POLYNOMIALS AND COMBINATORICS IN COMPETITIVE PROGRAMMING

LUIS FERRONI

ABSTRACT. In these notes we explore some combinatorial facts that can be expressed using certain polynomials or infinite series. We also give a brief overview of Burnside's Lemma and provide a very gentle introduction to generating functions. This was written as part of the lectures given by the author in the Argentinian Training Camp in Competitive Programming held virtually during July 2021.

1. BOXES WITH CAPACITIES

Problem 1.1. Assume that we have r balls and n boxes labeled from 1 to n, where $1 \leq n \leq 2000$ and $1 \leq r \leq 10^6$. Assume that for each $i = 1, \ldots, n$, the i-th box can contain at most a_i balls, where $1 \leq a_i \leq 1000$. We have to answer in how many distinct ways we can put all the r balls into the n boxes.

This statement *seems* to be a classic dynamic-programming problem but it is not. Playing a bit with it will reveal that any of the standard approachs using dynamic-programming seems not to be fast enough.

2. A REVIEW OF POLYNOMIALS

We all know how to operate with polynomials using the usual addition, multiplication and even division. Consider two polynomials A and B, say

$$A(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$$
$$B(x) = b_0 + b_1 x + b_2 x^2 + \cdots + b_m x^m.$$

If $a_n \neq 0$ we say that A has *degree* n, and we write $\deg A = n$. To denote the coefficient of x^k in A we will use the notation $[x^k]A(x)$. For example:

$$[x^3]A(x) = a_3,$$
$$[x^7]B(x) = b_7,$$
$$[x^1](1+x)^2 = 2,$$
$$[x^5](7+x)^3 = 0.$$

It is clear that if we wanted to compute the coefficients of $A(x) + B(x)$, it just amounts to find $a_i + b_i$ for each $i = 0, \ldots, \max(m, n)$. In particular, we see that computing sums of polynomials can be done in linear time. Things are somewhat trickier if we want to *multiply* polynomials. First, notice that:

$$[x^k](A(x)B(x)) = \sum_{j=0}^{k} [x^j]A(x) \cdot [x^{k-j}]B(x) = \sum_{j=0}^{k} a_j b_{k-j}.$$

We see very quickly that to compute all the coefficients of $A(x)B(x)$ naïvely, we would have to compute all the products $a_i b_j$ at some point, so that the whole time complexity would be $\Omega(mn)$, i.e. quadratic.

Things get even uglier if we wanted to *divide* two polynomials by brute force. Fortunately for us, there exists algorithms that provide us a much better time complexity.

Theorem 2.1 (Fast Fourier Transform)**.**

(a) There exists a way of computing all the coefficients of $A(x)B(x)$ using $O((m+n)\log(m+n))$ multiplications.

(b) Assume that we want to divide A by B. In other words, we want to know two polynomials Q and R such that $\deg R < \deg B$*, satisfying*

$$A(x) = Q(x)B(x) + R(x).$$

It is possible to compute Q and R in $O((m+n)\log(m+n)^2)$ multiplications.

Both of these two algorithms were addressed in last year's Training Camp, so we encourage you to take a look at the notes that were provided therein.

Interpolating a polynomial means to deduce the values of all of its coefficients by knowing the values of the polynomial when evaluated at several distinct values.

Consider the case of a linear polynomial (i.e., a polynomial of degree 1). It is clear that the graph of the polynomial would be a line, so that knowing two points that are part of the line allows us to recover all the coefficients of our polynomial. This property extends to an arbitrary degree.

Theorem 2.2 (Lagrange Interpolation)**.** *Assume that A is a polynomial of degree n, and assume that we know the value of A when evaluated at $\{x_1, \ldots, x_{n+1}\}$. Let us say that $A(x_i) = y_i$. Then A is completely determined as the following polynomial*

$$A(x) = \sum_{i=1}^{n+1} y_i \prod_{\substack{j=1 \\ j\neq i}}^{n} \frac{x - x_j}{x_i - x_j}.$$

Notice that if A has degree n, we need to know its value in *at least* $n+1$ distinct points to be able to fully determine the coefficients of A.

It is remarkable that using the Fast Fourier Transform it is possible to calculate the interpolating polynomial in $O(n\log(n)^2)$ multiplications. However, it depends on having a routine for *multi-evaluation*, i.e. calculating the values of a known polynomial of degree n into m distinct points (this can be achieved in $O((m+n)\log(m+n)^2)$ again using FFT, we refer to last year's TC talk for more details).

In many cases, just having the intuition that there is a polynomial making things happen in the background is all that one needs to get a solution that works.

Example 2.3. Assume that we are given $1 \leq n \leq 10^{18}$ and $1 \leq k \leq 1000$, and we are asked to compute

$$1^k + 2^k + \cdots + n^k \pmod{10^9+7}.$$

Doing it naïvely will not help since n can be too large. However, observe that

$$1 + 2 + \cdots + n = \frac{n(n+1)}{2} \qquad \text{(a polynomial of degree 2)},$$

$$1^2 + 2^2 + \cdots + n^2 = \frac{n(n+1)(2n+1)}{6} \qquad \text{(a polynomial of degree 3)},$$

$$1^3 + 2^3 + \cdots + n^3 = \frac{n^2(n+1)^2}{4} \qquad \text{(a polynomial of degree 4).}$$

So that it makes perfect sense to intuit that $1^k + \cdots + n^k$ is a polynomial of degree $k+1$ in n. Hence, if we compute the sums 1^k, $1^k + 2^k$, ..., $1^k + 2^k + \cdots + (k+1)^k$, then we would know the values of this polynomial into $k+1$ distinct points. By using Lagrange interpolation then we know who the polynomial is. After that, it remains only to evaluate it in the n given in the statement.

3. Solution to the boxes with capacities problem

Assume that we had only 3 boxes, with capacities 4, 1 and 3 respectively and we want to count in how many distinct ways we can put 5 balls into the boxes. Consider the following polynomial

$$A(x) = (1 + x + x^2 + x^3 + x^4) \cdot (1 + x) \cdot (1 + x + x^2 + x^3).$$

Let us invest a second into understanding what $[x^5]A(x)$ will be. To contribute to the coefficient of x^5, we have to choose some exponent i of x in the first factor, some exponent j of x in the second factor and some exponent k in the third factor such that $i + j + k$. Obviously $0 \leq i \leq 4$, $0 \leq j \leq 1$ and $0 \leq k \leq 3$. In other words, if we think of these three values as the number of balls in each of the three boxes (i.e. i in the first box, j in the second box, and k in the third box), then we can rest assured that there would be in total 5 balls into the boxes.

This gives a proof (actually a hint, but it is very close to a proof after putting a bit more of detail into it) of the fact that if we compute the coefficients of $A(x)$,

$$A(x) = 1 + 3x + 5x^2 + 7x^3 + 8x^4 + 7x^5 + 5x^6 + 3x^7 + x^8,$$

then $[x^5]A(x) = 7$ is the number of ways of putting 5 balls into the boxes with the given capacities.

Hence, using this reasoning, we see that the problem in general amounts just to do the following: compute $[x^r]A(x)$ where

$$A(x) = \prod_{j=1}^{n} (1 + x + \cdots + x^{a_i}).$$

We can multiply these n polynomials using FFT (to have a low complexity computing each product) and a divide and conquer approach (to do only $O(\log(n))$ products in total).

We leave a related problem to think about.

Problem 3.1. Let $S = \{a_1, \ldots, a_n\}$ be a set of $n \leq 1000$ integers $1 \leq a_i \leq 10^9$, and assume that we are given an integer $1 \leq k \leq n$. We are asked to compute the following sum (where the divisions are modulo $10^9 + 7$)

$$\sum_{i_1 < \cdots < i_k} 2^k \cdot \frac{a_{i_1} \cdots a_{i_k}}{i_1 \cdots i_k} \pmod{10^9 + 7},$$

where the sum runs over all possibilities of choosing k distinct indices between 1 and n.

4. Combinatorics of symmetries

Problem 4.1. Assume that we are given a $n \times n$ square board. Each 1×1 box has to be colored with one among c different colors that are also given. We are asked to report the number of distinct ways of coloring the board, in the following cases

- When two colorings are said to be equal if they coincide at each box.
- When two colorings are said to be equal if one is a reflection of the other.

We very quickly see that the first version of the problem is very easy. In fact, we can color each of the n^2 boxes using any color we want, so that we have c^{n^2} different colorings. For the other versions of the problem much more care is needed.

Observe that computing the number of objects in a certain set "*up to symmetries*" heavily depends on what the symmetry is. To get a flavour of what is going on here, let us focus on the second problem now.

In this particular case, we have four possible reflections

- Reflect the board along a horizontal line.
- Reflect the board along a vertical line.
- Reflect the board along both a vertical and a horizontal line.
- The reflection "doing nothing".

Let us consider how many colorings *are equivalent* under the first type of reflection (i.e. the horizontal line). Observe that if we color the top half of the board (which has $\lfloor \frac{n+1}{2} \rfloor \times n$ boxes in total), then the whole board will be determined. We have in total $c^{\lfloor \frac{n+1}{2} \rfloor \cdot n}$ such different boards.

Analogously, there are exactly $c^{\lfloor \frac{n+1}{2} \rfloor \cdot n}$ colorings that are equivalent under the second type of reflection (i.e. the vertical line).

For the third type of reflection a little more care is needed, since we are essentially reflecting *through the center* of the board. If n is even, then knowing the top $\lfloor \frac{n+1}{2} \rfloor \times n$ determines the whole board, so that there would be again $c^{\lfloor \frac{n+1}{2} \rfloor \cdot n}$ different colorings. If n is odd, however, the central row is symmetric with respect to the center, so we have to take into account the $\lfloor \frac{n+1}{2} \rfloor$ first boxes of the central line (we include the center of the board here), and the remaining boxes are determined, so that there are $c^{\lfloor \frac{n+1}{2} \rfloor \cdot n + \lfloor \frac{n+1}{2} \rfloor}$ colorings in this case.

In the fourth type of reflection, "doing nothing" means that we have c^{n^2} different boards that are different.

Why did we calculate these numbers? A priori they do not seem to be very useful, because it might be the case that a coloring is symmetric at the same time along the vertical line and along the horizontal line (for example if we color all the boxes of the board with the same color). The key point is that there is a Theorem (that we will state more precisely below) saying that the answer to our problem is exactly the *mean* of all the values we obtained. In other words,

$$\text{answer} = \begin{cases} \frac{1}{4}\left(c^{\lfloor \frac{n+1}{2} \rfloor \cdot n} + c^{\lfloor \frac{n+1}{2} \rfloor \cdot n} + c^{\lfloor \frac{n+1}{2} \rfloor \cdot n} + c^{n^2}\right) & \text{if } n \text{ is even} \\ \frac{1}{4}\left(c^{\lfloor \frac{n+1}{2} \rfloor \cdot n} + c^{\lfloor \frac{n+1}{2} \rfloor \cdot n} + c^{\lfloor \frac{n+1}{2} \rfloor \cdot n + \lfloor \frac{n+1}{2} \rfloor} + c^{n^2}\right) & \text{if } n \text{ is odd} \end{cases}$$

Let us state the result in general

Theorem 4.2 (Burnside's Lemma)**.** *Let X be a set, and let $s_1, \dots, s_k$ be the symmetries of the set. If the symmetries are nice (see the Remark below), then the number of ways of coloring the elements of X up to symmetries is*

$$\frac{1}{k} \sum_{i=1}^{k} \text{fix}(s_i),$$

where $\text{fix}(s_i)$ *is the number of colorings of X that are symmetric under the symmetry s_i.*

Remark 4.3. For the reader familiar with the technicalities. The above statement is a *down-to-Earth* version of Burnside's Lemma. In fact, as it is stated it will fail in some cases. The symmetries that we consider must conform *a group* (i.e. the composition of symmetries has to be a symmetry, there has to be a "doing nothing" symmetry and every symmetry has to admit an inverse symmetry).

There is a related fact that might be useful in some problems.

Theorem 4.4. *Let X be a set, and let $s_1, \ldots, s_k$ be the (nice) symmetries of this set. Assume that we have exactly c colors. Then the number of ways of coloring the elements of X up to symmetries is a polynomial in c of degree at most $|X|$.*

Observe that in our example X was the set of boxes of the $n \times n$ board, so that $|X| = n^2$. In fact, if we look at the answer, we see that it is a polynomial in c of degree exactly $|X|$. Why do we care about this? Because in some (weird and possibly *very* hard otherwise) problems it might happen that the size of the set X is relatively small whereas the number of colors can be huge. Hence, via Lagrange interpolation we can reduce the task to an easier problem.

5. Generating functions

Now we are going to introduce ourselves (very gently) to the world of generating functions. Essentially, a generating functions is just a way of talking about *sequences*. The true power of generating functions is that many operations that in the world of mathematical or complex analysis would completely *make sense*, still do when using the new language that we are going to discuss.

Assume that we have the sequence

$$a_0 = 1, \;\; a_1 = 2, \;\; a_2 = 4, \;\; \ldots, \;\; a_n = 2^n, \text{etc.}$$

Let us *encode* our sequence as follows. Consider:

$$f(x) = \sum_{k=0}^{\infty} a_k x^k = \sum_{k=0}^{\infty} 2^k x^k.$$

We will **absolutely never** evaluate f into any value of x. In other words, we don't care about what $f(8)$ is. We just write our sequence using this symbols that are nothing else than a *formal* expression. The above object is what we call the *generating function* of the sequence $\{a_n\}$.

In general if $f(x)$ is the generating function of $\{a_n\}$ we have that $[x^k]f(x) = a_k$ for every k. The key idea is that we can treat these objects as if they were just ordinary polynomials with *an infinite number of coefficients* and everything works just well. What does this mean? Let us see an example

Example 5.1. Consider the *all ones sequence*, $a_n = 1$ for all n. What is its generating function? Well, it is just

$$f(x) = \sum_{k=0}^{\infty} x^k \;\; \text{(here we talk about generating functions)}$$

If we put an effort into recalling what we know about series in the mathematical analysis world, we might recall that the expression in the right is just the Taylor expansion of the

following function around zero:

$$\frac{1}{1-x} = \sum_{k=0}^{\infty} x^k \quad \text{(here we talk about } \textit{analytic} \text{ functions)}$$

What we will do here is to say that $\boxed{\frac{1}{1-x} \textit{ is} \text{ the generating function of the sequence } \{a_n = 1\}}$.

Observe that using this trick we can write many sequences in a very compact form.

Example 5.2. Consider the sequence $a_n = 2^n$. We stated above that its generating function is just

$$f(x) = \sum_{k=0}^{\infty} 2^k x^k = \sum_{k=0}^{\infty} (2x)^k = \frac{1}{1-2x}$$

Example 5.3. Consider the sequence $a_n = \frac{1}{n!}$. We see that its generating function is

$$f(x) = \sum_{k=0}^{\infty} \frac{1}{k!} x^k = e^x$$

Observe that if we deal with more than one sequence at the same time, say that we have $\{a_n\}$ and $\{b_n\}$, with generating functions

$$f(x) = \sum_{k=0}^{\infty} a_k x^k$$

$$g(x) = \sum_{k=0}^{\infty} b_k x^k,$$

it is clear that the generating function of the sequence $c_n = a_n + b_n$ is just

$$h(x) = f(x) + g(x) = \sum_{k=0}^{\infty} (a_k + b_k) x^k$$

There are more operations that can be used in the world of generating functions.

Example 5.4. Assume that the sequence $\{a_n\}$ has generating function f. We can determine completely who the generating function of the sequence $b_n = n a_n$ is in terms of f. Look at the *derivative* of f

$$f'(x) = \sum_{k=1}^{\infty} k a_k x^{k-1},$$

we see that

$$x f'(x) = \sum_{k=0}^{\infty} k a_k x^k = \text{generating function of } \{b_n\}.$$

Example 5.5. Consider again $a_n = 1$ for all n, and call $f(x) = \frac{1}{1-x}$ its generating function. We have that

$$f'(x) = \left(\frac{1}{1-x}\right)' = \frac{1}{(1-x)^2} = \sum_{k=1}^{\infty} k x^{k-1}$$

$$f''(x) = \left(\frac{1}{1-x}\right)'' = \frac{2}{(1-x)^3} = \sum_{k=2}^{\infty} k(k-1)x^{k-2}$$
$$f'''(x) = \left(\frac{1}{1-x}\right)''' = \frac{6}{(1-x)^4} = \sum_{k=3}^{\infty} k(k-1)(k-2)x^{k-3}$$

In other words, we can see that

$$\frac{1}{(1-x)^2} = \sum_{k=0}^{\infty} k x^{k-1}$$
$$\frac{1}{(1-x)^3} = \sum_{k=2}^{\infty} \binom{k}{2} x^{k-2}$$
$$\frac{1}{(1-x)^4} = \sum_{k=3}^{\infty} \binom{k}{3} x^{k-3}$$

This is a general fact, the generating function of the m-th binomial numbers is

$$\frac{1}{(1-x)^{m+1}} = \sum_{k=m}^{\infty} \binom{k}{m} x^{k-m} = \sum_{j=0}^{\infty} \binom{m+j}{m} x^j$$

Example 5.6. Assume that $\{a_n\}$ has generating function f. We can determine completely who the generating function of $b_n = a_0 + a_1 + \cdots + a_n$ is in terms of f. In fact, consider the following product

$$g(x) = (1 + x + x^2 + \cdots) \cdot \sum_{k=0}^{\infty} a_k x^k,$$

it is not difficult to see that the expression in the right is just

$$a_0 + (a_0 + a_1)x + (a_0 + a_1 + a_2)x^2 + \cdots = \sum_{k=0}^{\infty} b_k x^k.$$

In other words, we see that g is the generating function of the sequence $\{b_n\}$. Moreover,

$$g(x) = (1 + x + x^2 + \cdots) \cdot \sum_{k=0}^{\infty} a_k x^k = \frac{1}{1-x} \sum_{k=0}^{\infty} a_k x^k = \frac{f(x)}{1-x}.$$

In other words, if f is the generating function of $\{a_n\}$ then $\frac{f(x)}{1-x}$ is the generating function of the sequence of partial sums.

6. Application of generating functions in theory

How can one apply these results? Well, let us see a very classic example. Assume that we want to *prove* that

$$1^2 + 2^2 + \cdots + n^2 = \frac{n(n+1)(2n+1)}{6}.$$

Let us consider the generating function u of the sequence $a_n = n^2$. So that:

$$u(x) = \sum_{k=0}^{\infty} k^2 x^k.$$

At first sight we cannot tell any nice expression for the infinite sum in the right. But, observe the following, if we define

$$f(x) = \frac{1}{1-x} = \sum_{k=0}^{\infty} x^k,$$

we see that the derivatives are respectively

$$f'(x) = \frac{1}{(1-x)^2} = \sum_{k=1}^{\infty} k x^{k-1} = \frac{1}{x} \sum_{k=0}^{\infty} k x^k.$$

Unfortunately, we did not achieve a k^2 but just a k in the right expression. But we see that we are very close. The above equalities translate into

$$xf'(x) = \frac{x}{(1-x)^2} = \sum_{k=0}^{\infty} k x^k$$

So, if we take derivatives again, we obtain

$$xf''(x) + f'(x) = \frac{1+x}{(1-x)^3} = \sum_{k=0}^{\infty} k^2 x^{k-1} = \frac{1}{x} \sum_{k=0}^{\infty} k^2 x^k$$

In other words, we know that the generating function of $a_n = n^2$ is

$$u(x) = \frac{x(1+x)}{(1-x)^3}.$$

Hence, the generating function for $b_n = 0^2 + 1^2 + 2^2 + \cdots + n^2$ has to be

$$v(x) = \frac{x(1+x)}{(1-x)^4}.$$

We see that $v(x) = (x + x^2) \cdot \frac{1}{(1-x)^4}$, which can be rewritten as

$$v(x) = (x + x^2) \sum_{j=0}^{\infty} \binom{3+j}{3} x^j = \sum_{j=0}^{\infty} \binom{3+j}{3} x^{j+1} + \sum_{j=0}^{\infty} \binom{3+j}{3} x^{j+2}$$

As $v(x)$ is the generating function of the sum of the first n squares, we see that:

$$1^2 + 2^2 + \cdots + k^2 = [x^k]v(x) = \binom{3+k-1}{3} + \binom{3+k-2}{3}.$$

Hence, we obtained that

$$1^2 + 2^2 + \cdots + k^2 = \binom{k+2}{3} + \binom{k+1}{3} = \frac{k(k+1)(2k+1)}{6}.$$

ZW

Università di Bologna, Dipartimento di Matematica, Piazza di Porta San Donato, 5, 40126 Bologna BO - Italia

E-mail address: luis.ferronirivetti2@unibo.it,