Codeforces Round #681 (Div. 2, based on VK Cup 2019-2020 - Final)

A. Kids Seating

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Today the kindergarten has a new group of n kids who need to be seated at the dinner table. The chairs at the table are numbered from 1 to $4n$. Two kids can't sit on the same chair. It is known that two kids who sit on chairs with numbers a and b ($a \neq b$) will indulge if:

1. $gcd(a, b) = 1$ or,
2. a divides b or b divides a.

$gcd(a, b)$ — the maximum number x such that a is divisible by x and b is divisible by x.

For example, if $n = 3$ and the kids sit on chairs with numbers 2, 3, 4, then they will indulge since 4 is divided by 2 and $gcd(2, 3) = 1$. If kids sit on chairs with numbers 4, 6, 10, then they will not indulge.

The teacher really doesn't want the mess at the table, so she wants to seat the kids so there are no 2 of the kid that can indulge. More formally, she wants no pair of chairs a and b that the kids occupy to fulfill the condition above.

Since the teacher is very busy with the entertainment of the kids, she asked you to solve this problem.

Input

The first line contains one integer t ($1 \leq t \leq 100$) — the number of test cases. Then t test cases follow.

Each test case consists of one line containing an integer n ($1 \leq n \leq 100$) — the number of kids.

Output

Output t lines, which contain n distinct integers from 1 to $4n$ — the numbers of chairs that the kids should occupy in the corresponding test case. If there are multiple answers, print any of them. You can print n numbers in any order.

Example

input

```
3
2
3
4
```

output

```
6 4
4 6 10
14 10 12 8
```

B. Saving the City

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Bertown is a city with n buildings in a straight line.

The city's security service discovered that some buildings were mined. A map was compiled, which is a string of length n, where the i-th character is "1" if there is a mine under the building number i and "0" otherwise.

Bertown's best sapper knows how to activate mines so that the buildings above them are not damaged. When a mine under the building numbered x is activated, it explodes and activates two adjacent mines under the buildings numbered $x - 1$ and $x + 1$ (if there were no mines under the building, then nothing happens). Thus, it is enough to activate any one mine on a continuous segment of mines to activate all the mines of this segment. For manual activation of one mine, the sapper takes a coins. He can repeat this operation as many times as you want.

Also, a sapper can place a mine under a building if it wasn't there. For such an operation, he takes b coins. He can also repeat this operation as many times as you want.

The sapper can carry out operations in any order.

You want to blow up all the mines in the city to make it safe. Find the minimum number of coins that the sapper will have to pay so that after his actions there are no mines left in the city.

Input

The first line contains one positive integer t ($1 \le t \le 10^5$) — the number of test cases. Then t test cases follow.

Each test case begins with a line containing two integers a and b ($1 \le a, b \le 1000$) — the cost of activating and placing one mine, respectively.

The next line contains a map of mines in the city — a string consisting of zeros and ones.

The sum of the string lengths for all test cases does not exceed 10^5.

Output

For each test case, output one integer — the minimum number of coins that the sapper will have to pay.

Example

input

```
2
1 1
01000010
5 1
01101110
```

output

```
2
6
```

Note

In the second test case, if we place a mine under the fourth building and then activate it, then all mines on the field are activated. The cost of such operations is six, $b = 1$ coin for placing a mine and $a = 5$ coins for activating.

C. The Delivery Dilemma

time limit per test: 2 seconds

memory limit per test: 256 megabytes

input: standard input

output: standard output

Petya is preparing for his birthday. He decided that there would be n different dishes on the dinner table, numbered from 1 to n. Since Petya doesn't like to cook, he wants to order these dishes in restaurants.

Unfortunately, all dishes are prepared in different restaurants and therefore Petya needs to pick up his orders from n different places. To speed up this process, he wants to order courier delivery at some restaurants. Thus, for each dish, there are two options for Petya how he can get it:

- the dish will be delivered by a courier from the restaurant i, in this case the courier will arrive in a_i minutes,
- Petya goes to the restaurant i on his own and picks up the dish, he will spend b_i minutes on this.

Each restaurant has its own couriers and they start delivering the order at the moment Petya leaves the house. In other words, all couriers work in parallel. Petya must visit all restaurants in which he has not chosen delivery, he does this consistently.

For example, if Petya wants to order $n = 4$ dishes and $a = [3, 7, 4, 5]$, and $b = [2, 1, 2, 4]$, then he can order delivery from the first and the fourth restaurant, and go to the second and third on your own. Then the courier of the first restaurant will bring the order in 3 minutes, the courier of the fourth restaurant will bring the order in 5 minutes, and Petya will pick up the remaining dishes in $1 + 2 = 3$ minutes. Thus, in 5 minutes all the dishes will be at Petya's house.

Find the minimum time after which all the dishes can be at Petya's home.

Input

The first line contains one positive integer t ($1 \le t \le 2 \cdot 10^5$) — the number of test cases. Then t test cases follow.

Each test case begins with a line containing one integer n ($1 \le n \le 2 \cdot 10^5$) — the number of dishes that Petya wants to order.

The second line of each test case contains n integers $a_1 \ldots a_n$ ($1 \le a_i \le 10^9$) — the time of courier delivery of the dish with the number i.

The third line of each test case contains n integers $b_1 \ldots b_n$ ($1 \le b_i \le 10^9$) — the time during which Petya will pick up the dish with the number i.

The sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case output one integer — the minimum time after which all dishes can be at Petya's home.

Example

input

```
4
4
3 7 4 5
```

```
2 1 2 4
4
1 2 3 4
3 3 3 3
2
1 2
10 10
2
10 10
1 2
```

output

```
5
3
2
3
```

D. Extreme Subtraction

time limit per test: 2 seconds

memory limit per test: 256 megabytes

input: standard input

output: standard output

You are given an array a of n positive integers.

You can use the following operation as many times as you like: select any integer $1 \le k \le n$ and do one of two things:

- decrement by one k of the first elements of the array.
- decrement by one k of the last elements of the array.

For example, if $n = 5$ and $a = [3, 2, 2, 1, 4]$, then you can apply one of the following operations to it (not all possible options are listed below):

- decrement from the first two elements of the array. After this operation $a = [2, 1, 2, 1, 4]$;
- decrement from the last three elements of the array. After this operation $a = [3, 2, 1, 0, 3]$;
- decrement from the first five elements of the array. After this operation $a = [2, 1, 1, 0, 3]$;

Determine if it is possible to make all the elements of the array equal to zero by applying a certain number of operations.

Input

The first line contains one positive integer t ($1 \le t \le 30000$) — the number of test cases. Then t test cases follow.

Each test case begins with a line containing one integer n ($1 \le n \le 30000$) — the number of elements in the array.

The second line of each test case contains n integers $a_1 \ldots a_n$ ($1 \le a_i \le 10^6$).

The sum of n over all test cases does not exceed 30000.

Output

For each test case, output on a separate line:

- YES, if it is possible to make all elements of the array equal to zero by applying a certain number of operations.
- NO, otherwise.

The letters in the words YES and NO can be outputed in any case.

Example

input

```
4
3
1 2 1
5
11 7 9 6 8
5
1 3 1 3 1
4
5 2 1 10
```

output

```
YES
YES
NO
YES
```

E. Long Permutation

time limit per test: 4 seconds

memory limit per test: 256 megabytes

input: standard input

output: standard output

A permutation is a sequence of integers from 1 to n of length n containing each number exactly once. For example, $[1]$, $[4, 3, 5, 1, 2]$, $[3, 2, 1]$ — are permutations, and $[1, 1]$, $[4, 3, 1]$, $[2, 3, 4]$ — no.

Permutation a is lexicographically smaller than permutation b (they have the same length n), if in the first index i in which they differ, $a[i] < b[i]$. For example, the permutation $[1, 3, 2, 4]$ is lexicographically smaller than the permutation $[1, 3, 4, 2]$, because the first two elements are equal, and the third element in the first permutation is smaller than in the second.

The next permutation for a permutation a of length n — is the lexicographically smallest permutation b of length n that lexicographically larger than a. For example:

- for permutation $[2, 1, 4, 3]$ the next permutation is $[2, 3, 1, 4]$;
- for permutation $[1, 2, 3]$ the next permutation is $[1, 3, 2]$;
- for permutation $[2, 1]$ next permutation does not exist.

You are given the number n — the length of the initial permutation. The initial permutation has the form $a = [1, 2, \ldots, n]$. In other words, $a[i] = i$ ($1 \le i \le n$).

You need to process q queries of two types:

- $1\ l\ r$: query for the sum of all elements on the segment $[l, r]$. More formally, you need to find $a[l] + a[l + 1] + \ldots + a[r]$.
- $2\ x$: x times replace the current permutation with the next permutation. For example, if $x = 2$ and the current permutation has the form $[1, 3, 4, 2]$, then we should perform such a chain of replacements $[1, 3, 4, 2] \rightarrow [1, 4, 2, 3] \rightarrow [1, 4, 3, 2]$.

For each query of the 1-st type output the required sum.

Input

The first line contains two integers n ($2 \le n \le 2 \cdot 10^5$) and q ($1 \le q \le 2 \cdot 10^5$), where n — the length of the initial permutation, and q — the number of queries.

The next q lines contain a single query of the 1-st or 2-nd type. The 1-st type query consists of three integers 1, l and r $(1 \le l \le r \le n)$, the 2-nd type query consists of two integers 2 and x $(1 \le x \le 10^5)$.

It is guaranteed that all requests of the 2-nd type are possible to process.

Output

For each query of the 1-st type, output on a separate line one integer — the required sum.

Example

input

```
4 4
1 2 4
2 3
1 1 2
1 3 4
```

output

```
9
4
6
```

Note

Initially, the permutation has the form $[1, 2, 3, 4]$. Queries processing is as follows:

1. $2 + 3 + 4 = 9$;
2. $[1, 2, 3, 4] \rightarrow [1, 2, 4, 3] \rightarrow [1, 3, 2, 4] \rightarrow [1, 3, 4, 2]$;
3. $1 + 3 = 4$;
4. $4 + 2 = 6$

F. Identify the Operations

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

We start with a permutation $a_1, a_2, \ldots, a_n$ and with an empty array b. We apply the following operation k times.

On the i-th iteration, we select an index t_i ($1 \le t_i \le n - i + 1$), remove a_{t_i} from the array, and append one of the numbers $a_{t_i - 1}$ or $a_{t_i + 1}$ (if $t_i - 1$ or $t_i + 1$ are within the array bounds) to the right end of the array b. Then we move elements $a_{t_i + 1}, \ldots, a_n$ to the left in order to fill in the empty space.

You are given the initial permutation $a_1, a_2, \ldots, a_n$ and the resulting array $b_1, b_2, \ldots, b_k$. All elements of an array b are **distinct**. Calculate the number of possible sequences of indices $t_1, t_2, \ldots, t_k$ modulo $998\,244\,353$.

Input

Each test contains multiple test cases. The first line contains an integer t ($1 \le t \le 100\,000$), denoting the number of test cases, followed by a description of the test cases.

The first line of each test case contains two integers n, k ($1 \le k < n \le 200\,000$): sizes of arrays a and b.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$): elements of a. All elements of a are **distinct**.

The third line of each test case contains k integers $b_1, b_2, \ldots, b_k$ ($1 \le b_i \le n$): elements of b. All elements of b are **distinct**.

The sum of all n among all test cases is guaranteed to not exceed $200\,000$.

Output

For each test case print one integer: the number of possible sequences modulo $998\,244\,353$.

Example

input

```
3
5 3
1 2 3 4 5
3 2 5
4 3
4 3 2 1
4 3 1
7 4
1 4 7 3 6 2 5
3 2 4 5
```

output

```
2
0
4
```

Note

Let's denote as $a_1 a_2 \ldots \cancel{a_i}\, \underline{a_{i+1}} \ldots a_n \to a_1 a_2 \ldots a_{i-1} a_{i+1} \ldots a_{n-1}$ an operation over an element with index i: removal of element a_i from array a and appending element a_{i+1} to array b.

In the first example test, the following two options can be used to produce the given array b:

- $12\underline{3}\cancel{4}5 \to 1\underline{2}\cancel{3}5 \to 1\cancel{2}\underline{5} \to 12$; $(t_1, t_2, t_3) = (4, 3, 2)$;
- $12\underline{3}\cancel{4}5 \to \cancel{1}\underline{2}35 \to 2\cancel{3}\underline{5} \to 15$; $(t_1, t_2, t_3) = (4, 1, 2)$.

In the second example test, it is impossible to achieve the given array no matter the operations used. That's because, on the first application, we removed the element next to 4, namely number 3, which means that it couldn't be added to array b on the second step.

In the third example test, there are four options to achieve the given array b:

- $14\cancel{7}\underline{3}625 \to 143\cancel{6}\underline{2}5 \to \cancel{1}\underline{4}325 \to 43\cancel{2}\underline{5} \to 435$;
- $14\cancel{7}\underline{3}625 \to 143\cancel{6}\underline{2}5 \to 1\underline{4}\cancel{3}25 \to 14\cancel{2}\underline{5} \to 145$;
- $147\underline{3}\cancel{6}25 \to 147\cancel{3}\underline{2}5 \to \cancel{1}\underline{4}725 \to 47\cancel{2}\underline{5} \to 475$;
- $147\underline{3}\cancel{6}25 \to 147\cancel{3}\underline{2}5 \to 1\underline{4}\cancel{7}25 \to 14\cancel{2}\underline{5} \to 145$;