CSCI 120 Introduction to Computation
Computation everyday (draft)

Saad Mneimneh
Visiting Professor
Hunter College of CUNY

1 Introduction

I had the intention of showing how we experience in our everyday life different aspects of math and computer science, without even knowing. My plan was to cover the following:

- Map coloring and the relation to the coloring of planar graphs (graph theory), in addition to an overview of the 4 color theorem which states that only 4 colors are needed to color any 2-dimensional map in a way that adjacent countries have different colors (google the 4 coloring theorem for very nice stories on the history of this theorem). Today, the only proof of this theorem is based on a computer program!
- Binary search trees and how they are used in everyday applications to search for things.
- The golden number and the Fibonacci sequence and the relation to all aspects of life including architecture, art, music, literature, biology, etc...
- Grammars and languages in the field of linguistics and the relation to programming languages.

I previously talked about map coloring, graphs, planar graphs, the 4 color theorem, and binary search trees. I will concentrate here on the golden number and the Fibonacci sequence, and on grammars and languages.

2 The golden number and the Fibonacci sequence

The golden number or the golden ratio is obtained in the following way: Consider a segment and a point on the segment. The point divides the segment in two parts, as illustrated in Figure 1 below:

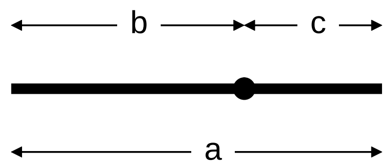

Figure 1: Segment divided by point

Referring to Figure 1 above, the placement of the point defines two ratios, namely a/b and b/c. When these two ratios are equal, we call the ratio the golden ratio, or simply the golden number. So how should we place the point in order to obtain $a/b = b/c$? It turns out that there is only one solution in which $a/b = b/c = 1.618033989....$ This number is given the symbol ϕ and is pronounced *phi*. Like π, ϕ is simply another kind of proportion. While π is the ratio of the circle perimeter to its diameter, ϕ is the ratio obtained by making $a/b = b/c$ for any line segment as shown in Figure 1.

You are not responsible for the detail on how to obtain ϕ, but here's a derivation for those of you who are interested:

We need $a/b = b/c$. But c is nothing but $a - b$. Therefore, we need

$$\frac{a}{b} = \frac{b}{a-b}$$

But

$$\frac{b}{a-b} = \frac{1}{a/b-1}$$

Therefore, we need:

$$\frac{a}{b} = \frac{1}{a/b-1}$$

Now letting $a/b = \phi$, we have:

$$\phi = \frac{1}{\phi - 1}$$

which gives the equation $\phi^2 - \phi - 1 = 0$. The only positive number that satisfies this equation is $(1+\sqrt{5})/2 = 1.618033989....$

It is not really known when ϕ was first discovered. It was probably rediscovered many times throughout history. For instance, ancient egyptian civilizations used ϕ in the construction of pyramids as illustrated below.

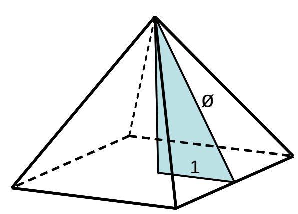

Figure 2: Egyptian pyramid

Similarly, the Greek civilizations used ϕ in the construction of their temples, as illustrated below.

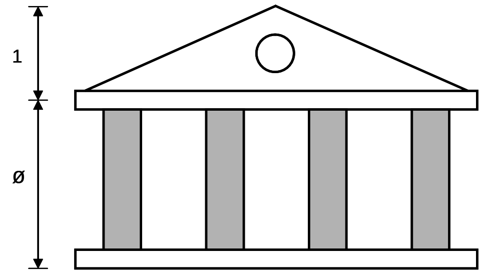

Figure 3: Greek temple

In general, ancient civilizations considered ϕ to be a *beautiful* ratio or proportion, even a divine one! As we will see later, ϕ is encoded in many aspects of nature and life.

In 1200 AD, an Italian mathematician, Leonardo Da Piza, also known as Leonardo Fibonacci (don't confuse him with Leonardo Da Vinci who is more recent), rediscovered ϕ using a hypothetical mathematical problem on rabbits. Here's the problem:

- We start with a new born pair of rabbits (a male and a female)
- A new born pair of rabbits, N, needs on month to become mature, M
- A mature pair of rabbits give birth to a new born pair of rabbits every month after one month of being mature
- What is the number of pairs of rabbits that we have in every month?

Of course there are some implicit assumptions here: (1) rabbits don't die, (2) rabbits are born in pairs (one male and one female) only, and (3) rabbits are lawful to their partners (and they always mate!).

The following tree shows the number of pairs of rabbits for each month. For a given month, each N becomes M in the following month, and each M gives N and M (itself) in the following month.

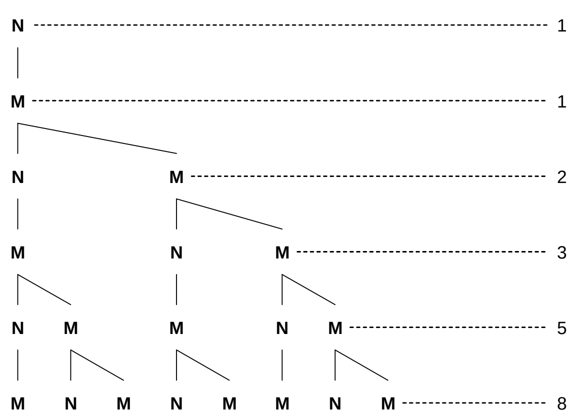

Figure 4: Fibonacci rabbits

Therefore, the number of pairs is 1 initially, 1 for the first month, 2 for the second month, 3 for the third month, 5 for the fourth month, 8 for the fifth month... If we continue, we find the sequence 1 1 2 3 5 8 13 21 34 55 ... where each number is the sum of the previous two numbers. This is known as the Fibonacci sequence.

$$Fib(0) = 1$$
$$Fib(1) = 1$$
$$Fib(n) = Fib(n-1) + Fib(n-2)$$

This is a recursive function, i.e. defined in terms of itself on smaller values. How does this sequence relate to ϕ? Let us compute $\frac{Fib(n+1)}{Fib(n)}$ for different values of n:

$n = 0$	$Fib(1)/Fib(0) = 1/1$	1
$n = 1$	$Fib(2)/Fib(1) = 2/1$	2
$n = 2$	$Fib(3)/Fib(2) = 3/2$	1.5
$n = 3$	$Fib(4)/Fib(3) = 5/3$	1.66667
$n = 4$	$Fib(5)/Fib(4) = 8/5$	1.6
$n = 5$	$Fib(6)/Fib(5) = 13/8$	1.625
$n = 6$	$Fib(7)/Fib(6) = 21/13$	1.61538
$n = 7$	$Fib(8)/Fib(7) = 34/21$	1.61904
$n = 8$	$Fib(9)/Fib(8) = 55/34$	1.61765
$n = 9$	$Fib(10)/Fib(9) = 89/55$	1.61818
$n = 10$	$Fib(11)/Fib(10) = 144/89$	1.61797
$n = 11$	$Fib(12)/Fib(11) = 233/144$	1.61805
$n = 12$	$Fib(13)/Fib(12) = 377/233$	1.61803

As it can be seen, the ratio of $Fib(n+1)/Fib(n)$ converges to ϕ, the golden number!

We experience ϕ every day. For instance, when we pay with a credit card, or when we show some kind of ID card, or when we use the metro card to take the subway. All these rectangular cards are designed to have a proportion of ϕ. This is known as the golden rectangle. If we divide the large side by the small side we get ϕ. These cards are designed this way because we believe (as ancient civilizations did) that this proportion reveals beauty (imagine if your credit card was narrower and longer, or wider and shorter).

Makers of musical instruments discovered that ϕ gives superior performance and acoustics in the design of the instruments.

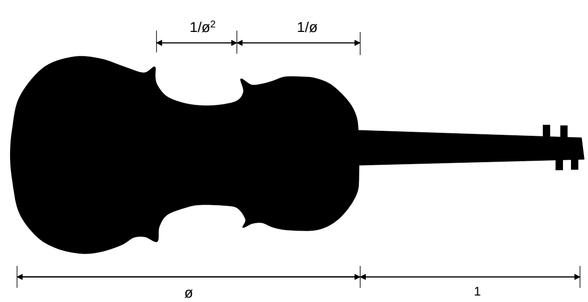

Figure 5: Violin

The Fibonacci sequence is also related to the musical scales. If we change the two initial values of the sequence to $Fib(0) = 2$ and $Fib(1) = 5$, we obtain the following sequence 2 5 7 12 19 31 .... Each musical note has a frequency. If we multiply that frequency by 2, we obtain the same note but one octave higher. Therefore, if we start with the note F for instance, and we repeatedly multiply the frequency by a ratio of 2/1, we don't get an interesting musical scale (same tonality at different octaves).

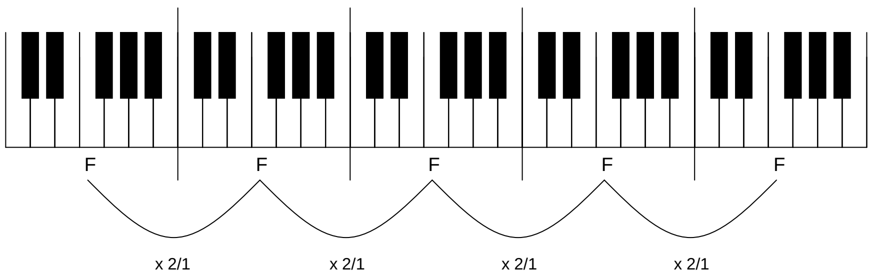

Figure 6: Octaves

The next simplest ratio to try is 3/2. Now if we base our scale on 5 notes (the second number in the above sequence), we get the following scale.

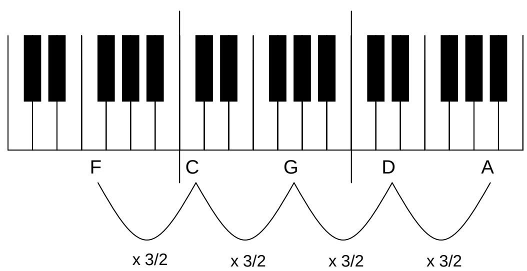

Figure 7: Pentatonic scale

This is the Pentatonic scale used in Chinese music. In fact, if we transpose each note to the next sharp, i.e. F#, C#, G#, D#, A#, we get all the black keys in an octave. Try to play on the black keys of a piano randomly, and you will hear a very nice Chinese music!

Now let us use the same technique and base our musical scale on 7 notes (the third number in the above sequence). Using the ratio of 3/2, we get the following musical scale: F C G D A E B. This is the 7 note major scale in western music.

Now let us try the next number in the sequence, 12. If we base our musical scale on 12 notes, and using the same ratio of 3/2, we get the following musical scale: F C G D A E B F# C# G# D# A#, which is a combination of both scales above. This gives all the keys on the piano (black and white).

Some people have experimented with a 19 note musical scale. We may get richer music with a musical scale composed of 19 notes, but so far no musical instruments are built to do that. Moreover, we are so used to the 12 note scale that it is hard to change at this point. But some groups experiment with different scales. Another important point worth mentioning is that the frequencies computed as shown above, are not the real frequencies that we experience when we play an instruments, because we have slightly changes those to give better tonality. But some people experiment with the original tuning (alternative tuning).

The golden number appears also in many artistic works. Painters have experimented with the golden number by dividing the painting area according to the golden ratio. The Fibonacci sequence also appears in literature. For instance, the recently written novel, The Da Vinci Code, uses the Fibonacci sequence as the code to reveal the secret of the story. Moreover, some poems are based on the Fibonacci sequence: the number of syllables in each line follow the Fibonacci sequence. Recently, this type of poems became very famous on the Internet, and such poems are called Fibs. Here's an example:

I 1
am 1
sitting 2
quietly 3
listening for the 5
quiet noises of the darkness 8

Here's another example by me:

thus 1
born 1
surely 2
from rabbits 3
the Fibonacci 5
of all sequences most divine 8

In nature, the Fibonacci sequence and the golden number are present as well. Here's how a shell forms:

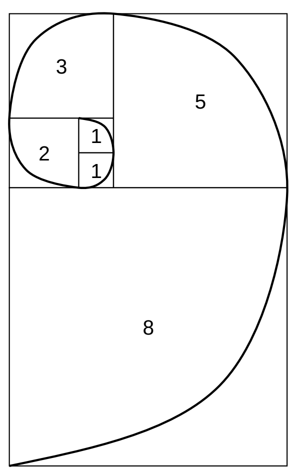

Figure 8: Shell

Here are some human proportions that are considered “perfect”. In fact, in plastic surgery, they use ϕ proportions for the face.

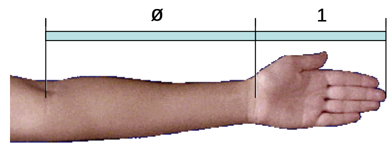

Figure 9: Human arm

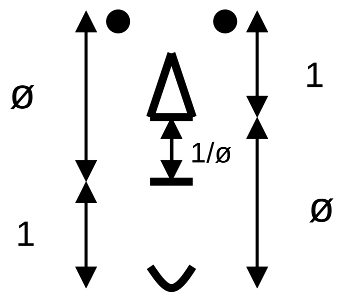

Figure 10: Human face

3 Grammars and languages

Grammars are used to describe languages. We can think of a grammar for a language as a finite representation of the language: while the language contains an infinite number of sentences and meanings, a grammar is just a finite collection of rules that govern the language. Therefore, a grammar constitutes a model for the language. This is why we have the ability to learn languages (although they are infinite). We actually learn the grammar and not the language explicitly. For instance, we don’t learn the meaning of each sentence individually, but we learn how to form sentences. In linguistics, there is a hypothesis that a universal grammar exists, which is part of knowledge that is encoded in our brain. The science of linguistics tries to identify what constitute the universal grammar and what beyond the universal grammar differentiates languages from one another.

So why do we mention grammars and linguistics here? The reason for this is that programming languages, like C, C++, Java, and others, are languages based on their own grammars. When we say the expression “programming language”, most people immediately think about the term “programming”. However, the way I see it at least, the correct approach is to think about the term “language”. After all, a programming language is nothing but another language!

In the same way you check that a sentence is grammatically correct, a programming language checks your code against its own grammar. In the programming world, this is called syntax. A syntax error means that your code does not conform to the grammar. One type of grammar that we use and that programming languages are based on is the context-free grammar. A context-free grammar is defined below (we will see later why we call it context-free).

A context-free grammar constitutes of a collection of production rules, also called substitution rules. Each rule appears as a line with a symbol and a string separated by an arrow.

$$symbol \rightarrow string$$

Such symbol appearing to the left of the arrow is called a variable. A string consists of variables and other symbols known as terminals (these do not appear on the left side of any rule). One of the variables is designated as a special variable called the start variable.

Here's an example context-free grammar:

$$S \rightarrow 0S1$$
$$S \rightarrow \#$$

In this grammar, the variables consists of only S, which is also the start variable. The terminals are 0, 1, and #. The grammar contains two rules. For convenience, one could combine multiple rules for the same symbol as follows (but this is just for the convenience of representation):

$$S \rightarrow 0S1 \mid \#$$

How does a context-free grammar for a language generate sentences in the language? Here's how:

1. write down the start variable
2. find a rule that applies and replace the corresponding variable with the corresponding string (this is why it is called context-free, because this replacement is done regardless of the position of the variable in the sentence)
3. repeat step 2 until no more variables

For instance, using the above grammar, we start by writing down the start variable S. Then we can apply any of the two rules $S \rightarrow 0S1$, or $S \rightarrow \#$. If we apply the latter, then we replace S by # and we are done because we have no more variables. Therefore # is part of the languages. We may choose, however, to apply the first rule. Therefore, we replace S by $0S1$. In this case, we are not done, because we still have variables, namely S. Applying the rule repeatedly will generate strings of the form $00...0S11...1$ with equal number of zeros and ones. When we finally choose the second rule, we end up with 00...0#11...1 and we stop. Therefore, this grammar generates all the strings (or sentences) of contain a number of zeros, followed by the pound sign, followed by an equal number of ones (where this number could be zero, i.e. just #).

Let us experiment with a more realistic, but very simplified, grammar for the English language.

$$<SENTENCE> \rightarrow <NOUN-PHRASE> <VERB-PHRASE>$$
$$<NOUN-PHRASE> \rightarrow <ARTICLE> <NOUN>$$
$$<VERB-PHRASE> \rightarrow <VERB> <NOUN-PHRASE>$$
$$<ARTICLE> \rightarrow a \mid the$$
$$<NOUN> \rightarrow boy \mid girl \mid flower$$
$$<VERB> \rightarrow touches \mid likes \mid sees$$

In this grammar, we have 6 variables, namely $< SENTENCE >$, $< NOUN-PHRASE >$, <VERB-PHRASE>, $< ARTICLE >$, $< NOUN >$, and $< VERB >$. We have 8 terminals, namely a, the, boy, $girl$, $flower$, $touches$, $likes$, and $sees$. We have 6 rules. The start variable is $< SENTENCE >$.

Let's try to generate a sentence using this grammar. We first write down the start variable, i.e. $< SENTENCE >$.

$$< SENTENCE >$$

Since we have a variable, we must apply a rule. There is only one rule that applies. Therefore, we replace $< SENTENCE >$ with $< NOUN - PHRASE > < VERB - PHRASE >$.

$$< NOUN - PHRASE > < VERB - PHRASE >$$

We still have variables. We need to choose a rule and apply it. Let us choose the second rule and replace $< NOUN - PHRASE >$ with its corresponding string $< ARTICLE > < NOUN >$.

$$< ARTICLE > < NOUN > < VERB - PHRASE >$$

We still have variables. Say we apply the rule for $< NOUN >$, namely the rule $< NOUN > \rightarrow boy$.

$$< ARTICLE > \ \ boy \ \ < VERB - PHRASE >$$

We still have variables. Let's apply the rule $< ARTICLE > \rightarrow a$.

$$a \ boy \ \ < VERB - PHRASE >$$

We still have variables, namely $< VERB - PHRASE >$. There is only one rule that applies, $< VERB - PHRASE > \rightarrow < VERB > < NOUN - PHRASE >$.

$$a \ boy \ \ < VERB > < NOUN - PHRASE >$$

Next, let us apply the rule $< VERB > \rightarrow sees$.

$$a \ boy \ sees \ \ < NOUN - PHRASE >$$

We still have the variable $< NOUN - PHRASE >$. Again, only one rule applies here.

$$a \ boy \ sees \ \ < ARTICLE > < NOUN >$$

Applying the rule $< ARTICLE \rightarrow the$, we get:

$$a \ boy \ sees \ the \ \ < NOUN >$$

Applying the rule $< NOUN > \rightarrow girl$, we get:

$$a \ boy \ sees \ the \ girl$$

We have no more variables. We stop. This is a sentence generated by the grammar. It is part of the language that this grammar represents. Any sentence obtained in this way is part of the language.

It is convenient to visualize this sequence of production rules as a tree, as follows (for a given variable producing a string, the variables and terminals in the string become its children in the tree):

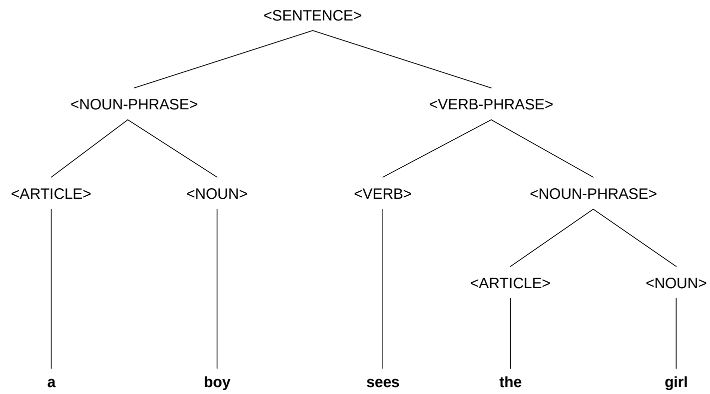

Figure 11: A parse tree for "a boy sees the girl"

This is known as a parse tree. The parse tree provides a better visualization of how the sentence was produced using the grammar. Moreover, it exhibits some structural knowledge about the sentence.

Any sentence that admits a parse tree can be generated by the grammar, and vice-versa. Although the parse tree provides meaningful structure, a sentence generated by the grammar (i.e. one that admits a parse tree) does not necessarily admit a correct meaning. For instance, "the flower sees a girl" can be generated by the above grammar and has a parse tree. However, one could argue that it does not carry a correct meaning.

Programming languages are similar to spoken language in terms of their grammar and parse trees. When we write a code, it is considered to be a very large sentence. The objective of the compiler is to first check whether the code is syntactically correct, i.e. can be generated by the grammar. To do so, the compiler attempts to build a parse tree for the code. If it can succeed in finding a parse tree, the code is syntactically correct. Otherwise, the compiler indicates that there is an error. As with a spoken language, the fact that a code is syntactically correct (grammatically), does not necessarily imply that it will perform the correct task (the meaning). For instance, we could have a logical bug in the code which is not a grammatical error (consider the sentence "the flower sees a girl" again as analogy).