2+n^2)$

So we therefore have order $O(n^2)$ (we ignore constants).

Relatives of Big-Oh

big-Omega: $f(n)$ is $\Omega(g(n))$ if there is $c > 0$ $n_0 \geq 1$:

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

big-Theta: $f(n)$ is $\Theta g(n)$ if there are $c', c'' > 0$, and $n_0 \geq 1$:

$$c'g(n) \leq f(n) \leq c''g(n) \quad n \geq n_0$$

Which big-? do I use?

Big-Oh

$f(n) \leq g(n)$

Big-Ω

$f(n) \geq g(n)$

E.g. $5n^2$ is $\Omega(n^2)$

- $c > 0$, $n_0 \geq 1$ so

$5n^2 \geq cn^2$ for $n \geq n_0$

Big-Θ

$f(n) = g(n)$

E.g. $f(n)$ is Ω and $O(n)$