Discrete Mathematics
Recurrences

Saad Mneimneh

1 What is a recurrence?

It often happens that, in studying a sequence of numbers a_n, a connection between a_n and a_{n-1}, or between a_n and several of the previous a_i, $i < n$, is obtained. This connection is called a recurrence relation. In spirit, a recurrence is similar to induction, but while induction is a proof technique, recurrence is more like a definition method.

2 Cutting the plane

Consider n lines in the plane such that:

- no two lines are parallel
- no three lines pass through the same point

We refer to such a configuration as n lines in *general position*. The lines divide the plane into a number of regions. How many regions do we get?

At the first glance, one might think that adding a line doubles the number of regions. For instance, one line divides the plane into two regions; two lines into four. However, it is easy to verify when $n = 3$ that this trend does not continue; otherwise, we end up with 2^n regions. At this juncture, it might be reasonable to start experimenting with $n = 3$ lines. Then you will invariably get 7 regions. With $n = 4$ lines, you will always get 11 regions. It looks like after adding the n^{th} line, the number of regions increases by n. Let's prove this fact: The n^{th} line intersects the other $n-1$ lines in exactly $n-1$ distinct points (because the lines are in general position). Tracing the n^{th} line as it passes through these $n-1$ points, we get to a new region every time we cross a point. So the number of regions the n^{th} line creates is one larger than the number of crossing points, thus n.

So if R_n is the number of regions we get with n lines in general position, then

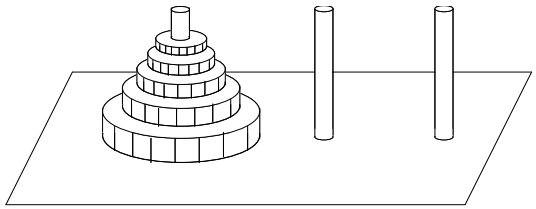
$$R_n = R_{n-1} + n$$

and we start with $R_1 = 2$. We can easily guess (relate this to the triangular numbers) that

$$R_n = \frac{n(n+1)}{2} + 1$$

which can be proved by induction.

3 Towers of Hanoi



The problem of the towers of Hanoi is based on a system of n discs of different sizes that fit over three fixed pegs. At the start we have all the discs arranged on peg one in size order smallest on top. The object is to transport this whole pile to peg three by a series of moves. A move consists of transporting the top disc from one peg to the top of a pile on either peg subject to the condition that no disc is ever placed on a smaller disc. Let a_n be the minimum number of moves required to complete the process. How big is a_n? It should be noted that the easiest way of proving that this process can be performed at all is to calculate a_n.

Eventually it will be necessary to move the bottom (largest) disc from peg one to peg three. The turn before this is done all the $(n-1)$ other discs are arranged on peg two. The minimum number of moves required to shift these to peg two is a_{n-1}. Then after the bottom disc is moved, the $(n-1)$ smaller discs must be reassembled on top of it. This again requires at least a_{n-1} moves. This proves the following recurrence:

$$a_n = 2a_{n-1} + 1$$

from which a_n can be determined.

This recurrence, along with the initial condition $a_1 = 1$, enables us to find a_n. We have $a_2 = 2 \cdot 1 + 1 = 3$, $a_3 = 2 \cdot 3 + 1 = 7$, $a_4 = 2 \cdot 7 + 1 = 15$. It appears that $a_n = 2^n - 1$. Let's confirm this by an induction proof. Our base case is $a_1 = 1$. For every $n > 1$, assume that the property holds up to $n-1$, and prove that it remains true for n.

$$a_n = 2a_{n-1} + 1 = 2(2^{n-1} - 1) + 1 = 2^n - 2 + 1 = 2^n - 1$$

Another way to obtaining this result is by repeatedly evaluating a_n using the recurrence:

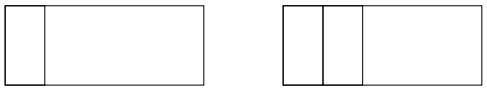
$$a_n = 2a_{n-1} + 1 = 2(2a_{n-2} + 1) + 1 = 2(2(2a_{n-3} + 1) + 1) + 1 = \ldots$$

Continuing this way, we get a geometric series (the following equality can be also proved by induction)

$$1 + 2 + 2^2 + \ldots + 2^{n-1} = \frac{2^n - 1}{2 - 1} = 2^n - 1$$

4 Fibonacci

Consider a staircase with n steps. We can walk up by either taking one or two steps at a time. In how many ways can we go up the stairs? Let a_n denote this number.Clearly $a_0 = 1$ and $a_1 = 1$. For $n > 1$, we must start by one of the two options shown below (top view):



In the first case we can continue in a_{n-1} ways, and in the second case we can continue in a_{n-2} ways. Since we start differently, by the addition principle:

$$a_n = a_{n-1} + a_{n-2}$$

Therefore, we obtain $a_2 = 1 + 1 = 2$, $a_3 = 1 + 2 = 3$, $a_4 = 3 + 2 = 5$, $a_5 = 5 + 3 = 8$, $a_6 = 8 + 5 = 13$, etc... The sequence a_n resembles the well known Fibonacci sequence.

n	0	1	2	3	4	5	6	7	8	9	10
F_n	0	1	1	2	3	5	8	13	21	34	55
a_n	1	1	2	3	5	8	13	21	34	55	89

Clearly, $a_n = F_{n+1}$. We will later obtain a formula for F_n.

5 Flags

A flag consists of n horizontal stripes, where each stripe can be any one of blue, red, and white, no two adjacent stripes having the same color. Under these conditions, the top stripe can be any of three colors, the second has two possibilities, the third has two, and so one (each stripe avoiding the color of the one above it). By the multiplication rule there are $3 \times 2^{n-1}$ possible flags.

Suppose now that, in order to avoid flying the flag upside down, the top and bottom stripes should be of different colors. How many such flags with n stripes are possible? Let a_n denote that number. If we let b_n be the number of flags with matching top and bottom, then $a_n + b_n = 3 \times 2^{n-1}$. Here's an interesting observation: there is a one-to-one correspondence between flags of n stripes with matching top and bottom, and flags of $n - 1$ stripes with different top and bottom. Here's why: Given a flag of n stripes with matching top and bottom, we can remove the bottom stripe to uniquely obtain a flag of $n - 1$

stripes with different top and bottom. Similarly, given a flag of $n-1$ stripes with different top and bottom, we can duplicate the top stripe at the bottom to uniquely obtain a flag of n stripes with matching top and bottom. By the one-to-one correspondence, $b_n = a_{n-1}$. Therefore,

$$a_n + a_{n-1} = 3 \times 2^{n-1}$$

Similarly,

$$a_{n-1} + a_{n-2} = 3 \times 2^{n-2}$$

By multiplying the second equation by 2 and subtracting it from the first we get:

$$a_n = a_{n-1} + 2a_{n-2}$$

Knowing that $a_1 = 0$ (why?) and $a_2 = 6$, we should be able to solve for a_n. It is now time to show how to solve this type of recurrence.

6 Solution for $a_n = Aa_{n-1} + Ba_{n-2}$

Given the recurrence

$$a_n = Aa_{n-1} + Ba_{n-2}$$

and two initial conditions, say a_1 and a_2, construct the following auxiliary equation:

$$x^2 = Ax + B$$

Let p and q be the solutions of the equation.

Case 1: $p \neq q$
Then $a_n = c_1 p^n + c_2 q^n$ satisfies the recurrence:

$$p^2 = Ap + B$$

$$p^n = Ap^{n-1} + Bp^{n-2}$$

$$c_1 p^n = c_1 Ap^{n-1} + c_1 Bp^{n-2}$$

Similarly,

$$c_2 q^n = c_2 Aq^{n-1} + c_2 Bq^{n-2}$$

Therefore, after adding both sides:

$$c_1 p^n + c_2 q^n = A(c_1 p^{n-1} + c_2 q^{n-1}) + B(c_1 p^{n-2} + c_2 q^{n-2})$$

$$a_n = Aa_{n-1} + Ba_{n-2}$$

Case 2: $p = q$

First note that if this is the case then $A = 2p$ and $B = -p^2$ (because $x^2 = Ax+B$ becomes equivalent to $(x-p)^2 = x^2 - 2px + p^2 = 0$). Then $a_n = c_1 p^n + c_2 n p^n$ satisfies the recurrence:

$$p^2 = Ap + B$$

$$p^n = Ap^{n-1} + Bp^{n-2}$$

$$c_1 p^n = c_1 A p^{n-1} + c_1 B p^{n-2}$$

Similarly,

$$c_2 n p^n = c_2 A n p^{n-1} + c_2 B n p^{n-2}$$

$$c_2 n p^n = c_2 A(n-1)p^{n-1} + c_2 B(n-2)p^{n-2} + \underbrace{(c_2 A p^{n-1} + 2c_2 B p^{n-2})}$$

The isolated above quantity is zero. Therefore, after adding both sides:

$$c_1 p^n + c_2 n p^n = A[c_1 p^{n-1} + c_2(n-1)p^{n-1}] + B[c_1 p^{n-2} + c_2(n-2)p^{n-2}]$$

$$a_n = A a_{n-1} + B a_{n-2}$$

In both cases, c_1 and c_2 can be obtained from the two initial conditions by equating the expressions for a_1 and a_2 to the values given. We end up with a solution for a_n that satisfies the recurrence and the two initial conditions. A similar technique can be used to solve recurrences of the form $a_n = A_1 a_{n-1} + A_2 a_{n-2} + \cdots + A_k a_{n-k}$ by forming the auxiliary equation $x^k = A_1 x^{k-1} + \ldots + A_{k-1} x + A_k$, but we will not consider it here.

7 Let's capture that rabbit!

The famous Fibonacci recurrence was associated by its inventor with a hypothetical rabbit problem. Rabbits obey the following law: every pair of rabbits, at least two months old, will produce one pair of rabbits every month. Starting with one pair of newly born rabbits, the n^{th} Fibonacci number, F_n, is the number of pairs at the beginning of the n^{th} month (by convention $F_0 = 0$.) Using the above technique, let's solve for F_n. Knowing that

$$F_n = F_{n-1} + F_{n-2}$$

we form the auxiliary equation:

$$x^2 = x + 1$$

which has two solutions $\phi = (1+\sqrt{5})/2$ and $1-\phi$. Therefore, our solution for F_n has the form:

$$c_1 \phi^n + c_2 (1-\phi)^n$$

Let's determine c_1 and c_2 from two initial conditions.

$$F_0 = c_1 \phi^0 + c_2 (1-\phi)^0 = c_1 + c_2 = 0$$

$$F_1 = c_1\phi^1 + c_2(1-\phi)^1 = c_1\phi + c_2(1-\phi) = 1$$

We obtain $c_1 = -c_2 = 1/(2\phi - 1) = 1/\sqrt{5}$. Therefore,

$$F_n = \frac{1}{\sqrt{5}}[\phi^n - (1-\phi)^n]$$

Observe that $(1-\phi)^n$ is approximately zero when n is large; therefore, F_n can be obtained for large n by finding the integer value closest to the following:

$$F_n \approx \frac{1}{\sqrt{5}}\phi^n$$

8 Let's fly the flag

We previously determined that

$$a_n = a_{n-1} + 2a_{n-2}$$

is the number of "good" flags. Let's solve for a_n. We form the auxiliary equation:

$$x^2 = x + 2$$

which has two solutions 2 and -1. Therefore, our solution for a_n has the form:

$$c_1 2^n + c_2(-1)^n$$

Let's determine c_1 and c_2 from two initial conditions.

$$a_1 = c_1 2^1 + c_2(-1)^1 = 2c_1 - c_2 = 0$$

$$a_2 = c_1 2^2 + c_2(-1)^2 = 4c_1 + c_2 = 6$$

We obtain $c_1 = 1$ and $c_2 = 2$. Therefore,

$$a_n = 2^n + 2(-1)^n$$

9 Another example

Let's solve the recurrence $a_n = 4a_{n-1} - 4a_{n-2}$ with $a_0 = 0$ and $a_1 = 2$. We form the auxiliary equation:

$$x^2 = 4x - 4$$

which has one solution 2 (technically two solutions that are equal). Therefore, our solution for a_n has the form:

$$c_1 2^n + c_2 n 2^n$$

Let's determine c_1 and c_2 from the two initial conditions.

$$a_0 = c_1 2^0 + c_2(0)2^0 = c_1 = 0$$

$$a_1 = c_1 2^1 + c_2(1)2^1 = 2c_1 + 2c_2 = 2$$

We obtain $c_1 = 0$ and $c_2 = 1$. Therefore,

$$a_n = n2^n$$

10 Generating functions

Going back to the flag problem, the first recurrence that we established was

$$a_n = -a_{n-1} + 3 \times 2^{n-1}$$

which does not have the form $a_n = Aa_{n-1} + Ba_{n-2}$. Generating functions represent another method for solving recurrences that we will not fully explore. The generating function of a sequence $a_0, a_1, a_2 \ldots$ is defined as

$$f(x) = \sum_{i=0}^{\infty} a_i x^i = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \ldots$$

For example, the generating function of the Fibonacci sequence is

$$x + x^2 + 2x^3 + 3x^4 + 5x^5 + \ldots$$

Sometimes, given a recurrence, it is possible to find the generating function of the sequence and then find a_n by reading off the coefficient of x^n. I will illustrate this for the flag problem.

$$f(x) = a_1 x + a_2 x^2 + a_3 x^3 + \ldots$$

$$= a_1 x + (3 \times 2 - a_1)x^2 + (3 \times 2^2 - a_2)x^3 + \ldots$$

$$= a_1 x + 3(2x^2 + 2^2 x^3 + \ldots) - (a_1 x^2 + a_2 x^3 + \ldots)$$

$$= 0 + 6x^2(1 + 2x + 2^2 x^2 + \ldots) - xf(x)$$

We can always assume that $|2x| < 1$ so the geometric series above will converge.

$$f(x) = 0 + 6x^2 \frac{1}{1-2x} - xf(x)$$

$$f(x) = \frac{6x^2}{(1+x)(1-2x)} = 2x^2\Big(\frac{2}{1-2x} + \frac{1}{1+x}\Big)$$

Expanding back into geometric series, we get:

$$f(x) = 4x^2(1 + 2x + 2^2 x^2 + \ldots) + 2x^2(1 - x + x^2 - \ldots)$$

Reading off the coefficient of x^n gives

$$a_n = 4(2)^{n-2} + 2(-1)^{n-2} = 2^n + 2(-1)^n$$

11 Sorting

Consider the task of sorting a list of n numbers from smallest to largest. One possible algorithm is the following: identify the smallest number in the list and swap it with the first number. This correctly places the smallest number at the beginning of the list. The process can be now repeated on the remaining numbers by considering them as a list of $n-1$ numbers.

To identify the smallest number in a list of n numbers one needs to make $n-1$ comparisons because each comparison can eliminate one number as being not the smallest. Therefore, the total number of comparisons is

$$(n-1)+(n-2)+\ldots+1=\frac{n(n-1)}{2}\approx\frac{n^2}{2}$$

This algorithm is called selection sort. We say that selection sort has an *order* n^2 running time, denoted $O(n^2)$.

A better way to sort is using the idea of merging sorted lists. Given a list of n numbers, split the list in two equal halves (or almost equal if n is not even). If we can manage to sort these two halves separately, then we can obtain a sorted list by merging them as follows. Compare the first (smallest) numbers in the lists, and take the smaller as the first number of a new list, crossing it out of its list. Repeat the process to find the second smallest, and so on until one of the two lists is exhausted. Append the other list to the end of the new list. The number of comparisons is at most $n-1$ because each time we make a comparison we cross out a number.

However, sorting the two halves of the original list using selection sort is not going to buy us any gain, because the number of comparisons would be approximately

$$\frac{(n/2)^2}{2}+\frac{(n/2)^2}{2}+n-1=n^2/4+n-1$$

This is still an $O(n^2)$ algorithm. But what happens if we recursively sort the two halves using the same method? For simplicity of illustration, let us assume that n is a power of 2, so we can always divide n by 2 until we get 1. Let t_n be the number of comparisons performed on a list of n numbers. Then

$$t_n=2t_{n/2}+n-1$$

Let $a_n=t_{2^n}$, then

$$a_n=2a_{n-1}+2^n-1$$

with $a_0=0$ (why?). Iterating this few times reveals that $a_n=2^n(n-1)+1$ which can then be proved by induction (try). This means

$$t_n=a_{\log_2 n}=2^{\log_2 n}(\log_2 n-1)+1=n(\log_2 n-1)+1$$

Another possibility is to sacrifice accuracy and seek an upper bound by saying that n comparisons are needed (as opposed to $n-1$) to merge two lists of size $n/2$. Then

$$t_n = 2t_{n/2} + n$$

$$a_n = 2a_{n-1} + 2^n$$

Similarly,

$$a_{n-1} = 2a_{n-2} + 2^{n-1}$$

By multiplying the second equation by 2 and subtracting it from the first we get:

$$a_n = 4a_{n-1} - 4a_{n-2}$$

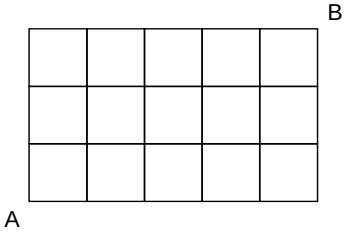
with $a_0 = 0$ and $a_1 = 2$ (why?). We already found the solution for this as $a_n = n2^n$. Therefore,

$$t_n = a_{\log_2 n} = \log_2 n 2^{\log_2 n} = n \log_2 n$$

This algorithm is called merge sort and it makes $O(n \log n)$ comparisons, an improvement over the $O(n^2)$ selection sort algorithm.

12 Traveling in Manhattan and the Catalan numbers

Imagine traveling in Manhattan. How many shortest paths can you take by walking on lines going from point A to point B shown below? Note that on a



shortest path you can only move to the right or up.

If we encode each path by a string of zeros (right moves) and ones (up moves), we will observe that every string has length 8 because every path must consist of 8 moves. Moreover, 5 of those moves must be to the right and 3 must be up. Therefore, our binary string of length 8 will contain exactly 5 zeros and 3 ones. It is not hard to see that there is a one-to-one correspondence between shortest paths and binary strings of 5 zeros and 3 ones. Every shortest path can uniquely be encoded by such a string, and conversely every such a string can uniquely identify a path. Therefore, the number of shortest paths is exactly the number of binary strings with 5 zeros and 3 ones. How many are there? Well, we have to choose 5 zeros from 8 bits (and the remaining 3 bits will be 1). Equivalently, we can choose 3 ones from 8 bits (and the remaining 5 bits will

be 0). Hopefully this should not be a surprise because you know by now that $\begin{pmatrix} n \\ k \end{pmatrix} = \begin{pmatrix} n \\ n-k \end{pmatrix}$. The number of shortest paths is

$$\begin{pmatrix} 8 \\ 5 \end{pmatrix} = \begin{pmatrix} 8 \\ 3 \end{pmatrix} = 56$$

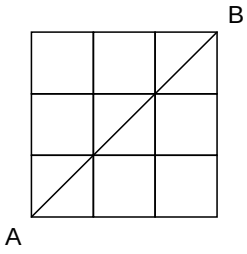
In general, if we have an $m \times n$ grid, the number of shortest paths will be

$$\begin{pmatrix} m+n \\ n \end{pmatrix} = \begin{pmatrix} m+n \\ m \end{pmatrix}$$

and if the grid is a square grid, i.e. $m = n$, then that number will be

$$\begin{pmatrix} 2n \\ n \end{pmatrix}$$

Suppose we now have a square $n \times n$ grid, and ask for the number C_n of shortest paths from point A to point B **which never go above the diagonal** AB. Let's call these the "good" paths. For the case of $n = 3$ shown below, there are 5 such paths; encoded as binary strings they are: 010101, 010011, 001101, 001011, 000111. Therefore, $C_3 = 5$.

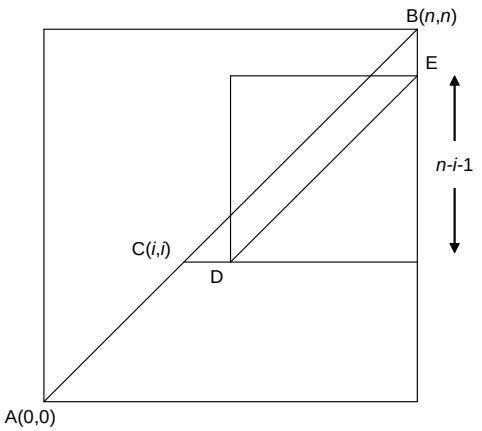


Any good path from A to B must touch the diagonal at some point prior to reaching B, even if this happens only at A. Let $C(i, i)$ be the **last** such point, where $0 \leq i < n$ (refer to the figure below). There are C_i good paths from A to C. The path must then proceed from C to D and eventually to E, but it must **never** go above the diagonal DE, since otherwise this would invalidate the choice of C. But D and E define a square grid of size $n - i - 1$, so there are C_{n-i-1} good paths from D to E. Therefore, by the multiplication principle, there are $C_i C_{n-i-1}$ good paths with (i, i) as the last contact with the diagonal AB.

Since i can take any value from 0 to $n - 1$, it follows from the addition principle that

$$C_n = C_0 C_{n-1} + C_1 C_{n-2} + C_2 C_{n-3} + \ldots + C_{n-1} C_0$$

with an initial condition $C_0 = 1$ because in a grid of size 0, where A and B coincide, there is only one way to go from A to B on a shortest path without going above the diagonal; simply stay there!



The sequence C_n defines the Catalan numbers:

$$1 \quad 1 \quad 2 \quad 5 \quad 14 \quad 42 \quad 139 \quad 429 \quad \ldots$$

but can we find a better expression for C_n? One could possibly solve the recurrence using the method of generating functions. But there is an easier way using a one-to-one correspondence. Observe that the number of good paths from A to B must be equal to the total number of paths $\begin{pmatrix} n \\ 2 \end{pmatrix}$ minus the number of shortest paths which do go above the diagonal AB (let's call these the bad paths).

Consider any bad path. There will be a first point on that path above the diagonal AB, say $R(i, i+1)$. If we replace the part of the path from A to R by its mirror image with respect to GF (refer to the figure below), then we get a shortest path from $H(-1, 1)$ to $B(n, n)$. Conversely, any shortest path from H to B must cross GF somewhere, and defines precisely one bad path from A to B (by mirroring).

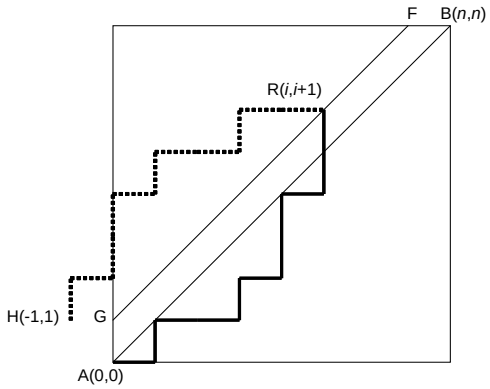
Therefore, the number of bad paths from A to B is equal to the number of shortest paths from H to B, which is

$$\begin{pmatrix} (n+1)+(n-1) \\ (n+1) \end{pmatrix} = \begin{pmatrix} 2n \\ n+1 \end{pmatrix}$$

Finally,

$$C_n = \begin{pmatrix} 2n \\ n \end{pmatrix} - \begin{pmatrix} 2n \\ n+1 \end{pmatrix} = \begin{pmatrix} 2n \\ n \end{pmatrix} - \frac{n}{n+1}\begin{pmatrix} 2n \\ n \end{pmatrix} = \frac{1}{n+1}\begin{pmatrix} 2n \\ n \end{pmatrix}$$

Catalan numbers appear in many situations. One immediate interpretation is that C_n is the number of binary strings of length $2n$ containing exactly n zeros and n ones, such that the number of 1s up to a point never exceeds the



number of 0s. If we replace 0 by '(' and 1 by ')', C_n gives us the number of valid parenthesized expressions of length $2n$ (with n pairs of parenthesis).

In how many ways can we carry out the sum $a_1 + a_2 + \ldots + a_n$? Let's take the example of $n = 4$, so we have $a_1 + a_2 + a_3 + a_4$. We can do the following:

$$a_1 + (a_2 + (a_3 + a_4))$$

$$a_1 + ((a_2 + a_3) + a_4)$$

$$(a_1 + a_2) + (a_3 + a_4)$$

$$(a_1 + (a_2 + a_3)) + a_4$$

$$((a_1 + a_2) + a_3) + a_4$$

If S_n is the number of ways of carrying out the sum $a_1 + a_2 + \ldots + a_n$, it is not hard to see that $S_1 = 1$ and $S_2 = 1$ and that

$$S_n = S_1 S_{n-1} + S_2 S_{n-2} + \ldots + S_{n-1} S_1$$

S_n behaves exactly like C_n with one index shift. Therefore, $S_n = C_{n-1}$.

13 Stirling numbers

Consider partitioning a set of size n into k subsets. This is similar to a problem that we have seen before, namely

$$x_1 + x_2 + \ldots + x_k = n$$

where $x_i \geq 0$ is an integer. But there is difference. In the equation above, the partitions are labeled. What is being partitioned is not. Now we have the opposite. The partitions are not labeled, and what is being partitioned is. For example, while 4 can be partitioned into two (labeled) parts as follows (5 ways):

|....

.|...

..|..

...|.

....|

the $S = \{1, 2, 3, 4\}$ can be partitioned into two subsets in the following ways: $\{1\} \cup \{2, 3, 4\}$, $\{2\} \cup \{1, 3, 4\}$, $\{3\} \cup \{1, 2, 4\}$, $\{4\} \cup \{1, 2, 3\}$, $\{1, 2\} \cup \{3, 4\}$, $\{1, 3\} \cup \{2, 4\}$, $\{1, 4\} \cup \{2, 3\}$.

Let $S(n, k)$ denote the number of ways we can partition a set of size n into k subsets. Then, $S(4, 2) = 7$. It is trivial that $S(n, 1) = S(n, n) = 1$. For other values of k, there is a nice recurrence for $S(n, k)$.

$$S(n, k) = S(n - 1, k - 1) + kS(n - 1, k)$$

where $1 < k < n$. Why is the above recurrence true? In any partition of S, the first element can appear in a subset by itself or can occur in a larger subset. If it appears by itself, then the remaining $n - 1$ elements are partitioned into $k - 1$ subsets, and there are $S(n - 1, k - 1)$ ways in which this can be done. On the other hand, if the first element occurs in a subset of size at least two, the remaining $n - 1$ elements are partitioned into k subsets - this can be done in $S(n-1, k)$ ways - and the first element is introduced into on of these k subsets - and this can be done in k ways. So by the addition and multiplication principles, we have $S(n, k) = S(n - 1, k - 1) + kS(n - 1, k)$.

It is not hard to prove the following interesting equality (define $S(m, 0) = 0$ for $m > 0$ and $S(0, 0) = 1$):

$$n^m = \sum_{k=0}^{n} S(m, k) \left(\begin{array}{c} n \\ k \end{array} \right) k!$$

We will do this by a combinatorial argument. Consider placing m labeled balls into n labeled bins. There are n^m ways of doing this (why?). Now imagine that we do it by first choosing a subset of bins of size k (can be done in $\left(\begin{array}{c} n \\ k \end{array} \right)$ ways), then partitioning the balls into k subsets (can be done in $S(m, k)$ ways), and finally assigning the k subsets to the k bins (can be done in $k!$ ways). By the multiplication principle, $S(m, k) \left(\begin{array}{c} n \\ k \end{array} \right) k!$ is the number of ways of placing m balls into **exactly** k of the n bins. By the addition principle, summing up from $k = 1$ to $k = n$ should give us the total n^m.

As a special case when $k = n$, $S(m, n)n!$ is the number of ways of placing m labeled balls into **exactly** n bins. It should be clear that this is also the number

of onto functions $f : X \to Y$ when $|X| = m$ and $|Y| = n$. It is possible to show that

$$S(m,n)n! = \sum_{k=0}^{n} (-1)^{n-k} \left(\begin{array}{c} n \\ k \end{array} \right) k^m$$

which is sort of an inverted form of the above formula for n^m. Since $S(m,n)n!$ is the number of onto functions from X to Y, it must be equal to n^m (the total number of functions) minus the number of functions that leave at least one element of Y unmapped. Observe that the number of functions that leave a given k elements of Y unmapped is $(n-k)^m$. Therefore, by the inclusion-exclusion principle:

$$S(m,n)n! = n^m - \underbrace{\left[\left(\begin{array}{c} n \\ 1 \end{array} \right) (n-1)^m - \left(\begin{array}{c} n \\ 2 \end{array} \right) (n-2)^m + \ldots + (-1)^{n-1} \left(\begin{array}{c} n \\ n \end{array} \right) (n-n)^m \right]}_{\text{number of functions that leave some elements of } Y \text{ unmapped}}$$

$$S(m,n)n! = \sum_{k=0}^{n} (-1)^k \left(\begin{array}{c} n \\ k \end{array} \right) (n-k)^m = \sum_{k=0}^{n} (-1)^{n-k} \left(\begin{array}{c} n \\ n-k \end{array} \right) k^m = \sum_{k=0}^{n} (-1)^{n-k} \left(\begin{array}{c} n \\ k \end{array} \right) k^m$$

Note that the above quantity is zero when $m < n$.

Example: consider the task of ranking n movies. You may put movies in the same rank and you may exclude movies from being ranked at all. In how many ways can we perform this task? Putting i movies into j ranks can be done by first partitioning the i movies into j sets (in $S(i,j)$ ways) and then ordering the j sets (in $j!$ ways), leading to $S(i,j)j!$ ways. Therefore, the total number of ways of ranking n movies is:

$$\sum_{i=0}^{n} \left(\begin{array}{c} n \\ i \end{array} \right) \sum_{j=0}^{i} S(i,j)j! = \sum_{i=0}^{n} \sum_{j=0}^{i} \sum_{k=0}^{j} \left(\begin{array}{c} n \\ i \end{array} \right) (-1)^{j-k} \left(\begin{array}{c} j \\ k \end{array} \right) k^i$$

14 Balls and bins

In this section we consider the different variations of placing n balls in k bins. The results are summarized in the table below:

Balls Labeled	Bins Labeled	Any Bin Empty	Number of Ways
No	Yes	No	$\left(\begin{array}{c} n-1 \\ n-k \end{array} \right)$
No	Yes	Yes	$\left(\begin{array}{c} n+k-1 \\ n \end{array} \right)$
Yes	No	No	$S(n,k)$
Yes	No	Yes	$S(n,1) + S(n,2) + \ldots + S(n,k)$
Yes	Yes	No	$S(n,k)k!$
Yes	Yes	Yes	k^n

The second entry refers to the problem

$$x_1 + x_2 + \ldots + x_k = n$$

where $x_i \geq 0$ are integers. The first entry refers to the problem

$$x_1 + x_2 + \ldots + x_k = n$$

where $x_i \geq 1$ are integers, which is equivalent to:

$$x_1 + x_2 + \ldots + x_k = n - k$$

where $x_i \geq 0$ are integers (why?).

The case where both balls and bins are unlabled is related to unordered integer partitions that we don't cover here. For instance, 5 can be partitioned in the following ways:

$1 + 1 + 1 + 1 + 1$
$1 + 1 + 1 + 2$
$1 + 2 + 2$
$1 + 1 + 3$
$2 + 3$
$1 + 4$
5