The 2nd Problem Set

January 2, 2018

Pre-requisites

You should do the followings

1. On a separate and clean paper, you need to describe your own strategy to solve the problems below, and to justify why your strategy is effective while handling each problem
2. On a new clean paper, transform your strategy into an algorithm, using your own form to express algorithms. Further, you need to analyze the total running steps of your algorithm and the required memory amount to finish your algorithm. Then express the total costs using Big-O notation.
3. Use the `Code ocean` (`https://codeocean.com`) platform; if necessary, you may invite me using my email address `lightsun.kim@gmail.com`.
4. Time limits for each problem
   - Problem #1: Within 4 hours
   - Problem #2: Within 5 hours
   - Problem #3: Within 4 hours

   Then you need to prepare two answer codes; one is a C code that you have made within each time limit, and the other is a C code augmented and fixed from the original code later.

Background

Let $\mathbb{Z}$ be the set of the integers. Doing arithmetic over $\mathbb{Z}$, there are two critical problems: (1) because $\mathbb{Z}$ is infinite, we cannot express the set $\mathbb{Z}$ on any computing machine, and (2) because $a \div b \notin \mathbb{Z}$ for some integers $a, b \in \mathbb{Z}$, division over $\mathbb{Z}$ is not closed. For example, when $a = 1, b = 2$, it is clear that $1 \div 2 \notin \mathbb{Z}$.

To solve these problems, we build the set $\mathbb{Z}_p = \{0, 1, \ldots, p-1\}$ for a positive integer $p > 1$. It is well known that for all integers $a, p \in \mathbb{Z}$ with $p > 0$, there exist unique q, r such that $a = pq + r$ and $0 \leqslant r < p$. Using the theorem, let us define a function $\mathrm{mod}(\cdot, \cdot)$ as follows.

$$\mathrm{mod}(a, p) := a \pmod{p} = r.$$

Next, we define four basic operations in $\mathbb{Z}_p$ for an integer $p > 2$. For all $a, b \in \mathbb{Z}_p$,

$$\begin{array}{rcl} a \boxplus b & := & (a + b) \pmod{p} = \mathrm{mod}(a + b, p), \\ a \boxminus b & := & (a + (p - b)) \pmod{p} = \mathrm{mod}(a + (p - b), p), \\ a \boxdot b & := & (a \cdot b) \pmod{p} = \mathrm{mod}(a \cdot b, p), \\ a \boxdiv b & := & (a \cdot b^{-1}) \pmod{p} = \mathrm{mod}(a \cdot b^{-1}, p) \end{array}$$

where b^{-1} is the multiplicative inverse of b modular p; in other words, $1 = \mathrm{mod}(b \cdot b^{-1}, p)$. According to Euler's theorem, when p is a prime, for all non-zero $b \in \mathbb{Z}_p$ there always exists unique $\beta \in \mathbb{Z}_p$ such that $\mathrm{mod}(b \cdot \beta, p) = 1$. In fact, we can find such a multiplicative inverse $\beta = b^{-1} \mod p$ of $b \in \mathbb{Z}_p$ using the *extended Euclidean algorithm* (EEA). Because $b, \beta \in \mathbb{Z}_p$ and multiplication is closed in $\mathbb{Z}_p$, It is easy to check that $a \boxdiv b = c \in \mathbb{Z}_p$.

Consider the set $\mathbb{Z}_p[x]$ which consists of all polynomials $f(x) = f_0 + f_1 x + \cdots + f_n x^n$ whose coefficients are in $\mathbb{Z}_p$. Then we will meet the same problems in doing arithmetics over $\mathbb{Z}_p[x]$ as calculating over $\mathbb{Z}$. Similarly, we see that $\mathbb{Z}_p[x]$ is an infinite set because the degree of $f(x) \in \mathbb{Z}_p[x]$ is not fixed. Moreover, because $f(x) \div g(x) \notin \mathbb{Z}_p[x]$, polynomial division is not closed in $\mathbb{Z}_p[x]$. To fix these problems, we introduce an irreducible polynomial $\delta(x) = \sum_{i=0}^{k} \delta_i x^i$ with $\delta_k = 1$ for a fixed integer $k > 1$. You may think of the irreducible polynomial $\delta(x)$ as a prime number p in $\mathbb{Z}_p$. Given a prime number p and an irreducible polynomial $\delta(x) \in \mathbb{Z}_p[x]$ of degree k, we can construct a finite set of polynomials $\mathbb{Z}_p[x]/\langle\delta(x)\rangle = \{f(x) | f(x) = f_0 + f_1 x + \cdots + f_{k-1} x^{k-1}$ for each $f_i \in \mathbb{Z}_p\}$. That is to say, $\mathbb{Z}_p[x]/\langle\delta(x)\rangle$ is the set of all polynomials whose degree $< k$ and coefficients are in $\mathbb{Z}_p$.

Now we can can compute polynomial arithmetic over $\mathbb{Z}_p[x]/\langle\delta(x)\rangle$ as follows. Let $f(x), g(x) \in \mathbb{Z}_p[x]/\langle\delta(x)\rangle$ and they are written by $f(x) = \sum_{i=0}^{n} f_i x^i, g(x) = \sum_{i=0}^{m} g_i x^i$ for integers $0 \leqslant n, m < k$ and with $f_n \neq 0, g_m \neq 0$. Assuming that $n \geqslant m$, we have

$$\begin{array}{rcl} f(x) \boxplus g(x) & := & \displaystyle\sum_{i=0}^{n} \mathrm{mod}(f_i + g_i, p) x^i, \\ f(x) \boxminus g(x) & := & \displaystyle\sum_{i=0}^{n} \mathrm{mod}(f_i + (p - g_i), p) x^i, \\ f(x) \boxdot g(x) & := & \displaystyle \mathrm{mod}\left(\sum_{i=0}^{n+m} \left(\sum_{j=0}^{i} f_j \boxdot g_{i-j}\right) x^i, \delta(x)\right) = \mathrm{mod}\left(\sum_{i=0}^{n+m} \mathrm{mod}\left(\sum_{j=0}^{i} f_j \cdot g_{i-j}, p\right) x^i, \delta(x)\right), \\ f(x) \boxdiv g(x) & := & f(x) \boxdot g(x)^{-1} \end{array}$$

where $g(x)^{-1}$ is the multiplicative inverse polynomial of $g(x)$ modular $\delta(x)$; in other words, $1 = g(x) \boxdot g(x)^{-1}$.

For example, consider a setting where $p = 3$ and $\delta(x) = x^3 + 2x + 1$. Let $f(x) = 2x^2 + x, g(x) = x^2 + 2$. Then $f(x) \boxplus g(x) = 2x^2 + x + x^2 + 2 = 3x^2 + x + 3 = x$ and $f(x) - g(x) = 2x^2 + x - x^2 - 2 = x^2 + x - 2 = x^2 + x + 1$. We have

$$\begin{array}{rcl} f(x) \boxdot g(x) & = & (2x^2 + x) \boxdot (x^2 + 2) = 2x^4 + x^3 + 4x^2 + 2x \mod \delta(x) \\ & = & x + 2 \end{array}$$

and

$$\begin{aligned} f(x) \boxplus g(x) &= (2x^2 + x) \boxdot (x^2 + 2)^{-1} \\ &= (2x^2 + x) \boxdot 2x \\ &= 4x^3 + 2x^2 \mod \delta(x) = x^3 + 2x^2 \mod \delta(x) \\ &= 2x^2 - 2x - 1 = 2x^2 + x + 2. \end{aligned}$$

Similarly, we can find the multiplicative inverse of $g(x)$ modular $\delta(x)$ over $\mathbb{Z}_p[x]/\langle\delta(x)\rangle$, using the EEA. See the below how the EEA runs over $\mathbb{Z}$.

1 Problem #1

Let a and p be non-zero positive integers with $a > p$. The greatest common divisor (GCD) is computed by the Euclidean algorithm. The Euclidean algorithm can be extended so that it outputs a multiplicative inverse of a modular p. Let us look at the extended Euclidean algorithm, firstly.

Algorithm The extended Euclidean algorithm

INPUT: two non-zero positive integers a, p with $a \geqslant p$
OUTPUT: (d, s, t) such that $d = \gcd(a, p)$ and $as + pt = d$

1. `if` $(n = 0)$ `then`
2. $\quad d \leftarrow a, s \leftarrow 1, t \leftarrow 0$
3. $\quad$ `return` (d, s, t)
4. $s_0 \leftarrow 1, t_0 \leftarrow 0, s_1 \leftarrow 0, t_1 \leftarrow 1$
5. $i \leftarrow 1$
6. `do`
7. $\quad i \leftarrow i + 1$
8. $\quad r_i \leftarrow r_{i-2} \mod r_{i-1}$
9. $\quad q_{i-1} \leftarrow \frac{r_{i-2} - r_i}{r_{i-1}}$
10. $\quad s_i \leftarrow s_{i-2} - q_{i-1} s_{i-1}$
11. $\quad t_i \leftarrow t_{i-2} - q_{i-1} t_{i-1}$
12. `while` $(r_i \neq 0)$
13. $d \leftarrow r_{i-1}, s \leftarrow s_{i-1}, t \leftarrow t_{i-1}$
14. `return` (d, s, t)

After running the EEA with (a, p), only when $d = 1$ you can compute the multiplicative inverse s of $a \mod n$. The reason is why if you apply "mod p" to both sides of $d = as + pt$, you have

$$\begin{aligned} d \mod p &= 1 \ (\because d = 1) \\ &= as + pt \mod p \\ &= as \mod p \ (\because 0 = pt \mod n) \end{aligned}$$

The result is that $1 = as \mod p$ and we see that s is the multiplicative inverse of $a \mod p$. When a and p are replaced with polynomials $F(x)$ and $G(x)$ respectively, the EEA can also make us compute the multiplicative inverse polynomial of $F(x)$ modular $G(x)$ only if $D(x) = \gcd(F(x), G(x)) = 1$.

Language requirements. During tackling this problem, you should follow the programming rules:

- You should use an ANSI C programming language whose source code can run on `Code ocean` platform.
- Function naming: Begin with the lower character, and every parameters are strong-typed variables (i.e., do not use `void` typed variables). All functions should have a single return value; thus even if a function will return no values; you should provide `return` keyword.
- Variable naming: Begin with a type-discriminating prefix. For example, if a variable name is for an age and is with an integer type, you need to declare the variable as `int iAge;` Especially for string-type variables you are strongly recommended to use the prefix `sz`. For example, if a variable name is for a name, then `szName` is a preferable choice.

Input format. The input is given a text-format file, named `input.txt` and all strings are separated by the blank character. You should take as input non-zero positive integers n, m with $n > m$. When assuming that $F(x) = \sum_{i=0}^{n} f_i x^i$ and $G(x) = \sum_{i=0}^{m} g_i x^i$, your input file writes only their coefficients of $F(x)$ and $G(x)$ in the form of vector, respectively.

In particular, you should randomly generate the integers n, m such that $n, m < 20$. Likewise, you should take two polynomials $F(x), G(x)$ at random except that $g_m = 1$. One way to handle it is to separately implement a program that creates an input file with correct values.

n m
[f_0 f_1 f_2 $\cdots$ f_n]
[g_0 g_1 g_2 $\cdots$ $1 = g_m$]

Output format. The output should be given as a text-format file, named `output.txt`. The output file writes (1) the GCD polynomial $D(x)$ such that $\gcd(F(x), G(x)) = D(x) = \sum_{i=0}^{u} d_i x^i$ and (2) if $D(x) = 1$, compute the multiplicative inverse $R(x) = \sum_{i=0}^{v} r_i x^i$ of $F(x) \mod G(x)$ such that $1 = R(x)F(x) \mod G(x)$.

gcd
[d_u d_{u-1} $\cdots$ d_1 d_0]
inverse
[r_v r_{v-1} $\cdots$ r_1 r_0]

2 Problem #2

We fix the prime $p = 3$ and the irreducible polynomial $\delta(x) = x^3 + 2x^2 + 1$.

Input format. The input is given a text-format file, named `input.txt` and all strings are separated by the blank character.

ℓ
$\mathtt{op}_1\ \mathtt{op}_2\ \mathtt{op}_3\ \mathtt{op}_4\ \cdots\ \mathtt{op}_\ell$
$[f_0\ \ f_1\ \ f_2\ \ \cdots\ \ f_n]$
$[g_{1,0}\ \ g_{1,1}\ \ g_{1,2}\ \ \cdots\ \ g_{1,m_1}]$
$[g_{2,0}\ \ g_{2,1}\ \ g_{2,2}\ \ \cdots\ \ g_{2,m_2}]$
$\vdots$
$[g_{\ell,0}\ \ g_{\ell,1}\ \ g_{\ell,2}\ \ \cdots\ \ g_{\ell,m_\ell}]$

where ℓ is the number of operators in the second line of the input file, and thus the total number of operators at the second line should amount to ℓ. Here because the operators of the second line are polynomial operators, $\mathtt{op}_i \in \{+, -, *\}$ and you should be able to consider $+$ as $\boxplus$, $-$ as $\boxminus$, and $*$ as $\boxdot$. In this problem, you do not consider division ($\div$), which will be visited later. The third line represents a polynomial $F(x)$ of degree $n (1 \leqslant n \leqslant 2)$; i.e., $F(x) = \sum_{i=0}^{n} f_i x^i$. The polynomials from the 4-th line to the last line mean in order

$$\begin{aligned} G_1(x) &= \sum_{i=0}^{m_1} g_{1,i} x^i, \\ G_2(x) &= \sum_{i=0}^{m_2} g_{2,i} x^i, \\ &\vdots \\ G_\ell(x) &= \sum_{i=0}^{m_\ell} g_{\ell,i} x^i. \end{aligned}$$

where for all $j \in \{1, \ldots, \ell\}$, the degrees $1 \leqslant m_i \leqslant 2$.

You should randomly generate $\ell + 1$ polynomials $F(x), G_1(X), \ldots, G_\ell(x)$.

Output format. The output should be given as a text-format file, named `output.txt`. The output file writes the resulting polynomials by performing an operation between $F(x)$ and $G_i(x)$ under the i-th operator $\mathtt{op}_i$ at the second line of the input file. For each $1 \leqslant i \leqslant \ell$ and $\mathtt{op}_i \in \{\boxplus, \boxminus, \boxdot\}$, when we write each resulting polynomial as $R_i(x) = F(x)\ \mathtt{op}_i\ G_i(x) = \sum_{j=0}^{v_i} r_{i,j} x^j$, the output file has to show the coefficients of every resulting polynomial in the form of vector.

$[r_{1,0}\ \ r_{1,1}\ \ r_{1,2}\ \ \cdots\ \ r_{1,v_1}]$
$[r_{2,0}\ \ r_{2,1}\ \ r_{2,2}\ \ \cdots\ \ r_{2,v_2}]$
$\vdots$
$[r_{\ell,0}\ \ r_{\ell,1}\ \ r_{\ell,2}\ \ \cdots\ \ r_{\ell,v_\ell}]$

3 Problem #3

In this problem, we again fix the prime $p = 3$, but can choose an irreducible polynomial from the below list:

Degree k	Irreducible polynomial $\delta(x) = \sum_{i=0}^{k} \delta_i x^i$
2	$x^2 + x + 2$
	$x^2 + 2x + 2$
3	$x^3 + 2x + 1$
	$x^3 + x^2 + 2x + 1$
4	$x^4 + x + 2$
	$x^4 + 2x^3 + 2$
5	$x^5 + 2x^2 + x + 1$
	$x^5 + x^4 + x^3 + x + 1$
6	$x^6 + x^5 + x^3 + 2$
	$x^6 + 2x^5 + 2x^4 + 2x^3 + 2x^2 + 2x + 2$

Input format. The input is given a text-format file, named `input.txt` and all strings are separated by the blank character.

k
$[\delta_0 \ \ \delta_1 \ \ \cdots \ \ \delta_k]$
ℓ
$[f_{1,0} \ \ f_{1,1} \ \ f_{1,2} \ \ \cdots \ \ f_{1,n_1}]$
$[f_{2,0} \ \ f_{2,1} \ \ f_{2,2} \ \ \cdots \ \ f_{2,n_2}]$
$\vdots$
$[f_{\ell,0} \ \ f_{\ell,1} \ \ f_{\ell,2} \ \ \cdots \ \ f_{\ell,n_\ell}]$
$[g_{1,0} \ \ g_{1,1} \ \ g_{1,2} \ \ \cdots \ \ g_{1,m_1}]$
$[g_{2,0} \ \ g_{2,1} \ \ g_{2,2} \ \ \cdots \ \ g_{2,m_2}]$
$\vdots$
$[g_{\ell,0} \ \ g_{\ell,1} \ \ g_{\ell,2} \ \ \cdots \ \ g_{\ell,m_\ell}]$

where k is the degree of your chosen irreducible polynomial $\delta(x) = \delta_k x^k + \delta_{k-1}x^{k-1} + \cdots + \delta_1 x + \delta_0$ which is given at the second line and ℓ is the number of target polynomials. For some $1 \leqslant i \leqslant \ell$, the ℓ polynomials from the 4-th line states $F_i(x) = \sum_{j=0}^{n_i} f_{i,j}x^j$ of degree n_i with each $0 \leqslant n_i \leqslant k$, the next ℓ polynomials to the last line states $G_i(x) = \sum_{j=0}^{m_i} g_{i,j}x^j$ of degree m_i with each $0 \leqslant m_i \leqslant k$.

You should randomly generate 2ℓ polynomials $F_1(x), \ldots, F_\ell(x), G_1(X), \ldots, G_\ell(x)$.

Output format. The output should be given as a text-format file, named `output.txt`. For every $1 \leqslant i \leqslant \ell$, the output file writes all resulting polynomials $R_i(x) = \sum_{j=0}^{v_i} r_{i,j}x^j$ of division between target polynomials so that $R_i(x) = F_i(x) \boxminus G_i(x)$ in the form of vector.

$[r_{1,0} \ \ r_{1,1} \ \ r_{1,2} \ \ \cdots \ \ r_{1,v_1}]$
$[r_{2,0} \ \ r_{2,1} \ \ r_{2,2} \ \ \cdots \ \ r_{2,v_2}]$
$\vdots$
$[r_{\ell,0} \ \ r_{\ell,1} \ \ r_{\ell,2} \ \ \cdots \ \ r_{\ell,v_\ell}]$