4A Enumerative combinatorics I

4A1 (Experimental design) A metallurgic laboratory is studying various additives that could be mixed with iron to make steel, in various proportions. For each possible mixture, the laboratory is going to perform a test (e.g. measure the tensile strength). Each test takes one hour.

(a) Two additives A and B are considered, and we want to study all mixtures where each additive has some integer percentage between 0 and 10 (endpoints included). So a mixture can be specified by some pair of integers $(a, b) \in \{0, 1, \ldots, 10\}^2$. Some examples of mixtures to test are $(0, 0)$, $(5, 3)$, $(9, 9)$, $(10, 0)$. How many tests are needed?

(b) From the previous experiment, we have found that mixtures with about 5% of A and 8% of B look promising. We want more precision around this point, so we now study all mixtures where the percentage of A is some number with one decimal, within $\{4.0, 4.1, \ldots, 5.9, 6.0\}$, and the percentage of B is some number in $\{7.0, 7.1, \ldots, 8.9, 9.0\}$. How many tests are needed now?

(c) Your boss has read about four other exciting additives C, D, E and F. He instructs you to study all mixtures with the six additives, each having integer percentages ranging from 0 to 10. "With six additives, it's only about triple of your first experiment", he claims. How many tests are you being told to conduct, and how much time will they take? Assume that the laboratory runs 24 hours a day, every day of the week.

Solution.

(a) Let A be the set of possible proportions of additive A, and B the set of possible proportions of additive B. Each additive has 11 possible proportions, that is $|A| = |B| = 11$. (Keep in mind the fencepost error). Using the multiplication principle, we get:

$$|A \times B| = |A| \times |B| = 11 \cdot 11 = 121$$

Thus 121 tests are needed.

(b) There are 21 different proportions of additives for both A and B. Using the multiplication principle, we get:

$$|A \times B| = |A| \times |B| = 21 \cdot 21 = 441$$

Thus 441 tests are needed.

A handy way to **count numbers that are equally spaced**, without listing them one by one, is as follows. If the smallest number is x and the largest is y, and the spacing (step) from one number to the next is s, then we are dividing the distance $y-x$ into s-size steps, so the number of steps is $(y-x)/s$. Keeping in mind the fencepost error, the number of *points* is one more: $[(y-x)/s]+1$. (Try $x=200$, $y=300$, $s=25$ for easy visualization.)

Here, for the first additive, we have $x=4.0$, $y=6.0$, $s=0.1$, so the number of possible values is $[(6.0-4.0)/0.1]+1=[2/0.1]+1=20+1=21$.

(c) There are 6 different additives with 11 different proportions as in part a). Using the multiplication principle, we get:

$$|A\times B\times C\times D\times E\times F| = |A|\times|B|\times|C|\times|D|\times|E|\times|F| = 11^6 = 1771561$$

This is equal to 1 771 561 hours, which is about 73 815 days, or 202 years.

4A2 (Files and bytes) A *byte* (in a modern computer) is a small integer between 0 and 255 (inclusive; thus 256 possible values). A *file* is a sequence (tuple) of bytes, for example $(72, 101, 108, 108, 111)$ is a possible file of five bytes.[1] For every $n\in\mathbb{N}$, let F_n be the set of all possible n-byte files, and M_n the set of all possible files of *at most* n bytes. (Note that F_0 has a single element, namely the empty file ( ), which you get if you create a file but store nothing in it.)

(a) Express the set M_n in terms of $F_0,\ldots,F_n$ using the "big union" notation.[2]

(b) What are the cardinalities of F_3 and M_3? (Hint: In M_3 do not forget the empty file.) Of all files whose length is 3 or less, what percentage have exactly 3 bytes?

(c) What are the cardinalities of F_n and M_n? Simplify the latter into a short expression (consisting of less than ten arithmetical operations, only using the basic operations $+-\times/$ and power, no big sums using $\sum$) so that it is easy to calculate for any n. (Hint: Exercise 2B4.)

(d) Approximately how many files are there of length exactly 500 bytes? Give an approximation in scientific notation (like 1.234×10^5), with four significant digits (three decimals after the decimal point). Hint: Usual rules of exponents, and the exact facts that $2^8=256$ and $2^{10}=1024=1.024\times10^3$. After some manipulations, a regular handheld calculator can do the rest.

[1] Practically, a byte can represent many things, for example English letters using the ASCII code, but that is not relevant in this exercise.

[2] A big $\bigcup$ symbol where a variable like i runs from a lower limit to an upper limit.

(e) (OPTIONAL — Not required for scoring the problem, and no extra points.) A magical fairy tells you of `https://sagecell.sagemath.org/`, an online calculator that does exact arithmetic on large integers (and many other things). The symbol `^` denotes a power, for example `3^4` outputs `81`. Using this, find the exact integer result for (d), and verify that your first four digits were correct.

Solution.

(a)

$$M_n = F_0 \cup F_1 \cup \ldots \cup F_n = \bigcup_{i=0}^{n} F_i$$

(b) For F_3, let each of the bytes be denoted by B_1, B_2, B_3. Now using the multiplication principle:

$$|B_1 \times B_2 \times B_3| = |B_1| \times |B_2| \times |B_3| = 256^3 = 16\,777\,216.$$

For M_3, we have to include also all shorter files, so

$$|M_3| = |F_0| + |F_1| + |F_2| + |F_3| = 1 + 256 + 256^2 + 256^3 = 16\,843\,009.$$

The percentage of files that have exactly three bytes out of all the files of length 3 or less is

$$\frac{F_3}{M_3} = \frac{16\,777\,216}{16\,843\,009} \approx 99.6\%.$$

(c) Let us denote the set of all possible bytes as $B = \{0, 1, \ldots, 255\}$. It contains $|B| = 256$ elements. Now F_n is the n-fold Cartesian product of B with itself,

$$F_n = \underbrace{B \times B \times \cdots \times B}_{n \text{ of them}} = B^n,$$

so its cardinality is (by the rule of product)

$$|F_n| = \underbrace{256 \times \ldots \times 256}_{n \text{ of them}} = 256^n.$$

For M_n we must also include all smaller files, just like before. Thus

$$\begin{aligned} |M_n| &= |F_0| + |F_1| + |F_2| + \ldots + |F_n| \\ &= 1 + 256 + 256^2 + \ldots + 256^n = \sum_{i=0}^{n} 256^i. \end{aligned}$$

This is a *finite geometric series*[3], that is, a sum whose terms grow by a constant factor (here 256), or in other words, a sum of consecutive powers of some number. The formula for the a finite geometric series is

$$\sum_{i=0}^{n} c^i = \frac{c^{n+1} - 1}{c - 1},$$

which can be proved e.g. by induction (in exercise 2B4 it was done in cases $c = 2$, $c = 3$ and $c = 10$, but the general proof is similar). Here we apply it with $c = 256$, so

$$|M_n| = \sum_{i=0}^{n} 256^i = \frac{256^{n+1} - 1}{255}.$$

The formula consists of four arithmetic operations, less than ten, as required. Let's double-check that the formula works in some small cases:

$$\begin{aligned}
M_0 &= (256^1 - 1)/255 = 1 && \text{clearly OK} \\
M_1 &= (256^2 - 1)/255 = 257 && \text{clearly OK, equals } 1 + 256 \\
M_3 &= (256^4 - 1)/255 = 16\,843\,009 && \text{OK, matches (b)}
\end{aligned}$$

Whenever you have deduced a general formula, it is a good idea to check (*even if you are not explicitly told to*) that it actually gives the values it should, in some small cases. Although this does not guarantee that the formula is correct, many simple mistakes are found this way.

(d)

$$\begin{aligned}
|F_{500}| &= 256^{500} = (2^8)^{500} = 2^{4000} = (2^{10})^{400} = 1024^{400} = (1.024 \times 10^3)^{400} \\
&= 1.024^{400} \times 10^{1200} \approx 13182.04 \times 10^{1200} \approx 1.318 \times 10^{1204}.
\end{aligned}$$

The only step where a calculator was needed was calculating 1.024^{400}, a high power of a number almost equal to one, and the result is in range for basically any calculator.

Some numbers are big (a quote from Captain Obvious). Scientific notation is handy, and you should know how to work with it, even when the numbers don't fit into your calculator. Also, if you don't *have* a calculator at hand, you can still "easily" see that $|F_{500}|$ is at least bigger than $1000^{400} = 10^{1200}$. It is good to remember $2^{10} \approx 1000$ as an approximation.

[3]Possibly already familiar from high school math.

4A3 (File compression) Continuing from the previous problem. A *file encoding scheme* is an injection f that maps each file x (in some specified set) into an encoded file $y = f(x)$ (possibly different from x). Typical uses are compression, encryption and error correction. If we have an encoded file y and we know the encoding scheme, the idea is that we could (by some method) find the original file x which had been encoded as $f(x) = y$. In compression, the idea is that at least *some* files (perhaps those that are likely to occur in practice) are encoded into shorter ones, saving storage space.

(a) Why is it essential that a file encoding scheme is an injection?

(b) Consider the following (very simplistic) scheme for files in M_{254}: If a file contains n bytes, all equal to a, the encoding is just two bytes (n, a). Otherwise, the encoding is $n + 1$ bytes, first the value 255 and then the original n bytes in order. Prove that this is an injection.

(c) Suppose that every file currently stored on our computer system has exactly 254 bytes, and 30% of them have the format "all bytes are equal". If we compress all our files using the scheme from (b), what is the *average length* of an encoded file?

(d) Of all files of *at most* n bytes, what fraction have length *smaller* than n? That is, find the fraction $|M_{n-1}|$ / $|M_n|$. First express it exactly (using the results from Problem 2), aiming for a simple expression. Then give an approximate value (one real number) when n is large.

(e) Professor Quentin U. Ackerman has devised an ingenious scheme that compresses *any* file of exactly 1000 bytes into a compressed file of some length strictly less than 1000 bytes. Also, he claims, there is a method of always recovering the original file from the compressed file. The scheme is very complicated and you have trouble understanding all its details. Is there a simple argument that the scheme cannot work?

Solution.

(a) We need to be able to "go backwards"to find the original file that has been encoded. If the encoding scheme was not an injection, that is if we had $f(x_1) = f(x_2)$ such that $x_1 \neq x_2$ then it would be impossible to determine whether the original file was x_1 or x_2.

(b) For injective functions it applies that: $f(x_1) = f(x_2) \Rightarrow x_1 = x_2$

Proof by cases:

Case 1: comparing files with only one type of byte We know that if we have (n,a) = (m,b) it must be so that n=m and a=b

Case 2: comparing files with different types of bytes The transformed version of the files is just the files themselves with the byte 255 at the beginning added to it. Thus if two transformed files are the same, their original file form must also be the same.

Case 3: comparing files with one type of byte to different types of bytes We still need to check if any of the transformed case 1) files can be equal to the transformed case 2) files. We quickly notice this is not possible, since in case 1) the transformed file is of length 2. The only way to achieve this with case 2 is if the original file was of length 1. However, all files of length 1 fall into the category of case 1) (files with bytes of only one kind in them). Thus it is not possible to have a case 2) file of length 2.

Conclusion: None of the transformed files will be equal unless the original files are the same. Thus the encoding scheme is an injection.

(c) After the compression, 70% of the files are of length 255 and 30% of the files are of length 2. The average length of a file is now

$$0.7 \cdot 255 + 0.3 \cdot 2 \approx 179.1.$$

(d) Using Problem 2:

$$M_n = (256^{n+1} - 1)/255$$
$$M_{n-1} = (256^n - 1)/255$$

Dividing and simplifying we get

$$\frac{M_{n-1}}{M_n} = \frac{256^n - 1}{256^{n+1} - 1} = \frac{1 - \frac{1}{256^n}}{256 - \frac{1}{256^n}} \approx \frac{\mathbf{1}}{\mathbf{256}},$$

when n is large. The second equality was obtained by dividing by 256^n on both sides of the division line, and the final approximation comes from the $1/256^n$ being tiny.

(e) Yes, there is such an argument. The professor claims to have an injection from the *set* of all 1000-byte files to the *set* of files that have 999 bytes or less, that is, an injection from F_{1000} to M_{999}.

But we know that

$$|F_{1000}| = 256^{1000}$$
$$|M_{999}| = \frac{256^{999+1} - 1}{255} < \frac{256^{1000}}{255}.$$

Clearly $|M_{999}|$ is smaller than $|F_{1000}|$, in fact approximately by a factor of 255.

There cannot be an injection from a finite set to smaller finite set. Certainly the professor may have devised a *function* that maps every file in F_{1000} to

something in M_{999}. But it cannot be an injection. Some (in fact many) files necessarily map to the same compressed files, and there is no way to recover the original files. The professor's scheme cannot work.

After this it may seem that file compression is impossible. Yes and no. It is not possible to map *all* n-byte files *injectively* to smaller files. In practice there are two ways out:

- Allow *some* files to become *longer*, like we did in (b)–(c). Then we can have an injection. Hopefully we devise a scheme that makes smaller *those* files that we have on our computer system in practice, at least on average.
- Allow the mapping to be non-injective. This is called *lossy compression*. An example is compression of pixel images (e.g. digital photographs) into JPG files. They can be compressed quite a lot, but the mapping is not injective. The original exact files cannot be recovered from the compressed ones.

4A4 (Binomial coefficient manipulation) Let a natural number n be given. We want to compute the whole nth row in Pascal's triangle, that is, the numbers $b_0, b_1, b_2, \ldots, b_n$, where $b_k = \binom{n}{k}$. Of course we know that $b_0 = b_n = 1$.

(a) Prove that for every $k \in \{0, 1, 2, \ldots, n-1\}$, if we know b_k, we can calculate b_{k+1} exactly, using a small constant number of simple arithmetic operations (in fact, the four operations $+ - \times /$) and the numbers n and k, without any factorials. Hint: Study the ratio b_{k+1}/b_k and simplify. Can you perform the calculation completely in integer arithmetic (so that none of your intermediate results is a noninteger)?

(b) Apply your scheme on rows $n = 3$ and $n = 6$, beginning in each case from b_0 and calculating successively the numbers up to b_n. You should be able to do this without a calculator. On each row, verify that you reached $b_n = 1$ (good for error-checking, if we have any doubt of our calculations). On each row, add up the numbers and verify that you got 2^n (here you can use a calculator).

Solution.

(a) We know that $b_k = \binom{n}{k} = \frac{n!}{k!(n-k)!}$, and thus $b_{k+1} = \frac{n!}{(k+1)!(n-k-1)!}$. Now

$$\frac{b_{k+1}}{b_k} = \frac{n!}{(k+1)!(n-k-1)!} \cdot \frac{k!(n-k)!}{n!} = \frac{k!(n-k)!}{(k+1)!(n-k-1)!}$$

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

(b) We know that $\frac{b_{k+1}}{b_k} = \frac{n-k}{k+1}$, so now we have $b_{k+1} = b_k \times \frac{n-k}{k+1}$. Thus:

Row $n = 3$:

$$b_0 = 1$$
$$b_1 = 1 \times (3/1) = 3$$
$$b_2 = 3 \times (2/2) = 3$$
$$b_3 = 3 \times (1/3) = 1.$$

Check: 1+3+3+1=8 and $2^3 = 8$.

Row $n = 6$:

$$\begin{aligned} b_0 &= 1 \\ b_1 &= 1 \times (6/1) = 6 \\ b_2 &= 6 \times (5/2) = 15 \\ b_3 &= 15 \times (4/3) = 20 \\ b_4 &= 20 \times (3/4) = 15 \\ b_5 &= 15 \times (2/5) = 6 \\ b_6 &= 6 \times (1/6) = 6. \end{aligned}$$

Check: 1+6+15+20+15+6+1=64 and $2^6 = 64$.

In every step, b_k is an integer, and we can first multiply by the numerator (which is an integer, so the result is integer), then divide by the denominator (also an integer, and the result is the binomial coefficent, so it is an integer always). No rational or real number arithmetic needed.

4A5 (Bounds and approximations) The *falling product* (also known by many other names) is the product of k consecutive decreasing numbers beginning from n,

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

(a) Prove the following *bounds* on the falling product (when $1 \le k \le n$):

$$(n-k+1)^k \le n^{\underline{k}} \le n^k.$$

(Hint: A simple arithmetical proof is sufficient, no need to use e.g. induction.)

(b) In (a) we have *bracketed* (or *sandwiched*) the quantity $n^{\underline{k}}$ between two bounds, and we know for sure that the two inequalities are true. The next interesting thing is how good (narrow) our bracketing is. Compute the two bounds, and the ratio of the upper bound to the lower bound, when $n = 100$ and $k = 3$. Do the same for $n = 100, k = 10$ and $n = 100, k = 50$.

(c) Using the bounds from (a), write lower and upper bounds for the binomial coefficients $\binom{n}{2}$ and $\binom{n}{3}$. The bounds must be in a simple form not involving factorials.

(d) Given a population of $n = 1\ 000\ 000$ people, we would like to count all subsets of three people. Calculate lower and upper bounds using the formula from (c). (You can use a calculator.) Let us call the bounds L and U. Calculate the relative difference $(U - L)/L$ and express it as a percentage. Do you think U is a good approximation for the exact value (which we did not calculate)?

Solution.

(a) First inequality:

$$(n-k+1) = (n-(k-1)) \leq (n-i) \forall i = 0, 1, ..., k-1$$

Now

$$(n-k+1)^k \leq \prod_{i=0}^{k-1} (n-i) = n^{\underline{k}}$$

Second inequality:

$$n - i \leq n, \forall i = 0, 1, ..., k-1$$

Now

$$n^{\underline{k}} = \prod_{i=0}^{k-1} (n-i) \leq n^k$$

(b) n = 100 and k = 3:

$$(n-k+1)^k = 98^3 = 941192$$

$$100^3 = 1000000$$

Their ratio is 0.94, so the bounds are reasonably close to each other.

n = 100 and k = 10:

$$(n-k+1)^k = 91^{10} \approx 3.894 \times 10^{19}$$

$$100^{10} = 1.0 \times 10^{20}$$

The ratio is 0.39.

Now the bounds are further apart, but they still give us a reasonable idea of the *magnitude* of the number.

n = 100 and k = 50:

$$(n-k+1)^k = 51^{50} \approx 2.391 \times 10^{85}$$

$$100^{50} = 1.0 \times 10^{100}$$

Their ratio is 2.391×10^{-15}.

The bounds are very far apart, and (with these simple methods) we know only that the true value is somewhere between them. But still we get at least some idea that the number must be pretty big. Even this precision might be enough in some applications!

(c)

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

k = 2:

$2! = 2$, so $\binom{n}{2} = n^{\underline{2}}/2$

We now have the bounds

$$\frac{(n-2+1)^2}{2} \leq \frac{n^{\underline{2}}}{2} \leq \frac{n^2}{2}$$
$$\Rightarrow \frac{n^2-2n+1}{2} \leq \binom{n}{2} \leq \frac{n^2}{2}$$

k = 3:

$3! = 3 \cdot 2 = 6$, so $\binom{n}{3} = \frac{n^{\underline{3}}}{6}$

We now have the bounds

$$\frac{(n-3+1)^3}{6} \leq \frac{n^{\underline{3}}}{6} \leq \frac{n^3}{6}$$
$$\frac{(n-2)^3}{6} \leq \binom{n}{3} \leq \frac{n^3}{6}$$

(d) n = 1 000 000, k = 3

Lower bound:

$$\frac{(1000000-2)^3}{6} \approx 1.667 \times 10^{17}$$

Upper bound:

$$\frac{(1000000)^3}{6} \approx 1.667 \times 10^{17}$$

The relative difference is $0.000006 = 0.006\%$. Thus U is a good approximation of the exact value.

Of course here we *could* also directly compute the exact value, even all its digits if we want. It is

$$\frac{1\,000\,000 \times 999\,999 \times 999\,998}{3} = 166\,666\,166\,667\,000\,000 \approx 1.667 \times 10^{17}.$$

The point is that often we do not need the exact value, but e.g. 4 significant digits is quite enough. In such cases, approximations can be quite convenient. It is also useful to know *how* good an approximation is. *Bounds* are one possible way of knowing it *even if we have not computed the exact value.*

Even if we *do* compute an exact value (perhaps because we can, or perhaps because we really need it), it is useful to have a *sense of proportion* to double-check that the exact value is at least *roughly correct*. If it came out of a long computation, quite often it is in fact wrong, perhaps by several orders of magnitude. Then it is a good sanity check to compare to a rough and easy estimate (such as U here).

4A6 (Nonattacking rooks) A standard chessboard is divided into 8 horizontal rows and 8 vertical columns, for a total of 64 squares. We place rooks (a particular chess piece) on some squares, with the condition that they are all *nonattacking*, meaning that no two rooks may be on the same row, and no two on the same column. The color of the rooks is irrelevant in this exercise.

(a) In how many different ways can you place two nonattacking, *numbered* rooks (rook 1 and rook 2)? Solve this using the multiplication principle. First consider how many possibilities you have for placing the first rook. Having placed it, consider how many valid possibilities you have for the second rook (so it is not on the same row nor on the same column).

(b) In how many different ways can you place two nonattacking, *identical* rooks? Use the result from (a), and make a simple manipulation to account for the fact that there is more than one way to reach the same final board position (with rooks on some two squares).

(c) Now consider a different way of counting. To place two nonattacking rooks, observe that they must be on two different rows, and two different columns. So, begin by counting the ways of choosing a set of two rows, out of the 8. Then count the ways of choosing a set of two columns, and multiply. Did you get the same result as in (b), or is there something more you need to do?

(d) (OPTIONAL — Not required for scoring the problem, and no extra points.) Count the ways of placing three identical nonattacking rooks, using a method similar to (a)–(b), i.e. first place three numbered rooks in order, then adjust the count to take into account that the same final result can be reached in more than one way.

(e) (OPTIONAL — Not required for scoring the problem, and no extra points.) Count the ways of placing three identical nonattacking rooks, using a method similar to (c), i.e. first choose three rows and three columns. Observe that the three rows intersect the three columns in 9 squares. Consider in how many ways you can put the three rooks into these nine squares. Combine your results to arrive at the final number, which should equal what you got in (d).

The points of this problem are: getting used to binomial coefficients; understanding the relation between *ordered choice* and *unordered choice*; and the general idea that there may be more than one method to calculate the same result, but all correct methods must lead to the same result.

Solution.

(a) The first rook can be placed on 64 different squares. Regardless of which squares we pick, the squares left that are not on the same row or column of r1 will be 49 in total. Thus by the addition principle we get $64 \cdot 49 = 3136$

(b) Since in the case of identical rooks we have to take into account the situations in which the rooks can be switched up, so we can simply divide the first part by 2. Thus we get $3136/2 = 1568$

(c) We can choose 2 out of 8 using the binomial coefficient. $\binom{8}{2} = 28$ for both the columns and rows. When multiplied $(28 \cdot 28)$, we get 784. However, we need to account for the fact that when choosing 2 rows and 2 columns, there are 4 squares that intersect. The two rooks can be arranged in 2 ways in those 4 squares. (This can be thought of as a 2x2 chess board). Thus we need to multiply $\binom{8}{2} \cdot \binom{8}{2} \cdot 2 = 784 \cdot 2 = 1568$

(d) The first rook can be placed in 64 ways. The next rook can be placed in 49 ways. The third rook can be placed in 36 ways. Thus we get $64 \cdot 49 \cdot 36 = 112896$ Since the rooks are identical, we need to take into account the situations in which the rooks can be switched up, so we can simply divide by 2 and then divide by 3 (for the third rook). We thus get $64 \cdot 49 \cdot 36 \cdot \frac{1}{2} \cdot \frac{1}{3} = 18816$

(e) Like in part c: $\binom{8}{3} = 56$ so $56 \cdot 56 = 3136$, but we need to again take into account the overlaps of the 3 columns and 3 rows. We get 9 squares in which they overlap (can be thought of as a 3x3 board). 3 identical rooks can be placed in the squares in 6 ways. Thus we have $3136 \cdot 6 = 18816$