A. The Closest Pair

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

Currently Tiny is learning Computational Geometry. When trying to solve a problem called "The Closest Pair Of Points In The Plane", he found that a code which gave a wrong time complexity got Accepted instead of Time Limit Exceeded.

The problem is the follows. Given n points in the plane, find a pair of points between which the distance is minimized. Distance between (x_1, y_1) and (x_2, y_2) is $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$.

The pseudo code of the unexpected code is as follows:

```
input n
for i from 1 to n
    input the i-th point's coordinates into p[i]
sort array p[] by increasing of x coordinate first and increasing of y coordinate second
d=INF        //here INF is a number big enough
tot=0
for i from 1 to n
    for j from (i+1) to n
        ++tot
        if (p[j].x-p[i].x>=d) then break    //notice that "break" is only to be
                                            //out of the loop "for j"
        d=min(d,distance(p[i],p[j]))
output d
```

Here, tot can be regarded as the running time of the code. Due to the fact that a computer can only run a limited number of operations per second, tot should not be more than k in order not to get Time Limit Exceeded.

You are a great hacker. Would you please help Tiny generate a test data and let the code get Time Limit Exceeded?

Input

A single line which contains two space-separated integers n and k $(2 \le n \le 2000, 1 \le k \le 10^9)$.

Output

If there doesn't exist such a data which let the given code get TLE, print "`no solution`" (without quotes); else print n lines, and the i-th line contains two integers x_i, y_i $(|x_i|, |y_i| \le 10^9)$ representing the coordinates of the i-th point.

The conditions below must be held:

- All the points must be distinct.
- $|x_i|, |y_i| \le 10^9$.
- After running the given code, the value of tot should be larger than k.

Sample test(s)

input
```
4 3
```
output
```
0 0
0 1
1 0
1 1
```

input
```
2 100
```
output
```
no solution
```

B. Cats Transport

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

Zxr960115 is owner of a large farm. He feeds m cute cats and employs p feeders. There's a straight road across the farm and n hills along the road, numbered from 1 to n from left to right. The distance between hill i and $(i - 1)$ is d_i meters. The feeders live in hill 1.

One day, the cats went out to play. Cat i went on a trip to hill h_i, finished its trip at time t_i, and then waited at hill h_i for a feeder. The feeders must take all the cats. Each feeder goes straightly from hill 1 to n without waiting at a hill and takes all the **waiting** cats at each hill away. Feeders walk at a speed of 1 meter per unit time and are strong enough to take as many cats as they want.

For example, suppose we have two hills $(d_2 = 1)$ and one cat that finished its trip at time 3 at hill 2 $(h_1 = 2)$. Then if the feeder leaves hill 1 at time 2 or at time 3, he can take this cat, but if he leaves hill 1 at time 1 he can't take it. If the feeder leaves hill 1 at time 2, the cat waits him for 0 time units, if the feeder leaves hill 1 at time 3, the cat waits him for 1 time units.

Your task is to schedule the time leaving from hill 1 for each feeder so that the sum of the waiting time of all cats is minimized.

Input

The first line of the input contains three integers n, m, p $(2 \le n \le 10^5, 1 \le m \le 10^5, 1 \le p \le 100)$.

The second line contains $n - 1$ positive integers $d_2, d_3, ..., d_n$ $(1 \le d_i < 10^4)$.

Each of the next m lines contains two integers h_i and t_i $(1 \le h_i \le n, 0 \le t_i \le 10^9)$.

Output

Output an integer, the minimum sum of waiting time of all cats.

Please, do not write the `%lld` specifier to read or write 64-bit integers in C++. It is preferred to use the `cin`, `cout` streams or the `%I64d` specifier.

Sample test(s)

input

```
4 6 2
1 3 5
1 0
2 1
4 9
1 10
2 10
3 12
```

output

```
3
```

C. Fetch the Treasure

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

Rainbow built h cells in a row that are numbered from 1 to h from left to right. There are n cells with treasure. We call each of these n cells "Treasure Cell". The i-th "Treasure Cell" is the a_i-th cell and the value of treasure in it is c_i dollars.

Then, Freda went in the first cell. For now, she can go just k cells forward, or return to the first cell. That means Freda was able to reach the 1st, $(k + 1)$-th, $(2 \cdot k + 1)$-th, $(3 \cdot k + 1)$-th cells and so on.

Then Rainbow gave Freda m operations. Each operation is one of the following three types:

1. Add another method x: she can also go just x cells forward at any moment. For example, initially she has only one method k. If at some moment she has methods $a_1, a_2, \ldots, a_r$ then she can reach all the cells with number in form $1 + \sum_{i=1}^{r} v_i \cdot a_i$, where v_i — some non-negative integer.
2. Reduce the value of the treasure in the x-th "Treasure Cell" by y dollars. In other words, to apply assignment $c_x = c_x - y$.
3. Ask the value of the most valuable treasure among the cells Freda can reach. If Freda cannot reach any cell with the treasure then consider the value of the most valuable treasure equal to 0, and do nothing. Otherwise take the most valuable treasure away. If several "Treasure Cells" have the most valuable treasure, take the "Treasure Cell" with the minimum number (not necessarily with the minimum number of cell). After that the total number of cells with a treasure is decreased by one.

As a programmer, you are asked by Freda to write a program to answer each query.

Input

The first line of the input contains four integers: h $(1 \le h \le 10^{18})$, n, m $(1 \le n, m \le 10^5)$ and k $(1 \le k \le 10^4)$.

Each of the next n lines contains two integers: a_i $(1 \le a_i \le h)$, c_i $(1 \le c_i \le 10^9)$. That means the i-th "Treasure Cell" is the a_i-th cell and cost of the treasure in that cell is c_i dollars. All the a_i are distinct.

Each of the next m lines is in one of the three following formats:

- "1 x" — an operation of type 1, $1 \le x \le h$;
- "2 x y" — an operation of type 2, $1 \le x \le n, 0 \le y < c_x$;
- "3" — an operation of type 3.

There are at most 20 operations of type 1. It's guaranteed that at any moment treasure in each cell has positive value. It's guaranteed that all operations is correct (no operation can decrease the value of the taken tresure).

Please, do not use the `%lld` specifier to read 64-bit integers in C++. It is preferred to use the `cin`, `cout` streams or the `%I64d` specifier.

Output

For each operation of type 3, output an integer indicates the value (in dollars) of the most valuable treasure among the "Treasure Cells" Freda can reach. If there is no such treasure, output 0.

Sample test(s)

input

```
10 3 5 2
5 50
7 60
8 100
2 2 5
3
1 3
3
3
```

output

```
55
100
50
```

Note

In the sample, there are 10 cells and 3 "Treasure Cells". The first "Treasure Cell" is cell 5, having 50 dollars tresure in it. The second "Treasure Cell" is cell 7, having 60 dollars tresure in it. The third "Treasure Cell" is cell 8, having 100 dollars tresure in it.

At first, Freda can only reach cell 1, 3, 5, 7 and 9. In the first operation, we reduce the value in the second "Treasure Cell" from 60 to 55. Then the most valuable treasure among the "Treasure Cells" she can reach is max(50, 55) = 55. After the third operation, she can also go 3 cells forward each step, being able to reach cell 1, 3, 4, 5, 6, 7, 8, 9, 10. So the most valuable tresure is 100.

Noticed that she took the 55 dollars and 100 dollars treasure away, so the last answer is 50.

D. Interval Cubing

time limit per test: 6 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

While learning Computational Geometry, Tiny is simultaneously learning a useful data structure called segment tree or interval tree. He has scarcely grasped it when comes out a strange problem:

Given an integer sequence $a_1, a_2, ..., a_n$. You should run q queries of two types:

1. Given two integers l and r ($1 \le l \le r \le n$), ask the sum of all elements in the sequence $a_l, a_{l+1}, ..., a_r$.
2. Given two integers l and r ($1 \le l \le r \le n$), let each element x in the sequence $a_l, a_{l+1}, ..., a_r$ becomes x^3. In other words, apply an assignments $a_l = a_l^3, a_{l+1} = a_{l+1}^3, ..., a_r = a_r^3$.

For every query of type 1, output the answer to it.

Tiny himself surely cannot work it out, so he asks you for help. In addition, Tiny is a prime lover. He tells you that because the answer may be too huge, you should only output it modulo 95542721 (this number is a prime number).

Input

The first line contains an integer n ($1 \le n \le 10^5$), representing the length of the sequence. The second line contains n space-separated integers $a_1, a_2, ..., a_n$ ($0 \le a_i \le 10^9$).

The third line contains an integer q ($1 \le q \le 10^5$), representing the number of queries. Then follow q lines. Each line contains three integers t_i ($1 \le t_i \le 2$), l_i, r_i ($1 \le l_i \le r_i \le n$), where t_i stands for the type of the query while l_i and r_i is the parameters of the query, correspondingly.

Output

For each 1-type query, print the answer to it per line.

You should notice that each printed number should be non-negative and less than 95542721.

Sample test(s)

input

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

output

```
14
224
2215492
```

E. Biologist

time limit per test: 3 seconds

memory limit per test: 256 megabytes

input: standard input

output: standard output

SmallR is a biologist. Her latest research finding is how to change the sex of dogs. In other words, she can change female dogs into male dogs and vice versa.

She is going to demonstrate this technique. Now SmallR has n dogs, the costs of each dog's change may be different. The dogs are numbered from 1 to n. The cost of change for dog i is v_i RMB. By the way, this technique needs a kind of medicine which can be valid for only one day. So the experiment should be taken in one day and each dog can be changed at most once.

This experiment has aroused extensive attention from all sectors of society. There are m rich folks which are suspicious of this experiment. They all want to bet with SmallR forcibly. If SmallR succeeds, the i-th rich folk will pay SmallR w_i RMB. But it's strange that they have a special method to determine whether SmallR succeeds. For i-th rich folk, in advance, he will appoint certain k_i dogs and certain one gender. He will think SmallR succeeds if and only if on some day the k_i appointed dogs are all of the appointed gender. Otherwise, he will think SmallR fails.

If SmallR can't satisfy some folk that isn't her friend, she need not pay him, but if someone she can't satisfy is her good friend, she must pay g RMB to him as apologies for her fail.

Then, SmallR hope to acquire money as much as possible by this experiment. Please figure out the maximum money SmallR can acquire. By the way, it is possible that she can't obtain any money, even will lose money. Then, please give out the minimum money she should lose.

Input

The first line contains three integers n, m, g $(1 \le n \le 10^4, 0 \le m \le 2000, 0 \le g \le 10^4)$. The second line contains n integers, each is 0 or 1, the sex of each dog, 0 represent the female and 1 represent the male. The third line contains n integers $v_1, v_2, ..., v_n$ $(0 \le v_i \le 10^4)$.

Each of the next m lines describes a rich folk. On the i-th line the first number is the appointed sex of i-th folk (0 or 1), the next two integers are w_i and k_i $(0 \le w_i \le 10^4, 1 \le k_i \le 10)$, next k_i distinct integers are the indexes of appointed dogs (each index is between 1 and n). The last number of this line represents whether i-th folk is SmallR's good friend (0 — no or 1 — yes).

Output

Print a single integer, the maximum money SmallR can gain. Note that the integer is negative if SmallR will lose money.

Sample test(s)

input

```
5 5 9
0 1 1 1 0
1 8 6 2 3
0 7 3 3 2 1 1
1 8 1 5 1
1 0 3 2 1 4 1
0 8 3 4 2 1 0
1 7 2 4 1 1
```

output

```
2
```

input

```
5 5 8
1 0 1 1 1
6 5 4 2 8
0 6 3 2 3 4 0
0 8 3 3 2 4 0
0 0 3 3 4 1 1
0 10 3 4 3 1 1
0 4 3 3 4 1 1
```

output

```
16
```