Part 2: Time Complexity and Asymptotic Notation

Contents

- Time Complexity of Algorithms p.2
- Asymptotic Notation p.7
- Limit-Based Tools p.18

[document finalized]

Time Complexity of Algorithms

Goal:

Establish precise framework for classifying algorithms w.r.t. their efficiency

In this section, let P denote an infinite problem set

For algorithm A we define

$$T_A(p) = \text{runtime of } A \text{ on problem instance } p$$

(alternatively: space requirement)

Two abstractions will allow us to make general statements:

1. We consider the size of p, denoted $|p|$, rather than p itself
   a) worst-case time complexity:

$$T_A(n) = \max\{T_A(p) \mid p \in P, |p| = n\}$$

$\sim$ maximum runtime of A for inputs of size n

b) average-case time complexity:

$$T_A(n) = E(\{T_A(p) \mid p \in P, |p| = n\})$$

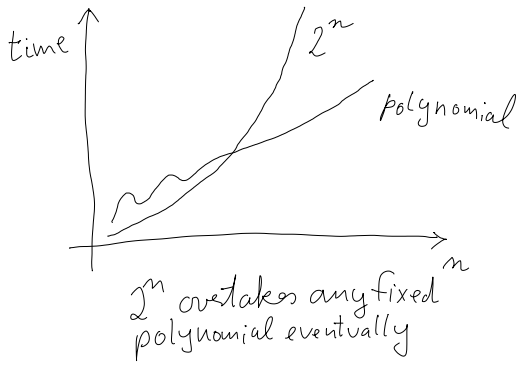
$\sim$ expected runtime of A for inputs of size n that have a certain probability distribution (usually uniform)

Example: Quicksort on average can sort n numbers in time $\approx n \log n$. But in the worst case it runs $\approx n^2$ steps.

In case a) could say A_1 is better than A_2 if

$$\forall n : T_{A_1}(n) < T_{A_2}(n)$$

but this is too restrictive, as the following graph illustrates:



2. Consider asymptotic behaviour of T_A:

   A_1 better than A_2 if

$$\lim_{n\to\infty} \frac{T_{A_1}(n)}{T_{A_2}(n)} = 0$$

   $\sim$ eventually $T_{A_2}(n)$ grows faster than $T_{A_1}(n)$

Lecture 5

Measuring Runtime

In general: sum up runtimes for each executed instruction

Depends on input (size)

Two common measures:

1. Unit-cost measure
   cost per instruction is 1 time unit
   realistic if operands fit in machine word size
   otherwise:
2. Logarithmic-cost measure
   consider size of operands in binary representation

$$l(i) = \begin{cases} 0, & i = 0 \\ \lfloor \log(i) \rfloor + 1, & i > 0 \end{cases}$$

($\lfloor x \rfloor$: round down to nearest integer – "floor" operation)
E.g., $l(17) = \lfloor \log(17) \rfloor + 1 = \lfloor 4.087 \rfloor + 1 = 4 + 1 = 5$. So we need 5 bits to represent decimal number 17 in binary, which is right because $17_{10} = 10001_2$.

Now the runtime of say $i \leftarrow j + k$ would be $l(j) + l(k)$ rather than just 1 when using the unit-cost measure, which is more accurate when dealing with large numbers.

Asymptotic Notation

Runtime Analysis with the Unit-Cost Measure

```
// input:  array A[0..n-1] and integer x
// output:  position of x in A, or n if not found

   function SeqSearch(A[0,...,n-1], x)
1. i ← 0                                  1            1
2. while i < n and A[i] ≠ x do            (j+1)2       (n+1)+n
3.    i ← i+1                             j            n
4. end
5. return i                               1            1
                                          ---          ---
                                          3j+5         3n+4
                                          x found at j not found
```

For Boolean connectives we use the "short-cut" semantics, i.e. in line 2. if $i < n$ fails, $A[i] \neq x$ is not evaluated, because the result of the and operation is false regardless. Because of this shortcut, the program will not try to access $A[n]$ – which would be out of bounds!

The worst-case runtime for SeqSearch on inputs of length n is $3n + 4$. We call this a linear-time algorithm.

SeqSearch is runtime-optimal up to a constant factor, because in order to be correct, any algorithm that solves the problem has to read each input number at least once, i.e. it has to run at least n steps.

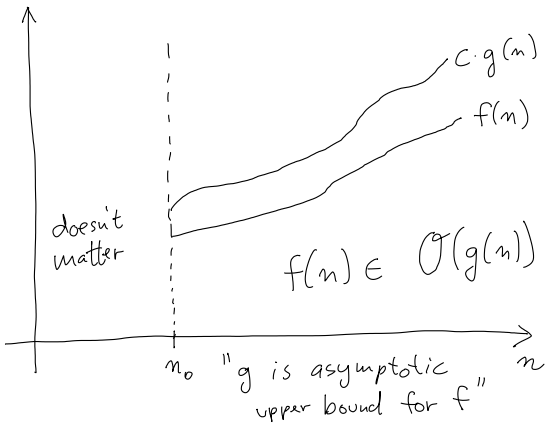
Growth Rate of Functions

When comparing runtime functions we are interested only in their growth rate which is the asymptotic behaviour for large n. We concentrate on the leading term because lower order terms become insignificant for large n. Also, constant coefficients are less significant than the rate.

Will use "big-O" notation to compare functions:

$O(g(n))$ — read as "big O of $g(n)$"

- roughly: The set of functions which, as n gets large, grow no faster than a constant times $g(n)$.
- precisely: The set of functions $f : \mathbb{N} \to \mathbb{R}$ such that for each f there are constants $c > 0$ and $n_0 \in \mathbb{N}$ with $|f(n)| \leq c\,|g(n)|$ for all $n \geq n_0$.



Examples:

$h(n) = 10n - 1 \in O(n)$

because $|10n - 1| \leq |10n|$ for $n \geq 1$.

$\Rightarrow$ Setting $c = 10$ and $n_0 = 1$ in the definition works.

$h(n) = (n+1)^2 \ \in O(n^2)$

because $|(n+1)^2| = |n^2+2n+1| \leq |n^2+2n^2+n^2| = 4n^2$ for $n \geq 1$

$\Rightarrow c = 4$ and $n_0 = 1$ works

$$h(n) = \begin{cases} 5^n, & n \leq 10^{120} \\ n^2, & n > 10^{120} \end{cases} \quad \in O(n^2)\ ?$$

Yes: $c = 1, n_0 = 10^{120} + 1$.

Lecture 6

$h(n) = 1 + 1/(n+1) \ \in O(1)$

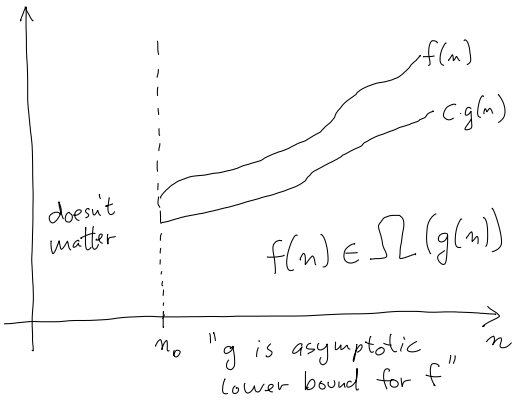
because $|h(n)| = 1 + 1/(n+1) \leq 1 + 1 = 2$ for $n \geq 0$

$\Rightarrow c = 2$ and $n_0 = 0$ works

More Asymptotic Growth Rate Sets

$\Omega(g(n))$ ("big-Omega") is the set of functions $f(n)$ that

- roughly, grow at least as fast as $g(n)$, namely
- $\exists c > 0, n_0$, such that $|f(n)| \geq c|g(n)|$ for all $n \geq n_0$

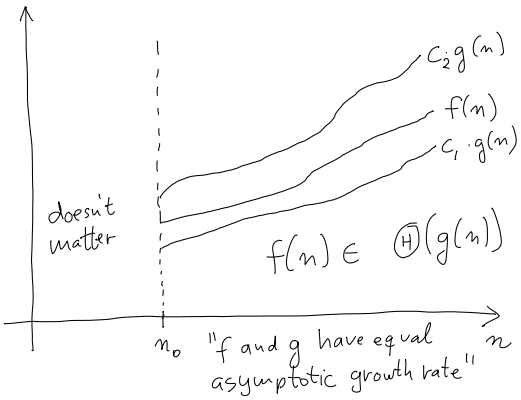


Example: $6n^3 \in \Omega(n)$ (grows at least as fast)

because $|6n^3| \geq |6n|$ for $n \geq 1$

$\Theta(g(n))$ ("big-Theta") is the set of functions $f(n)$ that

- roughly, grow with the same rate as $g(n)$, namely
- $\exists c_1 > 0, c_2 > 0, n_0$, such that $c_1|g(n)| \leq |f(n)| \leq c_2|g(n)|$ for all $n \geq n_0$



Example: $5n^2 + n \in \Theta(n^2)$ (equal growth rate)

because $|5n^2 + n| \leq |5n^2 + n^2| = 6n^2$ for $n \geq 0$

and $|5n^2 + n| \geq |5n^2|$ for $n \geq 0$

$o(g(n))$ ("small-o") is the set of functions $f(n)$ that

- roughly, grow slower than $g(n)$, namely
- $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$

Example: $n \in o(n^2)$

because $\lim_{n\to\infty} \frac{n}{n^2} = 0$

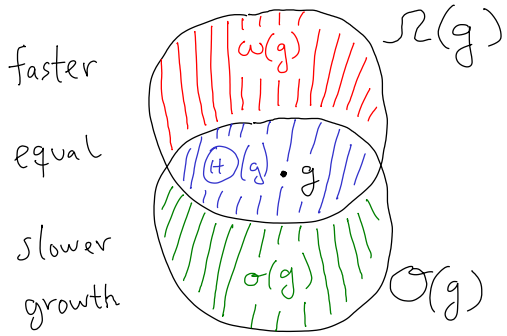
$\omega(g(n))$ ("small-omega") is the set of functions $f(n)$ that

- roughly, grow faster than $g(n)$, namely
- $\lim_{n\to\infty} |\frac{f(n)}{g(n)}| = \infty$

Example: $n^2 \in \omega(n)$

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

Venn Diagram of Growth Rate Sets:



Useful facts:

1. $\Theta(g) = O(g) \cap \Omega(g)$
2. $o(g) = O(g) - \Theta(g)$ (in $O(g)$, but not in $\Theta(g)$)
3. $\omega(g) = \Omega(g) - \Theta(g)$
4. $f \in o(g) \Rightarrow f \notin \Omega(g)$
5. $f \in \omega(g) \Rightarrow f \notin O(g)$

6. $f \in O(g)$ if and only if $g \in \Omega(f)$
7. $f \in \omega(g)$ if and only if $g \in o(f)$
8. $f \in O(1)$ if and only if f is bounded
9. If $f \in O(g)$ and $g \in O(h)$ then $f \in O(h)$

   I.e., $f \in O(g)$ is a transitive relation

10. For $f_1 \in O(g_1)$ and $f_2 \in O(g_2)$:

   (a) $f_1(n) + f_2(n) \in O(|g_1(n)| + |g_2(n)|)$

   (b) $f_1(n) + f_2(n) \in O(\max(|g_1(n)|, |g_2(n)|))$

   This lets us concentrate on leading terms when analysing growth rates.

   E.g. $f(n) = 3n^4 + n^4 + n^3 \log n + 5$.

   Then $f(n) \in O(n^4)$, because eventually $3n^4$ is the biggest term.

   (c) $f_1(n) \cdot f_2(n) \in O(g_1(n) \cdot g_2(n))$

Mnemonics for growth rate sets:

$f \in O(g)$: "$f \leq g$" $\quad$ $f \in o(g)$: "$f < g$"

$f \in \Omega(g)$: "$f \geq g$" $\quad$ $f \in \omega(g)$: "$f > g$"

$f \in \Theta(g)$: "$f = g$"

Proof of selected facts (others: exercise)

9) Prove: If $f \in O(g)$ and $g \in O(h)$ then $f \in O(h)$.

The premise says:

$\exists n_1 \in \mathbb{N}$ and $\exists c_1 > 0$ such that $|f(n)| \leq c_1 \cdot |g(n)|$ for all $n \geq n_1$ and

$\exists n_2 \in \mathbb{N}$ and $\exists c_2 > 0$ with $|g(n)| \leq c_2 \cdot |h(n)|$ for all $n \geq n_2$.

Therefore,

$$|f(n)| \;\leq\; c_1|g(n)| \;\leq\; c_1 \cdot c_2 \cdot |h(n)|,$$

for all n with $n \geq n_1$ and $n \geq n_2$. Choosing $c = c_1 \cdot c_2$ and $n_0 = \max(n_1, n_2)$ in the definition of $f \in O(h)$, we see that it holds. $\square$

10c) Prove: If

$$f_1 \in O(g_1) \text{ and } f_2 \in O(g_2)$$

then

$$f_1(n) \cdot f_2(n) \in O(g_1(n) \cdot g_2(n)).$$

The premise says that there exists $n_1, n_2 \in \mathbb{N}$ and $c_1, c_2 > 0$ such that

$$|f_1(n)| \leq c_1 \, |g_1(n)|$$

and

$$|f_2(n)| \leq c_2 \, |g_2(n)|$$

for $n \geq \max(n_1, n_2)$.

Hence,

$$|f_1(n) f_2(n)| \leq c_1 \cdot c_2 \, |g_1(n) g_2(n)|$$

holds for $n \geq \max(n_1, n_2)$, which means

$$f_1(n) \cdot f_2(n) \in O(g_1(n) \cdot g_2(n)).$$

□

Limit-Based Tools

L1. $\lim_{n\to\infty} |f(n)/g(n)| = 0 \overset{\text{def}}{\Leftrightarrow} f \in o(g)$

$(\overset{\text{facts 2+4}}{\Rightarrow} f \in O(g) \text{ and } f \notin \Omega(g))$

L2. $\lim_{n\to\infty} |f(n)/g(n)| = \infty \overset{\text{def}}{\Leftrightarrow} f \in \omega(g)$

$(\overset{\text{facts 3+5}}{\Rightarrow} f \in \Omega(g) \text{ and } f \notin O(g))$

L3. If $\lim_{n\to\infty} |f(n)/g(n)| = c > 0$, then $f \in \Theta(g)$
(note that the converse is not true, as $|f(n)/g(n)|$ not necessarily converges, if $f \in \Theta(g)$)

Examples:

$n^k \in o(n^{k+1})$: $\lim_{n\to\infty} \frac{n^k}{n^{k+1}} = \lim_{n\to\infty} \frac{1}{n} = 0$ (L1)

$n! \in o(n^n)$: $\frac{n!}{n^n} = \frac{1 \cdot 2 \cdot 3 \cdots n}{n \cdot n \cdot n \cdots n} \leq \frac{1 \cdot n \cdots n}{n \cdot n \cdots n} = \frac{1}{n}$

for $n \geq 1 \Rightarrow \lim_{n\to\infty} \frac{n!}{n^n} \leq \lim_{n\to\infty} \frac{1}{n} = 0$ (L1)

$n! \in \Theta(\sqrt{n}(\frac{n}{e})^n)$: We use Stirling's approximation:

$$\lim_{n\to\infty} \frac{n!}{\sqrt{2\pi n}(n/e)^n} = 1,$$

and apply L3.

Exercise: what is the growth rate of $\ln(n!)$?

Lecture 7

Sometimes, determining the limit of function ratios is not straight forward.

E.g. What is $\lim_{n\to\infty} \frac{\log n}{n}$?

In situations where f and g are differentiable functions from $\mathbb{R}$ to $\mathbb{R}$, the following theorem often helps:

L'Hôpital's rule [L'H]:

If $\lim_{n\to\infty} f(n) = \infty$, $\lim_{n\to\infty} g(n) = \infty$, and

$$\lim_{n\to\infty} \frac{f'(n)}{g'(n)} = L \in \mathbb{R},$$

then

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = L$$

Note that L'H can be applied multiple times as long as intermediate ratios have the form $\frac{\infty}{\infty}$, but only if the last ratio converges, we know that the original ratio converges to the same value.

Example: $\ln n \in o(n)$

We want to show that $\lim_{n\to\infty} |\frac{\ln n}{n}| = 0$

Both $f(n) = \ln n$ and $g(n) = n$ go to ∞ for $n \to \infty$

Apply L'H:

$$\lim_{n\to\infty} \frac{f'(n)}{g'(n)} = \lim_{n\to\infty} \frac{1/n}{1} = 0,$$

using the facts $\ln'(x) = \frac{1}{x}$ and $g'(n) = 1$.

Thus, $\lim_{n\to\infty} |\frac{\ln n}{n}| = 0$, and with L1, $\ln n \in o(n)$. $\square$

Example: Every monomial n^k grows slower than e^n, i.e.

$$\forall k \in \mathbb{N} : \; n^k \in o(e^n) :$$

We prove this by fixing k and applying L'H to $f(n) = n^k$ and $g(n) = e^n$ k times. Recall from calculus that $f'(n) = kn^{k-1}$ and $g'(n) = e^n$, so

$$\lim_{n\to\infty} \frac{n^k}{e^n} = \lim_{n\to\infty} \frac{f'(n)}{g'(n)}$$

$$= \lim_{n\to\infty} \frac{kn^{k-1}}{e^n}$$

$$= \lim_{n\to\infty} \frac{k(k-1)n^{k-2}}{e^n}$$

...

$= \lim_{n\to\infty} \frac{k(k-1)(k-2)\cdots 1}{e^n} = 0$, because k is a fixed constant.

So, with L1: $\forall k \in \mathbb{N} : \; n^k \in o(e^n)$. $\square$

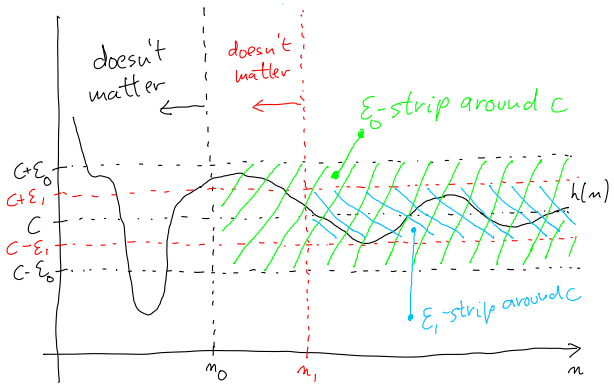
Proof of L3:

$$\lim_{n\to\infty} |f(n)/g(n)| = c > 0 \;\Rightarrow\; f \in \Theta(g)$$

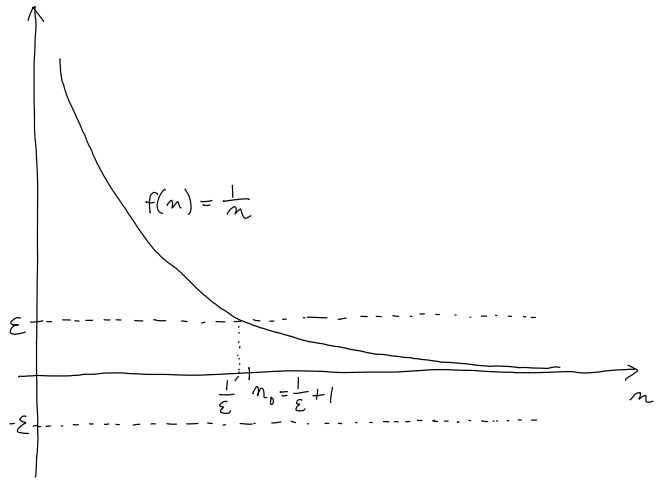
Definition of $\lim_{n\to\infty} h(n) = c \in \mathbb{R}$:

$$\forall \epsilon > 0 \; \exists n_0 \in \mathbb{N} \; \forall n \geq n_0 : |h(n) - c| < \epsilon$$

"For sufficiently large values of n, all function values $h(n)$ lie in the $\pm\epsilon$ strip around c, where ϵ can be arbitrarily small."



Example: $\lim_{n\to\infty} \frac{1}{n} = 0$.



Proof: Given $\epsilon > 0$ choose $n_0 = \frac{1}{\epsilon} + 1 = \frac{1+\epsilon}{\epsilon}$.

Then $|\frac{1}{n} - 0| < \epsilon$ for all $n \geq n_0$, because

$|\frac{1}{n} - 0| = \frac{1}{n} \overset{(n\geq n_0)}{\leq} \frac{1}{n_0} = \frac{\epsilon}{1+\epsilon} < \epsilon$ because $1 + \epsilon > 1$.

So, in summary we have proved $|\frac{1}{n}| < \epsilon$. Note, that

choosing $n_0 = \frac{1}{\epsilon} + 1$ we can prove $|\frac{1}{n}| < \epsilon$, which is required by our limit definition, otherwise it would have been $|\frac{1}{n}| \leq \epsilon$.

Also, the process of finding n_0 that works for a given ϵ actually starts with the conclusion, like so: we want $\frac{1}{n}$ to be less than ϵ, how do we chose n_0? ...

Now back to the proof of L3:

Fact: $|x| < \epsilon \;\Leftrightarrow\; -\epsilon < x < \epsilon$

Plug in $(h(n) - c)$ for $x \;\rightsquigarrow$

$$|h(n) - c| < \epsilon \;\Leftrightarrow\; -\epsilon < h(n) - c < \epsilon$$

$$\overset{\text{add } c \text{ to all terms}}{\Leftrightarrow} \quad c - \epsilon < h(n) < c + \epsilon$$

So: using above derivation, $\lim_{n\to\infty} |f(n)/g(n)| = c$ means

$$\forall \epsilon > 0 \, \exists n_0 \in \mathbb{N} \, \forall n \geq n_0 : c - \epsilon < |f(n)/g(n)| < c + \epsilon$$

(*)

Therefore, fixing $\epsilon = 1$ (or any other value), we have

$$\exists n_0 \in \mathbb{N} \, \forall n \geq n_0 : |f(n)/g(n)| < c + 1$$

$\Rightarrow$ (multiply by $|g(n)|$)

$$\exists n_0 \in \mathbb{N} \, \forall n \geq n_0 : |f(n)| < (c + 1)|g(n)|$$

which means $f \in O(g)$, because $c + 1 > 0$.

$f \in \Omega(g)$ can be proved analogously (exercise)

□

Lecture 8

Important Growth Rates

$\forall k \geq 1, c > 1$:

$O(1) \subsetneq O((\log n)^k) \subsetneq O(n) \subsetneq O(n(\log n)^k) \subsetneq O(n^2) \subsetneq O(n^{k+2}) \subsetneq O(n^{k+3}) \subsetneq O(c^n) \subsetneq O(n!) \subsetneq O(n^n)$

Proofs: exercise.

For each $A \subsetneq B$ you need to prove two things:

1. A is a subset of B, $A \subseteq B$, meaning $f \in A \Rightarrow f \in B$, and
2. $\exists f \in B : \; f \notin A$

Hint: first show $f \in o(g) \Rightarrow O(f) \subsetneq O(g)$.

Some more useful generalizations:

Theorem

- Any polynomial $p(n)$ with degree d is in $O(n^d)$
- For any polynomial $p(n)$ and any $c > 1$: $p(n) \in o(c^n)$
- $\forall k \in \mathbb{N}$ and $\epsilon > 0 : \; (\log n)^k \in o(n^\epsilon)$

Proofs: exercise