CSCB63 – Design and Analysis of Data Structures

Anya Tafliovich[1]

[1]with huge thanks to Anna Bretscher and Albert Lai

how long do things take

Remember this algorithm?

```
1       i = 1
2       while i < len(A):
3         v = A[i]
4         j = i
5         while j > 0 and A[j-1] > v:
6           A[j] = A[j-1]
7           j = j - 1
8         A[j] = v
9         i = i + 1
```

What do we count? Does it matter?

how long do things take

Let's try counting this way:

- get/set variables: 1 step
- function call: 1 + steps to evaluate each argument + steps to execute function
- return statement: 1 + steps to evaluate return value
- if/while condition: 1 + steps to evaluate the boolean expression
- assignment statement: 1 + steps to evaluate each side
- arithmetic/comparison/boolean operators: 1 + steps to evaluate each operand
- array access: 1 + steps to evaluate array index
- constants: free!

how long do things take

```
0 def InsertionSort (A):                    STEPS
1     i = 1                                   2 = 1 set var. + 1 assgn
2     while i < len(A):                       5
3       v = A[i]                              5
4       j = i                                 3
5       while j > 0 and A[j-1] > v:           10 or 3
6         A[j] = A[j-1]                       8
7         j = j - 1                           4
8       A[j] = v                              5
9       i = i + 1                             4
```

how long do things take

```
0 def InsertionSort (A):                          STEPS
1     i = 1                                         2
2     while i < len(A):                             5
3       v = A[i]                                    5
4       j = i                                       3
5       while j > 0 and A[j-1] > v:                 10 or 3
6         A[j] = A[j-1]                             8
7         j = j - 1                                 4
8       A[j] = v                                    5
9       i = i + 1                                   4
```

What assumptions did we make? Are they realistic?

let len(A)=n

So, what's the total number of steps?

steps for j>0 : i times

Total: $2+5n+5(n-1)+3(n-1)+\sum_{i=1}^{n-1}(10i+3+8i+4i)+5(n-1)+4(n-1)$

$= 11n^2+14n-18$

how long do things take

In the **worst case**: A is sorted in reverse

	steps	Total steps	
1	2	$2\cdot 1=2$	
2	5	$5n$	$i=1,\cdots,n-1$, n (true, false)
3	5	$5(n-1)$	$i=1,\cdots,n-1$
4	3	$3(n-1)$	
5	10 or 3	$10i+3$	$j=i, i-1, \cdots, 1, 0$ (true 10 times, false 3 times)
6	8	$8i$	$j=i, i-1, \cdots, 1$
7	4	$4i$	
8	5	$5(n-1)$	$i=1,\cdots,n-1$
9	4	$4(n-1)$	

how long do things take

In the **best case**: A is sorted

	steps	Total steps
1	2	2
2	5	5n
3	5	5(n-1)
4	3	3(n-1)
5	10 or 3	10(n-1)
6	8	0
7	4	0
8	5	5(n-1)
9	4	4(n-1)

Total:

32n-25

how long do things take

What if we write the same algorithm differently?

```
0 def InsertionSort (A):
1     n = len(A)
2     for (i = 1; i < n; i++):
3         for (j = i; j > 0 and A[j] < A[j-1]; j--):
4             swap A[j], A[j-1]
```

- line 1: once, 4 steps
- line 2: 2 steps (once) + 3 steps (n times) + 2 steps ($n-1$ times)
- line 3: for each i: 3 steps (once) + 11 steps ($i-1$ times) + 2 steps (once) + 2 steps ($i-1$ times)
- line 4: for each i: 9 steps ($i-1$ times)

how long do things take

- line 1: once, 4 steps
- line 2: 2 steps (once) + 3 steps (n times) + 2 steps ($n-1$ times)
- line 3: for each i: 3 steps (once) + 11 steps ($i-1$ times) + 2 steps (once) + 2 steps ($i-1$ times)
- line 4: for each i: 9 steps ($i-1$ times)

$= 11n^2 - 23n + 21$

Is this the same running time? In what sense?

how long do things take

Q. What if I now run this algorithm on a machine that is slower to perform variable look up and write?

Q. Should the complexity change?

Q. How important are those constants as the input size n gets large?

Q. How are our two results $11n^2 + 14n - 18$ and $11n^2 - 23n + 21$ similar?

Q. They are both quadratic polynomials.

how long do things take

We say...

- that a quadratic polynomial is of order n^2,
- that a cubic polynomial is of order n^3,
- that $4n\lg(n) + 2n + 10$ is of order $n\lg(n)$.

Why can we say this? With a little mathemagic:

$11n^2 + 14n - 18 \leq$ $11n^2+14n \leq 11n^2+14n^2 = 25n^2$ if $n \geq 1$

Another example:

$11n^2 - 21n + 19 \leq$ $11n^2+19 \leq 11n^2+n \leq 11n^2+n^2 = 12n^2$ if $n \geq 19$

or $\leq 11n^2$

how long do things take — formally

For all natural $n \geq 19$:

$$11n^2 - 21n + 19 \leq 12n^2$$

There exists an $n_0 \in \mathbb{N}$ such that, for all natural $n \geq n_0$,

$$11n^2 - 21n + 19 \leq 12n^2$$

We can take this even further and say, there exists real $c > 0$ and natural n_0 such that, for all natural $n \geq n_0$,

$$11n^2 - 21n + 19 \leq c \cdot n^2$$

which is exactly the definition of "Big-Oh"!

Big-Oh — Asymptotic Upper Bound

We denote:

- $\mathbb{N}$: the set of natural numbers
- $\mathbb{R}^+$: the set of positive real numbers
- $\mathcal{F}$: the set of functions $f : \mathbb{N} \to \mathbb{R}^+$

Let $g \in \mathcal{F}$. Define $\mathcal{O}(g)$ to be the set of functions $f \in \mathcal{F}$ such that

$$\exists c \in \mathbb{R}^+, \exists n_0 \in \mathbb{N}, \forall n \in \mathbb{N}, n \geq n_0 \Rightarrow f(n) \leq c \cdot g(n)$$

Big-Oh — Asymptotic Upper Bound

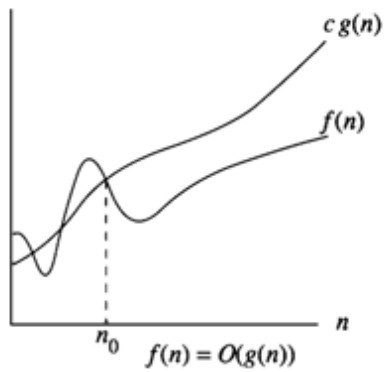
Let $g \in \mathcal{F}$. Define $\mathcal{O}(g)$ to be the set of functions $f \in \mathcal{F}$ such that

$$\exists c \in \mathbb{R}^+, \exists n_0 \in \mathbb{N}, \forall n \in \mathbb{N}, n \geq n_0 \Rightarrow f(n) \leq c \cdot g(n)$$

Big-Oh — Asymptotic Upper Bound

Let $g \in \mathcal{F}$. Define $\mathcal{O}(g)$ to be the set of functions $f \in \mathcal{F}$ such that

$$\exists c \in \mathbb{R}^+, \exists n_0 \in \mathbb{N}, \forall n \in \mathbb{N}, n \geq n_0 \Rightarrow f(n) \leq c \cdot g(n)$$



Let's practice proving a function belongs to big-Oh of another function.

Big-Oh practice

Suppose we determine an algorithm has running time

$$T(n) = n^3 - n^2 + 5$$

Prove. $T(n) \in O(n^3)$

Choose $c = 2$, $c \in \mathbb{R}^+$; $n_0 = 5$, $n_0 \in \mathbb{N}$

Claim: $\forall n \in \mathbb{N}.\ n \geq n_0 \Rightarrow n^3 - n^2 + 5 \leq c \cdot n^3$

Proof: $\forall n \geq 5.$

$$\begin{aligned} T(n) = n^3 - n^2 + 5 &\leq n^3 + 5 \\ &\leq n^3 + n \\ &\leq n^3 + n^3 = 2n^3 \end{aligned}$$

Is Big-Oh good enough?

Q. Is $12n^2 + 10n + 10 \in O(n^3)$?

Q. Is $12n^2 + 10n + 10 \in O(n^2 \lg n)$?

Q. Is $n \in O(n^2)$?

Q. Is $3 \in O(n^2)$?

$O(n^2)$ includes quadratic functions and "lesser" functions as well.

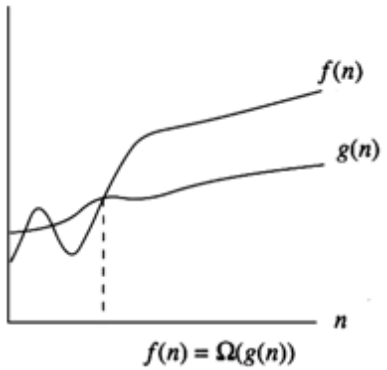
We need another definition to exclude "lesser" functions.

Big-Ω — Asymptotic Lower Bound

Idea. Want a function g such that for big enough n,

$$0 \leq b \cdot g(n) \leq f(n)$$

where b is a constant.



"Big Omega." Let $g \in \mathcal{F}$. Define $\Omega(g)$ to be the set of functions $f \in \mathcal{F}$ such that

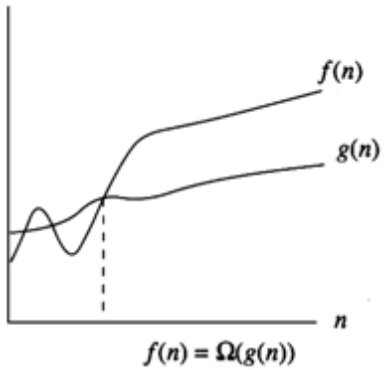
$$\exists b \in \mathbb{R}^+, \exists n_0 \in \mathbb{N}, \forall n \in \mathbb{N}, n \geq n_0 \Rightarrow f(n) \geq b \cdot g(n) \geq 0$$

Big-Ω — Asymptotic Lower Bound

Idea. Want a function g such that for big enough n,

$$0 \leq b \cdot g(n) \leq f(n)$$

where b is a constant.



"Big Omega." Let $g \in \mathcal{F}$. Define $\Omega(g)$ to be the set of functions $f \in \mathcal{F}$ such that

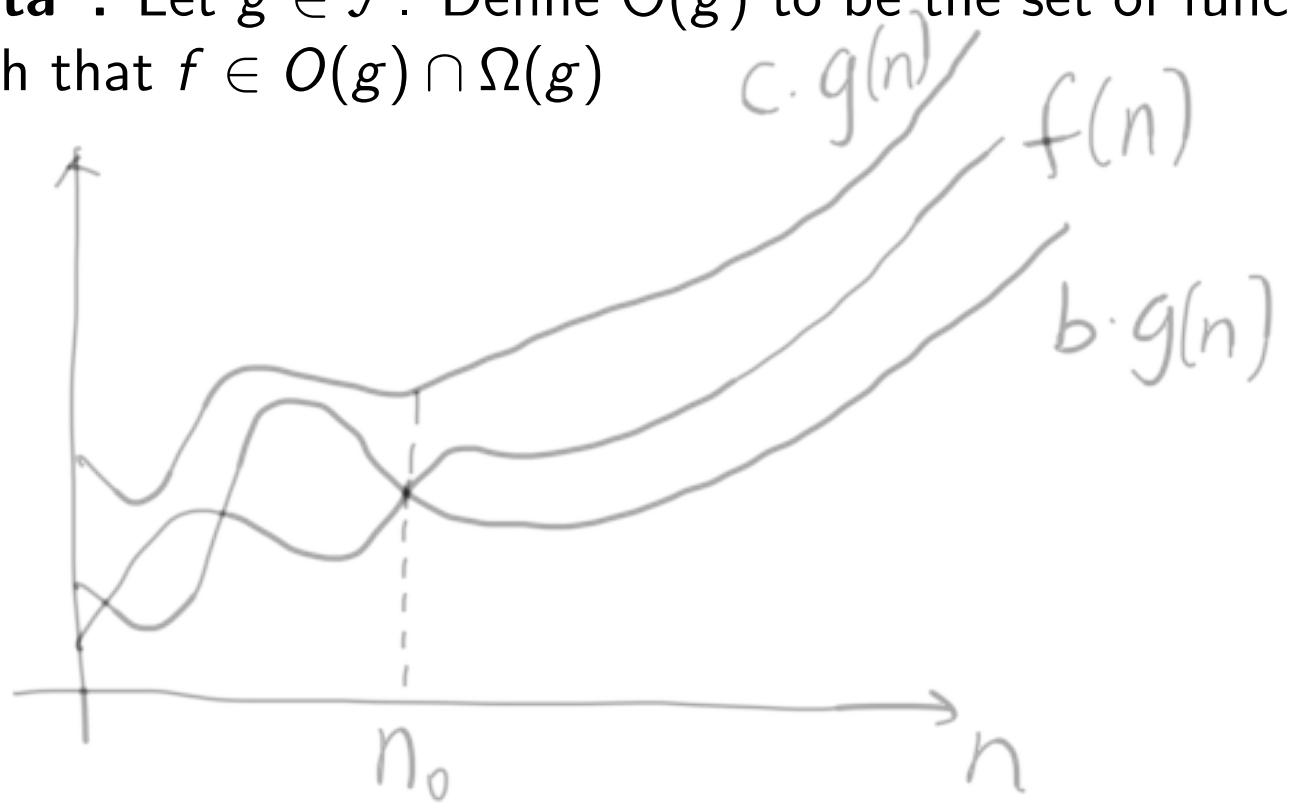
$$\exists b \in \mathbb{R}^+, \exists n_0 \in \mathbb{N}, \forall n \in \mathbb{N}, n \geq n_0 \Rightarrow f(n) \geq b \cdot g(n) \geq 0$$

Equivalently, $f \in \Omega(g)$ iff $g \in O(f)$.

Big-Θ — Asymptotic Tight Bound

What if it's both?
If $f \in O(g)$ and $f \in \Omega(g)$ then we say that $f \in \Theta(g)$.

"Big Theta". Let $g \in \mathcal{F}$. Define $\Theta(g)$ to be the set of functions $f \in \mathcal{F}$ such that $f \in O(g) \cap \Omega(g)$



Big-Θ — Asymptotic Tight Bound

What if it's both?
If $f \in O(g)$ and $f \in \Omega(g)$ then we say that $f \in \Theta(g)$.

"Big Theta". Let $g \in \mathcal{F}$. Define $\Theta(g)$ to be the set of functions $f \in \mathcal{F}$ such that $f \in O(g) \cap \Omega(g)$

or alternatively,

$$\exists b \in \mathbb{R}^+, \exists c \in \mathbb{R}^+, \exists n_0 \in \mathbb{N}, \forall n \in \mathbb{N},$$
$$n \geq n_0 \Rightarrow 0 \leq b \cdot g(n) \leq f(n) \leq c \cdot g(n)$$

Big-Θ practice

Show: $11n^2 + 14n - 18 \in \Theta(n^2)$

Choose $b = 2$, $b \in \mathbb{R}^+$; $c = 25$, $c \in \mathbb{R}^+$

$n_0 = 1$, $n_0 \in \mathbb{N}$

Then $\forall n \in \mathbb{N}.\ n \geq n_0 \Rightarrow 0 \leq b \cdot n^2 \leq 11n^2 + 14n - 18 \leq c \cdot n^2$

Proof: $11n^2 + 14n - 18 \leq 11n^2 + 14n$

$\forall n \geq 1$ $\leq 11n^2 + 14n^2 = 25n^2$

$11n^2 + 14n - 18 \geq 11n^2 - 18$

$\forall n \geq 2 \geq 1$ $\geq 11n^2 - 9n \geq 11n^2 - 9n^2 = 2n^2$

or $11n^2 - 18 \geq 11n^2 - n \geq 11n^2 - n^2 = 10n^2$

Big-Θ practice

Show: $11n^2 + 14n - 18 \in \Theta(n^2)$

Let $f(n) = n^3 - n^2 + 5$. Show: $f \in \Theta(n^3)$

$b = \frac{1}{2}$, $c = 2$, $n_0 = 2$

Show: $n \notin \Theta(n^2)$

$\Leftrightarrow n \in O(n^2)$ and $n \notin \Omega(n^2)$

$\Leftrightarrow \exists c \in \mathbb{R}^+, \exists n_0 \in \mathbb{N}, \forall n \in \mathbb{N}, n \geq n_0 \Rightarrow n \leq c \cdot n^2$

and $\forall b \in \mathbb{R}^+, \forall n_0 \in \mathbb{N}, \exists n \in \mathbb{N}, (n \geq n_0) \wedge (b \cdot n^2 > n)$

Choose $n = n_0 + 1$

negation of big-Ω

in summary

- concerned about the efficiency of an algorithm as the input size gets large
- not concerned about small constants as these are machine dependent
- therefore, use asymptotic notation: $\mathcal{O}, \Omega, \Theta$

using limits to prove Big-O

Assume. $\exists n_0 \in \mathbb{N} \colon \forall n \geq n_0 \colon f(n) \geq 0$ and $g(n) \geq 0$.

using limits to prove Big-O

Assume. $\exists n_0 \in \mathbb{N} : \forall n \geq n_0 : f(n) \geq 0$ and $g(n) \geq 0$.

Theorem. If $\lim_{n\to\infty} \frac{f(n)}{g(n)}$ exists and is finite, then $f \in O(g)$.

$\Leftrightarrow \deg(f) \leq \deg(g)$ $\quad$ $g \in \Omega(f)$

using limits to prove Big-O

Assume. $\exists n_0 \in \mathbb{N} : \forall n \geq n_0 : f(n) \geq 0$ and $g(n) \geq 0$.

Theorem. If $\lim_{n\to\infty} \frac{f(n)}{g(n)}$ exists and is finite, then $f \in O(g)$.

Example. Prove $n(n+1)/2 \in O(n^2)$

$$\lim_{n\to\infty} \frac{n(n+1)}{2n^2} = \frac{1}{2} \text{ is finite}$$

using limits to prove Big-O

Assume. $\exists n_0 \in \mathbb{N} : \forall n \geq n_0 : f(n) \geq 0$ and $g(n) \geq 0$.

Theorem. If $\lim_{n\to\infty} \frac{f(n)}{g(n)}$ exists and is finite, then $f \in O(g)$.

Example. Prove $n(n+1)/2 \in O(n^2)$

Example. Prove $\ln(n) \in O(n)$

$$\lim_{n\to\infty} \frac{\ln n}{n} = 0 \quad \text{is finite}$$

using limits to disprove Big-O

Assume. $\exists n_0 \in \mathbb{N} : \forall n \geq n_0 : f(n) \geq 0$ and $g(n) \geq 0$.

Theorem. If $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$, then $f \notin O(g)$.

$\Leftrightarrow \deg(f) > \deg(g)$

using limits to disprove Big-O

Assume. $\exists n_0 \in \mathbb{N} : \forall n \geq n_0 : f(n) \geq 0$ and $g(n) \geq 0$.

Theorem. If $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$, then $f \notin O(g)$.

$g \notin \Omega(f)$

Example. Disprove $n^2 \in O(n)$

$$\lim_{n\to\infty} \frac{n^2}{n} = n = \infty$$

using limits to disprove Big-O

Assume. $\exists n_0 \in \mathbb{N} : \forall n \geq n_0 : f(n) \geq 0$ and $g(n) \geq 0$.

Theorem. If $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$, then $f \notin O(g)$.

Example. Disprove $n^2 \in O(n)$

Example. Disprove $n \in O(\ln(n))$

$$\lim_{n\to\infty} \frac{n}{\ln(n)} = \infty$$

when limits don't help

Theorem. If $\lim_{n\to\infty} \frac{f(n)}{g(n)}$ exists and is finite, then ...

Theorem. If $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$, then ...

Q. Which case is not covered?

when limits don't help

Theorem. If $\lim_{n\to\infty} \frac{f(n)}{g(n)}$ exists and is finite, then ...

Theorem. If $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$, then ...

Q. Which case is not covered?

A. If $\lim_{n\to\infty} \frac{f(n)}{g(n)}$ does not exist and is not ∞, then no conclusion. (Hopefully this happens rarely.)

when limits don't help

Theorem. If $\lim_{n\to\infty} \frac{f(n)}{g(n)}$ exists and is finite, then ...

Theorem. If $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$, then ...

Q. Which case is not covered?

A. If $\lim_{n\to\infty} \frac{f(n)}{g(n)}$ does not exist and is not ∞, then no conclusion. (Hopefully this happens rarely.)

Q. Can you think of a function $crazy(n)$ where limits do not help to show $crazy(n) \notin O(1)$?

when limits don't help

Q. Can you think of a function $crazy(n)$ where limits do not help to show $crazy(n) \notin O(1)$?

$$f(n) = \begin{cases} 1, & n \text{ even} \\ n, & n \text{ odd} \end{cases}$$

using limits for Θ

Theorem. $f \in \Theta(g)$ iff $f \in O(g)$ and $g \in O(f)$.

(Handy when you want to use limits!)

Example. $n^2 + n^{3/2} \in \Theta(n^2)$

$$\lim_{n\to\infty} \frac{n^2+n^{\frac{3}{2}}}{n^2} = \lim_{n\to\infty}\left(1+\frac{1}{n^{\frac{1}{2}}}\right) \text{ exists and finite } \checkmark$$

$$\lim_{n\to\infty} \frac{n^2}{n^2+n^{\frac{3}{2}}} = \lim_{n\to\infty} \frac{2n}{2n+\frac{3}{2}n^{\frac{1}{2}}} = \lim_{n\to\infty} \frac{2}{2+\frac{3}{4}n^{-\frac{1}{2}}}$$

exists and finite $\checkmark$

using limits for Θ

Theorem. $f \in \Theta(g)$ iff $f \in O(g)$ and $g \in O(f)$.

(Handy when you want to use limits!)

Example. $n^2 + n^{3/2} \in \Theta(n^2)$

Example. $\ln(n) \notin \Theta(n)$

$\lim_{n\to\infty} \frac{\ln n}{n} = \lim_{n\to\infty} \frac{1}{n}$ exists and finite ✓

$\lim_{n\to\infty} \frac{n}{\ln n} = \infty$ ✗

Big-O, Big-Θ may miss something

Q. Can the Big-O definition be not at all useful?

A.

Big-O, Big-Θ may miss something

Q. Can the Big-O definition be not at all useful?

A.

$$n + 10^{100} \in \Theta(n)$$

$$10^{100} n \in \Theta(n)$$

Big-O, Big-Θ may miss something

Q. Can the Big-O definition be not at all useful?

A.

$$n + 10^{100} \in \Theta(n)$$

$$10^{100} n \in \Theta(n)$$

Can't say these are practical algorithm times, but O, Θ can't tell.

This is a price for ignoring constants (which we want to account for machine differences!)

Such pathological cases are rare. O and Θ are usually informative.

Myth Buster

Myth: O means worst-case time, Ω means best-case.

Truth: O, Ω, Θ classify functions, do not say what the functions stand for.

$9n^2 + 4n + 13$ may be best-case time, or worst-case time, or best-case space, or worst-case space, or just a polynomial from nowhere.

"Best case time is in $O(n^2)$" means:
Best case time is some function, that function is in $O(n^2)$.
Clearly a sensible statement and possible scenario.
O, Ω, Θ are good for any function from natural to non-negative real.