Counting, Coding, Sampling

Lecturer: Michel Goemans

In these notes we discuss techniques for counting, coding and sampling some classes of objects. We start by showing how to count several combinations and permutations, and extensions of it. We continue by presenting several classes of objects counted by the Catalan sequence $C_n = \frac{1}{n+1}\binom{2n}{n}$. This is an occasion to present several bijective techniques for counting, and simply beautiful mathematics. We then discuss some algorithmic application of (bijective) counting: some coding and random sampling algorithms.

1 Combinations and permutations

We will start by going over some of the most basic counting formulas: how to count the number of ways of coloring objects, and the number of ways of dividing a set of objects into subsets.

First, let's recall the formula

$$\binom{n}{k} = \frac{(n)(n-1)\dots(n-k+1)}{k!}$$

for choosing k things from n things. This is called the *binomial coefficient* and often pronounced n *choose* k.

How do you prove that this is the number of ways of choosing a set of k items from n items total? You can pick the first items in n ways, the second item in $n-1$ ways, the third item in $n-2$ ways, and so forth. This gives $n(n-1)(n-2)\dots(n-k+1)$ ways of picking k **ordered** sets out of n items. However, we're trying to count unordered sets. Each ordered set corresponds to $k!$ different unordered sets. For example, if we were trying to choose 2 numbers from $\{1, 2, \dots 7\}$, the unordered set $\{2, 5\}$ would correspond to the two ordered sets $[2, 5]$ and $[5, 2]$. We thus have to divide by $k!$ to obtain the number of unordered sets. This shows

$$\binom{n}{k} = \frac{(n)(n-1)\dots(n-k+1)}{k!}.$$

Note that we can multiply the numerator and denominator on the right-hand side by $(n-k)!$ to obtain

$$\binom{n}{k} = \frac{n!}{k!\,(n-k)!}.$$

This expression is symmetric in the number of chosen items, k and the number of unchosen items, $n-k$. Thus, we see that $\binom{n}{k} = \binom{n}{n-k}$. We should have expected this, since we could also single out a set of size k by choosing $n-k$ items to remove.

The next thing we want to cover is the *multinomial coefficient*. Suppose we have n items, and we want to color them with j different colors. Suppose furthermore that we want to color exactly k_i objects with color i. The number of ways of doing this is the multinomial coefficient

$$\binom{n}{k_1\, k_2\, \dots k_j} = \frac{n!}{k_1!k_2!k_3!\dots k_j!}$$

One can prove this by using the binomial coefficient formula for choosing k items from n and induction on the number of colors. We won't write this up in detail, but it follows from the formula

$$\binom{n}{k_1\, k_2\, \ldots\, k_j} = \binom{n}{k_j}\binom{n-k_j}{k_1\, k_2\, \ldots\, k_{j-1}}.$$

Now, let's discuss a different problem. Suppose you have n balls and j different colors. You can pick a color for each ball, and you want to know how many different sets of colored balls you can make. However, the balls are all identical so you only care about how many balls you have of each color.

For example, if you have three colors and two balls, the number of sets of colored balls you can make is six: RR, BB, GG, RB, RG, BG. (The order of the balls doesn't matter, so RG is the same as GR.) We will show that this is counted by the binomial coefficient $\binom{4}{2}$ by showing a bijection between ways of choosing two items out of four labeled items, and ways of choosing two balls of three colors.

A bijection is a one-to-one mapping from one set onto the other set, and if we have a bijection between two sets, we know that there are the same number of elements in each set.

One way to prove that a map is a bijection is to show that it is both an *injection* and a *surjection*. A function f mapping set A to set B is an *injection* if it doesn't map two different elements to the same element. In other words, if $f(x) = f(y)$, then $x = y$. A function f is a *surjection* if every element y of B has some element x of set A such that $f(x) = y$.

One way to prove that a map f is an injection is to show that it has an inverse. If there is a map g taking B to A such that $g(f(x)) = x$, then x must be an injection. If it is also true that $f(g(y)) = y$ for all $y \in B$, then it follows that f is also a surjection.

How does this bijection work? Let's illustrate it by an example. Suppose we have three colors, and seven balls. Let's sort them by color in some canonical order, say R, G, B.

$$R\ R\ R\ R\ G\ B\ B$$

Now, let's put dividers between the colors

$$R\ R\ R\ R \mid G \mid B\ B$$

Now, let's forget the colors.

We now have a sequence of $n + j - 1$ objects, n balls and $j - 1$ dividers. But the dividers tell us how to color the balls, so we can recover the information we started with, namely, how many balls were each color.

This shows that the mapping is a bijection, and thus that the number of ways of coloring n balls with j colors is $\binom{n+j-1}{j-1}$.

Now, let's count one more thing before moving on to Catalan numbers. Suppose we have 30 objects and want to divide them into three sets of ten objects each. How many ways are there of doing this? We had

$$\binom{30}{10\,10\,10} = \frac{30!}{10!10!10!}$$

ways to color 30 objects with 10 red objects, 10 blue objects, and 10 green objects. If we want to count the ways of dividing them into uncolored sets, we need to divide by $6 = 3!$, because there are 6 ways of coloring three uncolored sets. A similar argument shows that the number of ways of partitioning $n = \sum_i k_i n_i$ objects into k_i sets of size n_i with the n_i's distinct is:

$$\frac{1}{\prod_i k_i!} \cdot \frac{n!}{\prod_i (n_i!)^{k_i}}.$$

The last case, when you have n unlabeled objects and you want to count the number of ways of dividing them into unlabeled sets, is called the number of *partitions* of n. While there are lots of things you can say about partitions (including a nice generating function that counts them), there is no simple formula for the number of partitions.

2 Some Catalan families

We start by defining three classes of objects, and then discuss the relation between them.

A *plane tree* (a.k.a. ordered tree) is a rooted tree in which the order of the children matters. Let $\mathcal{T}_n$ be set of plane trees with n edges. The set $\mathcal{T}_3$ is represented in Figure 1. A *binary tree* is a plane tree in which vertices have either 0 or 2 children. Vertices with 2 children are called *nodes*, while vertices with 0 children are called *leaves*. Let $\mathcal{B}_n$ be the set of binary trees with n nodes. The set $\mathcal{B}_3$ is represented in Figure 2. A *Dyck path* is a *lattice path* (sequence of steps) made of steps +1 (up steps) and steps -1 (down steps) starting and ending at level 0 and remaining non-negative. Since the final level of a Dyck path is 0 the number of up steps and down steps are the same, and its length is even. Let $\mathcal{D}_n$ be the set of Dyck paths with $2n$ steps. The set $\mathcal{D}_3$ is represented in Figure 3.

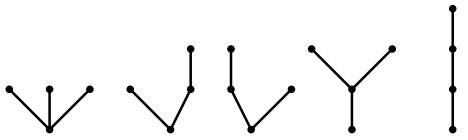

Figure 1: The set $\mathcal{T}_3$ of plane trees.

Figure 2: The set $\mathcal{B}_3$ of binary trees.

Figure 3: The set $\mathcal{D}_3$ of Dyck paths.

Observe that there is the same number of elements in $\mathcal{T}_3$, $\mathcal{B}_3$ and $\mathcal{D}_3$. This is no coincidence, as we will now prove that for all n, the sets $\mathcal{T}_n$, $\mathcal{B}_n$, $\mathcal{D}_n$ have the same number of elements. We now use the notation $|S|$ to denote the cardinality of a set S. We will now prove that

$$|\mathcal{T}_n| = |\mathcal{B}_n| = |\mathcal{D}_n| = \frac{1}{n+1}\binom{2n}{n}.$$

The number $\frac{1}{n+1}\binom{2n}{n}$ is the so-called nth *Catalan number*.

2.1 Counting Dyck paths

We first compute the number of Dyck paths. Let $\mathcal{P}_n^{(0)}$ be the set of paths of length $2n$ made of steps $+1$ steps and -1 steps starting and ending at level 0. Ending at level 0 is the same as having the same number of up steps and down steps, and any choice of order of such steps is allowed. Hence

$$\mathcal{P}_n^{(0)} = \binom{2n}{n}.$$

Now $\mathcal{D}_n$ is a subset of $\mathcal{P}_n^{(0)}$. It seems hard to find $|\mathcal{D}_n|$ because of the non-negativity constraint, but actually a trick will now allow us to compute the cardinality of the complement subset

$$\overline{\mathcal{D}}_n \equiv \mathcal{P}_n^{(0)} \setminus \mathcal{D}_n.$$

Indeed we claim that $|\overline{\mathcal{D}}_n| = \binom{2n}{n-1}$. To prove this claim we consider the set $\mathcal{P}_n^{(-2)}$ of paths of length $2n$ made of steps $+1$ steps and -1 steps starting at level 0 and ending at level -2. These paths have $n-1$ up steps and $n+1$ down steps, and any order of steps is possible, hence $|\mathcal{P}_n^{(-2)}| = \binom{2n}{n-1}$. So it suffices to give a bijection f between $\overline{\mathcal{D}}_n$ and $\mathcal{P}_n^{(-2)}$. This bijection is defined as follows: take a path D in $\overline{\mathcal{D}}_n$ consider the first time t it reaches level -1. The path $f(D)$ is obtained from D by flipping all the steps after time t with respect to the line $y = -1$. An example is shown in Figure 4. We let the reader check that f is a bijection between $\overline{\mathcal{D}}_n$ and $\mathcal{P}_n^{(-2)}$. (Do it! For example, explicitly give the inverse g of f, show that $g(f(D)) = D$ for all $D \in \overline{\mathcal{D}}_n$ and $f(g(P)) = P$ for all $P \in \mathcal{P}_n^{(-2)}$.) Since f is a bijection we have $|\overline{\mathcal{D}}_n| = |\mathcal{P}_n^{(-2)}| = \binom{2n}{n-1}$.

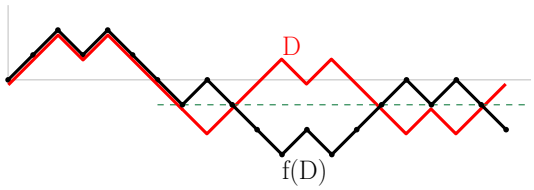

Figure 4: The bijection f: the path $D \in \overline{\mathcal{D}}$ in red, the path $f(D) \in \mathcal{P}_n^{(-2)}$ in black.

By the preceding, we have

$$|\mathcal{D}_n| = |\mathcal{P}_n^{(0)}| - |\overline{\mathcal{D}}_n| = \binom{2n}{n} - \binom{2n}{n-1} = \frac{(2n)!}{n!n!} - \frac{(2n)!}{(n+1)!(n-1)!}.$$

And by reducing to the same denominator we find

$$|\mathcal{D}_n| = \frac{(2n)!}{n+1!n!} = \frac{1}{n+1}\binom{2n}{n},$$

as wanted.

2.2 Bijection between plane trees, binary trees and Dyck paths

We now present bijections between the sets $\mathcal{T}_n$, $\mathcal{B}_n$ and $\mathcal{D}_n$.

We first present a bijection Φ between plane trees and Dyck paths as follows: given any tree T in $\mathcal{T}_n$, perform a *depth-first search* of the tree T (as illustrated in Figure 5) and define $\Phi(T)$ as the sequence of up and down steps performed during the search. A Dyke path is obtained from T because $\Phi(T)$ has n up steps and n down steps (one step in each direction for each edge of T), starts and end at level 0 and remains non-negative. A map Λ from Dyck paths to plane trees can also be easily defined in such a way that Φ and Λ are inverses of each other. Because Φ is a bijection between $\mathcal{T}_n$ and $\mathcal{D}_n$, we conclude

$$|\mathcal{T}_n| = |\mathcal{D}_n| = \frac{1}{n+1}\binom{2n}{n}.$$

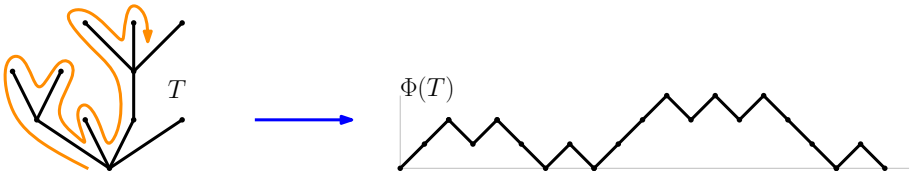

Figure 5: A plane tree T and the associated Dyck path $\Phi(T)$. The depth-first search of the tree T is represented graphically by a tour around the tree (drawn in orange).

We now present a bijection Ψ between binary trees and Dyck paths. Let B be a binary tree in $\mathcal{B}_n$. The tree B has n nodes. It can be shown that it has $n+1$ leaves (do it!). We can perform a depth-first search of the tree B and make a up step the first time we encounter each node and a down step each time we encounter a leaf. This makes a path with n up steps and $n+1$ down steps. The last step is a down step and we ignore it. We denote by $\Psi(B)$ the sequence of n up steps and n down steps obtained in this way. An example is represented in Figure 6. It is actually true that $\Psi(B)$ is always a Dyck path and that Ψ is a bijection between $\mathcal{B}_n$ and $\mathcal{D}_n$. We omit the proof of these facts. Since the sets $\mathcal{B}_n$ and $\mathcal{D}_n$ are in bijection we conclude

$$|\mathcal{B}_n| = |\mathcal{D}_n| = \frac{1}{n+1}\binom{2n}{n}.$$

3 Coding

Let S be a finite set of objects. A *coding function* for the set S is a function which associate a distinct binary sequence $f(s)$ to each element s in S. The binary sequence $f(S)$ is called code of S. Here are lower bounds for the length of codes.

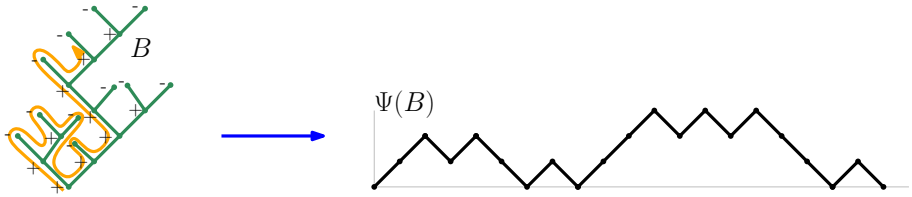

Figure 6: A binary tree B and the associated Dyck path $\Psi(B)$. The depth-first search of the tree B is represented graphically by a tour around the tree (drawn in orange).

Lemma 1. *If S contains N elements then at least one of the codes has length greater or equal to $\lfloor \log_2(N) \rfloor$. If one consider the uniform distribution for elements in S then the codes have length at least $\log_2(N) - 2$ in average.*

Exercise: Prove Lemma 1 for $N = 2^k - 1$.

Example 1: coding permutations. Let $\mathcal{S}_n$ be the set of permutations of $\{1, 2, \dots, n\}$. We now discuss a possible coding function f for the set $\mathcal{S}_n$. Recall that for any integer i, the binary representation of i has $\lceil \log_2(i+1) \rceil$ bits. Thus each number $i \in \{1, 2, \dots, n\}$ can be represented uniquely by binary sequences of length exactly $\lceil \log_2(n+1) \rceil$: it suffice to take their binary representations and add a few 0 in front if necessary to get this length. Let $\pi \in \mathcal{S}_n$ be a permutation seen as a sequence of distinct numbers $\pi = \pi_1 \pi_2 \dots \pi_n$. One can define $f(\pi)$ as the concatenation of the binary sequences (of length $\lceil \log_2(n) \rceil$) corresponding to each number $\pi_1 \pi_2 \dots \pi_n$. Then the length of the code $f(\pi)$ is $n \lceil \log_2(n+1) \rceil \sim n \log_2(n)$. We can recover the permutation from the code: if one has the code, it can cut it in subsequences of length $\log_2(n+1)$ each and then recover the numbers $\pi_1 \pi_2 \dots \pi_n$ making the permutation. Is it an efficient coding? Well according to Lemma 1 we cannot achieve codes shorter than $\log_2(n!) - 2$ in average. Moreover, $\log_2(n!) \sim n \log_2(n)$ (see Stirling's formula in the next example for a justification). Therefore our coding function f has length as short as possible asymptotically.

Example 2: coding Dyck paths. Consider the set $\mathcal{D}_n$ of Dyck path of length $2n$. There is an easy way of coding a Dyck path $D \in \mathcal{D}_n$ by a binary sequence of length $2n$. Simply encode down steps by "0" and up steps by "1" this give a binary sequence $f(D)$ of length $2n$. Could we hope for shorter codes? Certainly it would be possible to get a code of length $2n - 2$ because the first step is an up step and the last step is a down step, so these could be ignored. But could we do better than $2n + o(n)$ (where the "little o" notation means that the expression divided by n goes to zero as n goes to infinity)? We have seen that the set $\mathcal{D}_n$ has cardinality $N = \frac{2n!}{n!(n+1)!}$. Using the Stirling formula

$$n! \sim \sqrt{2n\pi} \left(\frac{n}{e}\right)^n$$

(where $a \sim b$ here means that $\frac{a}{b}$ tends to 1 as n tends to ∞) one gets $\log_2(n!) = n \log_2(n) - n \log_2(e) + o(n)$. Hence one can compute

$$\log_2(N) = \log_2(2n!) - \log_2(n!) - \log_2((n+1)!) = 2n + o(n).$$

Therefore, by Lemma 1 one cannot encode Dyck paths by codes of length less than $2n + o(n)$ on

average. So our naive coding is asymptotically optimal. Observe that this also gives a way of coding plane trees or binary trees optimally.

4 Random sampling

Let S be a finite set of objects. A (uniformly random) *sampling algorithm* for the set S is an algorithm which outputs an element in S uniformly at random from S. Here we suppose we dispose of a perfect random generator for integers. More precisely, let us suppose that one can generate a uniformly random integer in $\{1, 2, \ldots, n\}$ for any integer n.

Example 1: sampling permutations. How to sample a permutation in $\mathcal{S}_n$? Here is a solution written in pseudo-code.

Input an integer n.
• Initialize an array V of size n with value i at position i for $i = 1 \ldots n$.
• For $i = 1$ to n do
  Choose a integer r uniformly at random in $\{i, i+1, \ldots, n\}$.
  Swap the values at position i and r in V.
Output the array V.

The output of the above algorithm is an array of number which corresponds to a uniformly random permutation. Indeed, the first number of the array is chosen uniformly in $\{1, 2, \ldots, n\}$, the second number in the array is chosen uniformly randomly from the remaining numbers etc. Thus the above algorithm is indeed a sampling algorithm for the set $\mathcal{S}_n$.

If you make a small change to this program, and choose r uniformly in $\{1, 2, \ldots, n\}$ instead of $\{i, i+1, \ldots, n\}$, how non-uniform is the resulting random sampling algorithm? We will get back to that at the end of the notes.

Example 2: sampling Dyck paths. Sampling Dyck paths is a bit more difficult. We will need to first define an algorithm for sampling paths from another set. Let $\mathcal{P}_n^{(-1)}$ be the set of paths of length $2n+1$ with steps $+1$ and -1 starting at level 0 end ending at level -1. Hence a path $P \in \mathcal{P}_n^{(-1)}$ has steps "+1" and $n+1$ steps "-1" in any order. Here is a sampling algorithm for the set $\mathcal{P}_n^{(-1)}$.

Input an integer n.
• Initialize an array V of length $2n+1$ with value 1 in the first n entries and value -1 in the remaining $n+1$ entries.
• For $i = 1$ to $2n+1$ do
  Choose an integer r uniformly at random in $\{i, i+1, \ldots, 2n+1\}$.
  Swap the values at position i and r in V.
Output the array V.

Because the algorithm randomly permutes the steps $+1$ and -1, it indeed outputs a uniformly random path in $\mathcal{P}_n^{(-1)}$.

Now we will show how to obtain an Dyck path $D \in \mathcal{D}_n$ from a path $P \in \mathcal{P}_n^{(-1)}$. The trick we will use is known as the *cycle lemma*. Let $P \in \mathcal{P}_n^{(-1)}$. Let $\ell \leq 0$ be the lowest level of the path P, and let t be the first time the level ℓ is reached. This decomposes P as $P_1 P_2$ where P_1 is the path

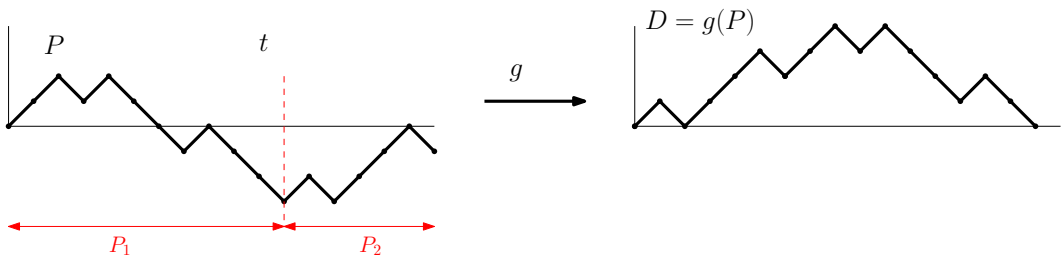

Figure 7: A path $P \in \mathcal{P}_n^{(-1)}$ and the resulting $g(P)$.

before time t and P_2 is the path after time t. Now consider the path P_2P_1. This path is ending with a -1 step. Then we define $g(P)$ as the path obtained from P_2P_1 by ignoring the last step. The mapping g is illustrated in Figure 7. The path $g(P)$ has n up steps and n down steps so it ends at level 0. In fact we claim that it is a Dyck path. Here is an even stronger claim.

Lemma 2. *For any path P is $P \in \mathcal{P}_n^{(-1)}$, the path $g(P)$ is a Dyck path. So g maps the set $\mathcal{P}_n^{(-1)}$ to the set $\mathcal{D}_n$. Moreover, any Dyck path in $\mathcal{D}_n$ is the image of exactly $2n+1$ paths in $\mathcal{P}_n^{(-1)}$.*

We will not prove this Lemma. However we argue that this gives a way of sampling Dyck paths. Indeed, by the above algorithm, one can sample a path P in $\mathcal{P}_n^{(-1)}$, and then apply the mapping g to obtain a Dyck path $g(P)$. Since every path in $\mathcal{P}_n^{(-1)}$ has the same probability of being sampled and every Dyck path in $\mathcal{D}_n$ has the same number of preimages, every Dyck path in $\mathcal{D}_n$ has the same probability of being sampled. We have thus found a sampling algorithm for Dyck paths. Observe that this also gives a way of sampling plane trees or binary trees.