Additional Notes for Algorithms and Data Structures about Notation for Analysis of Algorithms

Thore Husfeldt

Revision 6ffc465…, Mon May 2 21:01:45 2022 +0200, Thore Husfeldt

Chapter 1

Big Oh

This note is about notation for the growth of functions encountered in the analysis of algorithms, sometimes called asymptotic[1] *notation.*
Following Rutanen, we introduce Big-Oh notation as linear dominance, instead of asymptotic linear dominance. The current version is of a highly preliminary nature.

1.1 Asymptotic bound: Big Oh

The notation $f \in O(g)$, is a "fuzzy version of $f \leq g$." It is defined in terms of the *set of functions* $O(g)$, which one can think of as all function that are "at most g times a positive constant", i.e.,

$$O(g) = \{\ldots, \sqrt{g}, \ldots, \tfrac{1}{100}g, \ldots, g, \ldots, \pi g, \ldots, 235g \ldots\}.$$

We start with a precise definition for functions that are defined on the set of positive integers $\mathbf{N} = \{1, 2, \ldots,\}$. For many applications, this is completely sufficient.

Definition 1.1 (*O* **as linear dominance, monovariate on positive integers**) Let $f, g \colon \mathbf{N} \to \mathbf{R}_{\geq 0}$. Then $f \in O(g)$ if there exist $c > 0$ such that

$$f(n) \leq cg(n) \quad \text{for all } n \in \mathbf{N}.$$

The notation $f \in O(g)$ is often read as "eff is big-oh of gee", or just "eff is oh of gee", or "eff is order of gee."

Example 1.1 Some usage examples are:

$$n + 1 \in O(n) \quad \text{for } n \in \mathbf{N}$$

and

$$n^2 - n + 5 \in O(n^2) \quad \text{for } n \in \mathbf{N}$$

and also

$$2n \in O(n) \quad \text{for } n \in \mathbf{N},$$

[1] The term "asymptotic" is highly misleading, it means "not falling together", highlighting the fact that the function $x \mapsto 1/x$ never touches the x-axis (its "asymptote"). This aspect of *not touching* plays no role for us at all, it is in fact *false*: Our asymptotic notions are reflexive, so a function is asymptotically itself in that sense. You are advised to ignore your knowledge of Greek and stick to calling it "*O*-notation".

but

$$n^2 - n + 5 \notin O(n) \quad \text{for } n \in \mathbf{N}.$$

As a convention, we will often include the "$n \in \mathbf{N}$" for clarity (even though it is implicit in Definition 1.1) in this section.

Theorem 1.1 **(As a relation)** The relation between f and g when $f = O(g)$ is

reflexive: $f \in O(f)$

transitive: if $f \in O(g)$ and $g \in O(h)$ then $f \in O(h)$.

The O-relation is *not* symmetric. For instance, $n^2 \notin O(n)$ (even though $n \in O(n^2)$.)

Theorem 1.2 **(Properties) order-consistent** If $f \leq g$ then $f \in O(g)$.

scale-invariance $O(\alpha f) = O(f)$ for every real number $\alpha > 0$.

additive $O(f) + O(g) = O(f + g)$.

multiplicative $O(f) \cdot O(g) = O(f \cdot g)$.

homogenuous $f\,O(g) = O(fg)$.

monotone If $f \leq g$ then $O(f) \subseteq O(g)$.

sum is max $O(f + g) = O(\max\{f, g\})$

max is sum $\max\{O(f), O(g)\} = O(f) + O(g)$

Proof. If $f \leq g$ then in particular $f(x) \leq 1 \cdot g(x)$, so $f \in O(g)$.

The other proofs are tedious but easy. For instance, let us show $O(f) + O(g) \subseteq O(f + g)$. Consider $h \in O(f) + O(g)$. This means there is $f' \in O(f)$ and $g' \in O(g)$ such that $h = f' + g'$. Since $f' \in O(f)$ there exists $c_1 > 0$ such that $f'(n) \leq c_1 f(n)$ for all $n \in \mathbf{N}$. Similarly, there exists $c_2 > 0$ such that $g'(n) \leq c_2 g(n)$ for all $n \in \mathbf{N}$. Thus, we have $h(n) = f'(n) + g'(n) \leq c_1 f(n) + c_2 g(n) \leq c \cdot (f(n) + g(n)) = c \cdot (f + g)(x)$ for $c = \max\{c_1, c_2\}$. In particular, $h \in O(f + g)$ □

Example 1.2 For $n \in \mathbf{N}$,

1. $\frac{1}{2}n \in O(n)$
2. $n + 1 \in O(n)$
3. $3n \cdot \log(n^6) \in O(n \log n)$
4. $170n \in O(n)$
5. $170n + 10n \in O(n)$
6. $170n + \log n \in O(n)$
7. $\ln n \in O(\log n)$

8. $\log n \in O(\ln n)$
9. $\log n \in O(n)$
10. $\sqrt{n} \in O(n)$
11. $n \in O(1 + n \log n)$
12. $n \in O(n^2)$
13. $(\log_2 n)^3 \in O(n^{1/4})$
14. $\frac{1}{6}n^3 \in O(n^3)$
15. $\binom{n}{3} \in O(n^3)$
16. $2^n \in O(3^n)$.
17. $n^{16} \in O(2^n)$.

Most of these are mindless calculus exercises. For instance, $\log n \leq n$ for $n \in \mathbf{N}$ can be proved by induction or calculus (or eye-balling a function plot), at which point order-consistency applies.

Example 1.3
1. $n^2 \notin O(n)$
2. $n^2 \notin O(n \log n)$
3. $n \notin O(\log n)$
4. $n^{1/4} \notin \log n$
5. $3^n \notin O(2^n)$

The behaviour of O on polynomials can be summarised in the following useful result:

Theorem 1.3 $a_k n^k + a_{k-1} n^{k-1} + \cdots + a_1 n + a_0 \in O(n^k)$ for $n \geq 1$.

Proof. Let $a = \max\{a_0, \ldots, a_k\}$. Then every term is bounded as $a_i n^i \leq a n^i$. Moreover, $n^i = n^i 1^{k-i} \leq n^i \cdot n^{k-i} = n^k$ for all i because $n \geq 1$. Thus,

$$a_k n^k + a_{k-1} n^{k-1} + \cdots + a_1 n + a_0 \leq a n^k + a n^k + \cdots + k a n^k + k a n^k = k a n^k \in O(n^k).$$

□

Example 1.4 We have

$$\binom{n}{2} \in O(n^2) \quad n \in \mathbf{N},$$

because $\binom{n}{2} = \frac{1}{2} n(n-1)$.

Example 1.5 (Warning ⋆) Note that according to Definition 1.1, we have $n^2 \notin O(\binom{n}{2})$ for $n \in \mathbf{N}$. (This is because $\binom{1}{2} = 0$, so there is no way to make $1^2 \leq c\binom{1}{2}$ no matter how large c is chosen.)

Exercise 1.1 Give a O-estimate for $\binom{n}{3}$.

1.1.1 Non-examples

Here are some examples of $f \notin O(g)$.

Example 1.6 $4^n \notin O(2^n)$.
This is a popular misunderstanding, possibly induced by the (correct) claim that $2 \cdot 2^n \in O(2^n)$ and some confusion of $2 \cdot (2^n)$ and $(2 \cdot 2)^n$.
Another (faulty) argument is idea of applying the logarithm to both 4^n and 2^n, arriving at $n \log 4$ and $n \log 2$ and using the (correct) statement that $n \log 4 \in O(n \log 2)$. That reasoning is not valid.
Yet another mistaken argument is to reason as follows: Define function $f(n) = 2^n$. Then $f(2n) = 2^{2n} = 4^n$. (Correct so far.) But for every constant α, we have $f(\alpha n) =^? O(f(n))$, so $4^n = f(2n) =^? O(f(n)) = O(2^n)$. But no such rules exist, and I've marked the false statements with in the above argument with '?'.

To see that indeed $4^n \notin O(2^n)$, assume the opposite, namely $4^n \in O(2^n)$. Then, by definition there exists a constant $c > 0$ such that $4^n \leq c2^n$ for all $n \geq 1$. Dividing by 2^n we have $c \geq 2^n$, but this is false for $n \geq \lceil \log_2(c+1) \rceil$.

Example 1.7 $n^2 \notin O(n)$.
This is just plain false, despite its symmetrical claim, $n \in O(n^2)$ being true.

The above counterexamples hold no matter which definition of O you use. The following counterexample is specific for *our* definition.

Example 1.8 ⋆ Define O as in Definition 1.1. Then $n \notin O(n \log n)$ for $n \geq 1$.
(Why? If $n \leq cn \log n$ for all $n \neq 1$ then in particular, $1 \leq c1 \log 1 = 0$, which is absurd.)
Similarly, $O(\log n) \neq O(1 + \log n)$ for $n \geq 1$.

1.2 Asymptotic Equality: Big Theta

The notation $f \in \Theta(g)$ is a "fuzzy version of $f = g$" or a "symmetric version of O."

Definition 1.2 $f \in \Theta(g)$ if $f \in O(g)$ and $g \in O(f)$.

Example 1.9 For $n \in \mathbf{N}$, some usage examples are:

$$n + 1 \in \Theta(n), \quad n^2 - n + 5 \in \Theta(n^2),$$

and also

$$2n \in \Theta(n),$$

but

$$n^2 - n + 5 \notin \Theta(n).$$

Theorem 1.4 (as a relation) The relation between f and g when $f \in \Theta(g)$ is an equivalence relation:

reflexive: $f \in \Theta(f)$

symmetric: if $f \in \Theta(g)$ then $g \in \Theta(f)$

transitive: if $f \in \Theta(g)$ and $g = \Theta(h)$ then $f = \Theta(h)$.

1.3 Conventions

The curious expression $O(1)$ means "any positive constant." Directly from the defintion it is the set of functions f such that there is a number $c > 0$ with $f(n) \leq c \cdot 1$. Examples are cos and sin (both of which are bounded by 1. Typically, it is used for "some constant, maybe 17 or 18 or something."

Expressions like $O(3n)$ makes sense, but are confusing and unprofessional and should be avoided; the expression $O(3n)$ means the same as $O(n)$.

Similarly, don't write "the number of comparisons is $\sim 3n^2 - 5n + 7$." While it may not be *false*, this expression signals lack of mastery of the notation, you should write "the number of comparisons is $\sim 3n^2$."

Similarly, you should reduce expressions inside a O: The expression $O(n^3 + n^2(n + 16n + \log n))$ is equal to $O(n^3)$; the two expressions mean exaclty the same. Since the whole point of O was to reduce clutter, the latter form is strongly preferred. (The former form signals limited understand of how and why the notation is used.)

Most users of this notation are mainly interested in performance guarantees in terms of upper bounds, so one often sees O used where Θ would have been just as appropiate.

1.4 Daily Use

1.4.1 Basic programme analysis

Example 1.10 Here is a programme that counts the 0s in an array a of length $n \geq 1$:

```
c ← 0
for i ∈ {1,...,n}
    if a[i] = 0
        c ← c + 1
```

We want to determine the number of assignments.
The first line use exacly 1 assignment. The iteration is more interesting. Depending on the value of $a[i]$, the **if**-branch costs 0 or 1 assignments. In the worst case, this is

$\max\{0,1\} = 1$, which is the worst-case cost for the body of the loop. The for-expression implictly stands for n assigments $i \leftarrow 1, \ldots, i \leftarrow n$ and executes the body n times, for a total of $2n$ assignments.
(Arguably, these analyses are so banal that it's hard to see what the notation accomplishes. It gets more interesting now.)
Adding the 1 contributions from the first line we arrive at $2n + 1 = O(n)$ for $n \geq 1$. We used O-notation only at the end, to simplify the expression. We could also have written $2n + 1 = \Theta(n)$

Example 1.11 Let us make the above piece of code slighly more interesting to show off O. Assume now that a is some data structure containing n elements, with $n \geq 1$. Further assume that *a.popleft*() is known to take $O(m)$ time when a contains m elements. (Think of a as an array-based list.) The $O(m)$ bound may come from a previous analysis, or a textbook, or the library documentation.

```
c ← 0
for i ∈ {1,...,n}
    if a.popleft() = 0
        c ← c + 1
```

Evaluating the **if**-statement now costs $O(m)$ for $m \in \{n, n-1, \ldots, 1\}$. With foresight, we bound each of them as $O(n)$, using the fact that $m \leq n$ so that $O(m) \subseteq O(n)$ by monotonicity of O. Including the 1-contribution from the assigment, the whole **if**-statement costs $1 + O(n) = O(1) + O(n) = O(n+1) = O(\max n, 1) = O(n)$. The **for**-loop now costs $n \cdot (1 + O(n))$, which simplifies to $O(n^2)$ for $n \in \mathbf{N}$. Adding the contribution from the first line does not change this.

Note that in this example, it was crucial that we were freely able to manipulate the $O(m)$-expression for the beviour of *popleft* that we inherited.

1.4.2 Syntactic analysis

It is possible to formalise the syntactic approach by writing down general rules for determining the cost $\text{cost}_n(P)$ of programme P and parameter n.

Assignment: If $\text{cost}_n(x \leftarrow a)$. All other atomic statements cost 0.

Sequence: If $P = Q;R$ with $\text{cost}_n(Q) = f$, $\text{cost}_n(R) = g$ then $\text{cost}_n(P) = f + g$.

Selection: If $P =$ **if** b **then** Q **else** R with $\text{cost}_n(Q) = f$, $\text{cost}_n(R) = g$ and $\text{cost}_n(b) = h$ then $\text{cost}_n(P) = b + \max(f, g)$.

Iteration: If $P =$ **for** $i \in I$ **do** Q then $\text{cost}_n(P) = |I|(1 + \text{cost}_n(Q))$.

However, there is no strong tradition for this degree of formalism in the analysis of algorithms, so we will not pursue it further.

1.4.3 Conventions

Example 1.12 (Placeholder) When we say "Insertion sort on $n \geq 1$ elements costs $O(n^2)$", or "The running time of Insertion sort on $n \geq 1$ elements is $O(n^2)$" we mean "The cost of Insertion sort on $n \geq 1$ elements is given by a cost function $f \in O(n^2)$." This is just convenient type, despite the "type error" of pretending that a cost is a *set of functions* rather than a function. A way of expressing this both pleasantly and properly would be "The running time of Insertion sort on $n \geq 1$ elements *belongs to* $O(n^2)$."

1.5 More about Logarithms

We want to say something more general about the relationship between logarithms and power functions. Such expressions appear frequently in the analysis of algorithms, for instance when comparing shellsort $O(n^{3/2})$ to mergesort $O(n \log n)$, or $\sqrt{n}$-trees to balanced binary search trees.

Theorem 1.5 (ln versus power) For all exponents $a > 0$,

$$\ln n = O(n^a), \qquad (n \geq 1).$$

Unfortunately, I know no short and easy proof for these relationships.

Proof. We have

$$\frac{1}{y} \leq y \qquad (\text{for any } y \geq 1).$$

Thus, for any $z \geq 1$,

$$\int_1^z \frac{1}{y}\, dy \leq \int_1^z y\, dy,$$

so that

$$\ln z \leq \tfrac{1}{2} z^2.$$

In particular, if we choose $z = n^{a/2}$, then we have $\ln n^{a/2} \leq \frac{1}{2}(n^{a/2})^2$, which simplifies to

$$a \ln n \leq n^a \qquad (n \geq 1).$$

In other words, $\ln n \leq \frac{1}{a} n^a$. Conclusion by order-consistency and scale-invariance of O. □

An even more general bound is the following:

Theorem 1.6 (polylog versus power) For all bases $r > 1$ and exponents $a > 0$ and $k > 0$,

$$(\log_r x)^k = O(x^a).$$

Proof. The proof is almost identical to the previous; set $z = n^{a/2k}$, so that $\ln n^{a/2k} = \frac{1}{2}(n^{a/2k})^2 = \frac{1}{2} n^{a/k}$. We arrive at

$$\frac{a}{k} \ln n \leq n^{a/k},$$

so $\ln n \leq (k/a)n^{a/k}$ and thus

$$(\log_r n)^k = \left(\frac{\ln n}{\ln r}\right)^k \leq \left(\frac{k}{a \ln r}\right)^k n^a;$$

note that r, a, and k are constants so scale-invariance makes short work of them. □

1.6 Big Oh Defined in General

We now define O for the general case, in terms of linear dominance.

Definition 1.3 (O **as linear dominance**) Let X be a set and consider $f, g\colon X \to \mathbf{R}_{\geq 0}$. Then $f = O(g)$ if there exists a positive real number $c > 0$ such that that $f(x) \leq cg(x)$ for all $x \in X$.

Definition 1.1 is the special case $X = \mathbf{N}$ and suffices for many analyses. We need the full definition in two contexts that often arise.

Restrict the domain. For some functions, we need to be able to restric the domain, for instance, we want to talk about $O(\log n)$ or $O(\binom{n}{2})$, both of which are problematic for $n = 1$. (They are both 0, so under the simplified Definition 1.1, we unfortunately have both $1 \notin O(\log n)$ and $n^2 \notin O(\binom{n}{2})$, because of the behaviour at $n = 1$, neither of which is helpful.) We can solve this by setting $X = \{2, 3, \ldots\} = \mathbf{N} - \{1\}$.

Extend the domain. Many data structures and algorithms need expression in two variables, such as the number n of vertices and the number m of edges, or the number k of queries to a data structure of size n. In those situations, we want to set $X = \mathbf{N} \times \mathbf{N}$ and work with expressions such as $O(m(m+n))$ or $O(k(1+\log n))$.

1.6.1 Daily use

Specifying the domain X sometimes leads to clumsy expressions, and we will prefer to define X implictly by always stating bounds on our parameters.

Example 1.13 (**Implicit domain specifation**) We want two write human-readable expressions like the following:

> **Claim.** Mergesort of $n \geq 2$ elements takes time $O(n \log n)$.

What this claim expresses more formally is that if $X = \{2, 3, \ldots\}$ and $f\colon X \to \mathbf{R}_{\geq 0}$ is such that $f(n)$ denotes the cost of Merge Sort on instances of size $n \in X$ then $f \in O(n \log n)$.

Our use here aims to strike a compromise between on one side readibility and consistency with established notation, and on the other side precision and meaning. A formally satisfying development of Definition 1.3 would require separate O-sets for every domain X,encumbering the notation with expressions like O_X, $O_{\mathbf{N}\times N}$. The reader is referred to the thesis of Rutanen for such an approach.

Example 1.14 (Careful with small n) Note that the stronger claim that "Mergesort on $n \geq 1$ elements takes time $O(n \log n)$" in particular means that it takes *no* time for $n = 1$. This may be true or not, depending on the cost function or implementation. (For instance, an implementation may indeed spend zero array accessses when it can determine that $n = 1$.) But we have to be careful.

1.6.2 Important functions

Example 1.15 (Binomial coefficient revisited) We return to the example from 1.5 and finally establish

$$\binom{n}{2} = \Theta(n^2) \quad \text{for } n \geq 2.$$

It remains to show $n^2 \in O(\binom{n}{2})$ for $n \geq 2$. But if $n \geq 2$ then $n-1 \geq \frac{1}{2}n$, so $\binom{n}{2} = \frac{1}{2}n(n-1) \geq \frac{1}{4}n^2$. Thus, $n^2 \leq 4\binom{n}{2}$, so $n^2 \in O(\binom{n}{2})$ from the definition (with $c = 4$).

Example 1.16 (Expressions with logarithms) The following hold for $n \geq 2$:

1. $1 \in O(\log n)$
2. $1 \in O(\log^* n)$
3. $n \in O(n \log n)$

The following hold for $n \geq 3$:

1. $1 \in O(\log \log n)$
2. $n \in O(n \log \log n)$

1.6.3 Multivariate functions

In the following example, assume that a has length $m \geq n$ and recall that we assume *popleft* takes $O(s)$ time if a has s eleements.

```
c ← 0
for i ∈ {1,...,n}
    if a.popleft() = 0
        c ← c + 1
```

Assume $n \geq 1$. The costs for each of the n executions of the *if*-statement are $O(m)$, $O(m-1)$, …, $O(m-n)$, respectively, which we bound by $O(m)$. The cost for increasing c does not change this, since $m \geq n \geq 1$. The total cost becomes $1 + n \cdot O(m) = O(nm)$.

What happens *formally* in the above analysis is that $X = \mathbf{N} \times \mathbf{N}$ and the cost function has the form $f : \mathbf{N} \times \mathbf{N}$. Specifically, the cost of the **if**-statement is $f_{\mathbf{if}}(n, m) \in O(m-n) \subseteq O(m)$. The cost of the **for**-loop is $f_{\mathbf{for}}(n, m) = n \cdot f_{\mathbf{if}}(n, m) \in n \cdot O(m) = O(mn)$. This kind of precision is not normally used.

1.7 A word of warning

Definition 1.3 is *not* the standard definition of O found in most textbooks. (We define O in terms of "linear dominance"; the standard definition uses "asymptotic linear dominance on $\mathbf{N}$".)

For completeness, here is the standard definition:

Definition 1.4 **(Asymptotic linear dominance)** Let $f, g\colon \mathbf{N} \to \mathbf{R}_{\geq 0}$. Then $f = O_{\mathrm{a}}(g)$ if there exists a positive real number $c > 0$ and an integer $N \in \mathbf{N}$ such that that $f(n) \leq cg(n)$ for all $n \geq N$.

(There are a number of immaterial variations of the definition (using $\mathbf{N}_0 = \{0, 1, \ldots\}$ instead of $\mathbf{N}$, or requiring $|f(n)| \leq cg(n)$.))

For functions like n, n^2, it does not matter whether we use Definition 1.1 or 1.4, indeed $O(n^2) = O_{\mathrm{a}}(n^2)$ for $n \in \mathbf{N}$. The difference appears mainly in the behaviour of functions that vanish for small n, such as $\log n$, $n \log n$, and $\binom{n}{2}$. For instance $\log 1 = 0$, $1 \log 1 = 0$, and $\binom{1}{2} = 0$ so that $O(n \log n) \neq O_{\mathrm{a}}(n \log n)$ for $n \in \mathrm{N}$.

If we were using the standard definition 1.4, we could gloss over these details. However, in daily use, a stipulation of "for large enough n" is understood and few practictioners would ever raise an eyebrow at seeing the expression $n \in O(n \log n)$ without the (formally crucial) restriction to $n \geq 2$ or much care about which definition has been used.

Chapter 2

Tilde

This section was an attempt to carefully explain limits before tildes, because limits were also useful to define O (or so it seemed to be before 2022.)

By the much better definition in the previous chapter, this all may be a waste of effort. In any case, it's a draft. Don't trust a word of it.

2.1 Asymptotic equality: tilde

The notation $f(x) \sim g(x)$, sometimes abbreviated to $f \sim g$, is a "fuzzy version of $f = g$."

Definition 2.1 (Tilde, informally) We write $f(x) \sim g(x)$ when $f(x)/g(x)$ approaches 1 as x grows.

Example 2.1 Som usage examples are

$$n+1 \sim n, \quad \binom{n}{3} \sim \tfrac{1}{6}n^3, \quad 3n\log n + 100n \sim 3n\log n.$$

To get a feeling for this, evalute the corresponding fractions for $n = 1, 10, 100, 1000, \ldots$ and and use your good judgement. For instance, $n+1 \sim n$ because

$$\frac{n+1}{n} \quad \text{grows as} \quad \frac{2}{1} = 2, \frac{11}{10} = 1.1, \frac{101}{100} = 1.01, \frac{1001}{1000} = 1.001, \ldots.$$

Expecting no dramatic changes for larger n, we can confidently state that this sequence of numbers approaches 1.
On the other hand

$$2n \nsim n, \quad n^2 - n + 5 \nsim n.$$

For instance,

$$\frac{2n}{n} \quad \text{grows as} \quad \frac{2}{1}, \frac{20}{10}, \frac{200}{100}, \frac{2000}{1000}, \ldots$$

all terms of which are equal to 2. This does not approach 1. (Instead, it approaches 2.)

Exercise 2.1 Argue by by manual computation, using an electronic calculator, or writing a programme, that

1. $n \sim n-1$
2. $\binom{n}{3} \sim \frac{1}{6}n^3$.
3. $3n\log_2 n + 100n \sim 3n\log_2 n$.
4. $\sqrt{n} \nsim n$.
5. $n^2 - n + 5 \nsim n$.
6. $\log_2 n \nsim n$.
7. $7 \nsim 5$.

Exercise 2.2 True of false?

1. $n \sim n + \frac{1}{n}$
2. $\log_2 n \sim \ln n$
3. $n^{10000} \sim n^{10001}$

The notation $f \sim g$ is read as "eff is asymptotically equal to gee", or just "eff is tilde gee" among friends.

Theorem 2.1 **(as a relation)** The relation $\sim$ is an equivalence relation on the set of real functions. That is, it is

symmetric: $f \sim g$ if and only if $g \sim f$

reflexive: $f \sim f$

transitive: if $f \sim g$ and $g \sim h$ then $f \sim h$

Theorem 2.2 **(Rules)** Tilde notation satisfies the following rules:

1. If $f_1 \sim g_1$ and $f_2 \sim g_2$ then $f_1 \cdot f_2 \sim g_1 \cdot g_2$.
2. $n + C \sim n$ for every constant C
3. $n + \log_a n \sim n$ for any base $a > 1$.
4. $a_k n^k + a_{k-1} n^{k-1} + \cdots + a_1 n + a_0 \sim a_k n^k$
5. $\lfloor x \rfloor \sim \lceil x \rceil \sim x$

On the other hand,

1. $\log_b n \nsim \log_a n$ *for* $a \neq b$

2. $Cn \nsim n$ *for every constant* $C \neq 1$
3. $n^a \nsim n^b$ *for* $a \neq b$

In summary, think of $\sim$ as some fuzzy equality symbol. The most useful fuzziness is that we can disregard lower order terms (but have to pay attention to the constant in front of the highest-order term.)

Warning: the rest of this section is a draft and may contain errors.

Tilde notation is defined in terms of limits, so we consider limits first.

2.2 Limits

Let f be a function from $\mathbf{R}$ to $\mathbf{R}$. Then the *limit at infinity* of f (or just *limit* in these notes), written

$$\lim_{x\to\infty} f(x)$$

is one of four things: a real number, ∞, $-\infty$, or undefined. All our limits are at infinity, so we will drop the "$x \to \infty$." Inuitively, the limit is "what $f(x)$ approaches arbitrarily closely as x grows." For instance,

$$\lim \frac{1}{x} = 0, \qquad \lim\left(1 - \frac{1}{x}\right) = 1\,.$$

Pathologically, $\lim 7 = 7$. Not all functions have limits, for instance $\lim \sin x$ is meaningless and undefined. (The function oscillates between -1 and 1, never approaching a value.)

For all function we will encounter, the limit is easily determined.

The *formal* definition is as follows. For real number c we say that $\lim f(x) = c$ if for every $\delta > 0$, there is a real number b such that

$$|f(x) - c| \leq \delta \qquad \text{for } x > b\,.$$

We say that $\lim f(x) = \infty$ if for every B where is a real number b such that

$$f(x) > B \qquad \text{for } x > b\,.$$

Theorem 2.3 **(Laws for limits at infinity)** Assume that $\lim f(x)$ and $\lim g(x)$ exist.

Constant: $\lim c = c$ for any real c.

Constant factor: $\lim af(x) = a \lim f(x)$ for any real $a \neq 0$.

Additive: $\lim(f(x) + g(x)) = \lim f(x) + \lim g(x)$.

Multiplicative: $\lim(f(x) \cdot g(x)) = \lim f(x) \cdot \lim g(x)$.

Domination If $f(x) \leq g(x)$ for $x > b$ then $\lim f(x) \leq \lim g(x)$.

These laws sometimes give nonsensical results such as ∞/∞ or $5/0$, in which case you can deduce nothing from them.

Some proofs. For the second claim, let $c = \lim f(x)$ and let $\delta > 0$ be given.

Choose b such that $|f(x)-c| \le \delta/|a|$ for $x \ge b$. Then we have

$$|af(x)-ac| = |a(f(x)-c)| = |a| \cdot |f(x)-c| \le |a|\frac{\delta}{|a|} = \delta.$$

To establish additivity, let $\lim f_1(x) = c_1$ and $\lim f_2(x) = c_2$. We want to prove $\lim f_1(x) + f_2(x) = c_1 + c_2$. Let $\delta > 0$ be given and choose b_1 and b_2 such that

$$|f_i(x) - c_i| \le \tfrac{1}{2}\delta \qquad \text{for } x \ge b_i.$$

But then, for $x \ge \max b_1 + b_2$, we have

$$|(f_1(x)+f_2(x)) - (c_1+c_2)| = |f(x) - c_1 + f_2(x) - c_2| \le$$
$$|f(x)-c_1| + |f_2(x)-c_2| \le \tfrac{1}{2}\delta + \tfrac{1}{2}\delta = \delta,$$

using the triangle inequality (which states that $|x+y| \le |x| + |y|$.) The proof for multiplicativity is similar, but algebraically more involved. The exponentiation proof for integral n is induction. □

The calculation of limits is sometimes very easy (such as $\lim x$ or $\lim 7$ or $\lim 1/x$). We also have

$$\lim(a_k n^k + \cdots + a_1 n + a_0) =$$
$$\lim a_k n^k \cdot \Big(1 + \frac{a_{k-1}}{a_k n} + \cdots + \frac{a_1}{a_k n^{k-1}} + \frac{a_0}{a_k n^k}\Big),$$

and observing that each of the terms in the parenthesis involving n vanishes as n tends to infinity, the entire expression equals $\lim a_k n^k$.

Sometimes, these claim use nontrivial calculus, even for the relatively harmless functions that appear in the analysis of algorithms. For instance, depending on you mathematical maturity, arrogance, or sloppiness, the claim

$$\lim \frac{\ln n}{n} = 0,,$$

is either "clear," requires you to enter the expression into a graphing calculator, write a computer program to calculate the value for large n, or come up with the following:

Theorem 2.4

$$\lim \frac{\ln n}{n} = 0,.$$

Proof. Using calculus, we have

$$\ln n = \int_1^n \frac{1}{y}\,dy \le \int_1^n \frac{1}{\sqrt{y}}\,dy = \Big[2\sqrt{y}\Big]_1^n < 2\sqrt{n}.$$

Thus,

$$\frac{\ln n}{n} \le \frac{2\sqrt{n}}{n} = \frac{2}{\sqrt{n}},$$

which tends to 0 as n grows.

(There are many other ways to prove this.) □

Discussion Aren't we done? Can't we define the running time of an algorithm as $\lim T(x)$? For instance, because of the polynomial property, the complicated running time of three nested for-loops, $\binom{n}{3} = \frac{1}{6}n^3 - \frac{1}{2}n^2 + \frac{1}{3}n$, would be given as its limit,

$$\lim \tfrac{1}{6}n^3 - \tfrac{1}{2}n^2 + \tfrac{1}{3}n = \lim \tfrac{1}{6}n^3 .$$

Alas, this is not good enough, because $\lim \frac{1}{6}n^3 = \infty$. In fact, all nonconstant running times $(\log n, n, \frac{1}{6}n^3, 2^n, \ldots)$ have the *same* limit (namely, infinity). There would be only two different running times: constant and infinite.

2.3 Tilde

Definition 2.2 (Tilde) For functions, $f, g\colon \mathbf{R} \to \mathbf{R}$ we say that f is *asymptotically equal* to g, in symbols

$$f(x) \sim g(x)$$

if

$$\lim_{x\to\infty} \frac{f(x)}{g(x)} = 1 .$$

[Tilde]

Example 2.2 The above makes no sense whenever $g(x)$ is zero infinitely often as x tends to infinity. An example of such a function is $\sin(x)$; it vanishes whenever x is a multiple of π. Luckily for us, such functions do not appear in the analysis of algorithms (running times are not 0), and we could proceed by just ignoring this issue. Formalists will want to read every occurrence of "function" as a "function that is eventually nonzero," i.e., "function f for which there exists a value x_0 such that $f(x) \neq 0$ for $x \geq x_0$." Such formalists will want to convince themselves that all functions appearing the analysis of algorithms (polynomials, logarithms, exponentials and their compositions) satisfy this requirement, except for cases such as $n - n$. Developing this carefully is neither hard nor illuminating, so we will overlook it.

Theorem 2.5 $\sim$ is an equivalence relation on the set of real functions. That is,

1. $f(x) \sim g(x)$ iff $g(x) \sim f(x)$ (symmetry),
2. $f(x) \sim f(x)$ (reflexivity),
3. if $f(x) \sim g(x)$ and $g(x) \sim h(x)$ then $f(x) \sim h(x)$ (transitivity).

Proof. Directly from the definition and rules for limit. For instance,

$$\lim_{x\to\infty} \frac{f(x)}{h(x)} = \lim_{x\to\infty} \frac{f(x)}{h(x)}\frac{g(x)}{g(x)} = \lim_{x\to\infty} \frac{f(x)}{g(x)}\frac{g(x)}{h(x)} =$$

$$\left(\lim_{x\to\infty} \frac{f(x)}{g(x)}\right) \cdot \left(\lim_{x\to\infty} \frac{g(x)}{h(x)}\right) = 1 \cdot 1 = 1 .$$

establishes transitivity except if you worry about $g(x) = 0$. □

Theorem 2.6 **(multiplication, division)** If $f_1(x) \sim g_1(x)$ and $f_2(x) \sim g_2(x)$ then

1. $(f_1 \cdot f_2)(x) \sim (g_1 \cdot g_2)(x)$.
2. $(f_1/f_2)(x) \sim (g_1/g_2)(x)$.

Proof. Directly from the rules for limits. □

Theorem 2.7 **(logarithm)** If $\lim g(x) = \infty$ and $f \sim g$ then $\log f \sim \log g$.

Proof. Follows from Propositions 3.2 and **??** below. □

This does not work in the other direction: $f \sim g$ does *not* imply $2^f \sim 2^g$. It is not true in general that $f \sim g$ implies $\log f \sim \log g$.

Despite these handy tools, we often have to go back directly to the definition. To establish

$$n + \ln n \sim n,$$

we need the calculation

$$\lim \frac{n + \ln n}{n} = \lim \frac{n}{n} + \lim \frac{\ln n}{n} = 1 + 0 = 1,$$

using the laborious argument from Proposition 2.4.

Chapter 3

Retired stuff

Don't read this, it will burn your eyes and rot your soul.

3.1 Why do this?

We begin by introducing notation, intuition, and rules for use. Precise definitions and explanation are given later.

First, though: Why are we doing this at all? We need a toolset to compare functions, because we want to compare the running times of algorithms, which are given as functions (typically of the input size). We both want to be able to say "f is roughly equal to g," and "f is bounded by g" for a certain granularity of "roughly" that is both precise enough to be interesting and coarse enough to be useful.

Note that we *do* already a definition for have "precisely equal," which is typically written as $f = g$ and defined as

$$f(x) = g(x) \qquad \text{for all } x.$$

That was easy to define, but it is also quite useless for us; if $f(n) = n$ and $g(n) = n + 1$ then $f \neq g$ (but we want to treat those two running times as "the same"). Similarly we *do* already a definition for "bounded by," typically written as $f \leq g$ and defined as

$$f(x) \leq g(x) \qquad \text{for all } x.$$

This is also quite useless: if $f(n) = n^2$ and $g(n) = n^3$ then $f \not\leq g$ because $(-1)^2 \not\leq (-1)^3$. (But we really want to be able to say that a quadratic-time algorithm is better than a cubic-time one.)

3.1.1 Big Oh as a limit

Theorem 3.1 Let $f, g\colon \mathbf{R} \to \mathbf{R}$ with g nonnegative and such that $\lim f$ exists. Then

$$f = O(g)$$

if and only if

$$\lim \frac{|f(x)|}{g(x)} < \infty.$$

Proof. Assume $f(x) = O(g(x))$ and thus pick b such that $|f(x)| \leq |g(x)| = g(x)$ for all $x > b$. In particular

$$\frac{|f(x)|}{g(x)} \leq 1\,,$$

so

$$\lim \frac{|f(x)|}{g(x)} \leq 1 < \infty.$$

For the other direction, first assume $\lim |f(x)|/g(x) = B$ for some real number B. This means that there exist an integer b such that for $x > b$,

$$\left|\frac{|f(x)|}{g(x)} - B\right| \leq 1.$$

In particular,

$$\frac{|f(x)|}{g(x)} - B \leq 1\,,$$

which can be rearranged to

$$|f(x)| \leq (1 + B)g(x) = (1 + B)|g(x)|.$$

For the final case, i.e., $\lim |f(x)|/g(x) = -\infty$, we can repeat the above argument with $B = 0$. □

We can view this an alternative definition of O, closer in spirit to the defintion of $\sim$, in that it expresses the relationship between f and g in terms of the limit of f/g.[1]

After all this work, we can finally establish the relationship between $\sim$ and O:

Theorem 3.2 Let $f, g\colon \mathbf{R} \to \mathbf{R}$ with g nonnegative and such that $\lim f$ exists. If $f \sim g$ then $f = O(g)$.

We repeat that the other direction is not even close to being true: $2x = O(x)$ but $2x \not\sim x$.

3.2 Hints and pitfalls

When I read summaries like the above, my brain can't stop subvocalising "How very useful! Whatever makes the notation look neat is also be true; and somebody else has proved it for us. Why bother."

So here are some warnings.

Misapplied notation. Avoid writing "the number of comparisons is $\sim 3x^2 - 5x + 7$." While not *false*, this signals lack of mastery of the notation, you should write "the number of comparisons is $\sim 3x^2$." (The whole point of the notation is to allow you to avoid the smaller terms.) Similarly, don't write $O(n^2 + n)$, or $O(7n^2)$; do write $O(n^2)$, even though these expressions all mean the same. (As does $O(100034245n^2 + n \log n + \sqrt{n})$.) You're not doing yourself any favour (nor your reader) by not cleaning that up.

[1] A veritable curnucopia of related concepts exists and is used For instance, when $\lim f/g = 0$ we write $f = o(g)$. When $\lim g/f < \infty$ we write $f = \Omega(g)$. When $\lim f/g = B$ for some real number we write $f = \Theta(g)$.

Everything is a constant. Check this out: "$1 + 2 + 3 + \cdot + n$ is just a sum of constants. Each can be bounded by $O(1)$, so we can rewrite the sum to $O(1) + O(1) + \cdot + O(1) = O(n)$." The argument is false, and the conclusion is false as well. One thing wrong with the argument is that the terms are not constant, they depend on n. Indeed, half of them are larger than $\frac{n}{7}2$, which is certainly not $O(1)$. The other mistake is using O-expressions to the left of an equality sign:

Reading big-Oh backwards. It is true that $2n = O(n)$ and $17n + 4 = O(n)$. But this "=" does not behave as the one you're used to. So you cannot infer $2n = 17n + 4$, that's still false. In general, aim at having only a single O-in your calculations, and always at the end. Never write $O(n) = 3n$, it means nothing.

Exponents don't like asymptotics. The argument "$6^x = O(2^x)$ because 6 is only a constant times 2" sounds tempting but the claim is simply *false*. (In particular $6^x \neq 3 \cdot 2^x$.)

Broken logarithm rule. This may be nit-picking, but: Define $f, g\colon \mathbf{R} \to \mathbf{R}$ as $f(n) = 1 + \frac{1}{n}$ and $g(n) = 1 + \exp(-n)$ and note that

$$\lim f(n) = \lim g(n) = \lim \frac{f(n)}{g(n)} = 1$$

so that both $f(n) \sim g(n)$ and $f(n) = O(g(n))$. However, using calculus one can verify

$$\lim \frac{\log(1 + \frac{1}{n})}{\log(1 + \exp(-n))} = \lim \frac{\log(\frac{1}{n})}{\log \exp(-n)} = \lim \frac{\exp(n)}{n} = \infty$$

so $\log f(x)$ is not $O(\log g(n))$. Compare this to Propositions **??** and **??**. (It was important that $g(x) \to \infty$.)

3.3 Multivariate asymptotics

To be written.

The motivation is that we freely use expressions like $O(n+m)$ in the analysis of, say, graph algorihtms, but the concept of a *bivariate* asymptotic notation is never defined.