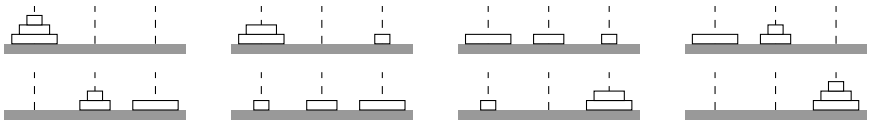
Recurrences are formulas that describe the value of a function of positive integers at an input in terms of the value of the same function at smaller inputs. We will see examples of problems in which recurrences come about and show how to solve them.

1 Towers of Hanoi

You have three posts and a stack of n disks of different sizes, stacked up from largest to smallest, on the first post.

The objective is to move the stack on to the third post, one disk at a time, so that a larger disk is never placed on top of a smaller one. Here is a way to do it for three disks:



What about four disks? You can try doing some experiments, but you might find the process quite involved. Instead, let us think inductively and use our solution for 3 disks as a "subroutine":

More generally, suppose we have solved the Towers of Hanoi problem for n disks. Here is how to solve it for $n+1$ disks: Ignore the largest disk and apply the solution for n disks to move the remaining tower to the middle pole. Then move the largest disk to the rightmost pole. Ignore the largest disk again and the apply the solution for n disks to move the remaining tower to the rightmost pole.

Let $T(n)$ be the number of moves we performed to move n disks. Then $T(n)$ satisfies the equation

$$T(n+1) = 2T(n) + 1. \tag{1}$$

Here, each of the two $T(n)$s accounts for the steps taken in each of the inductive moves applied to the tower of n smaller blocks, and the extra one is for the move of the largest block from the left pole to the right pole.

This type of equation is a *recurrence*. If we know $T(1)$, it allows us to calculate $T(2)$, $T(3)$, and so on. In our case, $T(1) = 1$ since a one block tower can be rearranged in one move.

2 Solving recurrences

One objective is to understand how a recurrence like (1) behaves for large values of n. The best thing to do is to get a closed-form formula for $T(n)$, if we can.

To do this I recommend you work backwards. From equation (1) we get

$$\begin{aligned} T(n) &= 2T(n-1)+1 \\ &= 2(2T(n-2)+1)+1 = 2^2T(n-2)+(2+1) \\ &= 2^2(2T(n-3)+1)+(2+1) = 2^3T(n-3)+(2^2+2+1). \end{aligned}$$

You start to spot a pattern: If we continue this process for enough steps until we get a $T(1)$ on the right hand side – this is $n-1$ steps – we get

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

Since $T(1)=1$, it seems that

$$T(n) = 2^{n-1}+2^{n-2}+\cdots+2+1 = 2^n-1$$

using the formula for geometric series from last lecture.

We can indeed verify that this guess is correct:

Theorem 1. *The assignment $T(n)=2^n-1$ satisfies equation (1) for all $n \geq 1$.*

Proof. Assume $T(n)=2^n-1$ and $n \geq 1$. Then

$$2T(n)+1 = 2\cdot(2^n-1)+1 = 2^{n+1}-2+1 = 2^{n+1}-1 = T(n+1). \qquad \square$$

Let's do another example. Recall the Beneš network B_n from Lecture 6. How many edges does the digraph B_n have? Recall that B_n was constructed recursively as follows. B_1 is a network with two sources, two sinks, and all four possible directed edges between them. The network B_n was constructed by taking two disjoint copies of B_{n-1}, adding 2^n new sources, 2^n new sinks, and two new incoming/outgoing edges for each source/sink. The number of edges $E(n)$ of B_n is given by the recurrence

$$\begin{aligned} E(n) &= 2E(n-1)+2^{n+2} \\ E(1) &= 4. \end{aligned} \tag{2}$$

Let us try to solve this recurrence by working backwards:

$$\begin{aligned} E(n) &= 2E(n-1)+2^{n+2} \\ &= 2(2E(n-2)+2^{n+1})+2^{n+2} = 2^2E(n-2)+2\cdot 2^{n+2} \\ &= 2^2(2E(n-3)+2^n)+2\cdot 2^{n+2} = 2^3E(n-3)+3\cdot 2^{n+2} \end{aligned}$$

Continuing this reasoning for $n-1$ steps we obtain the guess

$$E(n) = 2^{n-1}E(1)+(n-1)\cdot 2^{n+2} = 2^{n-1}\cdot 4+(n-1)\cdot 2^{n+2} = n\cdot 2^{n+2}-2^{n+1}.$$

We can now verify that our guess is correct:

Theorem 2. *For all $n \geq 1$, the digraph B_n has $E(n) = n\cdot 2^{n+2}-2^{n+1}$ edges.*

Proof. We prove the theorem by induction on n. The digraph B_1 has $4 = 1\cdot 2^3-2^2$ edges as desired. Now assume B_n has $n\cdot 2^{n+2}-2^{n+1}$ vertices. By recurrence (1), B_{n+1} then has

$$2\cdot(n\cdot 2^{n+2}-2^{n+1})+2^{n+3} = n2^{n+3}+2^{n+2} = (n+1)2^{n+3}-2^{n+2}$$

edges as desired. $\square$

3 Merge sort

Merge sort is the following procedure for sorting a sequence of n numbers in nondecreasing order:

Input: A sequence of n numbers.
Merge Sort Procedure:
If $n = 1$, do nothing. Otherwise,
Recursively sort the left half of the sequence.
Recursively sort the right half of the sequence.
Merge the two sorted sequences in increasing order.

For example, if the input sequence is

10 7 15 3 6 8 1 9

the sequence is split in the first half 10 7 15 3 and the second half 6 8 1 9. Each half is sorted recursively to obtain

3 7 10 15 and 1 6 8 9

Then the two sequences are merged. To merge the first and second half, we compare the leftmost numbers in both sequences, take out the smaller of the two and write it as the next term in the output sequence until both halves become empty.

first half	second half	output
3 7 10 15	**1** 6 8 9	
3 7 10 15	6 8 9	1
7 10 15	**6** 8 9	1 3
7 10 15	8 9	1 3 6
10 15	**8** 9	1 3 6 7
10 15	**9**	1 3 6 7 8
10 15		1 3 6 7 8 9
15		1 3 6 7 8 9 10
		1 3 6 7 8 9 10 15

For example, in the first line, the leftmost numbers in the first and the second half are 1 and 3, respectively. They are compared to each other, and since 1 is smaller, it is taken out and written in the output.

In this example, exactly seven pairwise comparisons were made in the merging phase. In general, if the length of the sequence is n, the number of comparisons when merging the two halves is $n-1$.

We want to count the total number $C(n)$ of comparisons that Merge Sort performs when sorting a sequence of n numbers. We will assume that n is a power of two so that any time the sequence is split in half in a recursive call the two halves are equal.

To sort n numbers (for $n > 1$), Merge Sort makes two recursive calls to sequences of length $n/2$ and performs exactly $n - 1$ comparisons in the merging phase. This gives the recurrence

$$C(n) = 2C(n/2) + (n - 1) \qquad \text{for } n > 1 \tag{3}$$

with the base case $C(1) = 0$.

Let's try to guess a solution for this recurrence by working backwards as usual:

$$\begin{aligned} C(n) &= 2C(n/2) + (n-1) \\ &= 2(2C(n/2^2) + (n/2-1)) + (n-1) = 2^2C(n/2^2) + 2n - (2+1) \\ &= 2^2(2C(n/2^3) + n/2^2 - 1) + 2n - (2+1) = 2^3C(n/2^3) + 3n - (2^2+2+1). \end{aligned}$$

We reach $C(1)$ on the right hand side after $\log n$ steps.[1] The expression we get on the right is

$$nC(1) + (\log n)n - (2^{\log n - 1} + \cdots + 2 + 1).$$

By our base case, $C(1) = 0$. By the formula for geometric sums, the last term equals $2^{\log n} - 1 = n - 1$. This suggests the guess

$$C(n) = n \log n - n + 1.$$

This formula indeed sets $C(1)$ to zero as desired. You can now verify on your own that this guess solves the recursion by a calculation.

Theorem 3. *The assignment $C(n) = n \log n - n + 1$ satisfies the equations (3).*

4 Homogeneous linear recurrences

You are given an unlimited supply of 1×1 and 2×1 tiles. In how many ways can you tile a hallway of dimension $n \times 1$ using the tiles? For example, here are all five possible tilings for $n = 4$:

Let $f(n)$ denote the number of desired tilings of an $n \times 1$ hallway. If $n = 0$ there is exactly one possible tiling — use no tiles — so $f(0) = 1$. If $n = 1$ there is also one tiling — use a single 1×1 tile — so $f(1) = 1$.

When $n \geq 2$, we distinguish two possible types of tilings. If the tiling starts with a 1×1 tile, then the remaining part of the corridor can be tiled in $f(n-1)$ ways. If the tiling starts with a 2×1 tile, then there are $f(n-2)$ possible tilings. Since these two possibilities cover each possibility exactly once, we obtain the recurrence

$$f(n) = f(n-1) + f(n-2) \qquad \text{for every } n \geq 2. \tag{4}$$

This is an example of a *homogeneous linear recurrence*. Such recurrences can be solved using the following guess verify method.

Initially, we forget about the "base cases" $f(0) = 1, f(1) = 1$ and focus on the equations (4). We look for solutions of the type $f(n) = x^n$ for some nonzero real number x. The reason we expect such solutions to work out is because for every $n \geq 2$, the equation $x^n = x^{n-1} + x^{n-2}$ is the same as

$$x^2 = x + 1$$

since all we did was scale down both sides by a factor of x^{n-2}. This is a quadratic equation in x and we can use the quadratic formula to calculate its solutions