COCI 2016/2017

Round #6, February 4th, 2017

Tasks

Task	Time limit	Memory limit	Score
Hindeks	1 s	64 MB	50
Telefoni	1 s	64 MB	80
Turnir	3 s	64 MB	100
Savrsen	3 s	128 MB	120
Sirni	5 s	768 MB	140
Gauss	2 s	256 MB	160
Total			650

How do we evaluate the success of a scientist? By the number of published papers or by their impact - more precisely, the number of citations? Both elements matter. We say that a scientific paper has a **citation score** C if other scientists cited the paper in question in their paper (referred to it) a total of C times. One of the possible metrics of the success of scientists is their **h-index** that takes into account both the amount of papers and their citation scores.

A scientist's h-index is defined as the largest number H with the following properties: the scientist can choose H papers such that their citation score is at least H. For example, if a scientist wrote 10 papers such that each of them has been cited 10 or more times, their h-index is (at least) 10.

Write a programme that inputs the citation scores of all papers of a given scientist and outputs their h-index.

INPUT

The first line of input contains the positive integer N ($1 \le N \le 500\,000$), the number of papers of a given scientist.
The following line contains N non-negative integers from the interval [0, 1 000 000], the citation scores of the respective papers.

OUTPUT

The first and only line of output must contain the required h-index.

SAMPLE TESTS

input

```
5
1 1 4 8 1
```

output

```
2
```

input

```
5
8 5 3 4 10
```

output

```
4
```

Clarification of the first test case: The scientist has two papers with citation scores larger than or equal to 2 (the papers with citation scores 4 and 8).
Clarification of the second test case: The scientist has four papers with citation scores larger than or equal to 4 (the papers with citation scores 8, 5, 4 and 10).

There are N desks in a room, placed from left to right, one next to each other. Some desks have phones on them, whereas some desks are empty. All phones are broken, so the phone on the i^{th} desk will ring if the phone at j^{th} desk rings, which is at most D desks away from the i^{th} desk. In other words, it holds $|j - i| \le D$. The first and the last desk will always have a phone on them. In the beginning the leftmost phone rings. What is the minimal amount of new phones to be placed on the desks so that the last phone rings?

INPUT

The first line of input contains two positive integers, N ($1 \le N \le 300\,000$) and D ($1 \le D \le N$). The following line contains N numbers 0 or 1. If the i^{th} number is 1, then the i^{th} desk from the left has a phone on it, otherwise the i^{th} desk is empty.

OUTPUT

The first and only line of output must contain the required minimal number of phones.

SCORING

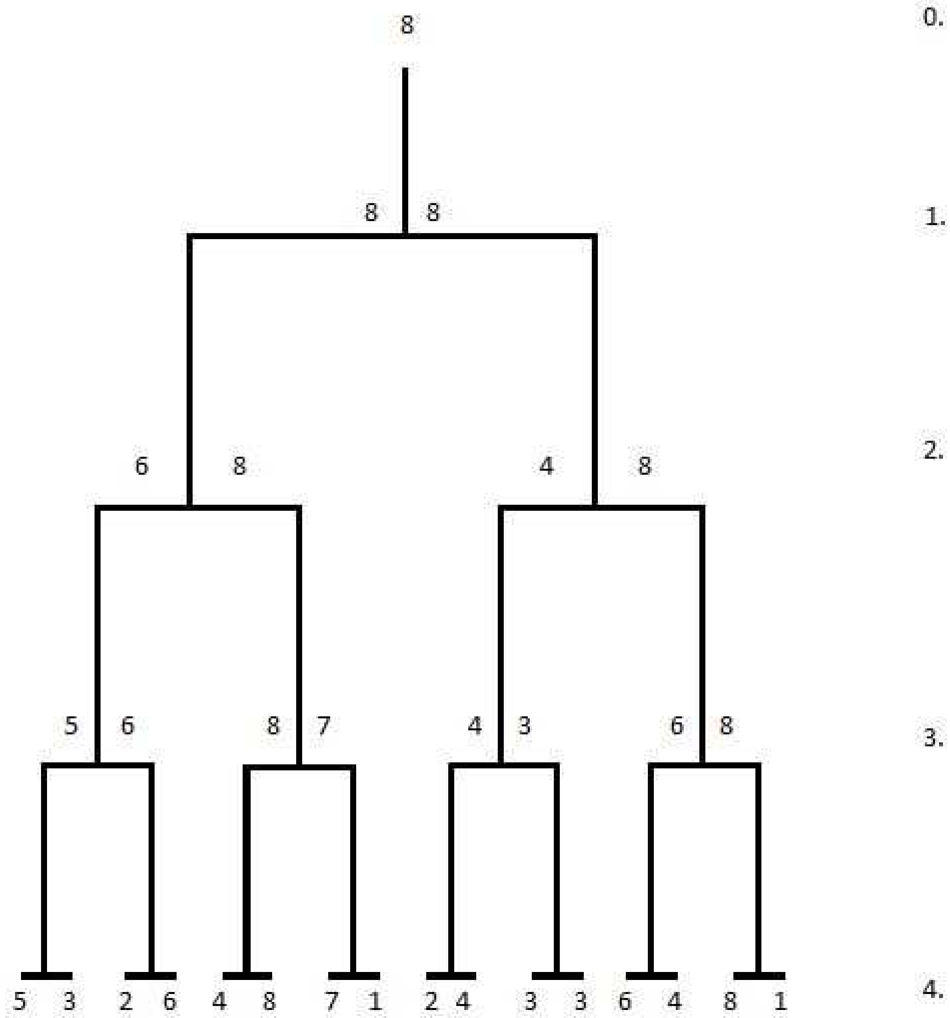
In test cases worth 40 points in total, it will hold $1 \le N \le 20$.

SAMPLE TESTS

input	input	input
`4 1` `1 0 1 1`	`5 2` `1 0 0 0 1`	`8 2` `1 1 0 0 1 0 0 1`
output	**output**	**output**
`1`	`1`	`2`

Young Jozef was given a set consisting of 2^N positive integers as a gift. Considering the fact that Jozef often takes part in football tournaments, he decided to organize a tournament for his 2^N positive integers.

The numbers tournament is depicted below; the tournament takes place in pairs, where the higher of two numbers advances to the upper level. The levels are denoted with numbers from 1 to *N*, where the highest level is given the number 0.



Since Jozef doesn't have time to organize all tournaments, he wants to know, for each number from the initial set, the highest level (the smallest level number) at which the number can end up in, for any permutation of the input array.

INPUT

The first line of input contains the positive integer *N* ($1 \le N \le 20$).
The following line contains 2^N positive integers from the interval $[1, 10^9]$, the elements of the set.

OUTPUT

The first and only line of output must contain 2^N numbers, the labels of the highest level (the smallest level labels) at which a number can end up in, in the order the numbers were given in the input.

SAMPLE TESTS

input	input	input
2 1 4 3 2	4 5 3 2 6 4 8 7 1 2 4 3 3 6 4 8 1	1 1 1
output	**output**	**output**
2 0 1 1	1 2 2 1 1 0 1 3 2 1 2 2 1 1 0 3	0 0

A number is **perfect** if it is equal to the sum of its divisors, the ones that are smaller than it. For example, number 28 is perfect because 28 = 1 + 2 + 4 + 7 + 14.

Motivated by this definition, we introduce the metric of **imperfection** of number *N*, denoted with *f(N)*, as the absolute difference between *N* and the sum of its divisors less than *N*. It follows that perfect numbers' imperfection score is 0, and the rest of natural numbers have a higher imperfection score. For example:

- f(6) = |6 - 1 - 2 - 3| = 0,
- f(11) = |11 - 1| = 10,
- f(24) = |24 - 1 - 2 - 3 - 4 - 6 - 8 - 12| = |-12| = 12.

Write a programme that, for positive integers *A* and *B,* calculates the sum of imperfections of all numbers between *A* and *B*: *f(A) + f(A + 1) + ... + f(B)*.

INPUT

The first line of input contains the positive integers *A* and *B* ($1 \le A \le B \le 10^7$).

OUTPUT

The first and only line of output must contain the required sum.

SAMPLE TESTS

input	input
`1 9`	`24 24`
output	**output**
`21`	`12`

Clarification of the first test case: 1 + 1 + 2 + 1 + 4 + 0 + 6 + 1 + 5.

Little Daniel has a bag of candy and *N* cards.

Each of the cards has a positive integer P_i written on it. While Daniel was eating his candy, he thought of a fun game. He can tie together two cards with labels *a* and *b*, and then he must eat $min(P_a \% P_b, P_b \% P_a)$ of candy, where operation $x \% y$ denotes the remainder when dividing x with y.

He wants to tie together pairs of cards in a way that, when he lifts one of them, all the rest are also lifted up. Each card can be directly connected with a tie to any number of other cards. As Daniel is watching his figure, he doesn't want to consume too much, so he is asking you to calculate the minimal number of candy he must eat so all cards are connected.

INPUT

The first line of input contains the positive integer *N*. $(1 \le N \le 10^5)$
Each of the following *N* lines contains a positive integer P_i $(1 \le P_i \le 10^7)$.

OUTPUT

The first and only line of output must contain the required value from the task.

SCORING

In test cases worth 30% of total points, it will hold $N \le 10^3$.
In test cases worth 40% of total points, it will hold $P_i \le 10^6$.
In test cases worth 70% of total points, at least one of the two conditions will hold.

SAMPLE TESTS

input	input	input
4 2 6 3 11	4 1 2 3 4	3 4 9 15
output	**output**	**output**
1	0	4

Clarification of the first test case:
Daniel can connect the first and second card and eat 0 candy, the second and third and eat 0 candy, and the first and fourth and eat 1 candy.

It is a little-known story that the young Carl Friedrich Gauss was restless in class, so his teacher came up with a task to keep him preoccupied.

The teacher gave him a series of positive integers F(1), F(2), …, F(K). We consider F(t) = 0 for t > K. Aditionally, she has given him a set of lucky numbers and the price of each lucky number. If X is a lucky number, then C(X) denotes its price.

Initially, there's a positive integer A written on the board. In each move, Carl must make one of the following things:

- If number N is currently written on the board, then Carl can write one of its divisors M, smaller than N, instead of N. If he writes the number M, the price of the move is F(d(N / M)), where d(N/M) is the number of divisors of the positive integer N/M (inlucluding N/M).
- If N is a lucky number, Carl can leave that number on the board, and the price of the move is C(N).

Carl must make **exactly** L moves, and after he has made all of his moves, the number B must be written on the board. Let's denote G(A, B, L) as the minimal price with which Carl can achieve this.
If it is not possible to make L such moves, we define G(A, B, L) = -1.

The teacher has given Carl Q queries. In each query, Carl gets numbers A and B and must calculate the value G(A, B, L_1) + G(A, B, L_2) + … + G(A, B, L_M), where numbers L_1, …, L_M are the same for all queries.

INPUT

The first line of input contains the positive integer K (1 ≤ K ≤ 10 000).
The second line contains K positive integers F(1), F(2), …, F(K) that are less than or equal to 1 000.
The following line contains the positive integer M (1 ≤ M ≤ 1 000).
The following line contains M positive integers L_1, L_2, …, L_M that are less than or equal to 10 000.
The following line contains the positive integer T, the total number of lucky numbers (1 ≤ T ≤ 50).
Each of the following T lines contains numbers X and C(X) that denote that X is a lucky number, and C(X) is his price (1 ≤ X ≤ 1 000 000, 1 ≤ C(X) ≤ 1 000).
Each lucky number appears at most once.
The following line contains the positive integer Q (1 ≤ Q ≤ 50 000).
Each of the following Q lines contains 2 positive integers A and B (1 ≤ A, B ≤ 1 000 000).

OUTPUT

You must output Q lines. The i^{th} line contains the answer to the i^{th} query defined in the task.

SAMPLE TESTS

input

```
4
1 1 1 1
2
1 2
2
2 5
4 10
1
4 2
```

output

```
7
```

input

```
3
6 9 4
2
5 7
3
1 1
7 8
6 10
2
6 2
70 68
```

output

```
118
-2
```

input

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

output

```
16
66
```

Clarification of the first test case:

L_1 = 1, so Carl can make exactly one move - replace number 4 with number 2, so G(4, 2, 1) = F(d(2)) = 1.

L_2 = 2 so Carl has two options:

- He can replace number 4 with number 2 and then leave number 2 (because it's a lucky number), so he pays the price F(d(4/2)) + C(2) = 1 + 5 = 6
- He can leave number 4 in the first move, and replace it in the second move with number 2, so the price is C(4) + F(d(4/2)) = 10 + 1 = 11

The first option costs less, so G(4, 2, 2) = 6.

The answer to the query is G(4,2,1) + G(4,2,2) = 7.