1070. Fill the line

n segments are painted on the line. The coordinates of the left and right ends of each segment l_i and r_i are known. Find the length of the colored part of the line.

Input. The first line contains the number n ($1 \le n \le 15000$), the following n lines contains the pair of integers l_i and r_i ($-10^9 \le l_i \le r_i \le 10^9$).

Output. Print the length of the colored part of the line.

Sample input	Sample output
5 6 8 1 2 0 3 7 9 2 4	7

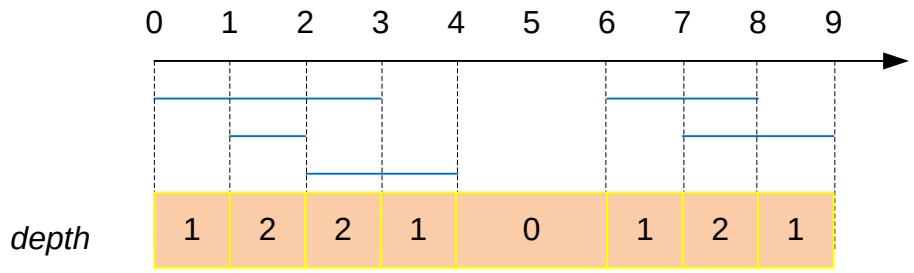
Algorithm analysis

Store the points – the ends of n segments into the array v, marking which end (left or right) they are. Then sort $2 * n$ points by the abscissa v[i]. Now move from left to right along the intervals (v[i], v[i + 1]) between the points (we'll organize the movement of the sweeping line). Keep the variable *depth* equal to the number of segments covering the interval (v[i], v[i + 1]). Initially, set it to 0. The value *depth* will be increased by 1 if the point is the start of segment, and decreased by 1 if the point is the end of segment.

The length of the colored part of the straight line equals to the sum of the lengths of the intervals $x_{i+1} - x_i$, where *depth* is not zero.

Example

Consider a set of the following line segments:



The answer is the sum of the lengths of the intervals $x_{i+1} - x_i$, where *depth* is not zero.

853. Sereja and Array

Sereja has got an array, consisting of n integers $a_1, a_2, ..., a_n$. Sereja is an active boy, so he is now going to complete m operations. Each operation will have one of the three forms:

- Make v_i-th array element equal to x_i. In other words, perform the assignment $a_{vi} = x_i$.
- Increase each array element by y_i. In other words, perform n assignments $a_i = a_i + y_i$ $(1 \le i \le n)$.
- Take a piece of paper and write out the q_i-th array element. That is, the element a_{qi}.

Help Sereja, complete all his operations.

Input. The first line contains integers n, m $(1 \le n, m \le 10^5)$. The second line contains n integers $a_1, a_2, ..., a_n$ $(1 \le a_i \le 10^9)$ – the original array.

Next m lines describe operations, the i-th line describes the i-th operation. The first number in the i-th line is integer t_i $(1 \le t_i \le 3)$, that represents the operation type. If $t_i = 1$, then it is followed by two integers v_i and x_i $(1 \le v_i \le n, 1 \le x_i \le 10^9)$. If $t_i = 2$, then it is followed by integer y_i $(1 \le y_i \le 10^4)$. And if $t_i = 3$, then it is followed by integer q_i $(1 \le q_i \le n)$.

Output. For each third type operation print value a_{qi}. Print the values in the order, in which the corresponding queries follow in the input.

Sample input 1

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

Sample output 1

```
1
5
15
```

Sample input 2

```
10 11
1 2 3 4 5 6 7 8 9 10
3 2
3 9
2 10
3 1
3 10
1 1 10
2 10
2 10
```

Sample output 2

```
2
9
11
20
30
40
39
```

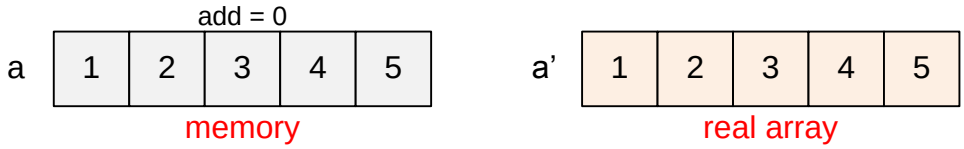
```
3 1
3 10
3 9
```

Algorithm analysis

Read the input array a. Create a variable add, initialize it to zero. When performing an operation of the second type (increasing each element of the array by y_i), add add += y_i. Thus, in the absence of operations of the first type (assignments $a_{vi} = x_i$), the final state of the element a_i will be equal to the initial state of a_i plus add. Then, to preserve the last property in the case of the first operation, we must assign $a_{vi} = x_i - add$.

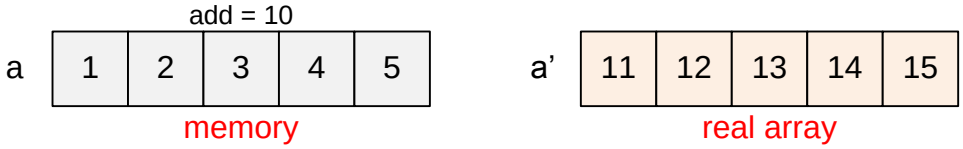
Example

Let's look at the first example. Let a_i be the state of the array in memory, a_i' be the real state of the array. Initial state of the array:



Since $a_i' = a_i + add$, but initially $add = 0$, then $a_i' = a_i$. That is, initially physically each element of the array equals to itself.

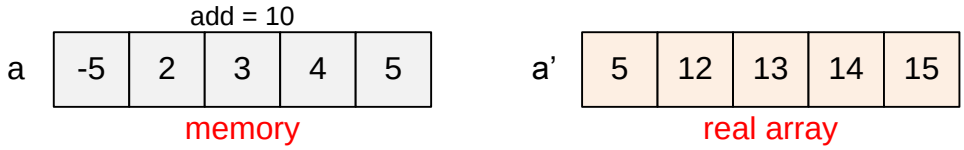
Add 10 to all the numbers in the array, $add = add + 10 = 10$. The following relation holds: $a_i' = a_i + add$.



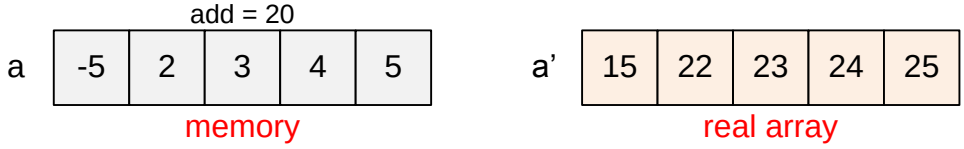
The next query: $a_1' = 5$. Physically you should make the assignment

$$a_1 = 5 - add,$$

after which the array will take the form:



Next query is to print the first element. We get $a_1' = a_1 + add = -5 + 10 = 5$.
Again add 10 to all the numbers of array: $add = add + 10 = 20$.



First element equals to $a_1' = a_1 + add = -5 + 20 = 15$.

5071. Roy and Coin Boxes

Roy has *n* coin boxes numbered from 1 to *n*. Every day he selects two indices [*l*, *r*] and adds 1 coin to each coin box starting from *l* to *r* (both inclusive). He does this for *m* number of days.

After *m* days, Roy has a query: How many coin boxes have at least *x* coins. He has *q* such queries.

Input. First line contains number of coin boxes *n* ($1 \le n \le 10^6$). Second line contains number of days *m* ($1 \le m \le 10^6$). Each of the next m lines consists of two space separated integers *l* and *r* ($1 \le l \le r \le n$). Followed number is the number of queries *q* ($1 \le q \le 10^6$). Each of next q lines contain a single integer *x* ($1 \le x \le n$).

Output. For each query output the result in a new line.

Sample input	Sample output
7 4 1 3 2 5 1 2 5 6 4 1 7 4 2	6 0 0 4

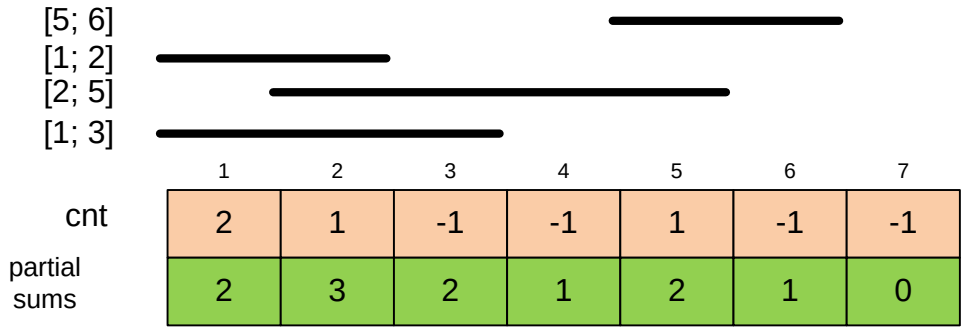
Algorithm analysis

Let's create an array – counter *cnt*. For each request [*l*, *r*], we'll perform two operations: cnt[*l*]++ and cnt[*r* + 1]--. Thus, the partial sum $\sum_{j=1}^{i} cnt[j]$ equals to the number of coins in the *i*-th coin box.

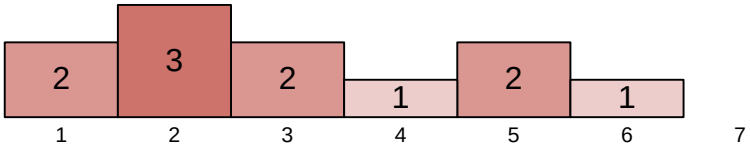
The number of days that Roy performs actions is no more than 10^6. Each day he puts no more than one coin in each coin box. After completing all the actions, each coin box will contain no more than 10^6 coins. Create an array of coins of size 10^6 and assgn to coins[*i*] the number of coin boxes with *i* coins. To do this, for each partial sum *sum* = $\sum_{j=1}^{i} cnt[j]$ execute coins[*sum*]++.

Now let's find the number of coin boxes that contain at least *x* coins. To do this, in the array coins, compute its partial sums from right to left (that is, from the end). Then the number of coin boxes that contain at least *x* coins equals to coins[*x*].

Example



We get the following distribution of coins by coin boxes:



The number of coin boxes with the sum of *s*. Partial sums (counted from right to left) contain the number of coin boxes with at least *s* coins.

s	0	1	2	3
coin[s]	1	2	3	1
partial sums	7	6	4	1

There is one coin box with sum 0 (coin box number 7), two coin boxes with sum 1 (coin boxes with numbers 4 and 6), three coin boxes with sum 2 (coin boxes with numbers 1, 3 and 5), one coin box with sum 3 (coin box number 2).

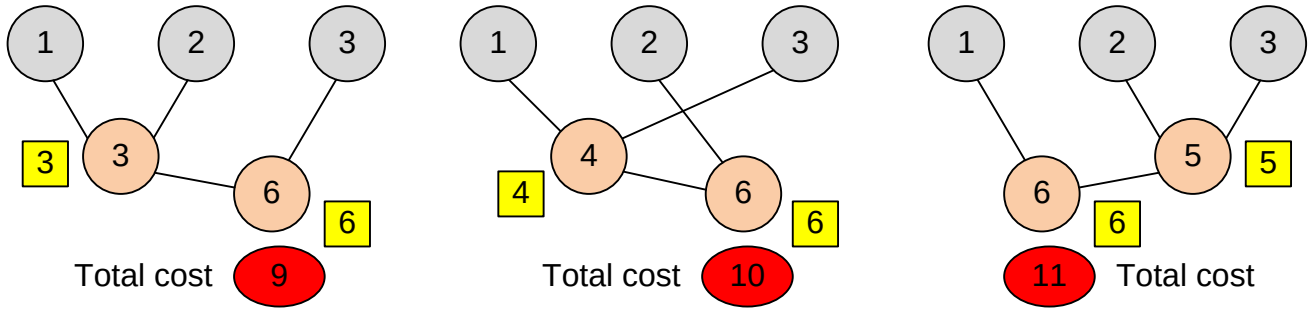
After computing the partial sums of the coins array, it can be stated, for example, that there are 6 coin boxes contain at least 1 coin. Or that there are 4 coin boxes contain at least 2 coins.

1228. Add All

The cost of adding two numbers equals to their sum. For example to add 1 and 10 cost 11. The cost of addition 1 and 2 is 3. We can add numbers in several ways:

- 1 + 2 = 3 (cost = 3), 3 + 3 = 6 (cost = 6), Total = 9
- 1 + 3 = 4 (cost = 4), 2 + 4 = 6 (cost = 6), Total = 10
- 2 + 3 = 5 (cost = 5), 1 + 5 = 6 (cost = 6), Total = 11

We hope you understood the task. You must add all numbers so that the total cost of summation will be the smallest.



Input. First line contains positive integer n ($2 \le n \le 10^5$). Second line contains n nonnegative integers, each no more than 10^5.

Output. Print the minimum total cost of summation.

Sample input	**Sample output**
4 1 2 3 4	19

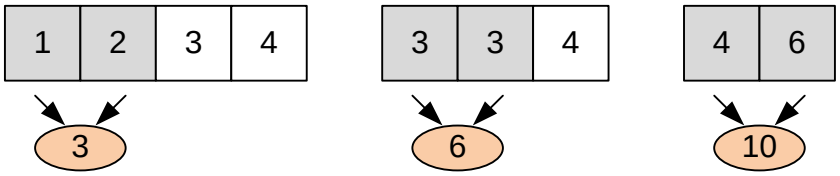
Algorithm analysis

Add the smallest two numbers each time. Then the total cost of summation for all n integers will be the minimum. Since numbers can be repeated, will store them in a multiset.

Example

To minimize the cost of addition, add the numbers in the following order:

1. Add 1 and 2, sum is 3. Cost of addition is 3.
2. Add 3 and 3, sum is 6. Cost of addition is 6.
3. Add 4 and 6, sum is 10. Cost of addition is 10.



Total cost of summation is 3 + 6 + 10 = 19.

4359. Sweets for mushrooms

It seems strange, but mushrooms like soda. Moreover, Michael likes to make experiments with it. Each kind of soda has it's own level of sweetness. Michael has n containers with soda of different levels. If Michael mix two containers with levels x and y, instead of these two he will get soda with level 2 * min(x, y).

Help Michael to get soda with the highest possible level of sweetness.

Input. The first line contains the number of containers n ($1 \le n \le 10^6$). The second line contains n integers: the level of sweetness x_i ($-10^9 \le x_i \le 10^9$).

Output. Print the highest possible level of sweetness that can be obtained by mixing some of the available sodas.

Sample input	Sample output
3 1 3 6	6

Algorithm analysis

Add all containers to the multiset (containers with the same sweetness level may appear during the mixing process). It makes sense to mix water with sweets x and y if the level of resulting sweetness is reater than $\max(x, y)$.

Consider two waters with the lowest levels of sweetness x and y.

- If $2x \le y$, then after mixing them you get water with a level of sweetness $2 * \min(x, y) = 2x$, which is not more than y. In this case, there is no point in mixing: we will remove the sweetness x from the multiset and the corresponding capacity will not be considered further.
- If $2x > y$, then after mixing them you get water with a level of sweetness $2 * \min(x, y) = 2x$, which is more than y. Remove x and y sweets from the multiset and add sweets $2x$.

Perform the described operation with the two smallest sweets x and y while the multiset contains more than one element.

3004. Queue

In civilized countries k ticket offices are working at the train station, but the queue to them is just one. The service works as follows. Initially, when all the ticket offices are free, the first k people from the queue go to the offices. The others are waiting their turn. As soon as someone is served and the corresponding office becomes free, the next person in the queue comes to this office. This continues until all the people will be served.

Find the time to serve all the clients.

Input. The first line contains two integers: the queue size n and the number of ticket offices k ($1 \le n \le 10^5$, $1 \le k \le 10^4$). n positive integers are given in the second line. The i-th number gives the time t_i ($1 \le t_i \le 10^5$) to serve the i-th client in the queue.

Output. Print one integer – the time to serve all the people in the queue.

Sample input

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

Sample output

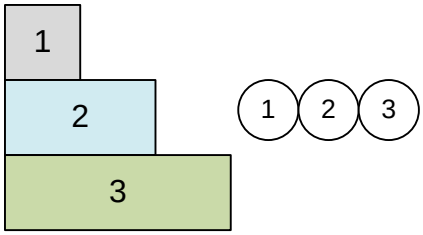
```
7
```

Algorithm analysis

Lets simulate the process of selling tickets using the multiset *s*. Bring the first *k* people to free cash desks and store their service time in the multiset. During further processing, the multiset will contain *k* elements. Each value in multiset reflects the time moment when the corresponding cash register will become free and the next person will be able to approach it. Obviously, each time a new person must come to the cash register for which this time is minimal. The time of the last served client will be the desired one.

Example

Consider the sample given. Put $k = 3$ first elements to the heap.



Then, at each iteration, we take the next element (the next person from the queue) and put it instead of the smallest one (we put the person to the ticket office that will be released earlier). We put to the queue the time at which this new person leaves the ticket office. Next to each figure the numbers in the heap are given.