CMSC 250: Sequences, Sums, Products, Recursively Defined Sets and Binary Trees

Justin Wyss-Gallifent

November 2, 2021

1 Sequences
   1.1 Intuition and Formal Definition
   1.2 Traditional Notation
   1.3 Recursive Sequences
   1.4 Conversion Between Definitions
   1.5 Shifting Starting Indices

2 Sums
   2.1 Notation
   2.2 Evaluation - Just Do It!
   2.3 Special Case Formulas
   2.4 Telescoping Sums

3 Products
   3.1 Notation
   3.2 Evaluation - Just Do It!
   3.3 Telescoping Products

4 Recursively Defined Sets
   4.1 Introduction
   4.2 Examples

5 Binary Trees
   5.1 Introduction
   5.2 Recursive Definition
   5.3 Properties of Binary Trees
1 Sequences

1.1 Intuition and Formal Definition

Sequences of numbers arise all over the place in both mathematics and computer science, in both limits and for loops. Informally a sequence is just an infinite list of numbers, one after another.

Example 1.1. Here are some examples:

1, 2, 3, 4, ...  
Looks like it increases by 1 each time!

42, 43, 44, 45, ... 
Same as above but starts higher!

0, 6, 23, −8.2, π, e^2, ... 
Pattern Not Clear!

0.1, 0.01, 0.001, 0.0001, ... 
Looks like it divides by 10 each time!

There’s nothing wrong with listing the elements of the sequence when the pattern is clear. However sometimes it isn’t.

There is a formal definition, however.

Definition 1.1.1. A sequence is a function f whose domain is a set of the form \( D = \{n_0, n_0 + 1, n_0 + 2, \ldots\} \) where \( n_0 \) is a nonnegative integer and which outputs real numbers. The domain is called the set of indices and \( n_0 \) is the starting index.

Example 1.2. If \( D = \{3, 4, 5, 6, \ldots\} \) and \( f(n) = n^2 \) then we get the sequence:

\[ f(3), f(4), f(5), \ldots = 3^2, 4^2, 5^2, \ldots \]  

1.2 Traditional Notation

It’s not very traditional to denote sequences the formal way and instead of \( f(n) \) we usually write \( a_n \) and then simply give the starting index.

Example 1.3. The sequence above would traditionally be defined by:

\[ a_n = n^2 \text{ for } n \geq 3 \]

Then we would have \( a_3 = 3^2 = 9, a_4 = 4^2 = 16, \) and so on.

Alternately we can also use curly bracket notation.

Example 1.4. The sequence above can also be defined by:

\[ \{n^2\}_{n=3} \]

The only downside to this notation is that it looks vaguely set-like. I’ll avoid it here in the notes for this reason.
1.3 Recursive Sequences

Another common way to define a sequence is recursively. A recursive definition involves giving one or more starting terms and then a formula for creating successive terms from previous terms.

Example 1.5. We may define a sequence as follows:

\[ a_1 = 4 \]
\[ a_k = 2a_{k-1} + 1 \text{ for } k \geq 2 \]

We can then calculate each term in turn:

\[ a_2 = 2a_1 + 1 = 2(4) + 1 = 9 \]
\[ a_3 = 2a_2 + 1 = 2(9) + 1 = 19 \]

\[ \vdots \]

We can also start by giving more than just the first term.

Example 1.6. We may define a sequence as follows:

\[ a_1 = 4 \]
\[ a_2 = -1 \]
\[ a_k = a_{k-1}^2 - a_{k-2} \text{ for } k \geq 3 \]

We can then calculate each term in turn:

\[ a_3 = a_2^2 - a_1 = 1 - 4 = -3 \]
\[ a_4 = a_3^2 - a_2 = 9 - (-1) = 10 \]

\[ \vdots \]

1.4 Conversion Between Definitions

In general it is hard to take a recursively defined sequence and give an \( a_n \) formula for it and it is also hard to do the reverse.

However in some cases we can. There are some algebraically technical approaches to this but we just want to be able to see the pattern in some examples.

Example 1.7. The sequence \( a_i = 3i \) for \( i \geq 1 \) has terms 3, 6, 9, 12, ... and can also be defined recursively by \( a_0 = 3 \) and \( a_k = 3 + a_{k-1} \) for \( k \geq 1 \).

Example 1.8. The sequence \( a_i = 3^i \) for \( i \geq 1 \) has terms 3, 9, 27, 81, ... and can also be defined recursively by \( a_0 = 3 \) and \( a_k = 3a_{k-1} \) for \( k \geq 1 \).
1.5 Shifting Starting Indices

It may be helpful (as we will see) to shift the indices of a sequence so that it starts at a different index.

Example 1.9. Suppose $a_k = k^2$ for $k \geq 4$. Suppose we wanted to have $k \geq 0$ instead. What could we do to the $a_k$? Well, it’s not hard to see that we would need to have $a_k = (k + 4)^2$ so that the first number we’re squaring is still 4.

There is a way to be formulaic about this. Let’s assume that $k$ is the sequence variable.

**Theorem 1.5.1.** To change the starting index from $a$ to $b$, replace all the $k$ by $k + (a - b)$.

Example 1.10. Suppose $a_k = 5^k + k - 7$ for $k \geq 4$. To change the starting index to 1 we replace $k$ by $k + (4 - 1) = k + 3$ to get $a_k = 5^{k+3} + (k+3) + 7 = 5^{k+3} + k + 10$.
2 Sums

2.1 Notation

Suppose we have a sequence \(a_n\) and wish to add up some finite number of terms, for example:

We use summation notation:

\[
\sum_{n=\text{Starting Index}}^{\text{Ending Index}} a_n
\]

**Example 2.1.** Here are some examples:

\[
\sum_{n=2}^{10} \frac{1}{n} = \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{10}
\]

\[
\sum_{i=3}^{42} n^2 + n = (3^2 + 3) + (4^2 + 4) + \ldots + (42^2 + 42)
\]

2.2 Evaluation - Just Do It!

In simple cases if we wish to evaluate a sum we can simply add up the numbers.

**Example 2.2.** For example:

\[
\sum_{n=2}^{5} n^2 = 2^2 + 3^2 + 4^2 + 5^2 = 4 + 9 + 16 + 25 = 54
\]

2.3 Special Case Formulas

We have several common sums which occur frequently:
\[
\sum_{k=1}^{n} 1 = 1 + 1 + \ldots + 1 = n
\]
\[
\sum_{k=1}^{n} n = 1 + 2 + \ldots + n = \frac{n(n + 1)}{2} \quad \text{Gauss’ Sum}
\]
\[
\sum_{k=1}^{n} n^2 = 1^2 + 2^2 + \ldots + n^2 = \frac{n(n + 1)(2n + 1)}{6} \quad \text{Sum of Squares}
\]
\[
\sum_{k=0}^{n} r^k = 1 + r + r^2 + \ldots + r^n = \frac{1 - r^{n+1}}{1 - r} \quad \text{Geometric Sum}
\]

There are various ways to prove these but for now we’ll wait until we have mathematical induction.

However it is worth noting that these formulas can be used to figure out more complicated sums.

**Example 2.3.** Consider the sum:

\[
\sum_{k=0}^{50} 3(2)^{4k+1}
\]

This looks a bit like the Geometric Sum but in order to use the formula we need to do some rewriting:

\[
\sum_{k=0}^{50} 3(2)^{4k+1} = 3 \sum_{k=0}^{50} (2)^{4k+1}
\]
\[
= 3 \sum_{k=0}^{50} 2^1 (2^4)^k
\]
\[
= 6 \sum_{k=0}^{50} (16)^k
\]
\[
= 6 \left( \frac{1 - 16^{51}}{1 - 16} \right)
\]

Here is a more complex example:

**Example 2.4.** Consider the sum:

\[
\sum_{k=2}^{20} 1 + k + 5(0.3)^k
\]

6
First note we can split it up and factor out the 5:

\[
\sum_{k=2}^{20} 1 + \sum_{k=2}^{20} k + 5 \sum_{k=2}^{20} (0.3)^k
\]

Each of these is familiar but the starting indices are not quite right. For the first, it’s easy to calculate anyway:

\[
\sum_{k=2}^{20} 1 = 19
\]

For the second two we can change the starting indices as long as we subtract the parts we’re adding:

\[
\sum_{k=2}^{20} k = \left[ \sum_{k=1}^{20} k \right] - 1 = \frac{20(20 + 1)}{2} - 1
\]

and:

\[
\sum_{k=2}^{20} (0.3)^k = \left[ \sum_{k=0}^{20} (0.3)^k \right] - 1 - 0.3 = \frac{1 - (0.3)^{21}}{1 - 0.3} - (0.3)^0 - (0.3)^1
\]

All together:

\[
\sum_{k=2}^{20} 1 + k + 5(0.3)^k = 19 + \frac{20(20 + 1)}{2} - 1 + 5 \left( \frac{1 - (0.3)^{21}}{1 - 0.3} - 1 - 0.3 \right)
\]

Here is a nested example:

**Example 2.5.** Consider the sum:

\[
\sum_{n=1}^{10} \sum_{i=1}^{n} 1
\]

We evaluate this from the inside out. Parentheses may help:

\[
\sum_{n=1}^{10} \left[ \sum_{i=1}^{n} 1 \right] = \sum_{n=1}^{10} [n] = \frac{10(10 + 1)}{2}
\]

Here is a more complicated nested example:

**Example 2.6.** Consider the sum:

\[
\sum_{n=5}^{50} \sum_{i=1}^{n+1} i
\]

\[7\]
We evaluate this from the inside out. Parentheses may help:

\[
\sum_{n=5}^{50} \left( \sum_{i=1}^{n+1} \right) = \sum_{n=5}^{50} \frac{(n+1)(n+1+1)}{2} \\
= \sum_{n=5}^{50} \frac{1}{2} \left[ n^2 + 3n + 2 \right] \\
= \frac{1}{2} \left[ \sum_{n=5}^{50} n^2 + 3 \sum_{n=5}^{50} n + 2 \sum_{n=5}^{50} \right]
\]

At this point we might want to do the remaining sums individually:

\[
\sum_{n=5}^{50} n^2 = \left( \sum_{n=1}^{50} n^2 \right) - 1^2 - 2^2 - 3^2 - 4^2 = \frac{50(50+1)(2(50)+1)}{6} - 30
\]

\[
\sum_{n=5}^{50} n = \left( \sum_{n=1}^{50} n \right) - 1 - 2 - 3 - 4 - 5 = \frac{50(50+1)}{2} - 15
\]

\[
\sum_{n=5}^{50} 1 = 51
\]

Thus together the answer is:

\[
\frac{1}{2} \left[ \frac{50(50+1)(2(50)+1)}{6} - 30 + 3 \left( \frac{50(50+1)}{2} - 15 \right) + 2(51) \right]
\]

\[\square\]

### 2.4 Telescoping Sums

Sometimes we find that when we write out a sum almost all the terms will cancel. These are known as telescoping sums.

Here is an example:

**Example 2.7.** Consider the sum:

\[
\sum_{i=1}^{100} \left( \frac{1}{i} - \frac{1}{i+1} \right)
\]

If we write out a number of the terms in this sum:

\[
\sum_{i=1}^{100} \left( \frac{1}{i} - \frac{1}{i+1} \right) = \left( \frac{1}{1} - \frac{1}{2} \right) + \left( \frac{1}{2} - \frac{1}{3} \right) + \left( \frac{1}{3} - \frac{1}{4} \right) + \ldots + \left( \frac{1}{99} - \frac{1}{100} \right) + \left( \frac{1}{100} - \frac{1}{101} \right)
\]

We see that all but the first and last fractions cancel, leaving a result of:

\[
\frac{1}{1} - \frac{1}{101}
\]

\[\square\]
3 Products

3.1 Notation
Suppose we have a sequence $a_n$ and wish to multiply some finite number of terms, for example:

We use product notation:

$$\prod_{n=\text{Starting Index}}^{\text{Ending Index}} a_n$$

**Example 3.1.** Here is some notation and what it means:

$$\prod_{n=2}^{10} \frac{1}{n} = \left( \frac{1}{2} \right) \left( \frac{1}{3} \right) \ldots \left( \frac{1}{10} \right)$$

$$\prod_{i=3}^{42} n^2 + n = (3^2 + 3)(4^2 + 4)...(42^2 + 42)$$

□

3.2 Evaluation - Just Do It!
In simple cases if we wish to evaluate a product we can simply multiply the numbers.

**Example 3.2.** For example:

$$\prod_{n=2}^{5} n^2 = (2^2)(3^2)(4^2)(5^2) = 14400$$

□

3.3 Telescoping Products
Sometimes we find that when we write out a product almost all the terms will cancel. These are known as telescoping products.

Here is an example:

**Example 3.3.** Consider the product:

$$\prod_{i=5}^{100} \frac{i}{i+1}$$

If we write out a number of the terms in this sum:

$$\prod_{i=5}^{100} \frac{i}{i+1} = \left( \frac{5}{6} \right) \left( \frac{6}{7} \right) \left( \frac{7}{8} \right) + \ldots + \left( \frac{99}{100} \right) \left( \frac{100}{101} \right)$$

We see that most of it cancels, leaving a result of:

$$\frac{5}{101}$$

□
4 Recursively Defined Sets

4.1 Introduction
Recursive definitions may be used not just to define sequences of numbers but to define sets.

The basic idea is that we create a set $S$ as follows:

(a) We put some collection of things in $S$.
(b) We give one or more rules which say: If certain things are in $S$, then certain other things must be, too.

4.2 Examples

Example 4.1. We can build a set $S$ of all the even numbers as follows:

(a) We put the number 0 in $S$.
(b) Whenever $a \in S$ we also put $a + 2$ and $a - 2$ in $S$.

It’s fairly obvious (but can be proved!) that this set is the set of all even numbers.

Example 4.2. Suppose we build a set $S$ as follows:

(a) We put the number 1 in $S$.
(b) Whenever $x \in S$ we also put $2x$ in $S$.

It’s fairly obvious (but can be proved!) that this set is the set of all powers of 2 greater than or equal to 1.

We don’t have to just build sets of numbers. Here are two examples.

Example 4.3. Suppose we build a set $S$ as follows:

(a) We put the pair $(0,0)$ in $S$.
(b) Whenever $(a,b) \in S$ we also put $(a,b+1)$ and $(a+1,b+1)$, and $(a+2,b+1)$ in $S$.

What else is in the set other than $(0,0)$?

Well since $(0,0) \in S$ we know that $(0,1), (1,1), (2,1) \in S$. But then since $(0,1) \in S$ we know that $(0,2), (1,2), (2,2) \in S$.

This goes on forever and it’s not entirely clear what is and is not in the set. For example is $(10,10)$ in the set? How about $(10,20)$? How about $(20,10)$?
Here is one with strings.

**Example 4.4.** Suppose we build a set $S$ as follows:

(a) We put the string $X$ in $S$.

(b) Whenever a string $s \in S$ we also put $Xs$ and $YsY$ in $S$.

What else is in the set other than $X$?

Well since $X \in S$ we know that $XX, YXY \in S$. But then since $YXY \in S$ we know that $XYXY, YYXYY \in S$.

This goes on forever and it’s not entirely clear what is and is not in the set.

Here is a somewhat familiar example.

**Example 4.5.** Suppose $p, q, r$ are the only statement variables we have. The set $S$ of sentences in propositional logic involving these statement variables can be be constructed this way:

(a) We put $p, q, r$ in $S$.

(b) Then:

• If $X, Y \in S$ then we put $(X \land Y)$ in $S$.
• If $X, Y \in S$ then we put $(X \lor Y)$ in $S$.
• If $X \in S$ then we put $(\sim X)$ in $S$.

So now we know that for example:

• $p, q \in S$ so $(p \land q) \in S$.
• $(p \land q) \in S$ so $(\sim(p \land q)) \in S$.
• $(\sim(p \land q)), r \in S$ so $((\sim(p \land q)) \lor r) \in S$.
• Etc...

This actually then allows us to construct all statements in propositional logic (or statements equivalent to them) using $p, q,$ and $r$. 

□
5 Binary Trees

5.1 Introduction

Binary trees are another thing which can be built recursively.

Informally a binary tree starts with a root node. The root node can then be
a parent node and connect down to either one or two child nodes. Each child
node can then connect down to either one or two more child nodes. This keeps
going until the tree ends at the leaf nodes, these are the child nodes without
children of their own.

Formally speaking one way to define a binary tree is as a specific type of graph.

Example 5.1. Here is binary tree:

5.2 Recursive Definition

The set of binary trees $B$ can also be defined recursively as follows:

(a) A single node is a binary tree, so a single node is in $B$.

(b) If $T_1$ and $T_2$ are binary trees (are in $B$) then we can create a new binary
tree by taking a new node as the root and attaching $T_1$ and $T_2$ on branches
below it.

(c) If $T_1$ is a binary tree (is in $B$) then we can create a new binary tree by
taking a new node as the root and attaching $T_1$ on a branch below it.

According to this definition we get new binary trees as follows:

and are trees and therefore so is

and are trees and therefore so is
is a tree and therefore so is

(This last one is the first tree listed in the section.)

5.3 Properties of Binary Trees

Binary trees are heavily used in various algorithms and have several properties which are critical to know.

We'll just mention them here but we'll prove some of them via structural induction.

**Definition 5.3.1.** For a binary tree $T$ define:

- $N(T)$ The number of nodes in the tree.
- $T(E)$ The number of edges (connections) in the tree.
- $L(T)$ The number of leaves in the tree.
- $H(T)$ The height of the tree. A single node has height 0.

In addition:

**Definition 5.3.2.** A binary tree is **perfect** if all the notes (except the leaves) have exactly two children and if all the leaves are at exactly the same depth.

**Example 5.2.** Here is a perfect tree:

Theorem 5.3.1. For a binary tree $T$ we have:

- $N(T) = E(T) + 1$
- $L(T) \leq 2^{H(T)}$
- $N(T) \leq 2^{H(T)+1} - 1$

With $= \text{iff } T$ is perfect.

Proof. See the notes on structural induction. QED

It’s worth noting that as a result of the latter two we have the following:
Theorem 5.3.2. For a binary tree $T$ we have:

\begin{align*}
H(t) &\geq \lg L(t) \quad \text{With } = \text{ iff } T \text{ is perfect.} \\
H(t) &\geq \lg(N(t) - 1) - 1 \quad \text{With } = \text{ iff } T \text{ is perfect.}
\end{align*}