Candyland

Source: WC2013

I’ve asked at least 2 people on the backend of Winter Camp 2013 already, no one remembers who wrote it. I suppose that’s what you get for writing the \sout{easier data structure problem} point pinata on a contest.

(The story was heavily edited during the translation process. Candy rain is a meme that got banned due to a series of unfortunate events around NOI`08.)



Candies are raining all over a kingdom, which is a connected tree with n vertices/cities, connected by n - 1 edges/roads. Each vertex/city only gets one of m types of candies. We denote the type of candy at city i by C\_i. Due to unpredictable weather patterns, these C\_i values also change regularly.

Some wanderers travel the kingdom, each along simple paths between two vertices. At each city they pass through (including start and finish), they receive one candy of the type currently raining on that city. That is, if the wander starts at x, ends at y, and the shortest path along the tree is x = u\_{1}, u\_{2}, ... u\_{L} = y, then the candies received are C\_{u\_1}, C\_{u\_2}, ... C\_{u\_{L}}.

Different types of candies bring different happiness values. A candy of type k has an inherent tastiness value of V\_k. However, repeatedly receiving candies gets boring, so the jth time one receives a candy of a type, one get a curiosity value of W\_j. The happiness of receiving a candy is the product of the curiosity value and the tastiness value: that is, if one receives a candy of type k for the jth time, ones happiness increases by V\_k \* W\_j.

You are asked to compute, from a log of candy type changes and wanderers’ travels given in chronological order, the happiness of each wanderer.

Input

Line 1 contains 3 numbers, n, m, q, the number of cities, the types of candies, and the number of operations.

Line 2 contains m integers, V\_1...V\_m, the inherent tastiness of the m types of candies.

Line 3 contains n integers, W\_1...W\_n, the curiosity value of receiving a candy the first, second, third, ... nth time respectively.

The next n-1 lines contain two numbers (between 1~n each): the vertices connected by an edge.

The next line contains n numbers, C\_1...C\_n, the types of candies (labeled 1...k) available at each of the cities initially.

Finally q lines follow, each describing an operation via Type x y:

If type is 0, then 1 <= x <= n, 1 <= y <= m, indicating that the type of candy at city x is now y. (aka. C\_x <-- y)

If type is 1, then 1 <= x, y<= n, and you need to output the total happiness (for the current configuration of candies) of a trip along the shortest path from x to y.

Output

For each type 1 query, output its happiness value.

Sample Input

4 3 5

1 9 2

7 6 5 1

2 3

3 1

3 4

1 2 3 2

1 1 2

1 4 2

0 2 1

1 1 2

1 4 2

Sample Output

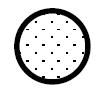
84

131

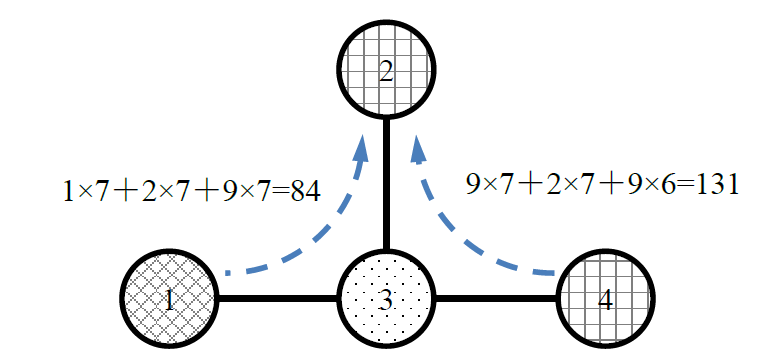
27

84

Explanation of sample output:

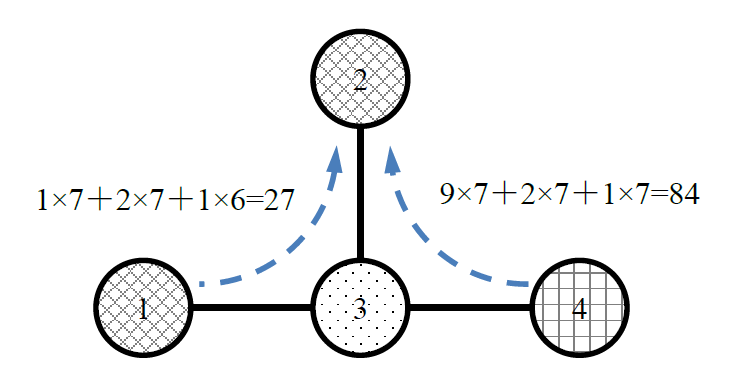
We use to represent a city with C\_i = 1, 2, 3 respectively.

The first two queries (12 & 42):



Note that for the trip from 4 to 2, the second time a type 2 candy is received with a curiority value of 6.

And after C\_2 is changed to 1:



Constraints:

n, m, q <= 10^5

1 <= V\_k, W\_j <= 10^6,

1 <= A\_i, B\_i <= n

1 <= C\_i <= m

W\_1....W\_n is monotonically non-increasing, that is, for each 1 <= j < n, W\_{j} >= W\_{j + 1}