Lab 8 - Advanced Tree

#A

Description

You need to implement a heap.

There are Q queries, each of the following 3 forms:

1 x: insert a integer x.

2: print the smallest number.

3: delete the smallest number.

Input

Q

Q queries.

Output

When a query of type 2 appears, output a line as its answer.

Sample Test Data

Input #1

```
6
1 7
1 8
1 3
2
3
2
```

Output #1

```
3
7
```

Data Limit

$2 \leq N \leq 1000000$

x is in the range of `int`.

#B

Description

You need to implement a naive binary search tree without any operations to maintain the balance of the tree.

There are Q queries, each of the following 3 forms:

1 x: insert a integer x.

2 k: print the k-th smallest number.

3: print all numbers that have been inserted so far in ascending order.

Finally, you need to print the preorder traversal of the naive BST. (This answer will be different if the tree you implemented is not naive BST)

Input

Q

Q queries.

Output

When a query of type 2 or 3 appears, output a line as its answer.

Sample Test Data

Input #1

```
12
1 5
3
1 4
1 6
3
2 1
2 2
2 3
1 7
1 8
3
2 4
```

Output #1

```
5
4 5 6
4
5
6
4 5 6 7 8
7
5 4 6 7 8
```

Data Limit

$2 \leq N \leq 1000$

All *x* are distinct.

k is legal.

#C

Description

You have a function $f(x)$, initially $f(x) = 0$

There are *Q* queries, each of the following 2 forms:

1 *a b*: *a*, *b* are integers. let $f_{new}(x) = f(x) + |x - a| + b$. And replace $f(x)$ with $f_{new}(x)$.

2: print $min\{\arg\min_x f(x)\}$ (The minimum *x* that minimizes) $f(x)$, and the minimum value of $f(x)$.

Input

Q

Q queries.

Output

When a query of type 2 appears, output a line as its answer.

Sample Test Data

Input #1

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

Output #1

```
4 2
1 -3
```

tips

In the two queries, f(x) are respectively

$f(x) = |x - 4| + 2$

$f(x) = |x - 1| + |x - 4| - 6$

Data Limit

$1 \le N \le 200000$

a,*b* is in the range of $(-10^9, 10^9)$.

Hint

Don't make this problem too complicated. Maybe you should notice how the problem is related to the median.

#D-裸平衡二叉搜索树AVL

Description

You need to implement a fully functional balanced binary search tree.

There are *Q* queries, each of the following 6 forms:

1. *x*. Insert *x*
2. *x*. Delete *x* (If there are multiple *x*, delete one of them.)
3. *x*. Print the ranking of *x* (Ranking is defined as (the number of numbers less than *x*) + 1)
4. *k*. Print the *k*-th smallest number. (See Sample Test Data 3)
5. *x*. Print the largest number in the BST less than *x*. (*x* may not be in the BST)
6. *x*. Print the smallest number in the BST greater than *x*. (*x* may not be in the BST)

Input

Q

Q queries.

Output

When a query of type 3,4,5,6 appears, output a line as its answer.

Sample Test Data

Input #1

```
9
1 10
4 1
1 8
1 9
1 12
1 2
1 3
6 9
5 8
```

Output #1

```
10
10
3
```

Input #2

```
19
1 9
1 7
1 3
1 12
```

```
1 13
5 7
5 12
1 14
6 13
6 3
2 3
2 12
4 3
4 4
3 13
3 9
2 7
2 14
3 9
```

Output #2

```
3
9
14
7
13
14
3
2
1
```

Input #3

```
23
1 3
1 4
1 4
1 4
1 5
4 1
4 2
4 3
4 4
4 5
3 3
3 4
3 5
5 4
6 4
2 4
4 1
4 2
4 3
4 4
3 3
3 4
3 5
```

Output #3

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

Data Limit

$1 \leq N \leq 100000$

a,*b* is in the range of $(-10^7, 10^7)$.

#E

Description

You have an integer array of length *n*

You need to select *k* intervals from *m* intervals.

For each interval you choose, add *x* to each number in that interval. (Do it for every interval, which means that there may be integers in the array that add multiples of *x*)

You need to maximize the minimum value of array after the adding.

Input

T. There are *T* testcases, for each testcase:

n m k x

n integers, representing the array.

m lines, each lines *l r*, representing the interval $[l, r]$

Output

The answer corresponding the problem.

Sample Test Data

Input #1

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

Output #1

```
3
```

Data Limit

$1 \leq \Sigma n \Sigma m \leq 200000$

$1 \leq T \leq 200000$

The integers in the array are in range of $[1, 10^8]$

$1 \leq x \leq 100$

Hint

You may find this problem difficult to get started. You may review some of the topics you learned in earlier lessons and turn the optimization problem into a decision problem(Consider a related problem, given a value *y*, can you tell if there is a way to make the final minimum value greater than *y*). Then some data structures you learned in this lecture will be put to good use. In addition, you can reduce time for adding numbers to intervals in a way similar to Lab7 problem E.