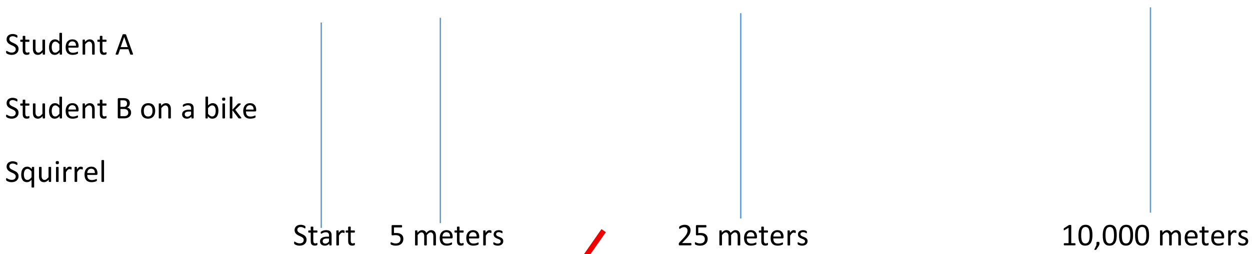
CS 315 - Day 08, Asymptotics

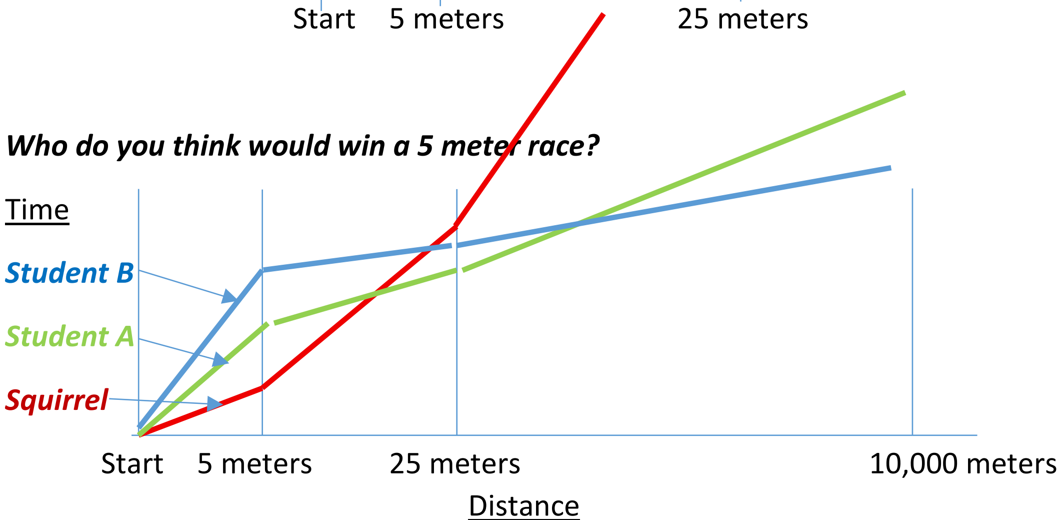
First, let's watch this (there is a pedagogical reason!): https://www.youtube.com/watch?v=P4Mpt_7g6U0

This is one of my favorites in this genre of ads (there really is no pedagogical reason to watch this): https://www.youtube.com/watch?v=m_MaJDK3VNE

So, back to algorithms… Let's picture a race, Student A, Student B on a bike, and a squirrel.



Who do you think would win a 5 meter race?



It depends on how long the race is. The Squirrel would dominate very short races, the running Student would dominate the middle distances, and then after a certain point, the bike riding Student would win all the races. When we get those arbitrarily large distances, where the ordering of the finishes is set, we are observing ***asymptotic*** behavior.

So when we describe functions which represent the running time of algorithms, we are interested in their asymptotic behavior, which means their behavior on large problem sizes.

We use ***bounds*** for the functions representing running time, since we're worried about the asymptotic effects and it wouldn't be worth the extra effort to calculate the exact function.

The notation we use: θ, O, Ω actually represent ***sets*** of functions.

Applications:

When is our first exam?

- "Sometime after next week" → lower bound.
- "Sometime before December 1" → upper bound.
- "On ***{actual date here}***." → tight bound.

What are the following in asymptotic notation? Remember we ***only*** look at the highest order term, and disregard coefficients.

So, yes, there *might be* a bit of algebra involved.

Oh, and usually on a closed expression we can pick a tight bound.

$\frac{k(k+1)}{2} = \frac{k^2+k}{2} = \frac{k^2}{2} + \frac{k}{2} \rightarrow \theta(k^2)$

$\frac{N(N+1)(2N+1)}{6}$ here, it's pretty obvious that the highest order term would be something involving N^3, so we can jump to $\theta(N^3)$.

So, the "punch line" is that:

$$\sum_{i=1}^{k} i = \theta(k^2)$$

$$\sum_{k=1}^{N} k^2 = \theta(N^3)$$

We skipped over the closed formulas, and went directly from the sigma notation to the asymptotic notation.

Note that we also stayed consistent, using the variables provided, and not the "dummy"/loop control variables. We will generate closed formulas when we analyze code.

First, let's watch this (there is a pedagogical reason!): https://www.youtube.com/watch?v=P4Mpt_7g6U0

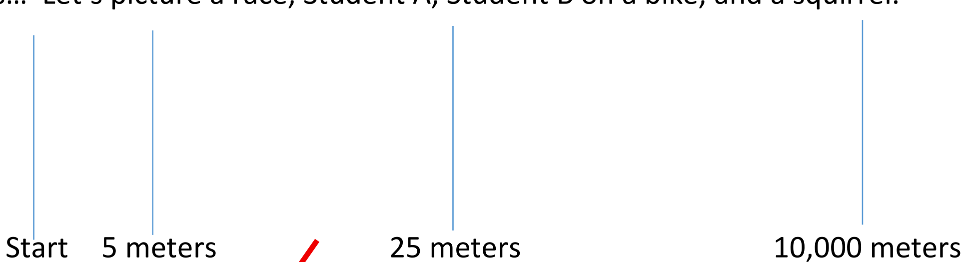
This is one of my favorites in this genre of ads (there really is no pedagogical reason to watch this): https://www.youtube.com/watch?v=m_MaJDK3VNE

So, back to algorithms… Let's picture a race, Student A, Student B on a bike, and a squirrel.

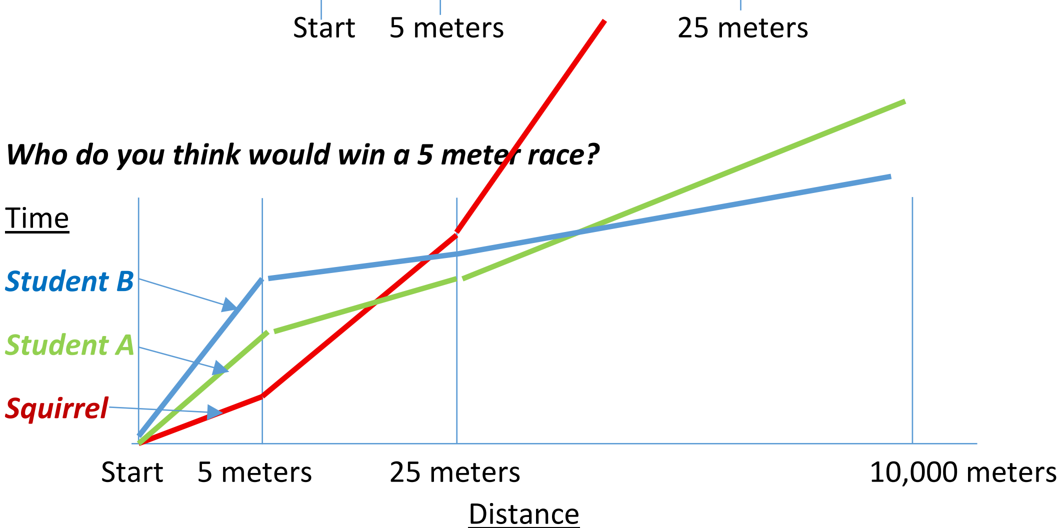
Student A

Student B on a bike

Squirrel



Who do you think would win a 5 meter race?



It depends on how long the race is. The Squirrel would dominate very short races, the running Student would dominate the middle distances, and then after a certain point, the bike riding Student would win all the races. When we get those arbitrarily large distances, where the ordering of the finishes is set, we are observing ***asymptotic*** behavior.

So when we describe functions which represent the running time of algorithms, we are interested in their asymptotic behavior, which means their behavior on large problem sizes.

We use ***bounds*** for the functions representing running time, since we're worried about the asymptotic effects and it wouldn't be worth the extra effort to calculate the exact function.

The notation we use: θ, O, Ω actually represent ***sets*** of functions.

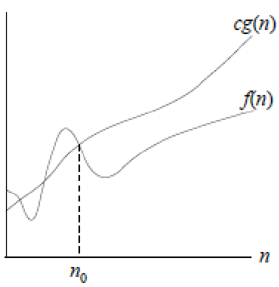
O-notation

O-notation characterizes an *upper bound* on the asymptotic behavior of a function: it says that a function grows *no faster* than a certain rate. This rate is based on the highest order term.

For example, $f(n) = 7n^3 + 100n^2 - 20n + 6$ is $O(n^3)$, since the highest order term is $7n^3$, and therefore the function grows no faster than n^3.

Te function $f(n)$ is also $O(n^5)$, $O(n^6)$, and $O(n^c)$ for any constant $c \geq 3$.

$O(g(n)) = \{f(n) :$ there exist positive constants c and n_0 such that $0 \leq f(n) \leq cg(n)$ for all $n \geq n_0\}$.



$g(n)$ is an ***asymptotic upper bound*** for $f(n)$.

If $f(n) \in O(g(n))$, we write $f(n) = O(g(n))$ (will precisely explain this soon).

Example

$2n^2 = O(n^3)$, with $c = 1$ and $n_0 = 2$.

Examples of functions in $O(n^2)$:

n^2
$n^2 + n$
$n^2 + 1000n$
$1000n^2 + 1000n$
Also,
n

Ω-notation

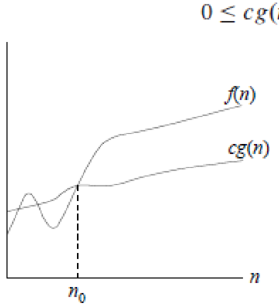
Ω-notation characterizes a *lower bound* on the asymptotic behavior of a function: it says that a function grows *at least as fast* as a certain rate. This rate is again based on the highest-order term.

For example, $f(n) = 7n^3 + 100n^2 - 20n + 6$ is $\Omega(n^3)$, since the highest-order term, n^3, grows at least as fast as n^3.

The function $f(n)$ is also $\Omega(n^2)$, $\Omega(n)$, and $\Omega(n^c)$ for any constant $c \leq 3$.

Ω-notation

$\Omega(g(n)) = \{f(n) :$ there exist positive constants c and n_0 such that $0 \leq cg(n) \leq f(n)$ for all $n \geq n_0\}$.



$g(n)$ is an ***asymptotic lower bound*** for $f(n)$.

Example

$\sqrt{n} = \Omega(\lg n)$, with $c = 1$ and $n_0 = 16$.

Examples of functions in $\Omega(n^2)$:

n^2
$n^2 + n$
$n^2 - n$
$1000n^2 + 1000n$
$1000n^2 - 1000n$
Also,
n^3

Can a function be in both O and Ω ?

YES!

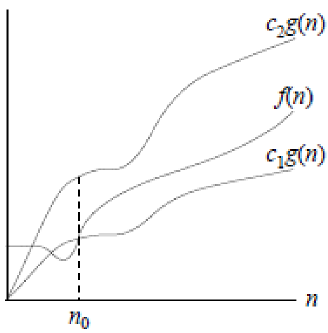
Θ-notation

Θ-notation characterizes a *tight bound* on the asymptotic behavior of a function: it says that a function grows *precisely* at a certain rate, again based on the highest-order term.

If a function is both $O(f(n))$ and $\Omega(f(n))$, then a function is $\Theta(f(n))$.

Θ-notation

$$\Theta(g(n)) = \{f(n) : \text{there exist positive constants } c_1, c_2, \text{ and } n_0 \text{ such that } 0 \le c_1 g(n) \le f(n) \le c_2 g(n) \text{ for all } n \ge n_0\}.$$



$g(n)$ is an ***asymptotically tight bound*** for $f(n)$.

Example

$n^2/2 - 2n = \Theta(n^2)$, with $c_1 = 1/4$, $c_2 = 1/2$, and $n_0 = 8$.

Theorem

$f(n) = \Theta(g(n))$ if and only if $f = O(g(n))$ and $f = \Omega(g(n))$.

So, if you can show that a function is both O and Ω, then it is automatically θ .

Note that $f(n) = \theta(n^2)$ really means that $f(n) \in \theta(n^2)$, and remember that in a mathematical expression, leading constants and low-order terms don't matter.

A constant factor can be expressed as $\theta(1)$ or $O(1)$, since it's within a constant factor of 1.

For example, the worst-case running time for insertion sort is $O(n^2)$, $\Omega(n^2)$, and $\Theta(n^2)$; all are correct. Prefer to use $\Theta(n^2)$ here, since it's the most precise. The best-case running time for insertion sort is $O(n)$, $\Omega(n)$, and $\Theta(n)$; prefer $\Theta(n)$.

But *cannot* say that the running time for insertion sort is $\Theta(n^2)$, with "worst-case" omitted. Omitting the case means making a blanket statement that covers *all* cases, and insertion sort does *not* run in $\Theta(n^2)$ time in all cases.

Can make the blanket statement that the running time for insertion sort is $O(n^2)$, or that it's $\Omega(n)$, because these asymptotic running times are true for all cases.

For merge sort, its running time is $\Theta(n \lg n)$ in all cases, so it's OK to omit which case.

Computer scientists typically mis-use notation, referring to something using O, but intending it to be a tight bound, which is θ. They cover by saying "the best" O bound.

Bounding functions asymptotics:

- "Tight Bound" $\theta(\,g(n)\,) = \{\,f(n) : \exists\, positive\ constants\ c_1, c_2, and\ n_0\ such\ that$ $0 \le c_1 \cdot g(n) \le f(n) \le c_2 \cdot g(n) \quad \forall\, n \ge n_0\,\}$

- "Upper Bound" $O(\,g(n)\,) = \{\,f(n) : \exists\, positive\ constants\ c\ and\ n_0\ such\ that$ $0 \le f(n) \le c \cdot g(n) \quad \forall\, n \ge n_0\,\}$

- "Lower Bound" $\Omega(\,g(n)\,) = \{\,f(n) : \exists\, positive\ constants\ c\ and\ n_0\ such\ that$ $0 \le c \cdot g(n) \le f(n) \quad \forall\, n \ge n_0\,\}$

So, $\sum_{i=1}^{n} i = \frac{n(n+1)}{2} = \frac{1}{2}n^2 + \frac{1}{2}n$ is $\theta(n^2)$. But let's take a deep dive and plug this into the definition.

$$0 \le c_1 n^2 \le \frac{1}{2}n^2 + \frac{1}{2}n \le c_2 n^2, \qquad \forall n_0 > n$$

We need to find c_1, c_2, and n_0.

$$c_1 n^2 \le \frac{1}{2}n^2 + \frac{1}{2}n \le c_2 n^2$$

$$c_1 \le \frac{1}{2} + \frac{1}{2n} \le c_2$$

We can just "guess and check" or inspect a graph, to determine which values of c_1, c_2, and n_0 would work.

(Hint: just try "easy" values).

$$c_1 = \frac{1}{2}$$

$$c_2 = 1$$

$$n_0 = 1$$

Next, let's show $\frac{1}{2}n + 14$ is $O(n^2)$ by plugging into the definition.

$$0 \le \frac{1}{2}n + 14 \le cn^2$$

Pick n_0 then find c	or	Pick c then find n_0
$n_0 = 10$		$c = 1$
$\frac{1}{2}10 + 14 \le c10^2$		$\frac{1}{2}n + 14 \le n^2$
$19 \le c10^2$		$\frac{1}{2} + \frac{14}{n} \le n$
$0.19 \le c$, so use $c = 0.20$		$n_0 = 10$ should work