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. In a mixture we can have several additives (and the rest is iron). 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 in total?

(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.

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.)

[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.

(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 $2^{10} = 1024 = 1.024 \times 10^3$. After some manipulations, a regular handheld calculator can do the rest.

(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.

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?

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).

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)?

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 idea of this problem is to get used to binomial coefficients; to understand the relation between *ordered choice* and *unordered choice*; and that there may be more than one method to calculate the same result, but all correct methods must lead to the same result.