

# Some Useful Math for the Analysis of Algorithms

Thore Husfeldt

Revision 2b39552... , Fri May 17 10:27:38 2013 +0200, Thore Husfeldt

This note covers some of the mathematical preliminaries needed to enjoy [SW], the textbook *Algorithms*, 4th ed. by Sedgewick and Wayne and is focussed on the B.Sc. course Algorithms and Data Structures (BADS) at ITU Copenhagen.

## Some functions encountered in the analysis of algorithms

### Exponentiation

For a positive integer  $n$  (the *exponent*) and a positive real number  $b$  (the *base*) we define a shorthand for iterated multiplication, called exponentiation:

$$b^n = \underbrace{b \times b \times \cdots \times b}_{n \text{ times}}.$$

(Most of the time  $b$  will just be 2, sometimes it's 10. In maths and physics, it's often the number  $e = 2.7182\dots$  and one then uses the notation  $\exp(n)$  for  $e^n$ , called *the exponential function*.)

**Exercise 1.** For  $b > 1$  and  $n, m \geq 1$ , which is true:  $b^{nm} = b^n + b^m$ ?  $b^{nm} = (b^n)^m$ ?

**Exercise 2.** For  $n, m \geq 1$ , true or false:  $b^{n+m} = b^n b^m$ ?  $b^{n-m} = b^n / b^m$ ?

**Exercise 3.** For  $a, b > 1$  and  $n, m \geq 1$ , which is true:  $(ab)^n = a^n + b^n$ ?  $(ab)^n = a^n b^n$ ?  $(a + b)^n = a^n + b^n$ ?

**Exercise 4.** What would be a useful convention for what  $b^0$  should mean?

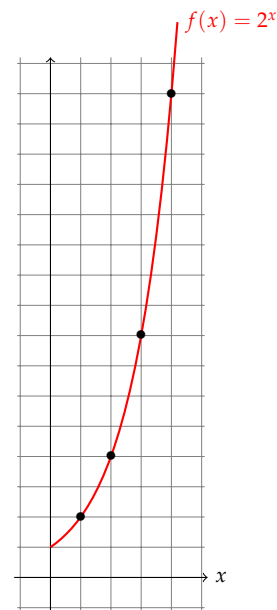
**Exercise 5.** Express as a power of 2:  $2 \cdot 2^2$ ,  $(2^2)^3$ ,  $2^{(2^2)}$ ,  $2^n + 2^n$ .

There is a way to define what  $b^x$  should mean even when  $x$  is negative, or not even an integer, so that  $x \mapsto b^x$  becomes a smooth function defined for all reals that agrees  $b^n$  when  $x = n$  is an integer. This definition requires calculus and lies outside of our scope. The notation makes good sense all the way, for example  $b^{-1}$  equals  $1/b$  so that  $b^{N+1}b^{-1} = b^N$ , and  $b^{\frac{1}{2}}$  equals  $\sqrt{b}$  so that  $(b^{\frac{1}{2}})^2 = b^{\frac{1}{2} \cdot 2} = b$ .

### Positional notation

One of Civilisation's big breakthroughs is to use exponentiation to denote numbers in a very compact way. When we write 7143 we mean

$$7 \cdot 10^3 + 1 \cdot 10^2 + 4 \cdot 10^1 + 3 \cdot 10^0.$$



To see how clever this is, compare to the notation used in ancient Rome (MMMMMMMCXLIII), or worse, to unary notation. Our notation is called “digital” or “base 10” (because  $b = 10$ ). Positional notation is particularly clever when performing arithmetic, and many people are comfortable using this, having been trained to find even complicated operations like  $100 - 1 = 99$  natural. (Compare to the more straightforward  $c - i = ic$ .)

Binary notation is the positional number system using the base  $b = 2$ . It has only two numerals (called “binary digits” or “bits”), 0 and 1. The binary number 1101111100111 has value

$$2^{12} + 2^{11} + 2^9 + 2^8 + 2^7 + 2^6 + 2^5 + 2^2 + 2^1 + 2^0$$

which happens to be 7143 in base 10.

**Exercise 6.** Express 8 in binary. What is the binary number 10010 in base 10?

**Exercise 7.** Add the binary numbers 100101 and 1011 using the school algorithm for addition.

**Exercise 8.** Multiply the binary numbers 100101 and 1011 using the school algorithm for multiplication.

### Logarithms

The *binary logarithm*  $\lg N$  is defined for every positive real number  $N$  and satisfies  $2^{\lg N} = N$ . Roughly, “how often you need to double 1 to get  $N$ .”

More generally, the *logarithm base  $b$*  for any real  $b > 1$  is a function defined for all positive reals  $N$  by  $\log_b N = x$  where  $b^x = N$ . (To see that this does indeed define  $\log_b N$  uniquely we would need to observe that  $x \mapsto b^x$  is a bijection from the reals to the positive reals for  $b > 1$ . But at this point we have stopped pretending to give a rigorous presentation anyway.)

Note that  $\log_1 5$  or  $\lg 0$  and  $\lg -1$  are just meaningless collections of ink, like  $8 + \times$  or  $\sqrt[4]{\phantom{x}}$ .

**Exercise 9.** What is  $\lg 2$ ? What is  $\log_b b$ ?

**Exercise 10.** What is  $\lg 2^x$ ? What is  $\log_b b^x$ ?

**Exercise 11.** For  $b > 1$  and  $N, M > 0$ , which is true:  $\log_b(NM) = (\log_b N) + (\log_b M)$ ?  $\log_b(NM) = (\log_b N)(\log_b M)$ ?  $\log_b(N + M) = (\log_b N)(\log_b M)$ ?  $\log_b(N + M) = (\log_b N) + (\log_b M)$ ?

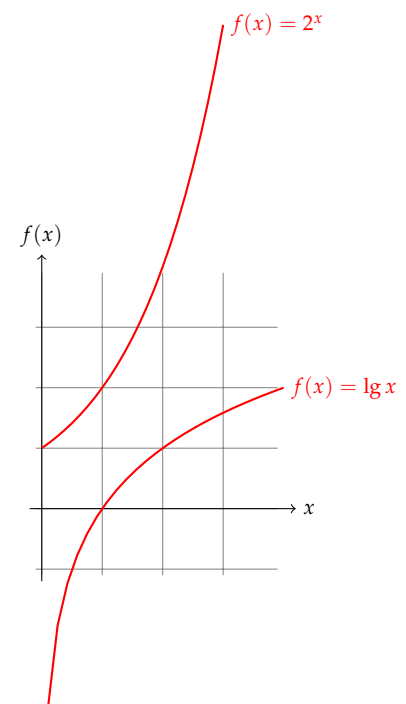
**Exercise 12.** Which is true:  $\log_b(N^x) = x \log_b N$ ?  $\log_b(N^x) = (\log_b N)^x$ ?

**Exercise 13.** True or false:  $\log_a N = \log_b N / \log_b a$ ?

**Exercise 14.** Find  $\lg 1024$ ,  $\lg \frac{1}{4}$ ,  $\lg 8$ .

**digit**, noun, from Latin *digitus* ‘finger, toe’. 1. Any of the numerals 0 to 9. 2. A finger or toe.

```
0 000
1 001
2 010
3 011
4 100
5 101
6 110
7 111
```



### Floors and ceilings

The *floor function*  $\lfloor x \rfloor$  maps any real number  $x$  to the largest integer less than or equal to  $x$ . Likewise, the *ceiling function*  $\lceil x \rceil$  maps any real number  $x$  to the smallest integer greater than or equal to  $x$ .

**Exercise 15.** Find  $\lfloor \frac{1}{2} \rfloor$ ,  $\lfloor -\frac{1}{2} \rfloor$ ,  $\lfloor \pi \rfloor$ ,  $\lfloor 7 \rfloor$ .

**Exercise 16.** Find  $\lceil \frac{1}{2} \rceil$ ,  $\lceil -\frac{1}{2} \rceil$ ,  $\lceil \pi \rceil$ ,  $\lceil 7 \rceil$ .

**Exercise 17.** Find  $\lfloor \frac{1}{2} + \lceil \frac{3}{2} \rceil \rfloor$ ,  $\lfloor \frac{1}{2} \cdot \lceil \frac{5}{2} \rceil \rfloor$ .

**Exercise 18.** For real  $x$ , true or false:  $x - 1 < \lfloor x \rfloor \leq x \leq \lceil x \rceil < x + 1$ .

**Exercise 19.** For all real  $x$ , true or false:  $\lceil -x \rceil = -\lfloor x \rfloor$ ,  $\lfloor -x \rfloor = -\lceil x \rceil$ .

**Exercise 20.** For reals  $x$  and  $y$ , true or false:  $\lceil x + y \rceil = \lceil x \rceil + \lceil y \rceil$ .  
 $\lceil xy \rceil = \lceil x \rceil \lceil y \rceil$ .

**Exercise 21.** For real  $x$  and integer  $n$ , true or false:  $\lfloor x + n \rfloor = \lfloor x \rfloor + n$ ,  
 $\lceil x + n \rceil = \lceil x \rceil + n$ ,

A particularly interesting function is  $\lfloor \log_b N \rfloor$ , the largest integer not greater than  $\log_b N$ . This is 1 less than the number of bits needed to express  $N$  in base  $b$ . For example, in base  $b = 10$  we need  $\lfloor \log_{10} 99 \rfloor + 1 = 2$  digits to express 99. (In other words,  $\lfloor \log_{10} N \rfloor$  function is *really easy* to compute in your head. Just count the number of digits and subtract 1. It's one of the most natural properties of a number. Similarly, if the number is written in binary,  $\lfloor \lg N \rfloor$  is just the number of bits minus 1.)

**Exercise 22.** How often can you divide a number  $N$  by two until you reach 1 or less?

### Factorials

The factorial function is another iterated multiplication, defined for integer  $N \geq 1$  as  $N! = N \times (N - 1) \times \cdots \times 1$ . By convention,  $0! = 1$ .

**Exercise 23.** How big is  $N!$  compared to  $2^N$  and  $N^N$ ?

**Exercise 24.** Find  $N$  such that  $N!$  is roughly the number of elementary particles in the universe.

**Exercise 25.** Find  $N$  such that  $N!$  is roughly the number of nanoseconds since the beginning to Time.

Except for “fricking’ huge” it’s difficult to get a feeling for the value of  $N!$ . A sometimes useful fact, the proof of which is beyond the scope of this note, is Stirling’s approximation,

$$N! \sim \sqrt{2\pi N} \left(\frac{N}{e}\right)^N.$$

### Binomial coefficients

The number of  $k$  element subsets of  $\{1, \dots, N\}$  is denoted  $C(N, k)$ . Note that we are talking about sets, and remember that the sets  $\{1, 2, 4\}$  and  $\{1, 4, 2\}$  are the same. For  $N = 3$  and  $k = 2$  there are 3 such sets:

$$\{1, 2\}, \{2, 3\}, \{1, 3\}.$$

A confusing way to define  $C(N, k)$  is to say “the number of ways to pick  $k$  elements from  $N$ ”. I find this confusing because we’re *not* actually interested in the fact that  $\{1, 2\}$  can be picked in two ways (“first pick 1, then 2”, or “first pick 2, then 1”.)

Nevertheless, many textbooks use this.

By contrast, consider number ordered sequences of  $k$  elements from  $\{1, \dots, N\}$ . For  $N = 3$  and  $k = 2$ ,

$$(1, 1), (1, 2), (1, 3), (2, 1), (2, 2), (2, 3), (3, 1), (3, 2), (3, 3).$$

In general, there are  $N^k$  such sequences, so this number does not deserve special notation.

**Exercise 26.** Which values do  $(r, s, t)$  take? What is the final value of  $C$  in terms of  $N$ ?

```
C = 0
for (int r = 0; r < N; r++)
    for (int s = 0; s < N; s++)
        for (int t = 0; t < N; t++)
            C++;
```

**Exercise 27.** Which values do  $(r, s, t)$  take? What is the final value of  $C$  in terms of  $N$ ?

```
C = 0
for (int r = 0; r < N-2; r++)
    for (int s = r+1; s < N-1; s++)
        for (int t = s+1; t < N; t++)
            C++;
```

Popular notation is  $C(N, k) = \binom{N}{k}$ . We’ll stick to that. By convention, we agree that the number of ways to pick 0 elements is 1 (namely “by not doing anything”), so  $\binom{N}{0} = 1$ . You can’t pick a nonempty set from an empty set, so  $\binom{0}{k} = 0$  for  $k \geq 1$ . And, perversely but consequently,  $\binom{0}{0} = 1$ . These conventions turn out to lead to pleasant notation and are consistent with our other choices, such as  $0! = 1$ .

The numbers also appear as coefficients in the evaluation of binomials, which are expressions of the form  $(x + y)^N$ . For example in  $(x + y)^2 = x^2 + 2xy + y^2$ , the coefficients are  $1 = \binom{2}{0}$ ,  $2 = \binom{2}{1}$ , and

$1 = \binom{2}{0}$ . In general, we have the Binomial Theorem,

$$(x + y)^N = \binom{N}{0}x^N y^0 + \binom{N}{1}x^{N-1}y^1 + \cdots + \binom{N}{k}x^{N-k}y^k + \cdots + \binom{N}{N-1}x^1y^{N-1} + \binom{N}{0}x^0y^N.$$

This is why  $\binom{N}{k}$  is often called a *binomial coefficient*.

**Exercise 28.** Find the values  $\binom{3}{0}$ ,  $\binom{3}{1}$ ,  $\binom{3}{2}$ , and  $\binom{3}{3}$  by factoring out  $(x + y)^3$ .

**Exercise 29.** Find  $\binom{N}{0} + \binom{N}{1} + \cdots + \binom{N}{N}$ .

One can give a closed formula for  $C(N, k)$  in terms of factorials:

$$\binom{N}{k} = \frac{N!}{k!(N-k)!}.$$

This formula is surprisingly useless both for getting an intuition about the size of  $\binom{N}{k}$  and for actually computing it (because the numbers are so huge). Still, it's good enough for some small numbers:

**Exercise 30.** Express  $\binom{N}{1}$ ,  $\binom{N}{2}$ , and  $\binom{N}{3}$  as polynomials in  $N$ .

*Harmonic numbers*

*Sums*

[To be written]

*Growth Rates*

To express the running time of algorithms we to say things like “Bottom-up Mergesort requires uses between  $\frac{1}{2}N \lg N$  and  $N \lg N$  compares and  $\sim 2N \lg N$  data moves” using plain English (“between”), tilde notation, and a well-defined cost model (comparisons, data moves).

Sometimes, this level of detail is exactly right. But in many instances, we are satisfied with more coarse-grained expressions and would like to say things like “Mergesort runs in linearithmic time,” without specifying what is counted (comparisons? moves? arithmetic? array accesses?) or what the leading coefficient is.

Let  $f$  and  $g$  be functions. Then  $f$  is  $O(g)$  if there exist positive constants  $C$  and  $N_0$  such that  $|f(N)| \leq |Cg(N)|$  for all  $N \geq N_0$ .

We pronounce this as “ $f$  is big oh of  $g$ .” It is often written  $f = O(g)$  or  $f \in O(g)$  and called *asymptotic*, *Laundau*, or just *big-O notation*.

**Exercise 31.** State the running times of the following code snippet as a function of  $N$  is asymptotic notation.

```

C = 0
for (int i = 0; i < N-2; i++)
    for (int j = i+1; j < N-1; j++)
        for (int k = j+1; k < N; k++)
            C++;

```

**Exercise 32.** As above:

```

C = 0
if (N > 10)
    for (int i = 0; i < N; i++)
        for (int j = 0; j < N; j++)
            C++;
else
    for (int i = 0; i < N; i++)
        C++;

```

**Exercise 33.** As above:

```

C = 0
if (N < 10)
    for (int i = 0; i < N; i++)
        for (int j = 0; j < N; j++)
            C++;
else
    for (int i = 0; i < N; i++)
        C++;

```

**Exercise 34.** True or false: if  $f \sim g$  then  $f$  is  $O(g)$ .

**Exercise 35.** True or false: if  $f$  is  $O(g)$  then  $f \sim g$ .

**Exercise 36.** True or false: if  $a_k N^k + a_{k-1} N^{k-1} + \cdots + a_1 N + a_0$  is  $O(N^k)$ .

**Exercise 37.** True or false:  $N$  is  $O(N^2)$ ,  $N$  is  $O(N)$ ,  $N + \log N$  is  $O(N)$ ,  $N \log N$  is  $O(N)$ .

**Exercise 38.** Order the following functions by their growth rate, i.e., write them on a list  $f_1(N), f_2(N), \dots$ , such that  $f_i(N) = O(f_{i+1}(N))$ :

$10^{10}, N^2, \log N, \log \log N, (\log N)^2, \sqrt{\log N}, N!, N^N,$   
 $N^2 \log N, N \log N \log \log N, N^{3/2}, 2^N, 2^{N^2}, 3^N, \log(N^2 + 1).$