Problem A. Arithmetic Subsequence

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 megabytes

Given an integer array $A = [a_1, a_2, \ldots, a_n]$ of length n, you need to determine if there exists an integer array $B = [b_1, b_2, \ldots, b_n]$ such that the followings hold:

- The array B is a rearrangement of A, i.e., there exists a **permutation** $p = [p_1, p_2, \ldots, p_n]$ of size n such that $b_i = a_{p_i}$ for each $1 \le i \le n$.
- The array B doesn't contain any **arithmetic subsequence** of length at least 3.

A sequence $C = [c_1, c_2, \ldots, c_k]$ is called an **arithmetic subsequence** of B if and only if the followings are satisfied:

- There exists a sequence of indices $1 \le i_1 < i_2 < \cdots < i_k \le N$, such that $c_j = b_{i_j}$ for each $1 \le j \le k$;
- C forms an arithmetic progression, i.e., for each $1 \le i \le k-2$, we have $c_{i+2} - c_{i+1} = c_{i+1} - c_i$.

Input

The first line contains an integer T $(1 \le T \le 25)$, denoting the number of test cases.

The first line of each test case contains an integer $n(1 \le n \le 5000)$, denoting the size of array A.

The next line contains n integers $a_1, a_2, \ldots, a_n (1 \le a_i \le 10^9)$, denoting the elements of array A.

Output

For each test case, if no such array B exists, output "NO"(without quotes) in a line. Otherwise, output "YES"(without quotes) in a line, and in the next line output a valid array B. If there are multiple arrays B that satisfy the requirement, outputting any of them would be considered correct.

Example

standard input	standard output
2 4 3 6 8 9 5 1 1 1 1 1	YES 8 6 9 3 NO

Problem B. Conquest of Masters Tour

Input file:	`standard input`
Output file:	`standard output`
Time limit:	10 seconds
Memory limit:	512 megabytes

The *Hearthstone Masters Tour* is a tournament for the famous card-collecting game Hearthstone, consisting of live and online events hosted every year in which Masters Qualifiers winners, Hearthstone Grandmasters, and other invitees compete for massive prize money.

In the Hearthstone Masters Tour, players are paired to play against each other in the format of *Conquest*, which is also the main match format used in official Hearthstone tournaments. The rule of the format is described as follows.

- Each player brings a specific number of decks, depending on tournament rules.
- To begin playing a round, each player chooses one of their decks to battle the opponent with, with the deck choice hidden from the opponent. The decks chosen by each player must satisfy the following rules:
  - Any deck that wins a game cannot be played again in the match;
  - Any deck that is defeated may be played again in the subsequent games.

  Then a game is played by the two players using the chosen deck.
- The first player to win with all of their decks wins the match.

Now you are playing a match in some Hearthstone Masters Tour where each player brings n decks and the decks are determined. You are given an n by n matrix A where $A_{i,j}$ denotes the probability of winning a game if you choose the i-th deck and your opponent chooses the j-th deck.

You want to know, what is the maximum probability you will win the match, supposing your opponent knows your strategy and chooses the deck **optimally** in each round.

Input

The first line contains an integer $T(1 \le T \le 5)$, denoting the number of test cases.

For each test case, the first line contains an integer $n(1 \le n \le 8)$, denoting the number of decks each player brings.

Then n lines describing the matrix A follow, where the $i(1 \le i \le n)$-th line contains n numbers with at most two decimal places $A_{i,1}, A_{i,2}, \ldots, A_{i,n}(0 \le A_{i,j} \le 1)$.

Output

For each test case, output a number in one line, denoting the answer. Your answer is considered correct, if its absolute or relative error does not exceed 10^{-6}.

Formally, let your answer be a, and the jury's answer be b. Your answer is considered correct if $|a - b| \min(1, |b|) \le 10^{-6}$.

Example

standard input	standard output
3 1 0.50 3 1.00 1.00 1.00 1.00 1.00 1.00 0.00 0.00 0.00 3 1.00 0.00 0.00 0.00 1.00 0.00 0.00 0.00 1.00	0.5000000000 0.0000000000 0.1666666667

Note

For the third test case in the sample test, if you use a determined strategy in each round, your opponent will also deterministically chooses a deck that wins with probability 1 if there remains any. Your optimal strategy is to pick any remaining deck uniformly at random in each round, and can win the match with probability $\frac{1}{6}$, no matter what strategy your opponent chooses in each round.

Problem C. Fast Bubble Sort

Input file:	`standard input`
Output file:	`standard output`
Time limit:	5 seconds
Memory limit:	512 megabytes

Given an array $A = (a_1, a_2, \ldots, a_m)$ of length m, denote by array $B(A)$ the output of a single iteration of bubble sort with input array A, i.e., the output of the following algorithm with input array A.

Algorithm 1: A single iteration of the bubble sort

```
Input: Array A of length m
for i ← 1 to m − 1 do
    if A_i > A_{i+1} then
        swap(A_i, A_{i+1})
    end
end
return A
```

A single iteration of the bubble sort

You may perform the following operation any number (including zero) of times on the array $A = (a_1, a_2, \ldots, a_m)$:

- Select an interval $[i, j]$ where $1 \le i \le j \le m$, and cyclically shift all elements of $a_i, a_{i+1}, \ldots, a_{j-1}, a_j$ in either direction, so that they become $a_j, a_i, a_{i+1}, \ldots, a_{j-1}$ or $a_{i+1}, \ldots, a_{j-1}, a_j, a_i$.

For example, if we cyclically shift the interval $[1, 4]$ of the array $A = [1, 2, 3, 4, 5]$ to the right, the resulting array would be $A' = [4, 1, 2, 3, 5]$.

You are now given a permutation $P = (p_1, p_2, \ldots, p_n)$ of length n and you need to answer q independent queries of the following form:

- In the i-th query, you are given parameters $1 \le l_i \le r_i \le n$ and you are supposed to find the minimum number of above operations needed to transform the subarray $P[l_i, r_i]$ to $B(P[l_i, r_i])$, where $P[l_i, r_i] = (p_{l_i}, p_{l_i+1}, \ldots, p_{r_i})$.

Input

The first line contains an integer $T (1 \le T \le 10)$, denoting the number of test cases.

For each test case, the first line contains two integers n, q $(1 \le n, q \le 10^5)$, denoting the length of permutation P and the number of queries, respectively.

The second line contains n distinct integers $p_1, p_2, \ldots, p_n$ $(1 \le p_i \le n)$.

Each of the following q lines contains two integers l_i, r_i $(1 \le l_i \le r_i \le n)$, denoting the parameters for the i-th query.

Output

For each query of each test case, output an integer in one line, denoting the answer.

Example

standard input	standard output
1 10 5 3 7 9 2 6 4 5 8 10 1 1 10 2 6 7 9 4 9 3 3	2 1 0 1 0

Note

For the second query of the sample test, we can transform $P[2,6]$ to $B(P[2,6])$ by performing a single cyclical shift of the interval $[2,5]$(corresponding to the interval $[3,6]$ in P) to the left:

$$[7,9,2,6,4] \to [7,2,6,4,9].$$

Problem D. If You Can't Beat Them, Join Them!

Input file:	`standard input`
Output file:	`standard output`
Time limit:	12 seconds
Memory limit:	512 megabytes

`Roundgod` and `kimoyami` are playing the **graph game** on a directed graph. Given a directed graph $G = (V, E)$ and a source vertex $s \in V$, the graph game (G, s) is defined as follows:

> Initially, there is a token on the source vertex s. `Roundgod` and `kimoyami` take turns moving the token through a directed edge, with `Roundgod` going first. The game ends when either player cannot make a valid move, and the player who cannot make a move loses. If the game lasts 10^{100} turns, then the game is considered a draw.

`Roundgod` is not an expert at this game and is often beaten by `kimoyami`. He remembered the famous quote, "If you can't beat them, join them!", and then came up with the notion of **join of graph games**. Given a nonempty collection of $k(k > 0)$ graph games $(G_1, s_1), \ldots, (G_k, s_k)$. The **join** of the k games is also a game with the following definition:

> `Roundgod` and `kimoyami` take turns moving the tokens in all k graph game **simultaneously**, with `Roundgod` going first. The game ends when either player cannot make a valid move **in any of the k games**, and the player who cannot make a move loses. If the game lasts 10^{100} turns, then the game is considered a draw.

Now, given a collection of k graph games, $(G_1, s_1), \ldots, (G_k, s_k)$, Roundgod then needs to choose a **nonempty subset** from the k games and play with `kimoyami` on the **join** of the chosen games. `Roundgod` wonders, how many ways are there to choose such a nonempty subset so that he may win the game under the optimal strategy of both players? As the answer may be too large, you need to output the answer modulo 998244353.

Input

The first line contains an integer $T(1 \le T \le 5)$, denoting the number of test cases.

For each test case, the first line of the input contains an integer $k(1 \le k \le 10^6)$, denoting the number of graph games in the collection.

Then the description of k graph games follows.

For the description of the $i(1 \le i \le k)$-th graph game, the first line contains three integers $n_i, m_i, s_i(1 \le n_i, m_i \le 10^6, 1 \le s_i \le n_i)$, denoting the number of vertices and edges of in the graph of the ith graph game, and the source vertex of the ith graph game, respectively.

Then m_i lines follow, each line contains two integers $u, v(1 \le u, v \le n_i, u \ne v)$, denoting an edge in the graph of the ith graph game. Note that it is possible for the graph to have **multiple edges**, but not **self loops**.

It is guaranteed that **for each test case**, $\sum_{i=1}^{k} n_i \le 2 \times 10^6$ and $\sum_{i=1}^{k} m_i \le 2 \times 10^6$.

Output

Output an integer in a line, denoting the answer taken modulo 998244353.

Example

standard input	standard output
2 4 3 1 1 2 1 3 3 4 3 2 2 1 1 2 2 1 2 1 1 1 2 2 1 2 1 2	1 2

Note

In the first test case of the sample test, Roundgod can choose the first graph game, and guarantee a win by moving the token from vertex 1 to vertex 2.

In the second test case of the sample test, Roundgod can either choose the second graph game, and guarantee a win by moving the token from vertex 1 to vertex 2; or choose both the first graph game and the second graph game, and guarantee a win by moving the token from vertex 1 to vertex 2 in both games.

Problem E. Leapfrogger

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2.5 seconds
Memory limit:	512 megabytes

Notice: If you have played Battlegrounds in Hearthstone and know the mechanism of leapfrogger and Baron Rivendare, you can start at the fourth paragraph, but note that the context for the problem is simplified and may not be the same as what is in-game.

In the famous card-collecting game *Hearthstone*, there is a mode named *Battlegrounds*, based on the auto battler genre, and allows eight players to compete in each match by recruiting minions over several rounds. In Battlegrounds, there is a special minion named **leapfrogger**, which is a second-tier beast with stats 3/3, and has the deathrattle of *"Give a friendly beast +1/+1 and this deathrattle"*, which means that when this minion dies, it will randomly give a friendly beast (if there exists any) the effect of +1/+1 in stats and this same effect (i.e., when that friendly beast dies, it will randomly give a friendly beast +1/+1 and the same effect, et cetera).

An interesting thing about leapfrogger is that its deathrattle is **stackable**, i.e., a minion is allowed to have multiple, say k, deathrattles of the leapfrogger at the same time, then when this minion dies, it will randomly give a friendly beast(if there exists any) the effect of +1/+1 in stats and this same effect, **repeated k times.** Note that in each of the k times, the friendly beast is chosen **independently** at random. What makes things more interesting is a minion in Battlegrounds called **Baron Rivendare** that can double the effect of deathrattles when it is on the board. When Baron Rivendare is on the board, and a leapfrogger dies, its deathrattle is triggered and given to a random friendly beast, **twice**. If both deathrattles are given to the same friendly beast and that beast dies, still with Baron Rivendare on the board, each of the two deathrattles of leapfrogger on the beast is triggered twice, so that a total of four copies of the deathrattle of leapfrogger are triggered and given to a random friendly beast... The speed the deathrattle of leapfrogger spread when Baron Rivendare is on the board is quite insane and is what makes the "leapfrogger build"a possible and quite powerful strategy in Battlegrounds.

That is quite a lot for the background. Let's then make some simplifications. We assume the deathrattle of leapfrogger can be given to **any minion** instead of **only to beasts**. We also assume that **ALL Deathrattles will be triggered k times** for the sake of the problem. The question is,

> There are n minions on the board with **exactly one** of them being leapfrogger. If you kill all the n minions in a random order, how many times in expectation will the deathrattle of the

leapfrogger be triggered? (Recall that all deathrattles will be triggered k times), answer the question for all $k = 2, 3, ..., m$ for some given parameter m. Note that the deathrattles on the last minion still count as triggered, even if they are not given to another minion.

Input

The first line contains an integer $T(1 \le T \le 10)$, denoting the number of testcases.

For each test case, the input consists of one line containing two integers $n(1 \le n < 998244353)$ and $m(2 \le m \le 10^5)$, denoting the number of minions and the parameter, respectively.

Output

The output consists of $m - 1$ lines, where on the i-th line, you should output a number denoting the expected number of times the deathrattle of leapfrogger is triggered when all deathrattles will be triggered $k = i + 1$ times. Under the input constraints of the problem, it can be shown that the answer can be written as a fraction $\frac{P}{Q}$, where P and Q are coprime integers and $Q \not\equiv 0 \pmod{998244353}$. You need to output $P \cdot Q^{-1} \pmod{998244353}$ as the answer, where $Q^{-1} \pmod{998244353}$ represents the modular inverse of Q with respect to 998244353.

Example

standard input	standard output
1 3 2	6

Note

For the first test case of the sample test, we only need to deal with the case when $k = 2$. We then have the following possibilities:

- Leapfrogger is the first minion to be killed, this happens with probability $\frac{1}{3}$ and is further divided into three cases:
  - The two deathrattles triggered are given to different minions, this happens with probability $\frac{1}{3} \times \frac{1}{2} = \frac{1}{6}$, and in this case the order you kill the remaining two minions doesn't matter, and the total number of deathrattles triggered is $1 \times 2 + 1 \times 2 + 3 \times 2 = 10$.
  - Both deathrattles are given to the same minion and this minion is the next to be killed, this happens with probability $\frac{1}{3} \times \frac{1}{4} = \frac{1}{12}$ and in this case the total number of deathrattles triggered is $1 \times 2 + 2 \times 2 + 4 \times 2 = 14$.
  - Both deathrattles are given to the same minion and this minion is the last to be killed, this happens with probability $\frac{1}{3} \times \frac{1}{4} = \frac{1}{12}$ and in this case the total number of deathrattles triggered is $1 \times 2 + 2 \times 2 = 6$.
- Leapfrogger is the second minion to be killed, this happens with probability $\frac{1}{3}$ and in this case the total number of deathrattles triggered is $1 \times 2 + 2 \times 2 = 6$.
- Leapfrogger is the last minion to be killed, this happens with probability $\frac{1}{3}$ and in this case the total number of deathrattles triggered is $1 \times 2 = 2$.

Overall, it can be shown the expected number of times the deathrattle is triggered is 6.

Problem F. Mario Party

Input file:	`standard input`
Output file:	`standard output`
Time limit:	15 seconds
Memory limit:	512 megabytes

Mario Party is a classic board game featuring numerous minigames. In this game, players possess coins and aim to collect stars at particular positions. For simplicity, we treat the board as a 1 by n grid with grids labeled with 1 to n from left to right, and there is an integer a_i in cell i. Suppose a player is currently in the cell i with x coins. He may perform the following operation:

> Move to cell $i+1$, and the number of coins he possesses becomes $x+a_{i+1}$ if $x+a_{i+1} \geq 0$, and remains the same otherwise .

You have to answer q independent queries of the following form:

> Suppose a player is currently in cell l with x coins. Compute the number of coins he possesses after he travels to cell r by performing the above operations $r-l$ times.

Input

The first line contains an integer T ($1 \leq T \leq 4$), denoting the number of test cases.

The first line of each test case contains two integers n, q ($1 \leq n, q \leq 5 \cdot 10^5$), denoting the number of cells in the grid and the number of queries, respectively.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($\sum_{i=1}^{n} |a_i| \leq 10^6$).

Each of the following q lines contains integers l_i, r_i, x_i ($1 \leq l_i \leq r_i \leq n$, $0 \leq x_i \leq 10^6$), denoting the parameters of the i-th query.

Output

For each query in each test case, output an integer in one line, denoting the answer.

Example

standard input	standard output
1 5 6 1 -2 3 -4 5 1 5 0 1 5 1 1 5 2 1 5 3 1 5 4 1 5 5	8 5 8 5 6 7

Problem G. Matryoshka Doll

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2.5 seconds
Memory limit:	512 megabytes

`zyb` bought n matryoshka dolls during his visit to Moscow, with sizes $a_1, a_2, \ldots, a_n$, respectively, **sorting from smallest to largest**.

A matryoshka of size i can be put into another matryoshka of size j iff $j - i \geq r$, where r is some given integer parameter.

`zyb` wishes to divide all n matryoshka dolls into k groups, such that one can form a **nested** matryoshka doll in each group, where a group of matryoshka dolls with indices $c_1, c_2, ..., c_m$ $(1 \leq c_1 < c_2 < ... < c_m \leq n)$ can form a **nested** matryoshka doll iff $\forall 1 \leq i < m, a_{c_i} + r \leq a_{c_{i+1}}$.

`zyb` wants to know how many ways there are to divide the n dolls into k groups satisfying the requirement above. Note that divisions such as $\{\{1,2\},\{3,4\}\}$ and $\{\{3,4\},\{1,2\}\}$ are considered the same way. As the answer may be too large, you only need to output the answer modulo 998244353.

Input

The first line contains an integer $T(1 \leq T \leq 20)$ denote the number of testcases.

For each test case, the first line of the input contains three integers $n, k, r(1 \leq k \leq n \leq 5000, 1 \leq r \leq 10^9)$, denoting the number of matryoshka dolls, the number of groups `zyb` wants to divide into, and the parameter, respectively.

The next line contains n integers $a_1, a_2, \ldots, a_n(1 \leq a_1 \leq a_2 \leq ... \leq a_n \leq 10^9)$, denoting the sizes of the matryoshka dolls.

It is guaranteed that $\sum n \leq 50000$ over all test cases.

Output

For each test case, output an integer in a line, denoting the answer taken modulo 998244353.

Example

standard input	standard output
2 4 3 2 1 2 3 4 4 2 1 1 1 2 2	3 2

Problem H. Shortest Path in GCD Graph

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2.5 seconds
Memory limit:	512 megabytes

There is an edge-weighted complete graph K_n with n vertices, where vertices are labeled through $1, 2, ..., n$. For each $1 \leq i < j \leq n$, the weight of the edge (i, j) between i and j is $gcd(i, j)$, the greatest common divisor of i and j.

You need to answer q queries. In each query, given two vertices u, v, you need to answer the **length of the shortest path** as well as the **number of shortest paths** between u, v. Since the **number of shortest paths** may be too large, you only need to output it modulo 998244353.

Input

The first line contains two integers $n, q (2 \leq n \leq 10^7, 1 \leq q \leq 50000)$, denoting the number vertices in the graph and the number of queries, respectively.

Then q lines follow, where each line contains two integers $u, v (1 \leq u, v \leq n, u \neq v)$, denoting a query between u and v.

Output

For each query, output one line contains two integers, denote the length and number of shortest path between given nodes, respectively. Note that only the **number of shortest paths** should be taken modulo 998244353.

Example

standard input	standard output
6 2 4 5 3 6	1 1 2 2

Problem I. Simple Math 4

Input file:	`standard input`
Output file:	`standard output`
Time limit:	9 seconds
Memory limit:	512 megabytes

Given nonnegative integers N, L, R and X, find the maximum value of $\sum_{i=1}^{N} A_i$ over all possible integer arrays A of length N satisfying

- $\bigoplus_{i=1}^{N} A_i = X$, where $\oplus$ denotes the exclusive-or operation.
- $\forall 1 \le i \le N$, $L \le A_i \le R$.

If there exists no valid array A satisfying above requirements, output -1.

Input

The first line contains a single integer T ($1 \le T \le 3000$), denoting the number of test cases.

For each test case, there is a line containing integers N, L, R, X ($1 \le N \le 10^9$, $0 \le L \le R \le 10^9$, $0 \le X \le 10^9$).

Output

For each test case, output an integer in a line, denoting the answer.

Example

standard input	standard output
1 5 890 970 768	4756

Problem J. Sum Plus Product

Input file:	`standard input`
Output file:	`standard output`
Time limit:	10 seconds
Memory limit:	512 megabytes

`triplea` has a box with $n(n \geq 1)$ balls inside, where on each of the balls there is an integer written. When there are **at least two** balls inside the box, `triplea` will do the following operation repeatedly:

- Take two balls from the box, uniformly and independently at random.
- Suppose the numbers written on the two balls are a and b, respectively, then `triplea` will put a new ball in the box on which a number $S + P$ is written, where $S = a + b$ is the sum of a and b, and $P = ab$ is the product of a and b.

The operation will end when there is only one ball in the box. `triplea` wonders, what is the expected value of the number written on the last ball? He gets the answer immediately, and leaves this as an exercise for the reader, namely, you.

Input

The first line of input consists of an integer $T(1 \leq T \leq 20)$, denoting the number of test cases.

For each test case, the first line of input consists of an integer $n(1 \leq n \leq 500)$, denoting the initial number of balls inside the box.

The next line contains n integers $a_1, a_2, \ldots, a_n(0 \leq a_i < 998244353)$, denoting the number written on each ball in the box, initially.

Output

For each test case, output an integer in one line, denoting the expected value of the number written on the last ball. Under the input constraints of this problem, it can be shown that the answer can be written as $\frac{P}{Q}$, where P and Q are coprime integers and $Q \not\equiv 0 \pmod{998244353}$. You need to output $P \cdot Q^{-1} \pmod{998244353}$ as an answer, where Q^{-1} is the modular inverse of Q with respect to 998244353.

Example

standard input	standard output
2 2 2 2 10 1 2 4 8 16 32 64 128 256 512	8 579063023

Note

For the first test case of the sample test, note that triplea can only take two balls with numbers 2 and 2 from the box, and then add into the box a ball with number $(2 + 2) + (2 \times 2) = 8$. This gives the answer.

Problem K. The Alchemist of the Discrete Mathematics

Input file:	`standard input`
Output file:	`standard output`
Time limit:	8 seconds
Memory limit:	512 megabytes

As a great alchemist, Sophie learns Discrete Mathematics very well.

One day, Sophie finds a magic prime number p. From her grandma's teaching, if a polynomial $F(x)$ satisfying that $F(x) \equiv 0 \pmod{p}$ has a solution, then the polynomial is mysterious.

To understand the property of mysterious polynomials, Sophie studies the roots of polynomial.

Given integer n, prime number p and parameter $c \in \mathbb{F}_p$, the set $S \subseteq \mathbb{F}_p$ is good, if there exist polynomial $f(x), g(x) \in \mathbb{F}_p[x]$ satisfying

- $\deg(f) = n$;
- $g(x) \mid f(x)$, i.e., $g(x)$ divides $f(x)$;
- Call T the set of roots of polynomial $h(x) = f(x)g(c \cdot x)$, then $S = T \cap \mathbb{F}_p$.

Sophie wonders the number of good sets given n, p and c. Since the answer may be too large, you should output the answer modulo 998244353.

If you are not familiar with those notations, please refer to the "Note section" for formal definition.

Input

The first line contains an integer $T(T \geq 1)$, denoting the number of test cases.

For each test case, there is a line containing three integers $p, c, n(1 \leq c < p \leq 5 \times 10^5, 0 \leq n \leq 10^6)$, whose meaning is already given in the previous statement.

It is guaranteed that the sum of p over all test cases won't exceed 10^7.

Output

For each test case, output an integer in a line, denoting the answer taken modulo 998244353.

Example

standard input	standard output
2 3 2 2 5 4 2	8 23

Note

Given a prime number p, $\mathbb{F}_p = \{0, 1, 2, \ldots, p-1\}$ is the set of integers with modular arithmetic and $\mathbb{F}_p[x] = \{f(x) = \sum_{i=0}^{n} a_i x^i \mid a_0, a_1, \ldots, a_n \in \mathbb{F}_p\}$.

The degree of polynomial $f(x) = \sum_{i=0}^{n} a_i x^i \in \mathbb{F}_p[x]$ is n if and only if $a_n \neq 0$. We note that the degree of the polynomial $h(x) \equiv 0$ is treated as $-\infty$.

Polynomial $f(x) = \sum_{i=0}^{n} f_i x^i$ divides $g(x)$ if and only if there exists a polynomial $h(x)$ satisfying $f(x) = g(x) \cdot h(x)$, i.e., the k-th coefficient of $f(x)$ is equal to that of $g(x) \cdot h(x)$ for all $k \geq 0$.