CSCE 411 Course Notes

Calvin Beideman

Fall 2026

1 Foundational Mathematical Notation

In this chapter we establish some important definitions and notation that will be used throughout the course. Hopefully many of these definitions will be familiar to you from previous courses, but there may be concepts that are new to you or slight differences between the notation used in this course and what you are familiar with.

1.1 Sets

A **set** is an unordered collection of distinct things. Often these things will be numbers, but they could also be strings, or algorithms, or other sets, or basically anything else you can think of. We call the things in a set its **elements**. Sets can be finite or infinite. We can represent a finite set with a list of its elements, separated by commas and enclosed in curly braces. So $\{1, 2, 3, 4, 5\}$ is a set, as is $\{\text{“apple”}, \text{“banana”}, \text{“orange”}\}$, as is $\{\text{“apple”}, \sqrt{3}, -100, \{\{2\}\}\}$.

The **size** or **cardinality** of a set is the number of elements that it contains. We use $|A|$ to denote the size of the set A. For example, $|\{1, 2, 5\}| = 3$ and $|\{\{\}, \{1, 2, 3, 4\}, \{7\}, \{\{5\}, \{6\}, \{9\}\}\}| = 4$. A set cannot contain multiple copies of the same element, $\{2, 2\}$ is the same set as $\{2\}$. A set can, however, have zero elements. The set $\{\}$, which contains no elements, is known as the **empty set**, and we denote it with the special symbol $\emptyset$.

We use the symbol $\in$ to say that an object is part of a set. For example $2 \in \{1, 2, 5\}$. To say that an object is not an element of a set, we can use the symbol $\notin$. For example, $4 \notin \{1, 2, 5\}$. For two sets A and B, if *all* of the elements of A are also in B, we say that A is a **subset** of B, written $A \subseteq B$ (conversely we say that B is a superset of A).

Theoretical computer science (including algorithm design and analysis) is built on discrete mathematics. Unlike ”continuous mathematics” (such as calculus) which focuses on real numbers, discrete mathematics is built on the integers. We use $\mathbb{Z}$ to denote the set of integers $\{\ldots, -3, -2, -1, 0, 1, 2, 3, \ldots\}$. In many contexts, we don't care about negative values or they don't make sense, so we restrict ourselves to the set of non-negative integers:

$$\mathbb{Z}_{\geq 0} := \{0, 1, 2, 3, \ldots\}.$$

Note that the notation $:=$ means “is defined as”. Sometimes we don't care about 0 either, and use the set of positive integers

$$\mathbb{Z}^+ := \{1, 2, 3, \ldots\}.$$

Many authors (including CLRS) use the term “natural numbers” for $\mathbb{Z}_{\geq 0}$, and denote it $\mathbb{N}$. However, this notation can be confusing, because other authors use ”natural numbers” and $\mathbb{N}$ to refer to $\mathbb{Z}^+$. To avoid confusion over whether 0 is included in the natural numbers, these notes and your homeworks/exams will avoid the term and instead use “non-negative integers” or “positive integers” and the symbols $\mathbb{Z}_{\geq 0}$ or $\mathbb{Z}^+$.

Listing all of the elements works great for describing small sets, but if a set has hundreds of elements, or worse, infinitely many, this notation is not very effective. In the previous paragraphs we defined an infinite set using ellipses i.e. we represented the infinite set of positive integers with $\{1, 2, 3, \ldots\}$. This approach can work when the pattern that the elements follow is very obvious, but

for more complicated sets it is likely to cause confusion. A better approach is to use "set builder notation" where we explicitly describe the pattern that the elements of the set follow. This notation typically takes the form $\{x \in S \mid P(x)\}$, where P is a function outputting true or false (a predicate) and S is some set which has already been defined. This expression is read as "The set of x in S such that $P(x)$". Sometimes you will also see this notation with a colon instead of a vertical bar (i.e. $\{x \in S : P(x)\}$). Set builder notation allows us to express any subset of a known set by describing some property that all of the elements of the set satisfy. For example, we can formally describe the set of even integers as $\{x \in \mathbb{Z} \,|\, x \bmod 2 = 0\}$.

Sometimes instead of describing a subset of an existing set, we want to create a new set by applying some function f to certain elements of an existing set. For this, we can use an alternative form of set builder notation: $\{f(x) \mid x \in S \text{ and } P(x)\}$. This defines a set, which contains $f(x)$ for each $x \in S$ which satisfies the property given by P. Note that some elements of our new set may be $f(x)$ for several different $x \in S$, but a set cannot contain the same element more than once. This notation gives us an alternative way of describing the set of even integers $\{2x \mid x \in \mathbb{Z}\}$, i.e. a number is an even integer if it can be written as $2x$ where x is an integer. Similarly, we can define the set of rational numbers by:

$$\mathbb{Q} := \{p/q \mid p, q \in \mathbb{Z}, q \neq 0\}.$$

That is, a number is a rational number if it can be written as p/q where p is an integer and q is a non-zero integer.

The real numbers "fill in the gaps" of the rational numbers by including infinitely many irrational values like $\sqrt{2}$ and π. We use $\mathbb{R}$ to denote the set of real numbers. We will not formally define the real numbers or make frequent use of them in this course. Just as we denote the positive integers with $\mathbb{Z}^+$, we denote the positive rationals with $\mathbb{Q}^+$, and the positive reals with $\mathbb{R}^+$. Similarly, we denote the non-negative rationals with $\mathbb{Q}_{\geq 0}$ and the non-negative reals with $\mathbb{R}_{\geq 0}$.

1.2 Combining Sets

We saw in the previous section how to define a new set in terms of an existing set using set builder notation. We can also use this notation to define a new set in terms of two or more existing sets. Certain ways of combining sets are common enough that they have their own special notation. For two sets S and T:

- The **union** of S and T is
$$S \cup T := \{x \mid x \in S \text{ or } x \in T\}.$$
- The **intersection** of S and T is
$$S \cap T := \{x \mid x \in S \textit{ and } x \in T\}.$$
- The **difference** of S and T is
$$S \setminus T := \{x \mid x \in S \text{ and } x \notin T\}.$$
Sometimes the difference is also written as $S - T$.
- The **(Cartesian) product** of S and T is
$$S \times T := \{(x, y) \mid x \in S \text{ and } y \in T\}.$$
That is, the product of S and T is the set of all pairs where the first element is from S and the second is from T.

One final useful set operation is the **powerset** of a set S:

$$2^S := \{T \mid T \subseteq S\}.$$

For example, if $S = \{1, 2, 3\}$,

$$2^S = \{\emptyset, \{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\}\}.$$

Note that if S is a finite set, $|2^S| = 2^{|S|}$. The notation $\mathcal{P}(S)$ is also sometimes used for the powerset.

1.3 Exponents and Logarithms

Exponents and logarithms arise frequently when analyzing algorithms, so you should be comfortable working with them.

For a real number x and a positive integer a, x^a is defined as the result of multiplying x with itself, a times. For example $5^3 = 5 \cdot 5 \cdot 5 = 125$. In the expression x^a, we call x the "base" and a the "exponent". We generally read x^a as "x to the a" with a few exceptions. In particular, x^2 is read as "x squared" and x^3 is read as "x cubed". Exponents satisfy the following properties which you should be familiar with. Don't treat these equations as facts to memorize—instead you should make sure that you understand *why* they're true so that when they're relevant you can apply them automatically.

Theorem 1. *If x and y are positive real numbers and a and b are positive integers then*

(a) $x^a x^b = x^{a+b}$

(b) $(x^a)^b = x^{ab}$

(c) $x^a y^a = (xy)^a$

Proof. (a) By definition x^a is the product of a copies of x and x^b is the product of b copies of x. So when we multiply them together we obtain the product of $a + b$ copies of x.

(b) $(x^a)^b$ is the product of b copies of x^a. Each x^a is the product of a copies of x. Thus, in total we are multiplying ab copies of x. Note that this equation also implies that $(x^a)^b = (x^b)^a$. Swapping the exponents like this is sometimes helpful for simplifying expressions.

(c) $x^a y^a$ is the result of multiplying x with itself a times, multiplying y with itself a times, and then multiplying the two results. Since multiplication is commutative (xy is the same as yx), we can rearrange the order of the multiplications without changing the result. In particular, we can pair up the x's and y's to rewrite our product of a copies of x and a copies of y as the product of a xy pairs, i.e. $(xy)^a$.

□

The definition of exponentiation as "repeated multiplication" makes sense for positive integer exponents, but it doesn't really make sense for other values. What would something like $5^{-2/3}$ mean? How can we multiply a number with itself negative two thirds of a time? To extend the definition of exponentiation beyond repeated multiplication, mathematicians looked at what definitions would preserve the useful properties given in Theorem 1. In particular, these properties give us values for all rational exponents as follows:

- For every real number x, we define $x^0 = 1$, because to preserve property (a) we need $x^1 \cdot x^0 = x^1$. (Note that in particular we define $0^0 = 1$, unlike in some branches of mathematics where it is considered indeterminate).
- For every non-negative real number x and every positive integer b, we define $x^{1/b}$ as the (non-negative real) solution to $y^b = x$ (i.e. the number which when multiplied with itself b times gives x). This is because to preserve property (b) we need $(x^{1/b})^b = x^1 = x$. $x^{1/b}$ is also known as "the b-th root of x" and can be written with the alternative notation $\sqrt[b]{x}$. Just as x^2 and x^3 are called "x squared" and "x cubed", we refer to $x^{1/2}$ as "the square root of x" (typically written as $\sqrt{x}$, note that we omit the 2) and we refer to $x^{1/3}$ as "the cube root of x".
- For every non-negative real number x and all positive integers a and b, we define $x^{a/b}$ as $(x^{1/b})^a$. Once again, this preserves property (b).
- For every positive real number x and every positive rational number a, we define $x^{-a} = 1/x^a$. This is because to preserve property (a) we need $x^a x^{-a} = x^0 = 1$.

We've now defined x^a whenever x is a positive real number and a is a rational number. We could extend our definition to cover every real number a using limits, but the details are outside the focus of this course (since we will not need tools from calculus elsewhere). While we will omit the formal definition, you may assume in this course that the properties in Theorem 1 hold for all real numbers a and b.

Logarithms are the inverse of exponents. $\log_b x$ is defined as the power that you have to raise b to in order to get x, i.e. $\log_b x$ is the value of y for which $b^y = x$.

The following theorem gives some properties of logarithms which you should understand and be comfortable using. Part (c) in particular allows us to convert a logarithm from one base to another, which can be useful for analyzing divide and conquer algorithms that we will discuss later in the course.

Theorem 2. *If a is a real number and b, c, x, and y are positive real numbers with $b, c \neq 1$ then*

(a) $\log_b(xy) = \log_b(x) + \log_b(y)$

(b) $\log_b(x^a) = a \log_b(x)$

(c) $\log_b x = (\log_c x) \log_b c$

Proof. (a)

$$\begin{aligned} b^{\log_b(x)+\log_b(y)} &= b^{\log_b(x)} b^{\log_b(y)} && \text{by part (a) of Theorem 1} \\ &= xy && \text{by the definition of log} \end{aligned}$$

Thus, since $b^{\log_b(x)+\log_b(y)} = xy$, by the definition of log, $\log_b(xy) = \log_b(x) + \log_b(y)$.

(b)

$$\begin{aligned} b^{a \log_b(x)} &= (b^{\log_b(x)})^a && \text{by part (b) of Theorem 1} \\ &= x^a && \text{by the definition of log} \end{aligned}$$

Thus, since $b^{a \log_b(x)} = x^a$, by the definition of log, $\log_b(x^a) = a \log_b(x)$.

(c)

$$\begin{aligned} b^{(\log_c x) \log_b c} &= (b^{\log_b(c)})^{\log_c x} && \text{by part (b) of Theorem 1} \\ &= c^{\log_c x} && \text{by the definition of log} \\ &= x && \text{by the definition of log} \end{aligned}$$

Thus, since $b^{(\log_c x) \log_b c} = x$, by the definition of log, $\log_b x = (\log_c x) \log_b c$.

□

When we have an exponent where the base is a logarithm, we will often use the following shorthand notation to avoid extra parentheses

$$\log_b^k n := (\log_b n)^k.$$

For example, $(\log_2 5)^3$ can be written as $\log_2^3 5$.

1.4 Summations and Products

We use the symbol $\sum$ (the capital Greek letter sigma) to represent a summation. You can think of this like a mathematician's version of a for loop. If f is some function which has been defined on integers and a and b are integers, then the expression

$$\sum_{i=a}^{b} f(i)$$

is equivalent to running the pseudocode

```
sum ← 0
For i from a to b:
   sum ← sum +f(i)
Return sum
```

For example, $\sum_{i=3}^{5} i^2 = 3^2 + 4^2 + 5^2 = 50$ (note that the upper and lower bounds for our iteration variable i can be written either above and below the $\sum$, as in the first example, or just to the right, as in the example in this paragraph).

Sometimes we want to iterate over something other than a range of integers. $\sum$ notation can also be used for looping over the elements of an arbitrary set:

$$\sum_{x \in S} f(x)$$

This expression represents applying the function f to every element of the set S and adding up the results.

Just as we use $\sum$ to denote sums, we use $\prod$ (the capital Greek letter pi) to denote products. The use is exactly the same, except with $\prod$ notation we are *multiplying* the results of each iteration of the loop instead of adding them. For example, $\prod_{i=3}^{5} i = 3 \cdot 4 \cdot 5 = 60$.

The summation in the following theorem shows up frequently enough in analyzing algorithms that you should be familiar with it and know how to simplify it.

Theorem 3.

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

Proof. We note that

$$2\sum_{i=1}^{n} i = 1 + 2 + \cdots + (n-1) + n + 1 + 2 + \cdots + (n-1) + n.$$

Since addition is commutative (we can rearrange the order in which we add numbers without changing the sum), we can rewrite this sum as

$$(1+n) + (2 + (n-1)) + (3 + (n-2)) + \cdots + ((n-2)+3) + ((n-1)+2) + (n+1).$$

This new version has n pairs of values and each pair adds up to $n+1$. Thus,

$$2\sum_{i=1}^{n} i = n(n+1),$$

and

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

□

We'll learn how to simplify some other summations in Chapter 3.

1.5 Other Important Mathematical Notation

For a real number x, we use $\lfloor x \rfloor$ to denote the *floor* of x, which is the largest integer less than or equal to x. Similarly we use $\lceil x \rceil$ to denote the *ceiling* of x, which is the smallest integer greater than or equal to x. For example:

- $\lfloor 4.5 \rfloor = 4$
- $\lceil 4.5 \rceil = 5$
- $\lfloor -2.7 \rfloor = -3$
- $\lceil -2.7 \rceil = -2$
- $\lfloor 7 \rfloor = 7$
- $\lceil 7 \rceil = 7$

For a non-negative integer n, we use $n!$ to denote n *factorial*, which is $n \cdot (n-1) \cdot (n-2) \cdot \cdots \cdot 2 \cdot 1$. For example, $5! = 120$. We define $0! := 1$.

1.6 Arrays and Strings

In this class, we will use many of the abstract data types and data structures you learned about in CSCE 221. The most important abstract data type is the **list** ADT, which represents an *ordered* collection of objects. Unlike a set, a list may contain multiple copies of the same value.

Lists are commonly implemented using the **array** data structure. The core property of arrays in contrast to other implementations of the list ADT (like linked lists) is that arrays allow for random access, meaning that we can get the element at any arbitrary position in the array in constant time, without having to first look through earlier elements of the array. As in many programming languages, we will use square brackets to denote indexing into an array. For example, $A[5]$ refers to the element at index 5 in the array A. Unlike some programming languages you may be familiar with (such as C++), in this course we will typically view arrays as beginning at index 1. Therefore we will use $A[1, \ldots, n]$ to denote an arbitrary array of size n.

We will sometimes use $A[i : j]$ to refer to the subarray starting at index i and ending at index j, i.e. $A[i, i+1, \ldots, j-1, j]$. For example, if A is the array $[3, 1, 4, 1, 5, 9]$ then $A[1] = 3$, and $A[2 : 4] = [1, 4, 1]$. Note that this convention is different from some programming languages like Python where $A[i : j]$ does not include $A[j]$.

An **alphabet** is a finite, nonempty set. We call the elements of the alphabet **symbols** or **characters**. Common choices in this course will be the English alphabet (the set containing the letters a through z) and the alphabet $\{0, 1\}$. We often use the character Σ as a variable for an alphabet (note that this is the same Greek letter used for describing summations, do not confuse the two uses). A **string** x over the alphabet Σ is a finite sequence of symbols $x_1 x_2 x_3 \ldots x_n$, where $x_i \in \Sigma$ for each $1 \leq i \leq n$. We typically write the symbols in the string right next to each other, but we can add commas or spaces if this would be ambiguous. We use $|x|$ to denote the length or size of x, which is the number of symbols that it contains. We use Σ^n to denote the set of all length-n strings over the alphabet Σ. We use Σ^* to denote the infinite set of all strings over the alphabet Σ (of any length). Thus $0011010 \in \{0, 1\}^7$, and therefore also $0011010 \in \{0, 1\}^*$ (we call strings over the alphabet $\{0, 1\}$ **binary strings**), and if Σ is the English alphabet then

$$\text{hello} \in \Sigma^*.$$

The empty string is a valid string over any alphabet, and we denote it with the special character ε. Given two strings x and y over alphabets Σ and Γ, their **concatenation** is the string over the alphabet $\Sigma \cup \Gamma$ which consists of all the symbols of x followed by all the symbols of y. We denote x concatenated with y as $x \cdot y$ or simply xy. For example, if x is the binary string 010 and y is the binary string 1011, xy is the binary string 0101011. We use x^n to denote the string obtained by

concatenating x with itself n times. For example, $(01)^4$ is 01010101. Be careful not to confuse this with exponentiation. To determine whether x^n means "x to the power n" or "the concatenation of n copies of x" you need to know whether x is being treated as a number or a string.

1.7 Comprehension Questions

1. What is the cardinality of the set $\{\{3, 7, 11\}, \{A, B\}\}$?
2. What is $|\{x \in \mathbb{Z} \mid x^2 < 10\}|$? (Note: don't confuse this with the set $\{x \in \mathbb{Z}_{\geq 0} \mid x^2 < 10\}$!)
3. What is $\{1, 3, 5\} \setminus \{1, 2, 3\}$?
4. What is $\{1, 2\} \times \{3, 4\}$?
5. What is $2^{\{1,2\}}$?
6. Let x and n be positive integers greater than 1 such that $(x^2x^3)^5 = x^n$. What is the value of n?
7. What is the value of $\log_2(8^{32})$?
8. What is the value of $\sum_{i=1}^{100} i$?
9. What is the value of $\prod_{i=1}^{5} i$?
10. What is the value of $\lfloor -2.1 \rfloor \cdot \lceil 3 \rceil$?
11. Which of these strings are in $\{0, 1\}^3$?
    - ε
    - 01
    - 101
    - 012
    - 1000
12. Let $\Sigma = \{a, b, c, d\}$, and consider the two strings ab and cd. What is $ab \cdot cd$?
13. Let $\Sigma = \{0, 1\}$, and consider the two strings 10 and 100. What is $10 \cdot 100$?
14. Compute $|\{s \in \{0, 1, 4\}^3 : s_1 = s_2\}|$.

1.8 Exercises

1. Let n be a positive integer greater than 1 and let x be a positive real number such that $\left(\frac{3}{2}\right)^{\log_2 n} n = n^x$. What is the value of x?
2. Let a, b, c, x, y be integers greater than 2 and let α and β be real numbers such that

   $$\log_b(x^a y^c) = \alpha \log_c(x) + \beta \log_b(y).$$

   What are the values of α and β? (Express your answer in terms of numeric constants and/or a, b, and c).
3. Let n be an integer greater than 99. Simplify

   $$\sum_{i=99}^{2n} (i - 3)$$

   (i.e. give an expression which is equivalent to this one but doesn't use a summation).