

Lecture 28: Recursion, lists and sequences

A list or sequence of objects from a set X is a function f from $\{1, 2, \dots, n\}$ to X , or (if infinite) from $\{1, 2, 3, \dots\}$ to X .

We usually write $f(k)$ as x_k and the list as $\langle x_1, x_2, \dots, x_n \rangle$, or $\langle x_1, x_2, x_3, \dots \rangle$. Thus

$$f(1) = x_1 = \text{first item on list}$$

$$f(2) = x_2 = \text{second item on list}$$

$$\vdots$$

The empty list is written $\langle \rangle$.

Example.

$\langle m, a, t, h, s \rangle$ is a function f from $\{1, 2, 3, 4, 5\}$ into the English alphabet, with $f(1) = m$, $f(2) = a$, etc.

Sequences

A sequence is also a list, but when we use the term sequence we are usually interested in the rule by which the successive terms t_1, t_2, \dots are defined.

Often, the rule is a recurrence relation.

Example. Arithmetic sequence

$$a, a + d, a + 2d, a + 3d, \dots$$

This is defined by

Initial value. $t_1 = a$

Recurrence relation. $t_{k+1} = t_k + d$

“Unfolding” this recurrence relation from t_n back to t_1 , we see that d gets added $n - 1$ times, hence

$$t_n = a + (n - 1)d.$$

Example. Geometric sequence

$$a, ar, ar^2, ar^3, \dots$$

Initial value. $t_1 = a$

Recurrence relation. $t_{k+1} = rt_k$

“Unfolding” t_n , we see that multiplication by r is done $n - 1$ times, hence

$$t_n = ar^{n-1}.$$

The above recurrence relations are called *first order* because t_{k+1} depends on only the previous value, t_k . (Or, because the values of all terms follow from one initial value.)

A *second order* recurrence relation requires *two* initial values, and is usually harder to unfold.

Example. A simple sequence in disguise

Initial values. $t_0 = 1, t_1 = 2$

Recurrence relation. $t_{k+1} = 2t_k - t_{k-1}$

Calculating the first values, we find

$$t_2 = 2t_1 - t_0 = 2 \times 2 - 1 = 3,$$

$$t_3 = 2t_2 - t_1 = 2 \times 3 - 2 = 4,$$

$$t_4 = 2t_3 - t_2 = 2 \times 4 - 3 = 5.$$

It looks like $t_n = n + 1$, and indeed we can prove this by induction.

Base step: $t_0 = 1 = 0 + 1$ and $t_1 = 2 = 1 + 1$.

We do the induction step by strong induction: assuming $t_n = n + 1$ for all $n \leq k$, we deduce that $t_{k+1} = k + 2$. We have

$$\begin{aligned} t_{k+1} &= 2t_k - t_{k-1} \quad \text{by the recurrence relation} \\ &= 2(k+1) - k \quad \text{by our assumption} \\ &= 2k + 2 - k = k + 2 \quad \text{as required.} \end{aligned}$$

This completes the induction.

Example. Fibonacci sequence

Initial values. $t_0 = 0, t_1 = 1$

Recurrence relation. $t_{k+1} = t_k + t_{k-1}$

It is possible to write t_n as a function of n .

The function is not obvious, because it involves $\sqrt{5}$...

$$t_n = \frac{1}{\sqrt{5}} \left(\left(\frac{1 + \sqrt{5}}{2} \right)^n - \left(\frac{1 - \sqrt{5}}{2} \right)^n \right).$$

We do not go into how to find such a formula in this unit.

However, if someone gave you such a formula you could prove it is correct by induction.

Relations - homogeneous and inhomogeneous

Recurrence relations such as

$$t_{k+1} = 2t_k$$

and

$$t_{k+1} = t_k + t_{k-1}$$

in which each term is a multiple of some t_j , are called *homogeneous*.

The characteristic property of any homogeneous equation is that if $t_n = f(n)$ is a solution, then so is $t_n = cf(n)$, for any constant c .

E.g. $t_n = 2^n$ is a solution of $t_{k+1} = 2t_k$, and so is $t_n = c2^n$, for any constant c .

Relations like $t_{k+1} = t_k + 3$, in which there is a term other than the t_j terms, are called *inhomogeneous*.

Homogeneous recurrence relations are usually easier to solve, and in fact there is a general method for solving them (which we will not cover in this unit).

There is no general method for solving inhomogeneous recurrence relations, though they can often be solved if the term other than the t_j terms is simple.

Questions

28.1 Find the next four values of each of the following recurrence relations. What order is each recurrence relation? Which are homogeneous and which are inhomogeneous?

(a) $r_{k+1} = r_k + k^2$, $r_0 = 0$.

(b) $s_{k+1} = 3s_k - 2s_{k-1}$, $s_0 = 1$, $s_1 = 2$.

(c) $t_{k+1} = t_k + t_{k-2} + 1$, $t_0 = 1$, $t_1 = 1$, $t_2 = 1$.

Order: (a) is first order, (b) is second order, and (c) is third order.

Homogeneous? (b) is homogeneous but (a) and (c) are not.

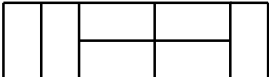
Next terms of (a): $r_1 = 0$, $r_2 = 1$, $r_3 = 5$, $r_4 = 14$.

Next terms of (b): $s_2 = 4$, $s_3 = 8$, $s_4 = 16$, $s_5 = 32$.

Next terms of (c): $t_3 = 3$, $t_4 = 5$, $t_5 = 7$, $t_6 = 11$.

Questions

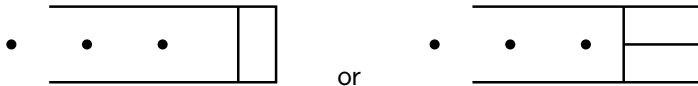
28.2 Let T_n be the number of ways of tiling a $2 \times n$ strip with 2×1 tiles (which may be rotated so they are 1×2). Find T_n for $n = 1, 2, 3, 4$. Find a recurrence relation for T_n .

E.g.  is one option counted by T_7 .

$$T_1 = 1, T_2 = 2, T_3 = 3, T_4 = 5.$$

How do we find a recurrence relation for T_n ?

Consider what happens at the right hand end:



In the first case removing the right most tile leaves us with a tiling of a $2 \times (n - 1)$ strip.

In the second case removing the two right most tiles leaves us with a tiling of a $2 \times (n - 2)$ strip.

Since these possibilities do not overlap, and between them include every option, we see that $T_n = T_{n-1} + T_{n-2}$.

Questions

28.3 Let C_n be the number of ways of writing down n pairs of brackets so that they are correctly matched. Find C_n for $n = 1, 2, 3, 4$. Can you explain why $C_{k+1} = \sum_{i=0}^k C_i C_{k-i}$?

E.g. $((()))()$ is counted by C_4 , whereas $((()))(())$ is not.

$$C_1 = 1 \quad ()$$

$$C_2 = 2 \quad ()() \text{ or } (())$$

$$C_3 = 5 \quad ()()(), ()(()), (())(), ((())), (()())$$

$$C_4 = 14 \quad ()()()(), ()()(()), ()()()(), ()((())), ()(()()), (())()(), ((()))(), (())()(), (())()(), (())()(), (())()(), (((()))), (((())))$$

$$\text{Why is } C_4 = \sum_{i=0}^3 C_i C_{3-i} = C_0 C_3 + C_1 C_2 + C_2 C_1 + C_3 C_0?$$

How can we split the problem up into smaller versions of itself?

From the suggested recurrence how many smaller pieces should we be looking for? 2

Using the parameter n as a measure of the size of the problem...

what is the total size of the pieces relative to the size overall? $n-1$.

The key is to think about the $)$ which matches the **first** $($.

This makes one pair of parenthesis. Inside it is a copy of the original problem... it's a correctly matched string of parentheses. To the right is another copy of the original problem... another correctly matched string of parentheses.

$((()())())$

There can be any number of pairs of parentheses inside the blue pair, and any number to the right of them. But overall the total number of pairs of parentheses should be n (including the blue pair).

Also, note that we need to consider C_0 , the number of ways to pair up 0 parentheses (this can be done in 1 way, namely the empty string). We need this because there might be nothing inside the blue pair; or there might be nothing to the right of it.