Upper and lower bounds

- The big-O notation gives an upper bound
  - $f(n) \in O(g(n))$ means that $f(n)$ has an upper bound $g(n)$
- But sometimes we need a lower bound, i.e., $f(n)$ is at least $g(n)$ (minimum work to do a computation)
  - We introduce a new concept: $f(n) \in \Omega(g(n))$
- And sometimes we would like to have both a lower and upper bound for $f(n)$
  - We introduce a new concept: $f(n) \in \Theta(g(n))$
- *We can use $O(g(n))$ to define $\Omega(g(n))$ and $\Theta(g(n))$*

Defining big-Ω and big-Θ

- Ω (Big Omega) denotes a *lower bound*:

$$f(n) \in \Omega(g(n)) \;\; \textbf{iff} \;\; g(n) \in O(f(n))$$

For example: $n^3 \in \Omega(n^2)$ since $n^2 \in O(n^3)$
Intuition: *g(n)* defines the floor and *f(n)* is always above the floor

- Θ (Big Theta) denotes lower and upper bounds at the same time (*asymptotic equivalence*):

$$f(n) \in \Theta(g(n)) \;\; \textbf{iff} \;\; f(n) \in O(g(n)) \;\; \text{and} \;\; f(n) \in \Omega(g(n))$$

For example: $400n\text{-}3 \in \Theta(n)$
Intuition: *g(n)* defines a “corridor” (with both floor and ceiling) and *f(n)* always stays in the corridor

What's the difference between big-O and big-Θ ?

- Let's say we have a program that takes a list of integers and returns the position of the first negative element
  - I={FirstNegative L}

- If L has size *n* then we can have
  - Worst case time $f_{worst}(n) \in \Theta(n) \Rightarrow$ *for the inputs we consider (all elements positive except for the last one), time is always proportional to n, never less*
  - Average case time $f_{average}(n) \in O(n) \Rightarrow$ *for the inputs we consider (all possible lists), time is bounded above by n, but it might be less for some inputs (say, if first element is negative)*