C. Build Permutation

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

A **0-indexed** array a of size n is called *good* if for all valid indices i ($0 \le i \le n-1$), $a_i + i$ is a perfect square[†].

Given an integer n. Find a permutation[‡] p of $[0, 1, 2, \ldots, n-1]$ that is good or determine that no such permutation exists.

[†] An integer x is said to be a perfect square if there exists an integer y such that $x = y^2$.

[‡] An array b is a permutation of an array a if b consists of the elements of a in arbitrary order. For example, $[4, 2, 3, 4]$ is a permutation of $[3, 2, 4, 4]$ while $[1, 2, 2]$ is not a permutation of $[1, 2, 3]$.

Input

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

The only line of each test case contains a single integer n ($1 \le n \le 10^5$) — the length of the permutation p.

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

Output

For each test case, output n distinct integers $p_0, p_1, \ldots, p_{n-1}$ ($0 \le p_i \le n-1$) — the permutation p — if the answer exists, and -1 otherwise.

Example

input Copy

```
3
3
4
7
```

output Copy

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

Note

In the first test case, we have $n = 3$. The array $p = [1, 0, 2]$ is good since $1 + 0 = 1^2$, $0 + 1 = 1^2$, and $2 + 2 = 2^2$

In the second test case, we have $n = 4$. The array $p = [0, 3, 2, 1]$ is good since $0 + 0 = 0^2$, $3 + 1 = 2^2$, $2 + 2 = 2^2$, and $1 + 3 = 2^2$.

Input:

3

3

4

7

Output:

1 0 2

0 3 2 1

1 0 2 6 5 4 3

https://codeforces.com/problemset/problem/1713/C

D. 0-1-Tree

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

You are given a tree (an undirected connected acyclic graph) consisting of n vertices and $n - 1$ edges. A number is written on each edge, each number is either 0 (let's call such edges 0-edges) or 1 (those are 1-edges).

Let's call an ordered pair of vertices (x, y) ($x \neq y$) **valid** if, while traversing the simple path from x to y, we never go through a 0-edge after going through a 1-edge. Your task is to calculate the number of **valid** pairs in the tree.

Input

The first line contains one integer n ($2 \leq n \leq 200000$) — the number of vertices in the tree.

Then $n - 1$ lines follow, each denoting an edge of the tree. Each edge is represented by three integers x_i, y_i and c_i ($1 \leq x_i, y_i \leq n$, $0 \leq c_i \leq 1$, $x_i \neq y_i$) — the vertices connected by this edge and the number written on it, respectively.

It is guaranteed that the given edges form a tree.

Output

Print one integer — the number of **valid** pairs of vertices.

Example

input Copy

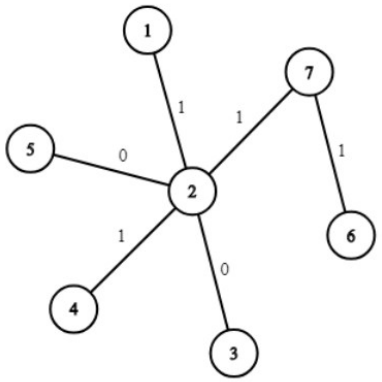
```
7
2 1 1
3 2 0
4 2 1
5 2 0
6 7 1
7 2 1
```

output Copy

```
34
```

Note

The picture corresponding to the first example:



Input:

7

2 1 1

3 2 0

4 2 1

5 2 0

6 7 1

7 2 1

Output:

1 0 2

0 3 2 1

1 0 2 6 5 4 3

https://codeforces.com/contest/1156/problem/D

B. Legacy

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

Rick and his co-workers have made a new radioactive formula and a lot of bad guys are after them. So Rick wants to give his legacy to Morty before bad guys catch them.

There are n planets in their universe numbered from 1 to n. Rick is in planet number s (the earth) and he doesn't know where Morty is. As we all know, Rick owns a portal gun. With this gun he can open one-way portal from a planet he is in to any other planet (including that planet). But there are limits on this gun because he's still using its free trial.

By default he can not open any portal by this gun. There are q plans in the website that sells these guns. Every time you purchase a plan you can only use it once but you can purchase it again if you want to use it more.

Plans on the website have three types:

1. With a plan of this type you can open a portal from planet v to planet u.
2. With a plan of this type you can open a portal from planet v to any planet with index in range $[l, r]$.
3. With a plan of this type you can open a portal from any planet with index in range $[l, r]$ to planet v.

Rick doesn't known where Morty is, but Unity is going to inform him and he wants to be prepared for when he finds and start his journey immediately. So for each planet (including earth itself) he wants to know the minimum amount of money he needs to get from earth to that planet.

Input

The first line of input contains three integers n, q and s ($1 \le n, q \le 10^5$, $1 \le s \le n$) — number of planets, number of plans and index of earth respectively.

The next q lines contain the plans. Each line starts with a number t, type of that plan ($1 \le t \le 3$). If $t = 1$ then it is followed by three integers v, u and w where w is the cost of that plan ($1 \le v, u \le n$, $1 \le w \le 10^9$). Otherwise it is followed by four integers v, l, r and w where w is the cost of that plan ($1 \le v \le n$, $1 \le l \le r \le n$, $1 \le w \le 10^9$).

Output

In the first and only line of output print n integers separated by spaces. i-th of them should be minimum money to get from earth to i-th planet, or -1 if it's impossible to get to that planet.

Examples

input Copy

```
3 5 1
2 3 2 3 17
2 3 2 2 16
2 2 2 3 3
3 3 1 1 12
1 3 3 17
```

output Copy

```
0 28 12
```

input Copy

```
4 3 1
3 4 1 3 12
2 2 3 4 10
1 2 4 16
```

output Copy

```
0 -1 -1 12
```

Note

In the first sample testcase, Rick can purchase 4th plan once and then 2nd plan in order to get to get to planet number 2.

Input:

3 5 1

2 3 2 3 17

2 3 2 2 16

2 2 2 3 3

3 3 1 1 12

1 3 3 17

Output:

0 28 12

Input:

4 3 1

3 4 1 3 12

2 2 3 4 10

1 2 4 16

Output:

0 -1 -1 12

https://codeforces.com/contest/786/problem/B

E. Easy Scheduling

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

Eonathan Eostar decided to learn the magic of multiprocessor systems. He has a full binary tree of tasks with height $$$h$$$. In the beginning, there is only one ready task in the tree — the task in the root. At each moment of time, $$$p$$$ processes choose at most $$$p$$$ ready tasks and perform them. After that, tasks whose parents were performed become ready for the next moment of time. Once the task becomes ready, it stays ready until it is performed.

You shall calculate the smallest number of time moments the system needs to perform all the tasks.

Input

The first line of the input contains the number of tests $$$t$$$ ($$$1 \leq t \leq 5\cdot 10^5$$$). Each of the next $$$t$$$ lines contains the description of a test. A test is described by two integers $$$h$$$ ($$$1 \leq h \leq 50$$$) and $$$p$$$ ($$$1 \leq p \leq 10^4$$$) — the height of the full binary tree and the number of processes. It is guaranteed that all the tests are different.

Output

For each test output one integer on a separate line — the smallest number of time moments the system needs to perform all the tasks.

Example

input Copy

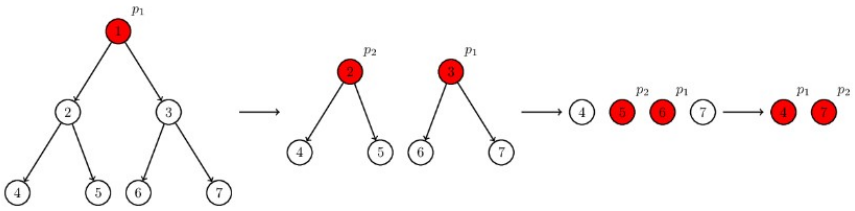
```
3
3 1
3 2
10 6
```

output Copy

```
7
4
173
```

Note

Let us consider the second test from the sample input. There is a full binary tree of height $$$3$$$ and there are two processes. At the first moment of time, there is only one ready task, $$$1$$$, and $$$p_1$$$ performs it. At the second moment of time, there are two ready tasks, $$$2$$$ and $$$3$$$, and the processes perform them. At the third moment of time, there are four ready tasks, $$$4$$$, $$$5$$$, $$$6$$$, and $$$7$$$, and $$$p_1$$$ performs $$$6$$$ and $$$p_2$$$ performs $$$5$$$. At the fourth moment of time, there are two ready tasks, $$$4$$$ and $$$7$$$, and the processes perform them. Thus, the system spends $$$4$$$ moments of time to perform all the tasks.



Input：

3

3 1

3 2

10 6

Output：

7

4

173

https://codeforces.com/problemset/problem/1578/E

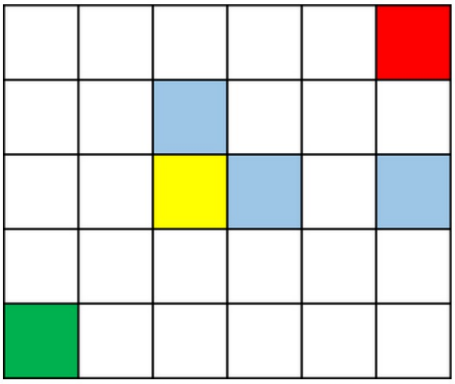
A. Chip Game

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

Burenka and Tonya are playing an old Buryat game with a chip on a board of $n \times m$ cells.

At the beginning of the game, the chip is located in the lower left corner of the board. In one move, the player can move the chip to the right or up by any **odd** number of cells (but you cannot move the chip both to the right and up in one move). The one who cannot make a move loses.

Burenka makes the first move, the players take turns. Burenka really wants to win the game, but she is too lazy to come up with a strategy, so you are invited to solve the difficult task of finding it. Name the winner of the game (it is believed that Burenka and Tonya are masters of playing with chips, so they always move in the optimal way).



Chip's starting cell is green, the only cell from which chip can't move is red. if the chip is in the yellow cell, then blue cells are all options to move the chip in one move.

Input

The first line contains one integer t ($1 \le t \le 10^4$) — the number of test cases. The following is a description of the input data sets.

The only line of each test case contains two integers n and m ($1 \le n, m \le 10^9$) — the dimensions of the game board.

Output

For each test case print a single line — the name of the winner of the game ("Burenka" or "Tonya").

Example

input

```
6
1 1
1 4
5 6
2 2
6 3
999999999 1000000000
```

output

```
Tonya
Burenka
Burenka
Tonya
Burenka
Burenka
```

Note

In the first case, Burenka has no move, so Tonya wins.

In the second case, Burenka can move 3 cells to the right, after which Tony will not be able to make a move, which means that Burenka wins.

In the third case, Burenka can move 5 squares to the right. Then we can say that we have a game on a board of 1×5 cells, and Tonya is the first player. In such game the second player wins, so in the original one Burenka will win.

Input:

6

1 1

1 4

5 6

2 2

6 3

999999999 1000000000

Output：

Tonya

Burenka

Burenka

Tonya

Burenka

Burenka

https://codeforces.com/problemset/problem/1719/A

D. Fill The Bag

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

You have a bag of size n. Also you have m boxes. The size of i-th box is a_i, where each a_i is an integer non-negative power of two.

You can divide boxes into two parts of equal size. Your goal is to fill the bag completely.

For example, if $n = 10$ and $a = [1, 1, 32]$ then you have to divide the box of size 32 into two parts of size 16, and then divide the box of size 16. So you can fill the bag with boxes of size 1, 1 and 8.

Calculate the minimum number of divisions required to fill the bag of size n.

Input

The first line contains one integer t ($1 \le t \le 1000$) — the number of test cases.

The first line of each test case contains two integers n and m ($1 \le n \le 10^{18}, 1 \le m \le 10^5$) — the size of bag and the number of boxes, respectively.

The second line of each test case contains m integers $a_1, a_2, \ldots, a_m$ ($1 \le a_i \le 10^9$) — the sizes of boxes. It is guaranteed that each a_i is a power of two.

It is also guaranteed that sum of all m over all test cases does not exceed 10^5.

Output

For each test case print one integer — the minimum number of divisions required to fill the bag of size n (or -1, if it is impossible).

Example

input Copy

```
3
10 3
1 32 1
23 4
16 1 4 1
20 5
2 1 16 1 8
```

output Copy

```
2
-1
0
```

Input：

3

10 3

1 32 1

23 4

16 1 4 1

20 5

2 1 16 1 8

Output：

2

-1

0

https://codeforces.com/problemset/problem/1303/D

C. Water the Trees

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

There are n trees in a park, numbered from 1 to n. The initial height of the i-th tree is h_i.

You want to water these trees, so they all grow to the **same** height.

The watering process goes as follows. You start watering trees at day 1. During the j-th day you can:

- Choose a tree and water it. If the day is odd (e.g. $1, 3, 5, 7, \ldots$), then the height of the tree increases by 1. If the day is even (e.g. $2, 4, 6, 8, \ldots$), then the height of the tree increases by 2.
- Or skip a day without watering any tree.

Note that you can't water more than one tree in a day.

Your task is to determine the **minimum** number of days required to water the trees so they grow to the same height.

You have to answer t independent test cases.

Input

The first line of the input contains one integer t ($1 \le t \le 2 \cdot 10^4$) — the number of test cases.

The first line of the test case contains one integer n ($1 \le n \le 3 \cdot 10^5$) — the number of trees.

The second line of the test case contains n integers $h_1, h_2, \ldots, h_n$ ($1 \le h_i \le 10^9$), where h_i is the height of the i-th tree.

It is guaranteed that the sum of n over all test cases does not exceed $3 \cdot 10^5$ ($\sum n \le 3 \cdot 10^5$).

Output

For each test case, print one integer — the **minimum** number of days required to water the trees, so they grow to the same height.

Example

input Copy

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

output Copy

```
4
3
16
```

Note

Consider the first test case of the example. The initial state of the trees is $[1, 2, 4]$.

1. During the first day, let's water the first tree, so the sequence of heights becomes $[2, 2, 4]$;
2. during the second day, let's water the second tree, so the sequence of heights becomes $[2, 4, 4]$;
3. let's skip the third day;
4. during the fourth day, let's water the first tree, so the sequence of heights becomes $[4, 4, 4]$.

Thus, the answer is 4.

Input：

3

3

1 2 4

5

4 4 3 5 5

7

2 5 4 8 3 7 4

Output：

4

3

16

https://codeforces.com/problemset/problem/1661/C

B. Optimal Reduction

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

Consider an array a of n positive integers.

You may perform the following operation:

- select two indices l and r ($1 \le l \le r \le n$), then
- decrease all elements $a_l, a_{l+1}, \ldots, a_r$ by 1.

Let's call $f(a)$ the minimum number of operations needed to change array a into an array of n zeros.

Determine if for all permutations[†] b of a, $f(a) \le f(b)$ is true.

[†] An array b is a permutation of an array a if b consists of the elements of a in arbitrary order. For example, $[4, 2, 3, 4]$ is a permutation of $[3, 2, 4, 4]$ while $[1, 2, 2]$ is not a permutation of $[1, 2, 3]$.

Input

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

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$) — description of the array a.

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

Output

For each test case, print "YES" (without quotes) if for all permutations b of a, $f(a) \le f(b)$ is true, and "NO" (without quotes) otherwise.

You can output "YES" and "NO" in any case (for example, strings "yEs", "yes" and "Yes" will be recognized as a positive response).

Example

input Copy

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

output Copy

```
YES
YES
NO
```

Note

In the first test case, we can change all elements to 0 in 5 operations. It can be shown that no permutation of $[2, 3, 5, 4]$ requires less than 5 operations to change all elements to 0.

In the third test case, we need 5 operations to change all elements to 0, while $[2, 3, 3, 1]$ only needs 3 operations.

Input:

3

4

2 3 5 4

3

1 2 3

4

3 1 3 2

Output:

YES

YES

NO

https://codeforces.com/problemset/problem/1713/B

C. Propagating tree

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

Iahub likes trees very much. Recently he discovered an interesting tree named propagating tree. The tree consists of n nodes numbered from 1 to n, each node i having an initial value a_i. The root of the tree is node 1.

This tree has a special property: when a value val is added to a value of node i, the value -val is added to values of all the children of node i. Note that when you add value -val to a child of node i, you also add -(-val) to all children of the child of node i and so on. Look an example explanation to understand better how it works.

This tree supports two types of queries:

- "$1\ x\ val$" — val is added to the value of node x;
- "$2\ x$" — print the current value of node x.

In order to help Iahub understand the tree better, you must answer m queries of the preceding type.

Input

The first line contains two integers n and m ($1 \le n, m \le 200000$). The second line contains n integers $a_1, a_2, ..., a_n$ ($1 \le a_i \le 1000$). Each of the next $n-1$ lines contains two integers v_i and u_i ($1 \le v_i, u_i \le n$), meaning that there is an edge between nodes v_i and u_i.

Each of the next m lines contains a query in the format described above. It is guaranteed that the following constraints hold for all queries: $1 \le x \le n$, $1 \le val \le 1000$.

Output

For each query of type two (print the value of node x) you must print the answer to the query on a separate line. The queries must be answered in the order given in the input.

Examples

input Copy

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

output Copy

```
3
3
0
```

Note

The values of the nodes are $[1, 2, 1, 1, 2]$ at the beginning.

Then value 3 is added to node 2. It propagates and value -3 is added to it's sons, node 4 and node 5. Then it cannot propagate any more. So the values of the nodes are $[1, 5, 1, -2, -1]$.

Then value 2 is added to node 1. It propagates and value -2 is added to it's sons, node 2 and node 3. From node 2 it propagates again, adding value 2 to it's sons, node 4 and node 5. Node 3 has no sons, so it cannot propagate from there. The values of the nodes are $[3, 3, -1, 0, 1]$.

You can see all the definitions about the tree at the following link: `http://en.wikipedia.org/wiki/Tree_(graph_theory)`

Input:

5 5

1 2 1 1 2

1 2

1 3

2 4

2 5

1 2 3

1 1 2

2 1

2 2

2 4

Output：

3

3

0

https://codeforces.com/problemset/problem/383/C

C. Strange Function

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

Let $f(i)$ denote the minimum positive integer x such that x is **not** a divisor of i.

Compute $\sum_{i=1}^{n} f(i)$ modulo 10^9+7. In other words, compute $f(1)+f(2)+\cdots+f(n)$ modulo 10^9+7.

Input

The first line contains a single integer t ($1 \le t \le 10^4$), the number of test cases. Then t cases follow.

The only line of each test case contains a single integer n ($1 \le n \le 10^{16}$).

Output

For each test case, output a single integer ans, where $ans = \sum_{i=1}^{n} f(i)$ modulo 10^9+7.

Example

input Copy

```
6
1
2
3
4
10
10000000000000000
```

output Copy

```
2
5
7
10
26
366580019
```

Note

In the fourth test case $n=4$, so $ans = f(1)+f(2)+f(3)+f(4)$.

- 1 is a divisor of 1 but 2 isn't, so 2 is the minimum positive integer that isn't a divisor of 1. Thus, $f(1)=2$.
- 1 and 2 are divisors of 2 but 3 isn't, so 3 is the minimum positive integer that isn't a divisor of 2. Thus, $f(2)=3$.
- 1 is a divisor of 3 but 2 isn't, so 2 is the minimum positive integer that isn't a divisor of 3. Thus, $f(3)=2$.
- 1 and 2 are divisors of 4 but 3 isn't, so 3 is the minimum positive integer that isn't a divisor of 4. Thus, $f(4)=3$.

Therefore, $ans = f(1)+f(2)+f(3)+f(4) = 2+3+2+3 = 10$.

Input：

6

1

2

3

4

10

10000000000000000

Output：

2

5

7

10

26

366580019

https://codeforces.com/problemset/problem/1542/C

D. Cut

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

This time Baby Ehab will only cut and not stick. He starts with a piece of paper with an array a of length n written on it, and then he does the following:

- he picks a range (l, r) and cuts the subsegment $a_l, a_{l+1}, \ldots, a_r$ out, removing the rest of the array.
- he then cuts this range into multiple subranges.
- to add a number theory spice to it, he requires that the elements of every subrange must have their product equal to their least common multiple (LCM).

Formally, he partitions the elements of $a_l, a_{l+1}, \ldots, a_r$ into contiguous subarrays such that the product of every subarray is equal to its LCM. Now, for q independent ranges (l, r), tell Baby Ehab the minimum number of subarrays he needs.

Input

The first line contains 2 integers n and q ($1 \le n, q \le 10^5$) — the length of the array a and the number of queries.

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

Each of the next q lines contains 2 integers l and r ($1 \le l \le r \le n$) — the endpoints of this query's interval.

Output

For each query, print its answer on a new line.

Example

input Copy

```
6 3
2 3 10 7 5 14
1 6
2 4
3 5
```

output Copy

```
3
1
2
```

Note

The first query asks about the whole array. You can partition it into $[2]$, $[3, 10, 7]$, and $[5, 14]$. The first subrange has product and LCM equal to 2. The second has product and LCM equal to 210. And the third has product and LCM equal to 70. Another possible partitioning is $[2, 3]$, $[10, 7]$, and $[5, 14]$.

The second query asks about the range $(2, 4)$. Its product is equal to its LCM, so you don't need to partition it further.

The last query asks about the range $(3, 5)$. You can partition it into $[10, 7]$ and $[5]$.

Input：

6 3

2 3 10 7 5 14

1 6

2 4

3 5

Output：

3

1

2

https://codeforces.com/problemset/problem/1516/D

B. Permutation

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

Recall that a permutation of length n is an array where each element from 1 to n occurs exactly once.

For a fixed positive integer d, let's define the cost of the permutation p of length n as the number of indices i $(1 \le i < n)$ such that $p_i \cdot d = p_{i+1}$.

For example, if $d = 3$ and $p = [5, 2, 6, 7, 1, 3, 4]$, then the cost of such a permutation is 2, because $p_2 \cdot 3 = p_3$ and $p_5 \cdot 3 = p_6$.

Your task is the following one: for a given value n, find the permutation of length n and the value d with maximum possible cost (over all ways to choose the permutation and d). If there are multiple answers, then print any of them.

Input

The first line contains a single integer t $(1 \le t \le 500)$ — the number of test cases.

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

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

Output

For each test case, print the value d in the first line, and n integers in the second line — the permutation itself. If there are multiple answers, then print any of them.

Example

input Copy

```
2
2
3
```

output Copy

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

Input：

2

2

3

Output：

2

1 2

3

2 1 3

https://codeforces.com/problemset/problem/1701/B