Asymptotes

Want to describe an algorithm’s running time. (or space, …)

Described by $O(n^2)$, $\Omega(n \log n)$, $\Theta(n)$, …

What do these mean? Why do we use these notations?

Upper Bounds

Creating an algorithm proves we can solve the problem within a given bound.

But another algorithm might be faster.

E.g., sorting an array. Insertion sort $\rightarrow O(n^2)$

Lower Bounds

Sometimes can prove that we cannot compute something without a sufficient amount of time.

That doesn't necessarily mean we know how to compute it in this lower bound.

E.g., sorting an array. # comparisons needed in worst case $\rightarrow \Omega(n \log n)$

Will prove this soon…

Upper & Lower Bounds: Informal Summary

Upper bounds:

\[ \leq O(\cdot) < o(\cdot) \]

Lower bounds:

\[ \geq \Omega(\cdot) > \omega(\cdot) \]

Upper & lower (“tight”) bounds:

\[ = \Theta(\cdot) \]
Definitions: $O$, $\Omega$

$T(n) \in O(g(n))$  
$\iff$  
$\exists$ constants $c,n_0 > 0$  
such that  
$\forall n \geq n_0, T(n) \leq c \times g(n)$

$T(n) \in \Omega(g'(n))$  
$\iff$  
$\exists$ constants $c',n'_0 > 0$  
such that  
$\forall n \geq n'_0, T(n) \geq c' \times g'(n)$

Bounding allows abstractions of details

$2n+13 \in O(\ ?)$  
$O(n)$  
Also, $O(n^2)$, $O(5n)$, … Can always weaken the bound.

$2n+13 \in \Omega(\ ?)$  
$\Omega(n)$, also $\Omega(\log n)$, $\Omega(1)$, ...

$2^n \in O(n) \ ? \ \Omega(n) \ ?$  
Given a $c$, $2^n \geq c \times n$, for all but small $n$.  
$\Omega(n)$, not $O(n)$.

$n^{\log n} \in O(n^5) \ ?$  
No. Given a $c$, $\log n \geq c \times 5$, for all large enough $n$.  
Thus, $\Omega(n^5)$.

Definitions: $o$, $\omega$

Might know that our upper & lower bounds aren’t tight.

$T(n) \in o(g(n))$  
$\iff$  
$\forall$ constants $c>0$, $\exists$ constant $n_0 > 0$,  
such that  
$\forall n \geq n_0, T(n) < c \times g(n)$

Also, $T(n) \in o(g(n))$  
$\iff$  
$\lim_{n \to \infty} \frac{T(n)}{g(n)} = 0$

$T(n) \in \omega(g'(n))$  
$\iff$  
$\forall$ constants $c'>0$, $\exists$ constant $n'_0 > 0$,  
such that  
$\forall n \geq n'_0, T(n) > c' \times g'(n)$

$\frac{1}{\log n} \in o(1) \ ?$  
Yes.  
$\lim_{n \to \infty} \frac{T(n)}{g(n)} \lim_{n \to \infty} \frac{1}{\log n} = \lim_{n \to \infty} \frac{1}{\log n} = 0$

Examples: $o$, $\omega$

$2n+13 \in o(n) \ ? \ \omega(n) \ ?$  
No to both!  
$2n+13 < c \times n$ fails for many $c$.  
$2n+13 > c \times n$ fails for many $c$.

$2n+13 \in o(\ ?) \ \omega(\ ?)$  
$o(n \log n)$, $o(n^2)$, …  
$\omega(\log n)$, $\omega(1)$, …

Definition: $\Theta$

$T(n) \in \Theta(g(n))$  
$\iff$  
$T(n) \in O(g(n))$  
$\land$  
$T(n) \in \Omega(g(n))$

Ideally, find algorithms that are asymptotically as good as possible.

Standard notational abuse

$O()$, $\Omega()$, … are sets of functions.

$T(n) = O(\ ?)$  
$T(n) \in O(\ ?)$

$T(n) = f(n) + O(\ ?)$  
$T(n) = f(n) + g(n)$  
$\exists g(n) \in O(\ ?)$
Three types of measures

Most common.

<table>
<thead>
<tr>
<th>Type</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>Worst case</td>
<td>$T(n) = \max { T(x) \mid x \text{ is an instance of size } n }$</td>
</tr>
<tr>
<td>Best case</td>
<td>$T(n) = \min { T(x) \mid x \text{ is an instance of size } n }$</td>
</tr>
<tr>
<td>Average case</td>
<td>$T(n) = \sum_{x \text{ of size } n} \Pr(x) \times T(x)$</td>
</tr>
</tbody>
</table>

Determining the input probability distribution can be difficult.