Algorithm Design >>
1846. GCD 2010

•••

6118149 heesun choi

Overview

1846. GCD 2010

Time limit: 0.5 second

Memory limit: 64 MB

Difficulty: 149

The problem

You have got a job offer from a secret project of the Agency of Federal Security under the code name “GCD 2010”. The subject of research is a collection of positive integer numbers. Your goal is to calculate how the greatest common divisor of all numbers in this collection changes as we insert numbers into this collection and remove them from it. At the beginning of the experiment, the collection is empty.

Input

The first line contains an integer q ($1 \le q \le 105$), which is the number of operations with the collection. Each of the next q lines has either the form “+ x” or “- x”. In the first case, number x is inserted into the collection, in the latter case it is removed from the collection. The number x is a positive integer not exceeding 109. It is guaranteed that operations remove only the integers which lie in the collection.

Output

Output the greatest common divisor of all numbers in the collection after each of the given operation.

According to the 190R order, the greatest common divisor of an empty collection is equal to one.

Sample

case 1. Input

```
5
+ 8
+ 6
+ 8
- 8
- 8
```

Output

```
8
2
2
2
6
```

case 2. Input

```
9
+ 10
+ 10
+ 10
- 10
- 10
+ 5
- 10
- 5
+ 123
```

Output

```
10
10
10
10
10
5
5
1
123
```

Code

```
import sys

def gcd(x,y):
    while(y):
        x, y = y, x % y
    return x

def add (val, pos, node, start, end):
    global t
    if start == end and start == pos:
        tree[node] = val
    else:
        mid = (start+end) >> 1
        if pos <= mid:
            add(val, pos, node*2, start,
mid)
        else:
            add(val, pos, node*2+1, mid+1,
end)
        tree[node] =
gcd(tree[node*2],tree[node*2+1])

def remove(pos, node, start, end):
    if start == end and start == pos:
        tree[node] = 0
    else:
        mid = (start+end) >> 1
        if pos <= mid:
            remove(pos, node*2, start, mid)
        else:
            remove(pos, node*2+1, mid+1,
end)
        tree[node] =
gcd(tree[node*2],tree[node*2+1])

k=0
size =0
default = -1

num = int(sys.stdin.readline())
inp = [[0 for col in range(3)] for row in
range(num)]
inpcopy = [[0 for col in range(3)] for row
in range(num)]
count = [0] * 100000
tree = [0] * (num * 4)
for i in range(num):
    temp = sys.stdin.readline().split()
    inp[i][0] = inpcopy[i][0] = int(temp[1])
    inp[i][1] = inpcopy[i][1] = temp[0]
    inp[i][2] = i
inp.sort(key = lambda x:x[0])
for i in range(num):
    if  default != inp[i][0]:
        k+=1
    default = inp[i][0]
    inpcopy[inp[i][2]][2] = k
for i in range(num):
    position = inpcopy[i][2]
    if inpcopy[i][1] == '-':
        if count[position] ==1:
            remove(position,1,1,num)
        count[position] -= 1
        size -= 1
    else:
        count[position] += 1
        if count[position] ==1:
            add(inpcopy[i][0],position,1,1,num)
        size += 1
    if size == 0:
        tree[1] = 1
    print(tree[1])
```

Euclidean-algorithm

method of obtaining GCD

GCD(A,B) = GCD(B, A%B)

GCD(A,B,C) = GCD(GCD(A,B),C)

[a > b, a % b = n]

```
If n == 0:
    gcd = b
If n != 0:
    a=b;   b=n;
    gcd(a,b)
```

Segment tree algorithm -binary tree

Store array elements on each leaf node and its parent node stores the gcd values of the child nodes. Finally, the root node stores the gcd value of the entire array.

idea

Arrange the input arrays in ascending order and assign the same mark to elements(inp[][3]) with the same number. The mark is used as the index of the tree.

cout array stores the number of duplicate numbers and add / remove those numbers to the tree only at the beginning(count[] == 1) / the end(count[] == 1).

Result

8886332	20:41:43 15 May 2020	heesun	1846. GCD 2010	Python 3.6	Time limit exceeded	15	0.515	17 504 KB

Reference

https://gist.github.com/wil93/47f9b7f8c409bf9de2c9 - idea

https://www.geeksforgeeks.org/segment-tree-set-1-sum-of-given-range/ - algorithm