Lecture outline: Recurrence relations: introduction, first-order recurrences

We introduce the general notion of recurrence, that allows to define an infinite sequence of integers $a_0, a_1, a_2 \dots$ via local relation between blocks of consecutive terms: we will see that if a_n is defined in terms of a block of previous terms $a_{n-1}, \dots, a_{n-k}$, we can often compute a closed formula for a_n. Often the sequence $a_0, a_1, \dots$ we are interested into is the counting sequence of a family of combinatorial objects A: a_n is the number of objects of A of size n.

In this first set of notes on recurrences, we will study various elements:

- introduction to recurrences with various examples of counting sequences;
- definitions of the different kinds of recurrences;
- introduction of the link between recurrences and proofs by induction;
- method to solve first-order linear recurrences.

Introductory example

We start by a general question inspired by the example we did not complete on slide 18 in the notes on strings: how many strings over the alphabet $\{A, C, G, T\}$ of size n are-there in which an A is always followed by a C.

We first introduce some notation. We denote by $\mathcal{C}$ the (infinite) set of all strings over $\{A, C, G, T\}$ where an A is always followed by a C. We denote by $\mathcal{C}_n$ the set of such strings of size n, and by c_n the cardinality $|\mathcal{C}_n|$ of this set of strings.

The question is then: find a formula for c_n.

The first thing that comes to mind when facing such a question, if the answer does not appear immediately, is to start by finding c_n explicitly for small values of n, by listing all objects of $\mathcal{C}_n$.

$$\mathcal{C}_0 = \{\epsilon\}, \text{ where } \epsilon \text{ as previously represents the empty string.} : \ c_0 = 1$$

$$\mathcal{C}_1 = \{C, G, T\} : \ c_1 = 3$$

$$\mathcal{C}_2 = \{CC, CG, CT, GC, GG, GT, TC, TG, TT, AC\} : \ c_2 = 10$$

$$\mathcal{C}_3 = \{CAC, CCC, CCG, CCT, CGC, CGG, CGT, CTC,$$
$$CTG, CTT, GAC, GCC, GCG, GCT, GGC, GGG, GGT, GTC, GTG, GTT,$$
$$TAC, TCC, TCG, TCT, TGC, TGG, TGT, TTC, TTG, TTT, ACC, ACG, ACT\} :$$
$$c_3 = 33$$

Then a pattern might (or might not) appear that allows to guess the formula for c_n.

If we can guess a formula, we can then try to prove it by induction (we will see that later). Otherwise, we can try to find a way to define c_n, for an arbitrary n, in terms of terms of smaller index of the counting sequence $c_0, c_1, \ldots$.

Here, if we look at the terms c_0, c_1, c_2, c_3, there is no obvious pattern that appears.

But we can understand/learn from the process of constructing the sets $\mathcal{C}_0, \mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3$, or also from just looking at them, the following principle:

Recursive decomposition of $\mathcal{C}_n$.

For $n \geq 2$, a string of $\mathcal{C}_n$ is

either a prefix C, G, T (three choices) followed by a string in $\mathcal{C}_{n-1}$

or the prefix AC (one choice) followed by a string in $\mathcal{C}_{n-2}$.

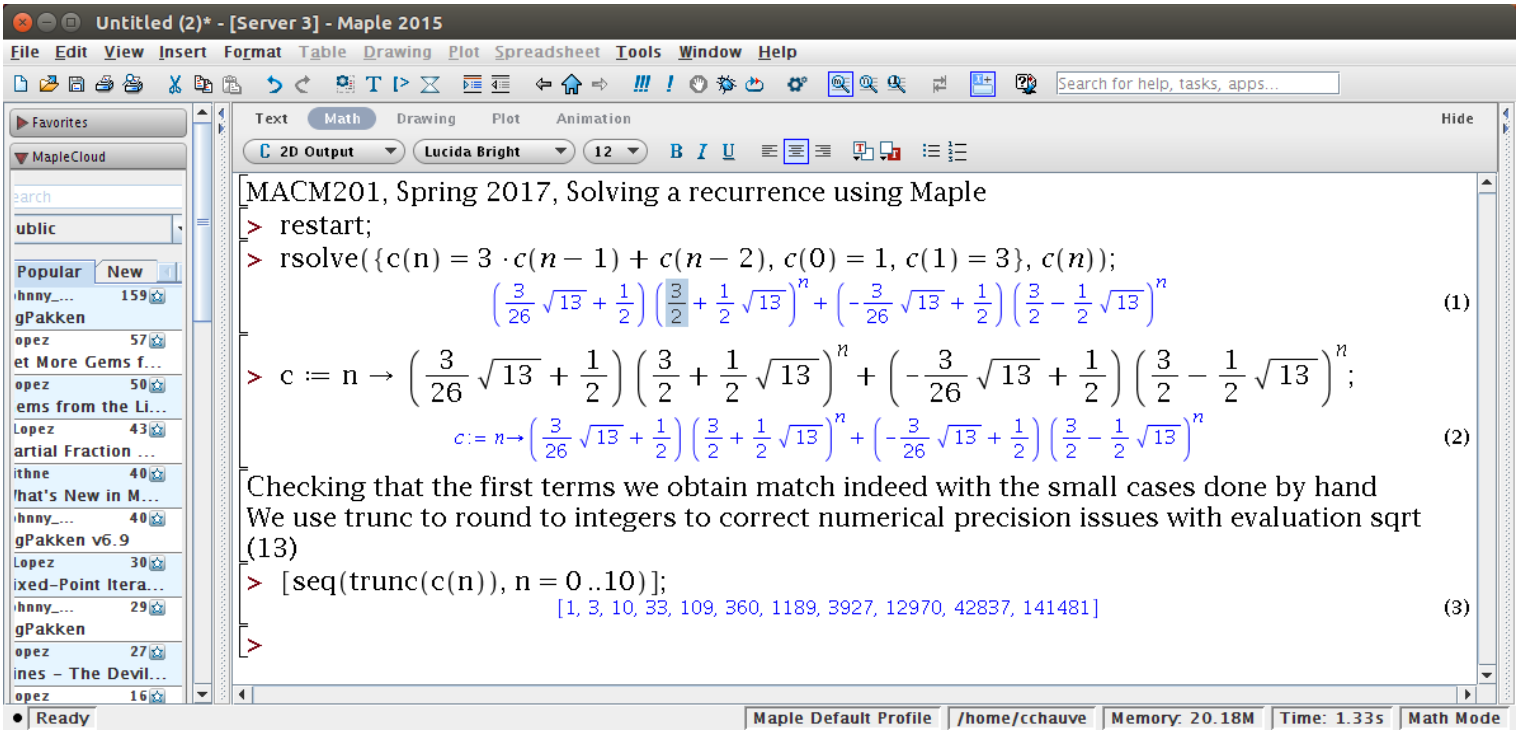
We can then translate this into a **recurrence relation** as follows:

$$\begin{cases} c_0 &= 1 \\ c_1 &= 3 \\ c_n &= 3c_{n-1} + 1c_{n-2} \quad n \geq 2 \end{cases}$$

The recurrence relation we obtained

$$\begin{cases} c_0 &= 1 \\ c_1 &= 3 \\ c_n &= 3c_{n-1} + 1c_{n-2}\,, \quad n \geq 2 \end{cases}$$

can then be solved, either manually (as we will learn to do it soon), or using a computer algebra software such as Maple:



So we can see that the solution we were looking for, which is a closed formula for c_n, is

$$c_n = \left(\frac{3}{26}\sqrt{13} + \frac{1}{2}\right)\left(\frac{3+\sqrt{13}}{2}\right)^n + \left(-\frac{3}{26}\sqrt{13} + \frac{1}{2}\right)\left(\frac{3-\sqrt{13}}{2}\right)^n$$

and that the first few terms obtained with this formula match the terms we computed by hand.

Remark 1. There were four different parts in the work we did to answer a counting question:

1. The **definition of the counting question** itself (here find a formula for c_n), which includes understanding which combinatorial objects we want to count, what is the notion of size, what are the properties of these objects, ...

2. **Setting-up the recurrence.** This is often the most difficult part because there is no general technique. It is more an art than a technique,that requires general skills in discrete mathematics, an ability to understand discrete structures, to see how they decompose.

   A general principle we used here however is that we often decompose strings by trying to identify a few possible cases for a **prefix** that are followed by smaller strings from the same family $\mathcal{C}$.

   Ideally, we will see that we should also **prove** that the recurrence is correct (*i.e.* defines well the counting sequence of interest). Often this proof will be by **induction**.

3. **Solving the recurrence.** As we illustrated it, this can often be done by computer, very easily, and in fact I encourage you strongly, if you want to check that your solution is correct, to use Maple to do so, as it is available on all SFU computers.

   However, we will learn to solve recurrences by hand. It is purely technical: you just need to learn the technique we will describe in later lectures.

4. **Checking the obtained formula.** This is a fourth step that is often forgotten. Once you have a formula for c_n, and assuming you have generated values of c_n for small values of n, it is important to check that the formula you obtained matches the values computed by hand from listing all objects of small size. If not, it points to an error in steps 2 or 3, that needs to be corrected.

So recurrence questions can be of various types, the most common being:

- The complete setup: you are given a counting question and you need to do all steps.
- You are given the counting question and you have to find the recurrence.
- You are given the counting question and the recurrence and you have to prove the recurrence is correct
- You are given a recurrence (that might not be related to a combinatorial counting question) and you have to solve it (and ideally check your solution is correct by looking at small cases).

Remark 2. The step 2, setting-up the recurrence, is important. If it is not done properly, then obviously the formula you will obtain will be wrong (which you will realize if you also do step 4, checking the obtained formula).

Important remark.

The ability to set-up properly a recurrence for a counting sequence is as important as a skill in discrete mathematics, than the ability to solve the recurrence.

Indeed, assuming you end-up alone having to solve a counting question, if you can not set-up a recurrence, you can do nothing. If you can set-up a recurrence, and it is correct, then you can very easily

- Generate quite a few terms by hand or by a computer program;
- Solve the recurrence using a computer algebra software;
- Solve the recurrence by hand;

but at least you can do something.

Basically, if you can set-up a recurrence for the counting sequence of a family of combinatorial objects (or for the analysis of some algorithm as we will see soon), this means that you have **understood the combinatorial structure of these objects** and this is very important.

In the skills about recurrences I expect you will acquire, the ability to set-up recurrences will be weighted (understand graded) as much at least than the ability to solve them.

The concept of recurrence

Informally, a recurrence is a way to define an infinite integer sequence $a_0, a_1, a_2, \ldots$ by giving initial k terms $a_0, \ldots, a_{k-1}$ (the number of initial terms to give depends on the **order** of the recurrence), and a functional relation between blocks of successive terms $a_{n-k}, \ldots, a_n$ (again k depends of the order of the recurrence). The nature of this functional relation is central to define the nature of a recurrence, in terms of **linearity** and **homogeneity**.

Definition. A recurrence relation of **order** k, for a sequence of integers $a_0, a_1, a_2, \ldots$, is defined by:

- Giving the values of the first k terms of the sequence, $a_0, \ldots, a_{k-1}$
- Giving a functional relation $f(a_{n-k}, \ldots, a_n) = g(n)$ satisfied by all a_n, $n \geq k$, for some function $g(n)$ that depends only of n.

For now we will assume that $f(a_{n-k}, \ldots, a_n)$ is a **polynomial** in the variables $\{a_{n-k}, \ldots, a_n\}$.

It is **linear** if f is a polynomial of degree 1, i.e do not contain any product of the a_is. i.e. a_n is a linear function of $a_{n-1}, \ldots a_{n-k}$

It is **homogeneous** if $g(n) = 0$.

It has **constant coefficients** if the coefficients of the a_is in $f(a_{n-k}, \ldots, a_n)$ are constant real numbers, *i.e.* do not depend on n.

Example 1: Counting binary strings.

Let b_n be the number of binary strings of size n, for $n \geq 0$. We know that $b_n = 2^n$. We can describe this formula through a recurrence relation

$$\begin{cases} b_0 &= 1 \\ b_n &= 2b_{n-1} \quad n \geq 1 \end{cases}$$

linear function; depends only on prev. term; const term of linear funct is zero

This is a first-order, linear, homogeneous recurrence relation with constant coefficients.

0 1 0 0 . . . 0 1 — $n-1$; 0 or 1

$b_n = 2 b_{n-1}$ — constant

closed form $b_n = 2^n$

Example 2: Factorial numbers.

Consider the sequence $a_0, a_1, a_2, \ldots$ where $a_n = n!$ for $n \geq 0$. It can also be defined as follows:

$$\begin{cases} a_0 &= 1 \\ a_n &= n a_{n-1} \quad n \geq 1 \end{cases}$$

This is a recurrence relation of first order, linear, homogeneous, but with variable coefficients.

$$\begin{aligned} a_n &= n\, a_{n-1} \\ &= n(n-1)\, a_{n-2} \\ &= n(n-1)(n-2)\, a_{n-3} \\ &\vdots \\ &= n! \end{aligned}$$

Example 3: Fibonacci numbers.

The celebrated Fibonacci numbers $f_0, f_1, f_2, \ldots$ are defined by the following recurrence:

$$\begin{cases} f_0 &= 0 \\ f_1 &= 1 \\ f_n &= f_{n-2} + f_{n-1} \quad n \geq 2 \end{cases}$$

This is a recurrence relation of second order, linear, homogeneous, with constant coefficients.

$0, 1, 1, 2, 3, 5, 8, 13, 21, 34, \ldots$

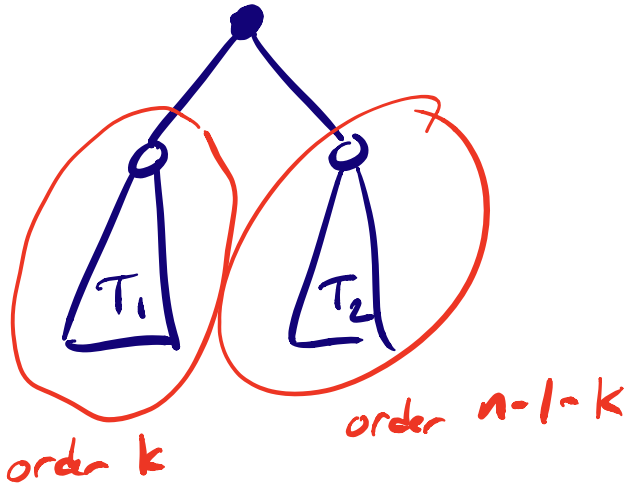
$$f_n = \frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right]$$

Example 4: Rooted, ordered, binary trees.

Let t_n denote the number of rooted ordered binary trees of ~~size~~ $n \geq 1$. We will see later that $t_1, t_2, \ldots$ satisfies the recurrence

$$\begin{cases} t_1 &= 1 \\ t_n &= \sum_{k=1}^{n-1} t_k t_{n-1-k} \quad n \geq 2 \end{cases}$$

This is a homogeneous but non-linear recurrence relation (quadratic in fact) with constant coefficients.

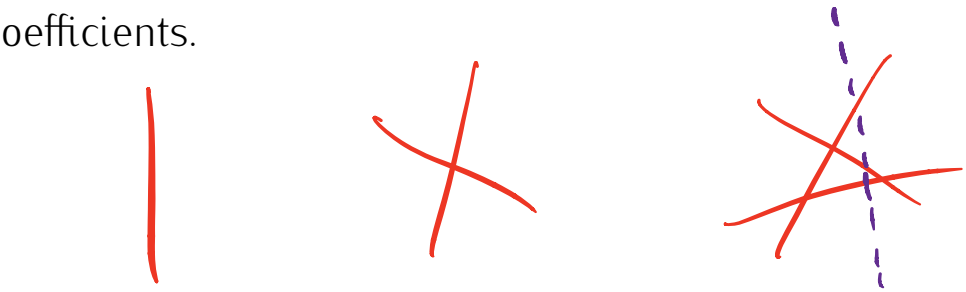


Example 5: Regions of a line arrangement.

Suppose we draw n lines in the plane, with no two lines being parallel and no three (or more) lines intersecting at the same point. The number r_n of regions of the plane defined by these lines satisfies the recurrence

$$\begin{cases} r_0 &= 1 \\ r_n &= r_{n-1} + n \quad n \geq 1 \end{cases}$$

This is a first-order non-homogeneous linear recurrence relation with constant coefficients.



$r_1 = 2$ $r_2 = 4$ $r_3 = 7$

each new line gives 1 new region
+ 1 more for each intersection pt
# of intersections

Homogeneous first order linear recurrence.

Theorem. The homogeneous first order linear recurrence with constant coefficients

$$\begin{cases} a_0 &= A \\ a_n &= d a_{n-1} \quad n \geq 1 \end{cases}$$

where d is a fixed real number, has a **unique solution**

$$a_n = A d^n.$$

Example. Solve the following recurrence.

$$\begin{cases} a_0 &= 3 \\ a_n &= 5 a_{n-1} \quad n \geq 1 \end{cases}$$

Ans $a_n = 3 \cdot 5^n$

$a_0 = A$

$a_1 = d a_0 = dA$

$a_2 = d a_1 = d^2 A$

$a_3 = d a_2 = d^3 A$

$\vdots$

$a_n = d a_{n-1}$

$= d^2 a_{n-2}$

$= d^3 a_{n-3}$

$\vdots$

$= d^n a_0$

$= A d^n$

Example. Solve the following recurrence.

$$\begin{cases} a_0 &= 3 \\ a_n^2 &= 5a_{n-1}^2 \quad n \geq 1 \end{cases}$$

This is **NOT** a linear recurrence, as it involves the square of a_n.

But we can do clever transformation: consider the numbers $b_n = a_n^2$, $n \geq 0$. They satisfy the following recurrence

$$\begin{cases} b_0 &= 9 \\ b_n &= 5b_{n-1} \quad n \geq 1 \end{cases}$$

From there we can apply the theorem.

$b_n = 9 \cdot 5^n$

Non-homogeneous first-order linear recurrences

Definition. A non-homogeneous first order linear recurrence with constant coefficients is a recurrence

$$\begin{cases} a_0 &= A \\ a_n &= a_{n-1} + g(n) \quad n \geq 1 \end{cases}$$

where $g(n)$ a non-zero function.

Important remark. So far we will not see a general method to solve non-homogeneous first-order linear recurrences, but we will use the following principle, that applies to all kinds of recurrences:

- Use the recurrence to generate the first few terms;
- Observe these terms to see if there is a pattern,if we can **guess** a solution to the recurrence;
- Once we have guessed a solution, **prove it is correct**. Often **induction** is the best way to prove it.

The most appropriate way to prove the guessed formula indeed is a solution to the recurrence is to use **induction** as follows:

1. Check base case If the recurrence is of order k, check that the first k terms of the counting sequence match with the formula. If not, the guessed formula is wrong.

2. State induction hypothesis. Assume the recurrence is true up to order $n-1$, $n \geq k$.

3. Prove the general case. Using the induction hypothesis and the recurrence relation, prove that the guessed formula is correct at order n, $n \geq k$.

Example 1. The following recurrence is a non-homogeneous first order linear recurrence with constant coefficients.

$$\begin{cases} a_1 &= 0 \\ a_n &= a_{n-1} + (n-1) \quad n \geq 2 \end{cases}$$

Let's compute the first few terms.

$a_1 = 0$

$a_2 = a_1 + 1 = 0 + 1$

$a_3 = a_2 + 2 = 0 + 1 + 2$

$a_4 = a_3 + 3 = 0 + 1 + 2 + 3$

So it seems that $a_n = 1 + 2 + 3 + \cdots + (n-1)$. We can prove it by **induction**:

Claim: $a_n = 0 + 1 + 2 + 3 + \cdots + (n-1) = n(n-1)/2, \ n \geq 1$

Base case: $a_1 = 0$ by hypothesis (i.e. the given recurrence).

Induction hypothesis: $a_{n-1} = 1 + 2 + 3 + \cdots + (n-2), \ n \geq 1$

General case: We want to prove that $a_n = 0 + 1 + 2 + 3 + \cdots + (n-1), \ n \geq 1$.
By induction hypothesis: $a_{n-1} = (n-1)(n-2)/2$
By the given recurrence: $a_n = a_{n-1} + (n-1)$
So, combining both we obtain $a_n = (n-1)(n-2)/2 + (n-1) = (n-1)n/2$.
This proves that the guessed formula was correct.

Example 2. Solve the recurrence

$$\begin{cases} a_0 &= 0 \\ a_n &= a_{n-1} + 2n \quad n \geq 1 \end{cases}$$

$a_0 = 0$

$a_1 = 0 + 2$

$a_2 = a_1 + 4 = 0 + 2 + 4$

$a_3 = a_2 + 6 = 0 + 2 + 4 + 6$

$a_n = 0 + 2 + 4 + \ldots + 2n$

$= 2(0 + 1 + \ldots n)$

$= n(n+1)$