Codeforces Round #757 (Div. 2)

A. Divan and a Store

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Businessman *Divan* loves chocolate! Today he came to a store to buy some chocolate. Like all businessmen, *Divan* knows the value of money, so he will not buy too expensive chocolate. At the same time, too cheap chocolate tastes bad, so he will not buy it as well.

The store he came to has n different chocolate bars, and the price of the i-th chocolate bar is a_i dollars. *Divan* considers a chocolate bar too expensive if it costs strictly more than r dollars. Similarly, he considers a bar of chocolate to be too cheap if it costs strictly less than l dollars. Divan will not buy too cheap or too expensive bars.

Divan is not going to spend all his money on chocolate bars, so he will spend at most k dollars on chocolates.

Please determine the maximum number of chocolate bars *Divan* can buy.

Input

Each test contains multiple test cases. The first line contains the number of test cases t $(1 \le t \le 100)$. Description of the test cases follows.

The description of each test case consists of two lines. The first line contains integers n, l, r, k $(1 \le n \le 100$, $1 \le l \le r \le 10^9$, $1 \le k \le 10^9)$ — the lowest acceptable price of a chocolate, the highest acceptable price of a chocolate and Divan's total budget, respectively.

The second line contains a sequence $a_1, a_2, \ldots, a_n$ $(1 \le a_i \le 10^9)$ integers — the prices of chocolate bars in the store.

Output

For each test case print a single integer — the maximum number of chocolate bars *Divan* can buy.

Example

input

```
8
3 1 100 100
50 100 50
6 3 5 10
1 2 3 4 5 6
6 3 5 21
1 2 3 4 5 6
10 50 69 100
20 30 40 77 1 1 12 4 70 10000
3 50 80 30
20 60 70
10 2 7 100
2 2 2 2 2 7 7 7 7 7
4 1000000000 1000000000 1000000000
1000000000 1000000000 1000000000 1000000000
1 1 1 1
1
```

output

```
2
2
3
0
0
10
1
1
```

Note

In the first example *Divan* can buy chocolate bars 1 and 3 and spend 100 dollars on them.

In the second example *Divan* can buy chocolate bars 3 and 4 and spend 7 dollars on them.

In the third example *Divan* can buy chocolate bars 3, 4, and 5 for 12 dollars.

In the fourth example *Divan* cannot buy any chocolate bar because each of them is either too cheap or too expensive.

In the fifth example *Divan* cannot buy any chocolate bar because he considers the first bar too cheap, and has no budget for the second or third.

In the sixth example *Divan* can buy all the chocolate bars in the shop.

B. Divan and a New Project

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

The company "Divan's Sofas" is planning to build $n + 1$ different buildings on a coordinate line so that:

- the coordinate of each building is an integer number;

- no two buildings stand at the same point.

Let x_i be the coordinate of the i-th building. To get from the building i to the building j, *Divan* spends $|x_i - x_j|$ minutes, where $|y|$ is the absolute value of y.

All buildings that *Divan* is going to build can be numbered from 0 to n. The businessman will live in the building 0, the new headquarters of "Divan's Sofas". In the first ten years after construction *Divan* will visit the i-th building a_i times, each time spending $2 \cdot |x_0 - x_i|$ minutes for walking.

Divan asks you to choose the coordinates for all $n+1$ buildings so that over the next ten years the businessman will spend as little time for walking as possible.

Input

Each test contains several test cases. The first line contains one integer number t ($1 \le t \le 10^3$) — the number of test cases.

The first line of each case contains an integer n ($1 \le n \le 2 \cdot 10^5$) — the number of buildings that "Divan's Sofas" is going to build, apart from the headquarters.

The second line contains the sequence $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 10^6$), where a_i is the number of visits to the i-th building.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, on the first line print the number T — the minimum time *Divan* will spend walking.

On the second line print the sequence $x_0, x_1, \ldots, x_n$ of $n+1$ integers, where x_i ($-10^6 \le x_i \le 10^6$) is the selected coordinate of the i-th building. It can be shown that an optimal answer exists with coordinates not exceeding 10^6.

If there are multiple answers, print any of them.

Example

input

```
4
3
1 2 3
5
3 8 10 6 1
5
1 1 1 1 1
1
0
```

output

```
14
2 4 1 3
78
1 -1 0 2 3 4
18
3 6 1 5 2 4
0
1 2
```

Note

Let's look at the first example.

Divan will visit the first building $a_1 = 1$ times, the second $a_2 = 2$ times and the third $a_3 = 3$ times. Then one of the optimal solution will be as follows:

1. the headquarters is located in $x_0 = 2$;
2. $x_1 = 4$: *Divan* will spend $2 \cdot |x_0 - x_1| \cdot a_1 = 2 \cdot |2-4| \cdot 1 = 4$ minutes walking to the first building;
3. $x_2 = 1$: *Divan* will spend $2 \cdot |x_0 - x_2| \cdot a_2 = 2 \cdot |2-1| \cdot 2 = 4$ minutes walking to the second building;
4. $x_3 = 3$: *Divan* will spend $2 \cdot |x_0 - x_3| \cdot a_3 = 2 \cdot |2-3| \cdot 3 = 6$ minutes walking to the third building.

In total, *Divan* will spend $4+4+6 = 14$ minutes. It can be shown that it is impossible to arrange buildings so that the businessman spends less time.

Among others, $x = [1, 3, 2, 0]$, $x = [-5, -3, -6, -4]$ are also correct answers for the first example.

C. Divan and bitwise operations

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Once *Divan* analyzed a sequence $a_1, a_2, \ldots, a_n$ consisting of n non-negative integers as follows. He considered each non-empty *subsequence* of the sequence a, computed the bitwise XOR of its elements and added up all the XORs, obtaining the *coziness* of the sequence a.

A sequence c is a *subsequence* of a sequence d if c can be obtained from d by deletion of several (possibly, zero or all) elements. For example, $[1,\ 2,\ 3,\ 4]$, $[2,\ 4]$, and $[2]$ are subsequences of $[1,\ 2,\ 3,\ 4]$, but $[4,\ 3]$ and $[0]$ are not.

Divan was very proud of his analysis, but now he lost the sequence a, and also the coziness value! However, *Divan* remembers the value of bitwise OR on m contiguous subsegments of the sequence a. It turns out that each element of the original sequence is contained in **at least one** of these m segments.

Divan asks you to help find the coziness of the sequence a using the information he remembers. If several coziness values are possible, print any.

As the result can be very large, print the value modulo $10^9 + 7$.

Input

The first line contains one integer number t ($1 \le t \le 10^3$) — the number of test cases.

The first line of each test case contains two integer numbers n and m ($1 \le n, m \le 2 \cdot 10^5$) — the length of the sequence and the number of contiguous segments whose bitwise OR values *Divan* remembers, respectively.

The following m lines describe the segments, one per line.

Each segment is described with three integers l, r, and x ($1 \le l \le r \le n$, $0 \le x \le 2^{30} - 1$) — the first and last elements of the segment and the bitwise OR of $a_l, a_{l+1}, \ldots, a_r$, respectively.

It is guaranteed that each element of the sequence is contained in at least one of the segments. It is guaranteed that there exists a sequence that satisfies all constraints.

It is guaranteed that the sum of n and the sum of m over all test cases do not exceed $2 \cdot 10^5$.

Output

For each test case print the coziness any suitable sequence a modulo $10^9 + 7$.

Example

input

```
3
2 1
1 2 2
3 2
1 3 5
2 3 5
5 4
1 2 7
3 3 7
4 4 0
4 5 2
```

output

```
4
20
112
```

Note

In first example, one of the sequences that fits the constraints is $[0, 2]$. Consider all its non-empty subsequences:

- $[0]$: the bitwise XOR of this subsequence is 0;
- $[2]$: the bitwise XOR of this subsequence is 2;
- $[0, 2]$: the bitwise XOR of this subsequence is 2.

The sum of all results is 4, so it is the answer.

In second example, one of the sequences that fits the constraints is $[0,\ 5,\ 5]$.

In third example, one of the sequences that fits the constraints is $[5,\ 6,\ 7,\ 0,\ 2]$.

D1. Divan and Kostomuksha (easy version)

time limit per test: 4 seconds
memory limit per test: 1024 megabytes
input: standard input
output: standard output

This is the easy version of the problem. The only difference is maximum value of a_i.

Once in *Kostomuksha Divan* found an array a consisting of positive integers. Now he wants to reorder the elements of a to maximize the value of the following function:

$$\sum_{i=1}^{n} \gcd(a_1,\ a_2,\ \ldots,\ a_i),$$

where $\gcd(x_1, x_2, \ldots, x_k)$ denotes the greatest common divisor of integers $x_1, x_2, \ldots, x_k$, and $\gcd(x) = x$ for any integer x.

Reordering elements of an array means changing the order of elements in the array arbitrary, or leaving the initial order.

Of course, *Divan* can solve this problem. However, he found it interesting, so he decided to share it with you.

Input

The first line contains a single integer n ($1 \le n \le 10^5$) — the size of the array a.

The second line contains n integers $a_1,\ a_2,\ \ldots,\ a_n$ ($1 \le a_i \le 5 \cdot 10^6$) — the array a.

Output

Output the maximum value of the function that you can get by reordering elements of the array a.

Examples

input

```
6
2 3 1 2 6 2
```

output

```
14
```

input

```
10
5 7 10 3 1 10 100 3 42 54
```

output

```
131
```

Note

In the first example, it's optimal to rearrange the elements of the given array in the following order: $[6,\ 2,\ 2,\ 2,\ 3,\ 1]$:

$$\gcd(a_1) + \gcd(a_1,\ a_2) + \gcd(a_1,\ a_2,\ a_3) + \gcd(a_1,\ a_2,\ a_3,\ a_4) + \gcd(a_1,\ a_2,\ a_3,\ a_4,\ a_5) + \gcd(a_1,\ a_2,\ a_3,\ a_4,\ a_5,\ a_6) = 6 + 2 + 2 + 2 + 1$$

It can be shown that it is impossible to get a better answer.

In the second example, it's optimal to rearrange the elements of a given array in the following order: $[100,\ 10,\ 10,\ 5,\ 1,\ 3,\ 3,\ 7,\ 42,\ 54]$.

D2. Divan and Kostomuksha (hard version)

time limit per test: 4 seconds
memory limit per test: 1024 megabytes
input: standard input
output: standard output

This is the hard version of the problem. The only difference is maximum value of a_i.

Once in *Kostomuksha Divan* found an array a consisting of positive integers. Now he wants to reorder the elements of a to maximize the value of the following function:

$$\sum_{i=1}^{n} \gcd(a_1,\ a_2,\ \dots,\ a_i),$$

where $\gcd(x_1, x_2, \ldots, x_k)$ denotes the greatest common divisor of integers $x_1, x_2, \ldots, x_k$, and $\gcd(x) = x$ for any integer x.

Reordering elements of an array means changing the order of elements in the array arbitrary, or leaving the initial order.

Of course, *Divan* can solve this problem. However, he found it interesting, so he decided to share it with you.

Input

The first line contains a single integer n $(1 \leq n \leq 10^5)$ — the size of the array a.

The second line contains n integers $a_1,\ a_2,\ \dots,\ a_n$ $(1 \leq a_i \leq 2 \cdot 10^7)$ — the array a.

Output

Output the maximum value of the function that you can get by reordering elements of the array a.

Examples

input

```
6
2 3 1 2 6 2
```

output

```
14
```

input

```
10
5 7 10 3 1 10 100 3 42 54
```

output

```
131
```

Note

In the first example, it's optimal to rearrange the elements of the given array in the following order: $[6,\ 2,\ 2,\ 2,\ 3,\ 1]$:

$$\gcd(a_1) + \gcd(a_1,\ a_2) + \gcd(a_1,\ a_2,\ a_3) + \gcd(a_1,\ a_2,\ a_3,\ a_4) + \gcd(a_1,\ a_2,\ a_3,\ a_4,\ a_5) + \gcd(a_1,\ a_2,\ a_3,\ a_4,\ a_5,\ a_6) = 6 + 2 + 2 + 2 + 1$$

It can be shown that it is impossible to get a better answer.

In the second example, it's optimal to rearrange the elements of a given array in the following order: $[100,\ 10,\ 10,\ 5,\ 1,\ 3,\ 3,\ 7,\ 42,\ 54]$.

E. Divan and a Cottage

time limit per test: 2 seconds
memory limit per test: 1024 megabytes
input: standard input
output: standard output

Divan's new cottage is finally complete! However, after a thorough inspection, it turned out that the workers had installed the insulation incorrectly, and now the temperature in the house directly depends on the temperature outside. More precisely, if the temperature in the house is P in the morning, and the street temperature is T, then by the next morning the temperature in the house changes according to the following rule:

- $P_{new} = P + 1$, if $P < T$,
- $P_{new} = P - 1$, if $P > T$,
- $P_{new} = P$, if $P = T$.

Here P_{new} is the temperature in the house next morning.

Divan is a very busy businessman, so sometimes he is not at home for long periods and does not know what the temperature is there now, so he hired you to find it. You will work for n days. In the beginning of the i-th day, the temperature outside T_i is first given to you. After that, on the i-th day, you will receive k_i queries. Each query asks the following: "if the temperature in the house was x_i at the morning of the **first** day, what would be the temperature in the house next morning (after day i)?"

Please answer all the businessman's queries.

Input

The first line of the input contains the number n ($1 \le n \le 2 \cdot 10^5$) — the number of days.

The following is a description of n days in the following format.

The first line of the description contains an integer T_i ($0 \le T_i \le 10^9$) — the temperature on that day.

The second line contains a non-negative integer k_i ($0 \le k_i \le 2 \cdot 10^5$) — the number of queries that day.

The third line contains k integers x'_i ($0 \le x'_i \le 10^9$) — the encrypted version of Divan's queries.

Let $lastans = 0$ initially. Divan's actual queries are given by $x_i = (x'_i + lastans) \bmod (10^9 + 1)$, where $a \bmod b$ is the reminder when a is divided by b. After answering the query, set $lastans$ to the answer.

It is guaranteed that the total number of queries (the sum of all k_i) does not exceed $2 \cdot 10^5$.

Output

For each query, output a single integer — the temperature in the house after day i.

Examples

input

```
3
50
3
1 2 3
50
3
4 5 6
0
3
7 8 9
```

output

```
2
5
9
15
22
30
38
47
53
```

input

```
4
728
3
859 1045 182
104
1
689
346
6
634 356 912 214 1 1
755
3
241 765 473
```

output

```
858
1902
2083
2770
3401
3754
4663
4874
4872
4870
5107
5868
6337
```

input

```
2
500000000
3
1000000000 999999999 888888888
250000000
5
777777777 666666666 555555555 444444444 333333333
```

output

```
999999999
999999996
888888882
666666656
333333321
888888874
333333317
```

```
666666648
```

Note

Let's look at the first four queries from the example input.

The temperature is 50 on the first day, 50 on the second day, and 0 on the third day.

Note that $lastans = 0$ initially.

- The initial temperature of the first query of the first day is $(1 \, + \, lastans) \bmod (10^9 + 1) = 1$. After the first day, the temperature rises by 1, because $1 < 50$. So the answer to the query is 2. Then, we set $lastans = 2$.
- The initial temperature of the second query of the first day is $(2 \, + \, lastans) \bmod (10^9 + 1) = 4$. After the first day, the temperature rises by 1, because $4 < 50$. So the answer to the query is 5. Then, we set $lastans = 5$.
- The initial temperature of the third query of the first day is $(3 \, + \, lastans) \bmod (10^9 + 1) = 8$. After the first day, the temperature rises by 1. So the answer to the query is 9. Then, we set $lastans = 9$.
- The initial temperature of the first query of the second day is $(4 \, + \, lastans) \bmod (10^9 + 1) = 13$. After the first day, the temperature rises by 1. After the second day, the temperature rises by 1. So the answer to the query is 15. Then, we set $lastans = 15$.