Codeforces Round #540 (Div. 3)

A. Water Buying

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

Polycarp wants to cook a soup. To do it, he needs to buy exactly n liters of water.

There are only two types of water bottles in the nearby shop — 1-liter bottles and 2-liter bottles. There are infinitely many bottles of these two types in the shop.

The bottle of the first type costs a burles and the bottle of the second type costs b burles correspondingly.

Polycarp wants to spend as few money as possible. Your task is to find the minimum amount of money (in burles) Polycarp needs to buy exactly n liters of water in the nearby shop if the bottle of the first type costs a burles and the bottle of the second type costs b burles.

You also have to answer q independent queries.

Input

The first line of the input contains one integer q $(1 \le q \le 500)$ — the number of queries.

The next q lines contain queries. The i-th query is given as three space-separated integers n_i, a_i and b_i ($1 \le n_i \le 10^{12}, 1 \le a_i, b_i \le 1000)$ — how many liters Polycarp needs in the i-th query, the cost (in burles) of the bottle of the first type in the i-th query and the cost (in burles) of the bottle of the second type in the i-th query, respectively.

Output

Print q integers. The i-th integer should be equal to the minimum amount of money (in burles) Polycarp needs to buy exactly n_i liters of water in the nearby shop if the bottle of the first type costs a_i burles and the bottle of the second type costs b_i burles.

Example

input

```
4
10 1 3
7 3 2
1 1000 1
1000000000000 42 88
```

output

```
10
9
1000
42000000000000
```

B. Tanya and Candies

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

Tanya has n candies numbered from 1 to n. The i-th candy has the weight a_i.

She plans to eat exactly $n-1$ candies and give the remaining candy to her dad. Tanya eats candies in order of increasing their numbers, **exactly one candy per day**.

Your task is to find the number of such candies i (let's call these candies **good**) that if dad gets the i-th candy then the sum of weights of candies Tanya eats in even days will be equal to the sum of weights of candies Tanya eats in odd days. Note that at first, she will give the candy, after it she will eat the remaining candies one by one.

For example, $n=4$ and weights are $[1,4,3,3]$. Consider all possible cases to give a candy to dad:

- Tanya gives the 1-st candy to dad ($a_1=1$), the remaining candies are $[4,3,3]$. She will eat $a_2=4$ in the first day, $a_3=3$ in the second day, $a_4=3$ in the third day. So in odd days she will eat $4+3=7$ and in even days she will eat 3. Since $7 \neq 3$ this case shouldn't be counted to the answer (this candy isn't **good**).
- Tanya gives the 2-nd candy to dad ($a_2=4$), the remaining candies are $[1,3,3]$. She will eat $a_1=1$ in the first day, $a_3=3$ in the second day, $a_4=3$ in the third day. So in odd days she will eat $1+3=4$ and in even days she will eat 3. Since $4 \neq 3$ this case shouldn't be counted to the answer (this candy isn't **good**).

- Tanya gives the 3-rd candy to dad ($a_3 = 3$), the remaining candies are $[1, 4, 3]$. She will eat $a_1 = 1$ in the first day, $a_2 = 4$ in the second day, $a_4 = 3$ in the third day. So in odd days she will eat $1 + 3 = 4$ and in even days she will eat 4. Since $4 = 4$ this case **should be counted** to the answer (this candy is **good**).
- Tanya gives the 4-th candy to dad ($a_4 = 3$), the remaining candies are $[1, 4, 3]$. She will eat $a_1 = 1$ in the first day, $a_2 = 4$ in the second day, $a_3 = 3$ in the third day. So in odd days she will eat $1 + 3 = 4$ and in even days she will eat 4. Since $4 = 4$ this case **should be counted** to the answer (this candy is **good**).

In total there 2 cases which should counted (these candies are **good**), so the answer is 2.

Input

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

The second line of the input contains n integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le 10^4$), where a_i is the weight of the i-th candy.

Output

Print one integer — the number of such candies i (**good** candies) that if dad gets the i-th candy then the sum of weights of candies Tanya eats in even days will be equal to the sum of weights of candies Tanya eats in odd days.

Examples

input
```
7
5 5 4 5 5 5 6
```
output
```
2
```

input
```
8
4 8 8 7 8 4 4 5
```
output
```
2
```

input
```
9
2 3 4 2 2 3 2 2 4
```
output
```
3
```

Note

In the first example indices of **good** candies are $[1, 2]$.

In the second example indices of **good** candies are $[2, 3]$.

In the third example indices of **good** candies are $[4, 5, 9]$.

C. Palindromic Matrix

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

Let's call some square matrix with integer values in its cells *palindromic* if it doesn't change after the order of rows is reversed and it doesn't change after the order of columns is reversed.

For example, the following matrices are ***palindromic***:

$$\begin{bmatrix} 1 & 3 & 1 \\ 3 & 1 & 3 \\ 1 & 3 & 1 \end{bmatrix} \qquad \begin{bmatrix} 1 & 2 & 2 & 1 \\ 8 & 2 & 2 & 8 \\ 8 & 2 & 2 & 8 \\ 1 & 2 & 2 & 1 \end{bmatrix}$$

The following matrices are **not *palindromic*** because they change after the order of rows is reversed:

$$\begin{bmatrix}1&3&1\\3&1&3\\2&3&2\end{bmatrix}\rightarrow\begin{bmatrix}2&3&2\\3&1&3\\1&3&1\end{bmatrix}\quad\begin{bmatrix}1&8&8&9\\2&4&3&2\\1&3&4&1\\9&8&8&1\end{bmatrix}\rightarrow\begin{bmatrix}9&8&8&1\\1&3&4&1\\2&4&3&2\\1&8&8&9\end{bmatrix}$$

The following matrices are **not *palindromic*** because they change after the order of columns is reversed:

$$\begin{bmatrix}1&3&2\\3&1&3\\1&3&2\end{bmatrix}\rightarrow\begin{bmatrix}2&3&1\\3&1&3\\2&3&1\end{bmatrix}\quad\begin{bmatrix}1&2&1&9\\8&4&3&8\\8&3&4&8\\9&2&1&1\end{bmatrix}\rightarrow\begin{bmatrix}9&1&2&1\\8&3&4&8\\8&4&3&8\\1&1&2&9\end{bmatrix}$$

You are given n^2 integers. Put them into a matrix of n rows and n columns so that each number is used exactly once, each cell contains exactly one number and the resulting matrix is *palindromic*. If there are multiple answers, print any. If there is no solution, print "NO".

Input

The first line contains one integer n ($1 \le n \le 20$).

The second line contains n^2 integers $a_1, a_2, \dots, a_{n^2}$ ($1 \le a_i \le 1000$) — the numbers to put into a matrix of n rows and n columns.

Output

If it is possible to put all of the n^2 numbers into a matrix of n rows and n columns so that each number is used exactly once, each cell contains exactly one number and the resulting matrix is *palindromic*, then print "YES". Then print n lines with n space-separated numbers — the resulting matrix.

If it's impossible to construct any matrix, then print "NO".

You can print each letter in any case (upper or lower). For example, "YeS", "no" and "yES" are all acceptable.

Examples

input

```
4
1 8 8 1 2 2 2 2 2 2 2 2 1 8 8 1
```

output

```
YES
1 2 2 1
8 2 2 8
8 2 2 8
1 2 2 1
```

input

```
3
1 1 1 1 1 3 3 3 3
```

output

```
YES
1 3 1
3 1 3
1 3 1
```

input

```
4
1 2 1 9 8 4 3 8 8 3 4 8 9 2 1 1
```

output

```
NO
```

input

```
1
10
```

output

```
YES
10
```

Note

Note that there exist multiple answers for the first two examples.

D1. Coffee and Coursework (Easy version)

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

The only difference between easy and hard versions is the constraints.

Polycarp has to write a coursework. The coursework consists of m pages.

Polycarp also has n cups of coffee. The coffee in the i-th cup has a_i caffeine in it. Polycarp can drink some cups of coffee (each one no more than once). He can drink cups in **any order**. Polycarp drinks each cup **instantly** and **completely** (i.e. he cannot split any cup into several days).

Surely, courseworks are not usually being written in a single day (in a perfect world of Berland, at least). Some of them require multiple days of hard work.

Let's consider some day of Polycarp's work. Consider Polycarp drinks k cups of coffee during this day and caffeine dosages of cups Polycarp drink during this day are $a_{i_1}, a_{i_2}, \dots, a_{i_k}$. Then the first cup he drinks gives him energy to write a_{i_1} pages of coursework, the second cup gives him energy to write $max(0, a_{i_2} - 1)$ pages, the third cup gives him energy to write $max(0, a_{i_3} - 2)$ pages, ..., the k-th cup gives him energy to write $max(0, a_{i_k} - k + 1)$ pages.

If Polycarp doesn't drink coffee during some day, he cannot write coursework at all that day.

Polycarp has to finish his coursework as soon as possible (spend the minimum number of days to do it). Your task is to find out this number of days or say that it is impossible.

Input

The first line of the input contains two integers n and m ($1 \le n \le 100$, $1 \le m \le 10^4$) — the number of cups of coffee and the number of pages in the coursework.

The second line of the input contains n integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le 100$), where a_i is the caffeine dosage of coffee in the i-th cup.

Output

If it is impossible to write the coursework, print -1. Otherwise print the minimum number of days Polycarp needs to do it.

Examples

input
```
5 8
2 3 1 1 2
```
output
```
4
```

input
```
7 10
1 3 4 2 1 4 2
```
output
```
2
```

input
```
5 15
5 5 5 5 5
```
output
```
1
```

input
```
5 16
5 5 5 5 5
```
output
```
2
```

input
```
5 26
5 5 5 5 5
```
output
```
-1
```

Note

In the first example Polycarp can drink fourth cup during first day (and write 1 page), first and second cups during second day (and write $2 + (3 - 1) = 4$ pages), fifth cup during the third day (and write 2 pages) and third cup during the fourth day (and write 1 page) so the answer is 4. It is obvious that there is no way to write the coursework in three or less days in this test.

In the second example Polycarp can drink third, fourth and second cups during first day (and write $4 + (2 - 1) + (3 - 2) = 6$ pages) and sixth cup during second day (and write 4 pages) so the answer is 2. It is obvious that Polycarp cannot write the whole coursework in one day in this test.

In the third example Polycarp can drink all cups of coffee during first day and write $5 + (5 - 1) + (5 - 2) + (5 - 3) + (5 - 4) = 15$ pages of coursework.

In the fourth example Polycarp cannot drink all cups during first day and should drink one of them during the second day. So during first day he will write $5 + (5 - 1) + (5 - 2) + (5 - 3) = 14$ pages of coursework and during second day he will write 5 pages of coursework. This is enough to complete it.

In the fifth example Polycarp cannot write the whole coursework at all, even if he will drink one cup of coffee during each day, so the answer is -1.

D2. Coffee and Coursework (Hard Version)

time limit per test: 2.5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

The only difference between easy and hard versions is the constraints.

Polycarp has to write a coursework. The coursework consists of m pages.

Polycarp also has n cups of coffee. The coffee in the i-th cup Polycarp has a_i caffeine in it. Polycarp can drink some cups of coffee (each one no more than once). He can drink cups in **any order**. Polycarp drinks each cup **instantly** and **completely** (i.e. he cannot split any cup into several days).

Surely, courseworks are not being written in a single day (in a perfect world of Berland, at least).

Let's consider some day of Polycarp's work. Consider Polycarp drinks k cups of coffee during this day and caffeine dosages of cups Polycarp drink during this day are $a_{i_1}, a_{i_2}, \dots, a_{i_k}$. Then the first cup he drinks gives him energy to write a_{i_1} pages of coursework, the second cup gives him energy to write $max(0, a_{i_2} - 1)$ pages, the third cup gives him energy to write $max(0, a_{i_3} - 2)$ pages, ..., the k-th cup gives him energy to write $max(0, a_{i_k} - k + 1)$ pages.

If Polycarp doesn't drink coffee during some day, he cannot write coursework at all that day.

Polycarp has to finish his coursework as soon as possible (spend the minimum number of days to do it). Your task is to find out this number of days or say that it is impossible.

Input

The first line of the input contains two integers n and m ($1 \le n \le 2 \cdot 10^5$, $1 \le m \le 10^9$) — the number of cups of coffee and the number of pages in the coursework.

The second line of the input contains n integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le 10^9$), where a_i is the caffeine dosage of coffee in the i-th cup.

Output

If it is impossible to write the coursework, print -1. Otherwise print the minimum number of days Polycarp needs to do it.

Examples

input

```
5 8
2 3 1 1 2
```

output

```
4
```

input

```
7 10
1 3 4 2 1 4 2
```

output

```
2
```

input

```
5 15
5 5 5 5 5
```

output

```
1
```

input

```
5 16
5 5 5 5 5
```

output

```
2
```

input

```
5 26
5 5 5 5 5
```

output

```
-1
```

Note

In the first example Polycarp can drink fourth cup during first day (and write 1 page), first and second cups during second day (and write $2 + (3 - 1) = 4$ pages), fifth cup during the third day (and write 2 pages) and third cup during the fourth day (and write 1 page) so the answer is 4. It is obvious that there is no way to write the coursework in three or less days.

In the second example Polycarp can drink third, fourth and second cups during first day (and write $4 + (2 - 1) + (3 - 2) = 6$ pages) and sixth cup during second day (and write 4 pages) so the answer is 2. It is obvious that Polycarp cannot write the whole coursework in one day in this test.

In the third example Polycarp can drink all cups of coffee during first day and write $5 + (5 - 1) + (5 - 2) + (5 - 3) + (5 - 4) = 15$ pages of coursework.

In the fourth example Polycarp cannot drink all cups during first day and should drink one of them during the second day. So during first day he will write $5 + (5 - 1) + (5 - 2) + (5 - 3) = 14$ pages of coursework and during second day he will write 5 pages of coursework. This is enough to complete it.

In the fifth example Polycarp cannot write the whole coursework at all, even if he will drink one cup of coffee during each day, so the answer is -1.

E. Yet Another Ball Problem

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

The king of Berland organizes a ball! n pair are invited to the ball, they are numbered from 1 to n. Each pair consists of one man and one woman. Each dancer (either man or woman) has a monochrome costume. The color of each costume is represented by an integer from 1 to k, inclusive.

Let b_i be the color of the man's costume and g_i be the color of the woman's costume in the i-th pair. You have to choose a color for each dancer's costume (i.e. values $b_1, b_2, \dots, b_n$ and $g_1, g_2, \dots g_n$) in such a way that:

1. for every i: b_i and g_i are integers between 1 and k, inclusive;
2. there are no two completely identical pairs, i.e. no two indices i, j ($i \neq j$) such that $b_i = b_j$ and $g_i = g_j$ at the same time;
3. there is no pair such that the color of the man's costume is the same as the color of the woman's costume in this pair, i.e. $b_i \neq g_i$ for every i;
4. for each two consecutive (adjacent) pairs both man's costume colors and woman's costume colors differ, i.e. for every i from 1 to $n - 1$ the conditions $b_i \neq b_{i+1}$ and $g_i \neq g_{i+1}$ hold.

Let's take a look at the examples of bad and good color choosing (for $n = 4$ and $k = 3$, man is the first in a pair and woman is the second):

Bad color choosing:

- $(1, 2)$, $(2, 3)$, $(3, 2)$, $(1, 2)$ — contradiction with the second rule (there are equal pairs);
- $(2, 3)$, $(1, 1)$, $(3, 2)$, $(1, 3)$ — contradiction with the third rule (there is a pair with costumes of the same color);
- $(1, 2)$, $(2, 3)$, $(1, 3)$, $(2, 1)$ — contradiction with the fourth rule (there are two consecutive pairs such that colors of costumes of men/women are the same).

Good color choosing:

- $(1, 2)$, $(2, 1)$, $(1, 3)$, $(3, 1)$;
- $(1, 2)$, $(3, 1)$, $(2, 3)$, $(3, 2)$;
- $(3, 1)$, $(1, 2)$, $(2, 3)$, $(3, 2)$.

You have to find **any** suitable color choosing or say that no suitable choosing exists.

Input

The only line of the input contains two integers n and k ($2 \leq n, k \leq 2 \cdot 10^5$) — the number of pairs and the number of colors.

Output

If it is impossible to find **any** suitable colors choosing, print "NO".

Otherwise print "YES" and then the colors of the costumes of pairs in the next n lines. The i-th line should contain two integers b_i and g_i — colors of costumes of man and woman in the i-th pair, respectively.

You can print each letter in any case (upper or lower). For example, "YeS", "no" and "yES" are all acceptable.

Examples

input

```
4 3
```

output

```
YES
3 1
1 3
3 2
2 3
```

input

```
10 4
```

output

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

input

```
13 4
```

output

```
NO
```

F1. Tree Cutting (Easy Version)

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

You are given an undirected tree of n vertices.

Some vertices are colored blue, some are colored red and some are uncolored. It is guaranteed that the tree contains at least one red vertex and at least one blue vertex.

You choose an edge and remove it from the tree. Tree falls apart into two connected components. Let's call an edge *nice* if neither of the resulting components contain vertices of both red and blue colors.

How many *nice* edges are there in the given tree?

Input

The first line contains a single integer n ($2 \le n \le 3 \cdot 10^5$) — the number of vertices in the tree.

The second line contains n integers $a_1, a_2, \dots, a_n$ ($0 \le a_i \le 2$) — the colors of the vertices. $a_i = 1$ means that vertex i is colored red, $a_i = 2$ means that vertex i is colored blue and $a_i = 0$ means that vertex i is uncolored.

The i-th of the next $n-1$ lines contains two integers v_i and u_i ($1 \le v_i, u_i \le n$, $v_i \ne u_i$) — the edges of the tree. It is guaranteed that the given edges form a tree. It is guaranteed that the tree contains at least one red vertex and at least one blue vertex.

Output

Print a single integer — the number of *nice* edges in the given tree.

Examples

input

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

output

```
1
```

input

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

output

```
4
```

input

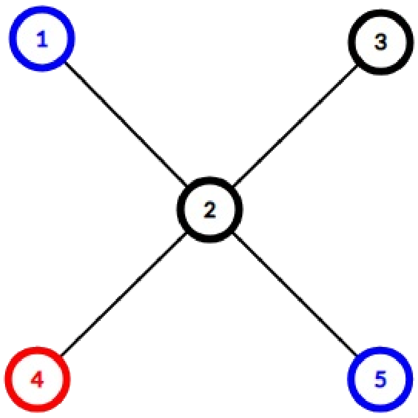
```
3
1 1 2
2 3
1 3
```

output

```
0
```

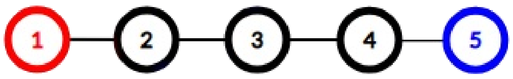
Note

Here is the tree from the first example:



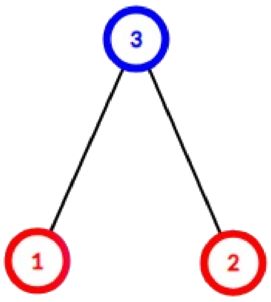
The only *nice* edge is edge $(2, 4)$. Removing it makes the tree fall apart into components $\{4\}$ and $\{1, 2, 3, 5\}$. The first component only includes a red vertex and the second component includes blue vertices and uncolored vertices.

Here is the tree from the second example:



Every edge is *nice* in it.

Here is the tree from the third example:



Edge $(1, 3)$ splits the into components $\{1\}$ and $\{3, 2\}$, the latter one includes both red and blue vertex, thus the edge isn't *nice*. Edge $(2, 3)$ splits the into components $\{1, 3\}$ and $\{2\}$, the former one includes both red and blue vertex, thus the edge also isn't *nice*. So the answer is 0.

F2. Tree Cutting (Hard Version)

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

You are given an undirected tree of n vertices.

Some vertices are colored one of the k colors, some are uncolored. It is guaranteed that the tree contains at least one vertex of each of the k colors. There might be no uncolored vertices.

You choose a subset of **exactly** $k-1$ **edges** and remove it from the tree. Tree falls apart into k connected components. Let's call this subset of edges *nice* if none of the resulting components contain vertices of different colors.

How many *nice* subsets of edges are there in the given tree? Two subsets are considered different if there is some edge that is present in one subset and absent in the other.

The answer may be large, so print it modulo 998244353.

Input

The first line contains two integers n and k ($2 \le n \le 3 \cdot 10^5$, $2 \le k \le n$) — the number of vertices in the tree and the number of colors, respectively.

The second line contains n integers $a_1, a_2, \dots, a_n$ ($0 \le a_i \le k$) — the colors of the vertices. $a_i = 0$ means that vertex i is uncolored, any other value means the vertex i is colored that color.

The i-th of the next $n-1$ lines contains two integers v_i and u_i ($1 \le v_i, u_i \le n$, $v_i \ne u_i$) — the edges of the tree. It is guaranteed that the given edges form a tree. It is guaranteed that the tree contains at least one vertex of each of the k colors. There might be no uncolored vertices.

Output

Print a single integer — the number of *nice* subsets of edges in the given tree. Two subsets are considered different if there is some edge that is present in one subset and absent in the other.

The answer may be large, so print it modulo 998244353.

Examples

input

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

output

```
1
```

input

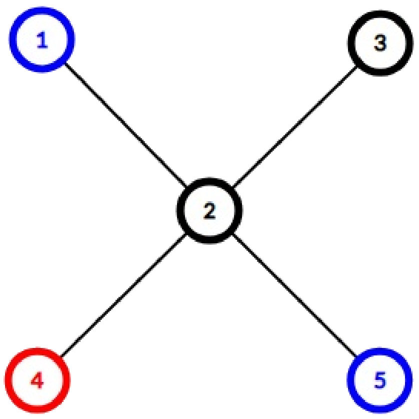
```
7 3
0 1 0 2 2 3 0
1 3
1 4
1 5
2 7
3 6
4 7
```

output

```
4
```

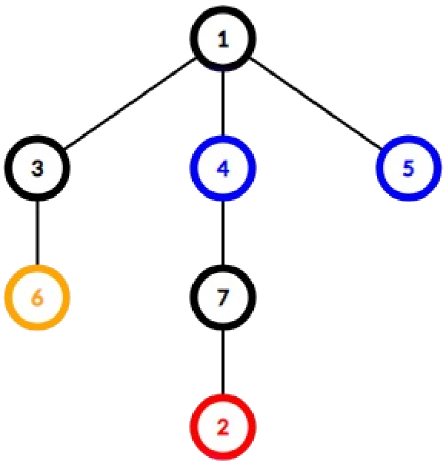
Note

Here is the tree from the first example:



The only *nice* subset is edge $(2, 4)$. Removing it makes the tree fall apart into components $\{4\}$ and $\{1, 2, 3, 5\}$. The first component only includes a vertex of color 1 and the second component includes only vertices of color 2 and uncolored vertices.

Here is the tree from the second example:



The *nice* subsets are $\{(1,3),(4,7)\}$, $\{(1,3),(7,2)\}$, $\{(3,6),(4,7)\}$ and $\{(3,6),(7,2)\}$.