A - End of Evolution

In the beginning Bob created the heavens and the earth. And Bob said, "Let there be species", and there was a species. Initially, there was just one species: the ancestor of all other species to roam the earth. He creatively called it species 1. After that, he made $n - 1$ more species. Each of these species evolved from an older species. Bob then numbered these species 2 to n, though this numbering is arbitrary and not necessarily in chronological order. Finally, he declared some of the species **mature**, and the remaining ones **young**. He did all of this in n days, and on the $(n + 1)$th day, he rested.

Bob wanted more species, but he grew tired of creating new ones manually, so instead, he decided to create more species *algorithmically*.

Here's the algorithm. Every night, the following happens:

- Each young species becomes mature.
- Each young species that became mature on this night generates k "children" species, where k is randomly chosen *nonnegative* integer, chosen with probability $(1 - p)p^k$. These k children species are considered young. Also, the integer 'k' is chosen independently of the 'k's of other species.

You may now notice that evolution ends as soon as all species become mature—an event Bob calls the "end of days".

Define the **generation count** as the largest distance from species 1 to some species. (Species 1 is considered to have distance 0 from species 1.) Formally, the generation count is the largest g such that there exists a sequence of species $s_0, s_1, \ldots, s_g$ satisfying:

- $s_0 = 1$; and
- for each $i > 0$, species s_i is a "child" of species s_{i-1}.

What is the probability that at some night, all species become mature, *and* that the generation count is exactly g when that happens?

You need to output this probability as follows: The probability can be uniquely written as a fraction a/b in lowest terms and b is positive. Output aB modulo $(10^9 + 7)$ where B is a *modular inverse* of b, defined as an integer B such that $bB \equiv 1 \mod (10^9 + 7)$. It can be shown that a modular inverse of b will always exist, and that the required output is unique and well-defined, under the constraints of this problem.

Input Format

The first line of input contains a single integer t, the number of test cases. The description of t test cases follow:

For each test case:

- The first line contains four space-separated integers n, x, y and g.
  - The number p in the problem statement is then defined as $p := x/y$.
- The second line contains n space-separated integers y_1, y_2, ..., y_n.
  - For $1 \le i \le n$, y_i is 1 if species i is young, and 0 if it is mature.
- Each of the next $n - 1$ lines contains two space-separated integers u and v.
  - This line says that either species u evolved from species v, or species v evolved from species u.
  - These $n - 1$ lines describe all evolutionary relationships among species 1 to n.

Output Format

For each test case, print a line containing an integer—the required answer as described in the problem statement.

Constraints

- $1 \le t \le 1000$
- $1 \le n \le 10^5$
- The sum of the ns across all test cases is $\le 3 \cdot 10^5$.
- $0 < x < y < 1000$
- $0 \le g \le 10^7$
- $1 \le u, v \le n$
- $u \ne v$
- It is guaranteed that species 1 is an ancestor of all species.

Sample

Input

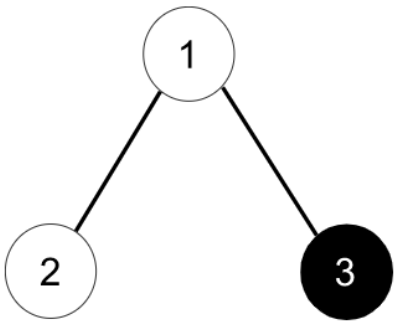
```
2
3 1 2 1
1 1 0
1 2
1 3
5 3 5 2
0 1 0 1 1
1 4
1 3
2 4
4 5
```

Output

```
333333336
852631585
```

Explanation

For the first test case, the situation looks like:



where a black circle represents a mature species, and a white circle represents a young species.

In order for the generation count to be `g = 2`:

- Species `2` should generate `k = 0` children species.
- Species `1` can generate any number `k ≥ 0` of children species (including `0`), but those species should each generate `k = 0` children species.

It can be shown that the probability of this happening is `1/3`.

B - Generalized Collatz Conjecture

You are a bright-eyed young undergrad who has ~~watched a Youtube video about~~ done extensive research into the Collatz conjecture, and have been struck with inspiration for several *brilliant* ideas of how to solve it! You can picture it now—for your undergraduate thesis, you will solve the Collatz conjecture, cementing your name in the textbooks as a genius on the level of von Neumann, Terence Tao, and Ramanujan. Your thesis advisor thinks this is a bad idea; they warn you that many others before you have tried and failed taking this path. But they just don't get it—that's them, and you're you. This is *different*. *You're special*.

Due to your persistence, your thesis advisor reluctantly agrees to support you, but *on the condition* that you prove yourself by solving this problem featuring an **even more difficult and generalized version** of the setup in the Collatz conjecture!

You are given a set `M` of distinct integers and an integer `n`. Your goal is to *transform* `n` into `1` using only operations of the following two kinds:

1. Choose an `m in M`, and replace `n → mn + 1`
2. Choose a prime factor `p` of `n`, and replace `n → n/p`

What is the *minimum* number of operations needed to transform `n` into `1`? If it is impossible to transform `n` to `1`, say so as well.

Input Format

The first line of input contains an integer `t`, the number of test cases. The descriptions of `t` test cases follow.

Each test case consists of one line containing `2+|M|` space-separated integers, where `|M|` denotes the size of `M`. The first two integers are `n` and `|M|`, and the remaining `|M|` ones are the elements of `M`.

Output Format

For each test case, output one line containing either:

- a single integer denoting the minimum number of operations needed to transform `n` to `1`, or
- the string `FIELDS MEDAL` if it is impossible to transform `n` to `1`.

Important Note: The output is **case-sensitive**, so you need to output in all-capital letters. Also, don't put leading or trailing spaces, two consecutive spaces, or tabs, in the output.

Constraints

- `1 ≤ t ≤ 262144`
- `2 ≤ n ≤ 2097152`

- `1 ≤ |M| ≤ 8`
- `1 ≤ m ≤ 64` for each `m` in `M`
- No two cases in each file are exactly the same.
- The elements of `M` are given in increasing order.

Sample

Input

```
2
84 2 3 6
18588 3 18 25 44
```

Output

```
3
4
```

Explanation

- In the first test case, one possible sequence of `3` operations could be: `84 -> 12 -> 37 -> 1`.
- In the second test case, one possible sequence of `4` operations could be: `18588 -> 12 -> 301 -> 5419 -> 1`.

C - Lost Luggage

You've just landed at Kanpur Airport, ready to rumble. You're now waiting to collect your check-in luggage at the airport carousel, but you have a problem: *you forgot what your bag looks like*! You even forgot to put identifying marks on it. (You were busy training, after all!) All you remember is that it is shaped like a cuboid with dimensions $q \times r \times s$ satisfying

- $A \le q \le B$
- $C \le r \le D$
- $E \le s \le F$

after possibly rotating it.

Given the dimensions of the bags in the carousel right now (which all have cuboid shapes), determine which ones among them could be yours. If there are no bags satisfying the conditions, say so as well. (And sorry about your lost luggage!)

Note that you may have to *rotate the bag first* before checking the conditions above.

Input Format

The first line of input contains an integer t denoting the number of test cases. The descriptions of t test cases follow.

The first line of each test case consists of seven space-separated integers n, A, B, C, D, E, F, where n is the number of bags in the carousel right now, and the meanings of the other six integers are given in the problem statement. The next n lines describe the dimensions of the bags. Specifically, the ith line contains three space-separated integers u_i, v_i and w_i. The dimensions of bag i are $u_i \times v_i \times w_i$. The bags are numbered 1 to n.

Output Format

Output t lines, one for each test case. For each test case, output one line containing either:

- The string `MY LUGGAGE IS LOST!` (including the exclamation mark) if there are no bags that could be yours; or,
- a sorted list of space-separated integers between 1 to n denoting the numbers of all the bags that could be yours.

Important Note: The output is **case-sensitive**, so you need to output in all-capital letters. Also, don't put leading or trailing spaces, two consecutive spaces, or tabs, in the output.

Constraints

- $1 \le t \le 15000$
- $1 \le n$

- The sum of the ns in a single test file is ≤ 1500000.
- $1 \leq A \leq B \leq 150$
- $1 \leq C \leq D \leq 150$
- $1 \leq E \leq F \leq 150$
- $1 \leq u_i, v_i, w_i \leq 150$

Sample

Input

```
2
3 20 40 30 55 55 65
56 36 23
10 30 40
35 22 60
6 20 40 30 55 55 65
100 100 100
150 150 150
143 69 42
100 80 100
75 75 100
1 1 1
```

Output

```
1 3
MY LUGGAGE IS LOST!
```

Explanation

In the first test case, bags 1 and 3 satisfy the condition, while bag 2 does not. For bag 1, by rotating it, we can write its dimensions as $23 \times 36 \times 56$, and we can verify that $20 \leq 23 \leq 40$, $30 \leq 36 \leq 55$, and $55 \leq 56 \leq 65$.

D - Mod Messages

This problem is a tribute to all the moderators who are working tirelessly to keep the trolls and bullies at bay, and to make sure that the internet is a friendly and safe space.

There are n moderators communicating with each other through a private network. The moderators are numbered 1 to n. The favorite number of moderator i is F_i.

There are $n - 1$ pairs of moderators that have direct contact with each other. These pairs have the property that there's a unique "path" from any moderator to another, where a **path** from moderator x to moderator y is defined as a sequence of *distinct* numbers $s[0], s[1], ..., s[k]$ satisfying:

- $s[0] = x$;
- $s[k] = y$;
- for each $i > 0$, moderators $s[i-1]$ and $s[i]$ have direct contact with each other.

When moderator x messages moderator y, x first sends a test message as a sort of virtual handshake. The message starts as F_x, but as it travels through the path from x to y, it gets *modded* along the way. (They're mods, after all.) Formally, the *handshake message that moderator y receives from x* is

$$(((F_{s[0]} \bmod F_{s[1]}) \bmod F_{s[2]}) \ldots) \bmod F_{s[k]}$$

where $s[0], s[1], ..., s[k]$ is the path from x to y.

You need to answer q queries. In each query, you are given two moderators x and y, and you need to find the handshake message that y receives from x.

Input Format

The first line contains a single integer n.

The second line contains n space-separated integers $F_1, F_2, ..., F_n$.

Each of the next $n-1$ lines contains two space-separated integers u and v denoting a pair of moderators that have direct contact with each other.

The next line contains a single integer q.

Each of the next q lines contains two space-separated integers x and y denoting a query.

Output Format

For each query, output a line containing an integer denoting the answer for that query.

Constraints

- $2 \le n, q \le 2 \cdot 10^5$
- $1 \le F_i \le 10^9$
- $1 \le u, v, x, y \le n$
- $u \ne v$
- $x \ne y$
- It is guaranteed that there's a unique path from any moderator to any other.

Sample

Input

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

Output

```
7
1
0
2
```

Explanation

- For the first query, the path from moderator 1 to moderator 2 is $1 \rightarrow 2$, so handshake message that moderator 2 receives from 1 is $7 \bmod 10 = 7$.
- For the second query, the path from moderator 1 to moderator 5 is $1 \rightarrow 4 \rightarrow 5$, so handshake message that moderator 5 receives from 1 is $(7 \bmod 3) \bmod 9 = 1 \bmod 9 = 1$.

E - Pairwise Perfect

Two distinct points on the Cartesian plane are said to form a **perfect pair** if the axis-aligned rectangle with those two points as opposite corners has an area that is a perfect square. Formally, two distinct points (x, y) and (X, Y) form a perfect pair if $|(x - X)(y - Y)|$ is a perfect square.

You're given an integer k ($0 \le k \le 10^6$). Find a set of **exactly 2023** points that satisfies the following properties:

- The x coordinates of the points are all distinct.
- The y coordinates of the points are all distinct.
- Exactly k pairs of distinct points form a perfect pair.

It can be shown that under the given constraints, there always exists a set of points satisfying these properties.

Notes:

- For this problem, we consider pairs to be unordered, so there are exactly $2023 \cdot 2022/2$ pairs among 2023 points.
- A number n is a perfect square iff there is an integer m such that $m^2 = n$.

Input Format

The first line contains an integer t denoting the number of test cases.

Each test case consists of a single line containing one integer k, the required number of pairs.

Output Format

For each test case, print 2024 lines.

- For $1 \le i \le 2023$, the ith line should contain two integers x_i and y_i separated by a space. This means that the ith point is (x_i, y_i).
- The 2024th line must contain the integer -1. (This doesn't represent a point and only serves as a delimiter.)

Your output must satisfy $1 \le x_i, y_i \le 2 \cdot 10^6$.

Constraints

- $1 \le t \le 100$
- $0 \le k \le 10^6$

Sample

Input

```
3
0
1
5
```

Output

```
1 2
8 3
2 7
-1
3 2
5 8
7 16
-1
2 7
8 2
11 9
9 5
5 10
4 1
7 11
15 19
-1
```

Explanation

Note that the samples merely showcase the output format, and don't actually constitute a valid output to this task—you **must** print `2023` points, always.

- In the first example, the areas of the `3` rectangles are `7·1 = 7, 1·5 = 5, 6·4 = 24`. None of them are perfect squares.
- In the second example, the areas of the `3` rectangles are `2·6 = 12, 2·8 = 16, 4·14 = 56`. Only `16` is a perfect square.

F - Palindromic Polygon

There are n points on the Cartesian plane. The ith point has coordinates (x_i, y_i) and value v_i.

No two points are at the same location, that is, for any $1 \le i < j \le n$, either $x_i \neq x_j$ or $y_i \neq y_j$.

Find any non-empty subset S of the points such that:

- The convex hull of S has positive area.
- There exists a point P lying on the convex hull boundary such that when you read the values of the points from S that lie on the convex hull boundary in clockwise order starting from P, the values form a *palindromic sequence*.

Note that when "going clockwise" around the polygon, *every* point of S that lies on the convex hull boundary must be included, and points not in S won't be included. See the examples below for further clarification.

Note:

- The **convex hull** of a set of points S is the smallest *convex* polygon containing all points in S in its interior or boundary.
- A polygon is **convex** if, for every two points P and Q inside or on the boundary of the polygon, the whole line segment PQ is also contained in the polygon.
- A sequence is **palindromic** if it reads the same forwards or backwards.

Input Format

The first line contains an integer t, the number of test cases.

Each test case consists of several lines of input.

- The first line of each test case contains n, the number of points.
- The next n lines describe the points. The ith line contains three space-separated integers x_i, y_i, v_i.

Output Format

For each test case:

- If no valid subset S exists, print -1.
- Otherwise,
  - First print a line containing a single integer k, the number of points in your chosen subset S.
  - After that, print a second line containing k **distinct** space-separated integers between 1 and n, the indices of the points chosen to be in your subset S. These can be printed in any

order.

If multiple solutions exist, you may output any of them.

Constraints

- $1 \le t \le 10^4$
- $1 \le n \le 3 \cdot 10^5$
- $1 \le x_i, y_i \le 10^9$
- $1 \le V_i \le n$
- All the input points are at distinct locations.
- The sum of n across all tests is $\le 3 \cdot 10^5$.

Sample

Input

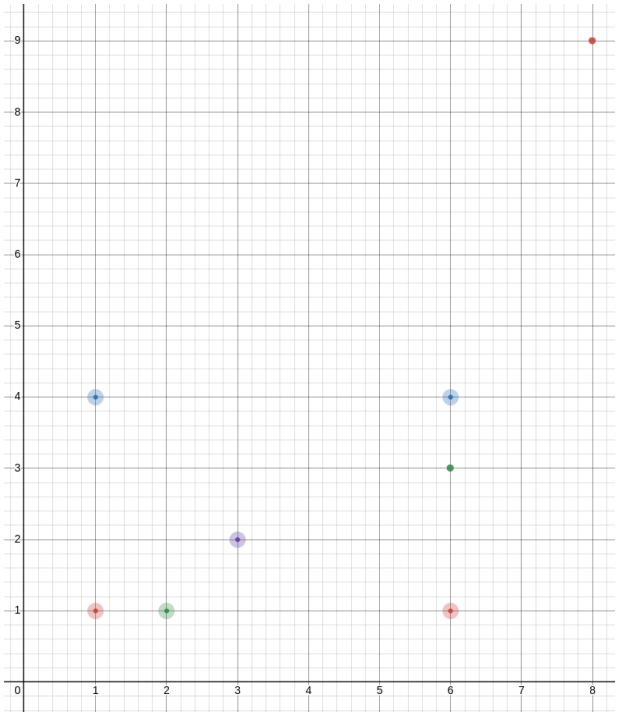
```
2
8
1 1 5
6 4 2
6 1 5
1 4 2
6 3 1
2 1 1
8 9 5
3 2 3
2
3 4 5
6 7 8
```

Output

```
6
1 2 3 4 6 8
-1
```

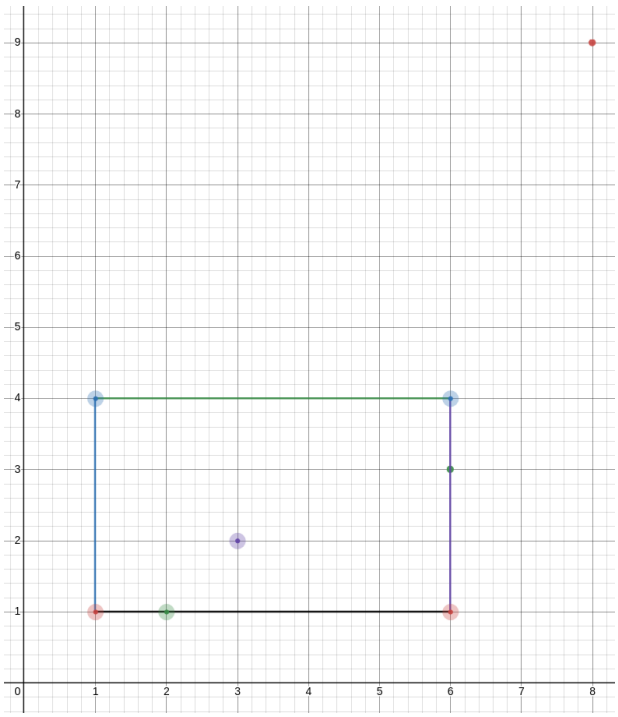
Explanation

In the first sample, the points look like this:



Points with the same color have the same V_i values, and the six highlighted points are the ones chosen to be in S.
Note that different colors might be indistinguishable on your printed copy, so refer to the version on your PC if necessary.

Their convex hull looks like this:



5 points of S lie on the convex hull (all but the one at $(3, 2)$), and it can be seen that their values are palindromic when read clockwise starting from $(6, 4)$ (with the array of values formed being

`[2, 5, 1, 5, 2]`).

Point `(6, 3)` lies on the hull of `S`, but is not taken into consideration because it doesn't lie in `S` itself.

In the second sample, the convex hull of any subset of points has zero area.

G - Protracted Parenthesization

You are given a string `S` of length `n` consisting of parenthesis characters `(` or `)`. It has an equal number of `(` and `)`. The string `S` may or may not be balanced.

While the string `S` is *not balanced*, you may perform the following operation:

- Choose two indices `i` and `j` such that `1 ≤ i < j ≤ n`, `S[i]` is a `)` and `S[j]` is a `(`.
- Swap `S[i]` and `S[j]`.

Once `S` becomes balanced, you cannot perform any more operations.

What is the **maximum** number of operations you can perform starting from `S`?

Note: A parenthesis string is *balanced* if every `(` is paired to a `)` in a natural way. Formally:

- The empty string is balanced.
- If `X` and `Y` are balanced, then `XY` is balanced.
- If `X` is balanced, then `(X)` is balanced.
- No other string is balanced, aside from those implied to be balanced by the above rules.

Input Format

The first line of input contains a single integer `t`, the number of test cases.

Each test case consists of two lines of input.

- The first line contains an integer `n`.
- The second line contains the string `S`.

Output Format

For each test case, print a line containing an integer denoting the maximum number of operations that can be performed.

Constraints

- $1 \le t \le 10^5$
- $2 \le n \le 5 \cdot 10^5$
- `n is even`
- The number of `(` and `)` characters in `S` are equal.
- The sum of `n` across all test cases is $\le 10^6$.

Sample

Input

```
3
4
()()
4
))((
6
)(()()
```

Output

```
0
3
2
```

Explanation

- For the first test case, the sequence is already balanced.
- For the second test case, at most three operations can be performed. One such sequence of three operations is `))((` → `)()(` → `())(` → `(())`.

H - Slothful Secretary

A school has n classrooms, labeled 1 to n. Each pair of rooms has exactly one corridor directly connecting them, but for ease of traffic, these corridors are *one-way* for students.

The school's secretary has to deliver a memo to the class president of each classroom (each classroom has exactly one). He has a brilliant idea that helps him avoid work—he can give *multiple* memos to some class president, and task that class president with doing the delivery for him. A class president can deliver a memo to some other classroom if they can *walk* through the corridors to reach that other classroom and then *walk* through the corridors to return to their original classroom. A class president is willing to do as many deliveries as is requested of them—again, so long as it is possible for them to return to their original classroom when all is done.

A set of class presidents is called *complete* if the secretary can distribute the memos among the class presidents in this set, and task them with deliveries such that every classroom can receive a memo. Find the minimum possible size of a complete set of class presidents.

Actually, you must process q updates. In the initial setup, for each pair (i, j) such that $1 \le i < j \le n$, the one-way corridor connecting classrooms i and j is oriented from i to j (that is, you can travel from the lower-numbered room to the higher one). Then, for each update, the direction of a one-way corridor is reversed—give the minimum possible size of a complete set of class presidents after each update.

Input Format

The first line contains two integers n and q.

The kth of the next q lines contains two integers u_k and v_k. The kth update reverses the direction of the one-way corridor connecting u_k and v_k.

Output Format

For each update, print one line containing a single integer denoting the answer right after the update.

Constraints

- $1 \le n, q \le 5 \cdot 10^5$
- $1 \le u_k < v_k \le n$

Sample

Input

```
4 5
2 3
3 4
```

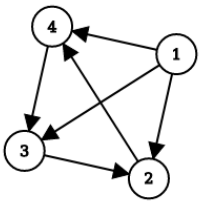
```
1 3
2 3
3 4
```

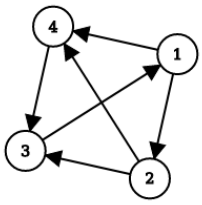
Output

```
4
2
1
1
2
```

Explanation

Please refer following images to see how the classroom pathway looks like after second and fourth updates, respectively:





- After the second update, the secretary can deliver 3 memos to the president of class 3, and 1 memo to the president of class 1. The president of class 3 can deliver memos to the presidents of classes 2 and 4 and then come back to class 3.
- After the fourth update, the secretary can deliver 4 memos to the president of class 4. The president of class 4 can deliver memos to all the other presidents and then come back to class 4.

I - Swirly Sort

You're given an array A containing n integers, and an integer k $(1 \le k \le n)$. You would like to transform it into a sorted array.

You can perform the following operations on A any number of times:

- Choose k indices $1 \le i_1 < i_2 < i_3 < \ldots < i_k \le n$ and cyclically shift the values at these indices.
  - That is, $A[i_1]$ moves to index i_2, $A[i_2]$ moves to index i_3, ..., $A[i_K]$ moves to index i_1.
  - Note that the indices you choose for the cyclic shift *must* always be increasing.
  - This operation has cost 0.
- Choose an index i, and either increment or decrement $A[i]$ by 1.
  - This operation has cost 1.

Find the minimum total cost of operations to transform the array into a sorted array.

Notes:

- The array A is one-indexed.
- An array X is sorted if $X[1] \le X[2] \le X[3] \le \ldots$.

Input Format

The first line of input contains a single integer t, denoting the number of test cases.

Each test case consists of two lines of input.

- The first line contains two space-separated integers n and k.
- The second line contains n space-separated integers $A[1], A[2], \ldots, A[n]$, the initial values of array A.

Output Format

For each test case, print a line containing an integer: the minimum cost to transform A into a sorted array.

Constraints

- $1 \le t \le 10^5$
- $1 \le n \le 3 \cdot 10^5$
- $1 \le k \le n$
- $1 \le A[i] \le 10^9$
- The sum of $n \cdot k$ across all test cases is $\le 3 \cdot 10^5$.

Sample

Input

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

Output

```
3
0
1
2
```

Explanation

In all samples, the initial array is `A = [6, 4, 3, 7]`.

- For `k = 1`, we subtract `2` from the first element and add `1` to the third element, turning the array into `A = [4, 4, 4, 7]`.
- For `k = 2`, we can choose $i_1 = 1$ and $i_2 = 3$, transforming the array into `[3, 4, 6, 7]` which is sorted. This has a cost of `0`.
- For `k = 3`, the following process is optimal:
  - Choose $i_1 = 1$, $i_2 = 2$, $i_3 = 3$. The array becomes `[3, 6, 4, 7]`.
  - Choose $i_1 = 1$, $i_2 = 2$, $i_3 = 3$ again. The array becomes `[4, 3, 6, 7]`.
  - Subtract `1` from the first element to obtain `[3, 3, 6, 7]`.
- For `k = 4`, the following is optimal:
  - Add `1` to the first element. The array becomes `[7, 4, 3, 7]`.
  - Subtract `1` from the second element. The array becomes `[7, 3, 3, 7]`.
  - Choose all four elements and cyclic shift to obtain `[7, 7, 3, 3]`.
  - Cyclic shift again and obtain `[3, 7, 7, 3]`.
  - Cyclic shift again and obtain `[3, 3, 7, 7]`.