A. Energy Crystals

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

There are three energy crystals numbered 1, 2, and 3; let's denote the energy level of the i-th crystal as a_i. Initially, all of them are discharged, meaning their energy levels are equal to 0. Each crystal needs to be charged to level x **(exactly x, not greater)**.

In one action, you can increase the energy level of any one crystal by any positive amount; however, the energy crystals are synchronized with each other, so an action can only be performed if the following condition is met afterward:

- for each pair of crystals i, j, it must hold that $a_i \geq \lfloor \frac{a_j}{2} \rfloor$.

What is the minimum number of actions required to charge all the crystals?

Input

Each test consists of several test cases. The first line contains a single integer t ($1 \leq t \leq 10^4$) — the number of test cases. The description of the test cases follows.

The only line of each test case contains a single integer x ($1 \leq x \leq 10^9$).

Output

For each test case, output a single integer — the minimum number of actions required to charge all energy crystals to level x.

Standard Input	Standard Output
7 1 5 14 2025 31415 536870910 1000000000	3 7 9 23 31 59 61

Note

In the first test case, one possible sequence of actions is:

$$[0, 0, 0] \to [1, 0, 0] \to [1, 0, 1] \to [1, 1, 1]$$

One of the possible sequences of actions in the second test case is:

$$[0, 0, 0] \to [1, 0, 0] \to [1, 1, 0] \to [1, 1, 2] \to [3, 1, 2] \to [3, 5, 2] \to [5, 5, 2] \to [5, 5, 5]$$

,

B. Fibonacci Cubes

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

There are n Fibonacci cubes, where the side of the i-th cube is equal to f_i, where f_i is the i-th Fibonacci number.

In this problem, the Fibonacci numbers are defined as follows:

- $f_1 = 1$
- $f_2 = 2$
- $f_i = f_{i-1} + f_{i-2}$ for $i > 2$

There are also m empty boxes, where the i-th box has a width of w_i, a length of l_i, and a height of h_i.

For each of the m boxes, you need to determine whether all the cubes can fit inside that box. The cubes must be placed in the box following these rules:

- The cubes can only be stacked in the box such that the sides of the cubes are parallel to the sides of the box;
- Every cube must be placed either on the bottom of the box or on top of other cubes in such a way that all space below the cube is occupied;
- A larger cube cannot be placed on top of a smaller cube.

Input

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

In the first line of each test case, there are two integers n and m ($2 \le n \le 10, 1 \le m \le 2 \cdot 10^5$) — the number of cubes and the number of empty boxes.

The next m lines of each test case contain 3 integers w_i, l_i, and h_i ($1 \le w_i, l_i, h_i \le 150$) — the dimensions of the i-th box.

Additional constraints on the input:

- The sum of m across all test cases does not exceed $2 \cdot 10^5$.

Output

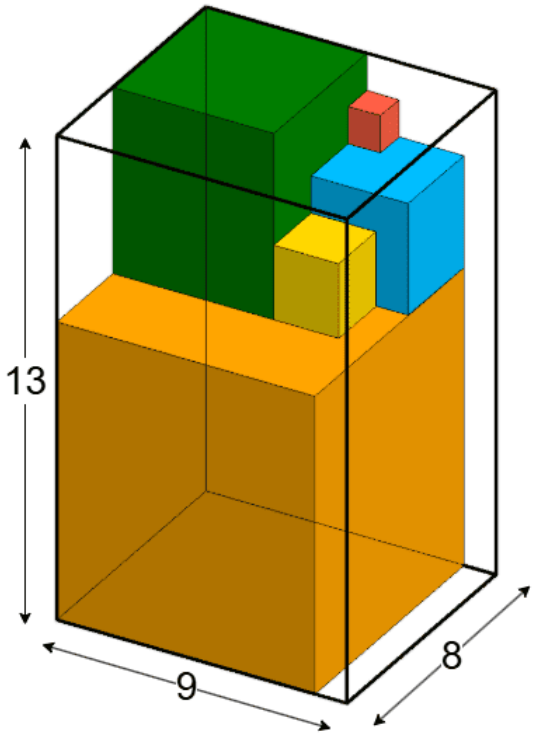
For each test case, output a string of length m, where the i-th character is equal to "1" if all n cubes can fit into the i-th box; otherwise, the i-th character is equal to "0".

Standard Input	Standard Output
2 5 4 3 1 2 10 10 10 9 8 13	0010 100101

14 7 20 2 6 3 3 3 1 2 1 2 1 2 3 2 2 2 3 1 3 2 4	

Note

In the first test case, only one box is suitable. The cubes can be placed in it as follows:



,

C. Equal Values

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

You are given an array $a_1, a_2, \ldots, a_n$, consisting of n integers.

In one operation, you are allowed to perform one of the following actions:

- Choose a position i ($1 < i \le n$) and make all elements to the left of i equal to a_i. That is, assign the value a_i to all a_j ($1 \le j < i$). The cost of such an operation is $(i - 1) \cdot a_i$.
- Choose a position i ($1 \le i < n$) and make all elements to the right of i equal to a_i. That is, assign the value a_i to all a_j ($i < j \le n$). The cost of such an operation is $(n - i) \cdot a_i$.

Note that the elements affected by an operation may already be equal to a_i, but that doesn't change the cost.

You are allowed to perform any number of operations (including zero). What is the minimum total cost to make all elements of the array equal?

Input

The first line contains one integer t ($1 \le t \le 10^4$) — the number of test cases.

The first line of each test case contains a single integer n ($2 \le n \le 5 \cdot 10^5$).

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$).

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

Output

For each test case, print a single integer — the minimum total cost of operations to make all elements of the array equal.

Standard Input	Standard Output
3 4 2 4 1 3 3 1 1 1 10 7 5 5 5 10 9 9 4 6 10	3 0 35

Note

In the first test case, you can perform the operation twice:

- choose $i = 3$ and make all elements to the left of it equal to it, the cost will be $2 \cdot 1 = 2$;
- choose $i = 3$ and make all elements to the right of it equal to it, the cost will be $1 \cdot 1 = 1$.

The total cost is $2 + 1 = 3$.

In the second test case, all elements are already equal, so no operations need to be performed.

,

D. Creating a Schedule

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

A new semester is about to begin, and it is necessary to create a schedule for the first day. There are a total of n groups and m classrooms in the faculty. It is also known that each group has exactly 6 classes on the first day, and the k-th class of each group takes place at the same time. Each class must be held in a classroom, and at the same time, there cannot be classes for more than one group in the same classroom.

Each classroom has its own index (at least three digits), and all digits of this index, except for the last two, indicate the floor on which the classroom is located. For example, classroom 479 is located on the 4-th floor, while classroom 31415 is on the 314-th floor. Between floors, one can move by stairs; for any floor $x > 1$, one can either go down to floor $x - 1$ or go up to floor $x + 1$; from the first floor, one can only go up to the second; from the floor 10^7 (which is the last one), it is possible to go only to the floor 9999999.

The faculty's dean's office has decided to create the schedule in such a way that students move as much as possible between floors, meaning that **the total number of movements between floors across all groups should be maximized**. When the students move from one floor to another floor, they take the shortest path.

For example, if there are $n = 2$ groups and $m = 4$ classrooms $[479, 290, 478, 293]$, the schedule can be arranged as follows:

Class No.	Group 1	Group 2
1	290	293
2	478	479
3	293	290
4	479	478
5	293	290
6	479	478

In such a schedule, the groups will move between the 2nd and 4th floors each time, resulting in a total of 20 movements between floors.

Help the dean's office create any suitable schedule!

Input

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

The first line of each test case contains two integers n and m ($1 \le n \le m \le 10^5$) — the number of groups and the number of available classrooms.

The second line of each test case contains m integers a_i ($100 \le a_i < 10^9$) — the indices of the available classrooms.

Additional constraints on the input:

- the numbers of all classrooms are pairwise distinct;
- the sum of m across all test cases does not exceed 10^5.

Output

For each test case, output n lines, where the i-th line should contain 6 integers — the indices of the classrooms where the classes for the i-th group will be held.

Each classroom must be occupied by at most one group during the k-th class.

Standard Input	Standard Output
3 2 4 479 290 478 293 1 1 31415 6 10 479 385 290 293 384 383 297 478 291 382	290 478 293 479 293 479 293 479 290 478 290 478 31415 31415 31415 31415 31415 31415 479 290 479 290 479 290 290 479 290 479 290 479 293 478 293 478 293 478 297 385 297 385 297 385 478 293 478 293 478 293 291 384 291 384 291 384

Note

In the third test case, the maximum number of moves between classrooms is 50.

,

E. Changing the String

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

Given a string s that consists only of the first three letters of the Latin alphabet, meaning each character of the string is either a, b, or c.

Also given are q operations that need to be performed on the string. In each operation, two letters x and y from the set of the first three letters of the Latin alphabet are provided, and for each operation, one of the following two actions must be taken:

- change any (one) occurrence of the letter x in the string s to the letter y (if at least one occurrence of the letter x exists);
- do nothing.

The goal is to perform all operations in the given order in such a way that the string s becomes lexicographically minimal.

Recall that a string a is lexicographically less than a string b if and only if one of the following conditions holds:

- a is a prefix of b, but $a \neq b$;
- at the first position where a and b differ, the string a has a letter that comes earlier in the alphabet than the corresponding letter in b.

Input

Each test consists of several test cases. The first line contains a single integer t ($1 \leq t \leq 10^3$) — the number of test cases. The description of the test cases follows.

In the first line of each test case, there are two integers n and q ($1 \leq n, q \leq 2 \cdot 10^5$) — the length of the string s and the number of operations.

In the second line of each test case, the string s is given — a string of exactly n characters, each of which is a, b, or c.

The next q lines of each test case contain the description of the operations. Each line contains two characters x and y, each of which is a, b, or c.

Additional constraints on the input:

- the sum of n across all test cases does not exceed $2 \cdot 10^5$;
- the sum of q across all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output the lexicographically minimal string that can be obtained from s using the given operations.

Standard Input	Standard Output

3 2 2 cb c b b a 10 10 bbbbbbbbbb b a b c c b b a c a b c b c b a a b c a 30 20 abcaababcbbcabcbbcabcbabbbbabc b c b c c a b c b c b a b c b c b a b a b a b a c a b c c a b c c a c a b c c b	ab aaaaabbbbb aaaaaaaaaaaaaaabbbabcbabbbbabc

Note

In the first test case, both operations need to be applied to the first letter:

1. after the first operation, $s =$ "bb"
2. after the second operation, $s =$ "ab"

In the second test case, the string could change as follows:

1. "bbbbabbbbb" (changed the 5-th letter)

2. "cbbbabbbbb" (changed the 1-st letter)
3. "cbbbabbbbb" (did nothing)
4. "cbbaabbbbb" (changed the 4-th letter)
5. "abbaabbbbb" (changed the 1-st letter)
6. "abcaabbbbb" (changed the 3-rd letter)
7. "abcaabbbbb" (did nothing)
8. "aacaabbbbb" (changed the 2-nd letter)
9. "aacaabbbbb" (did nothing)
10. "aaaaabbbbb" (changed the 3-rd letter)

,

F. Puzzle

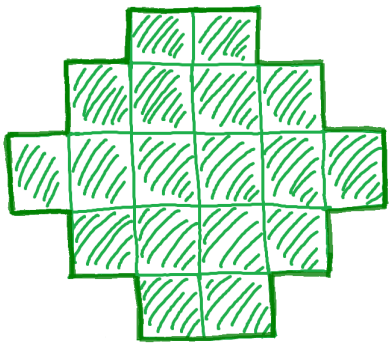
Input file:	standard input
Output file:	standard output
Time limit:	2 seconds
Memory limit:	512 megabytes

You have been gifted a puzzle, where each piece of this puzzle is a square with a side length of one. You can glue any picture onto this puzzle, cut it, and obtain an almost ordinary jigsaw puzzle.

Your friend is an avid mathematician, so he suggested you consider the following problem. Is it possible to arrange the puzzle pieces in such a way that the following conditions are met:

- the pieces are aligned parallel to the coordinate axes;
- the pieces do not overlap each other;
- all pieces form a single connected component (i.e., there exists a path from each piece to every other piece along the pieces, where each two consecutive pieces share a side);
- the ratio of the perimeter of this component to the area of this component equals $\frac{p}{s}$;
- the number of pieces used does not exceed $50\,000$.

Can you handle it?



For this figure, the ratio of the perimeter to the area is $\frac{11}{9}$

Input

Each test consists of several test cases. The first line contains a single integer t ($1 \le t \le 10$) — the number of test cases. The description of the test cases follows.

The only line of each test case contains two integers p and s ($1 \le p, s \le 50$).

Output

For each test case:

- if it is impossible to arrange the pieces as described above, output a single integer -1;
- otherwise, in the first line output a single integer k ($1 \le k \le 50\,000$), and then in k lines output the coordinates of the pieces: each line should contain two integers x_i and y_i ($-10^9 \le x_i, y_i \le 10^9$). If there are multiple suitable arrangements of the pieces, output any of them.

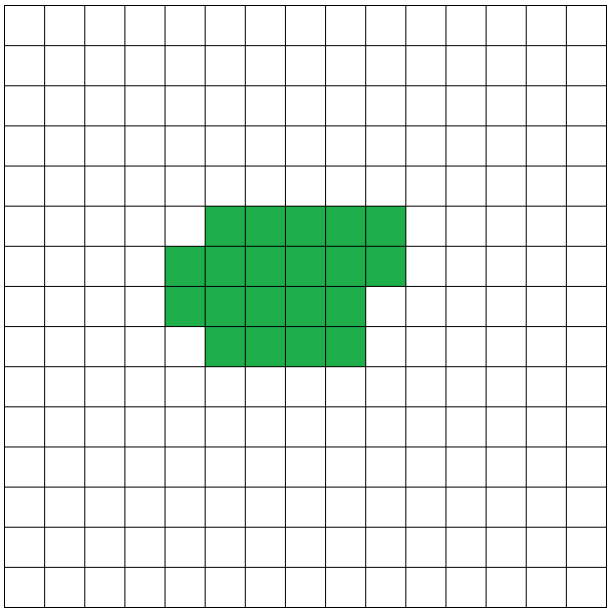
Standard Input	Standard Output
2 1 1	20 3 7

31 4	3 8 6 4 6 5 3 5 4 4 4 5 4 3 3 4 5 3 5 4 5 7 3 6 4 6 5 5 5 6 4 7 4 8 6 6 6 7 -1
2 4 2 12 5	24 -7 2 -3 -3 -7 -5 -7 1 -3 2 -7 -2 -3 -5 -7 -6 -5 -6 -3 -4 -3 -6 -7 0 -6 -6 -7 -3 -5 2 -7 -1 -3 1 -4 -6 -3 0 -7 -4 -6 2 -4 2 -3 -1 -3 -2 5 0 0 0 1 1 0

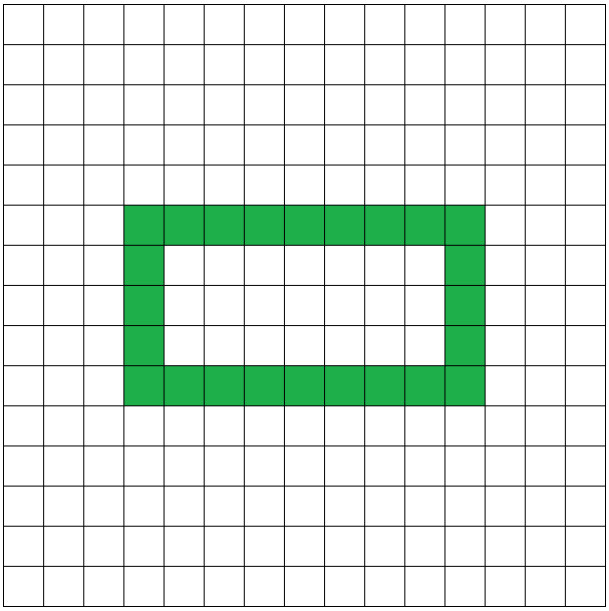
	-1 0 0 -1

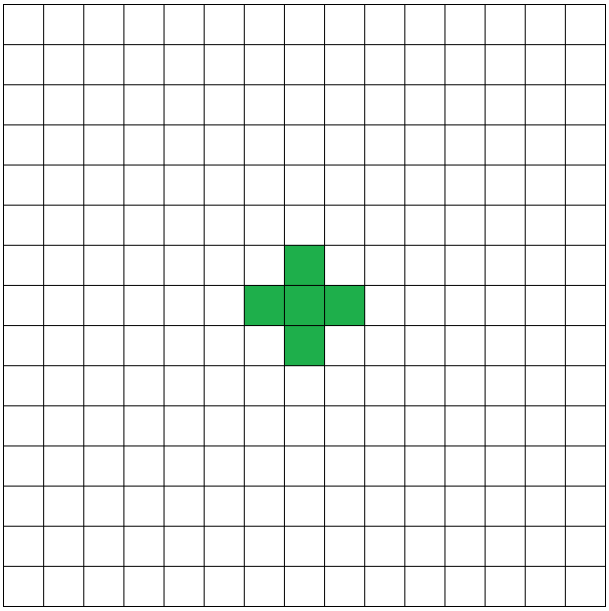
Note

In the first test case of the first test, the figure may look like this:



In the second test, the figures look like this:





Note that the internal perimeter is also taken into account!

,

G. Divisible Subarrays

Input file: standard input
Output file: standard output
Time limit: 7 seconds
Memory limit: 1024 megabytes

Technically, this is an interactive problem.

An array a of m numbers is called *divisible* if at least one of the following conditions holds:

- There exists an index i ($1 \le i < m$) and an integer x such that for all indices j ($j \le i$), it holds that $a_j \le x$ and for all indices k ($k > i$), it holds that $a_k > x$.
- There exists an index i ($1 \le i < m$) and an integer x such that for all indices j ($j \le i$), it holds that $a_j > x$ and for all indices k ($k > i$), it holds that $a_k \le x$.

You are given a permutation p of integers $1, 2, \ldots, n$. Your task is to answer queries of the following form fast: if we take only the segment $[l, r]$ from the permutation, that is, the numbers $p_l, p_{l+1}, \ldots, p_r$, is this subarray of numbers *divisible*?

Queries will be submitted in interactive mode in groups of 10, meaning you will not receive the next group of queries until you output all answers for the current group.

Input

The first line contains one integer n ($2 \le n \le 2 \cdot 10^5$) — the size of the permutation.

The second line contains n integers p_i ($1 \le p_i \le n$) — the permutation of natural numbers itself.

The third line contains one integer q ($10 \le q \le 10^6, q \bmod 10 = 0$) — the number of queries.

The following q lines contain two integers l and r ($1 \le l < r \le n$) — the parameters of the query.

Output

For each query, output the string "YES" if the subarray from this query is *divisible* and "NO" otherwise.

After printing the answers to a group of queries, do not forget to output the end of line and flush the output buffer. Otherwise, you may get the `Idleness Limit Exceeded` verdict. To flush the buffer, use:

- `fflush(stdout)` or `cout.flush()` in C++;
- `System.out.flush()` in Java;
- `flush(output)` in Pascal;
- `stdout.flush()` in Python;
- refer to the documentation for other languages.

You have to flush the output buffer after the 10-th, 20-th, 30-th query (and so on), i. e. after each query with index divisible by 10. After that, you can read the next group of queries.

Standard Input	Standard Output
7 4 2 3 6 1 5 7 20 1 2	YES YES YES YES

1 3	NO
1 4	YES
1 5	YES
1 6	YES
2 3	NO
2 4	YES
2 5	YES
2 6	NO
3 4	YES
3 5	YES
3 6	YES
4 5	YES
4 6	YES
5 6	YES
1 7	YES
2 7	YES
3 7	
4 7	
5 7	