Super Tree

You are given a rooted tree with n vertices, identified by indices $0, \ldots, n-1$. The root has index 0. For each $i \in \{0, \ldots, n-1\}$, the vertex i (i.e., the vertex with index i) has an integer a_i assigned to it. Let f_v be the value of the bitwise AND (henceforth denoted by $\&$) of the values a_i on the simple path from the vertex v to the root. (Note that the simple path from a vertex x to a vertex y includes both x and y.) Let the *power* of the tree be the value of

$$\sum_{0 \le u, v < n} f_u \cdot f_v,$$

and let the *superpower* of the tree be the value of (note the difference in ranges)

$$\sum_{0 \le u < v < n} f_u \cdot f_v.$$

For a clarifying example, see the explanation of the sample test cases below.

We will say that a vertex u belongs to *the subtree of a vertex* v if v belongs to the simple path from the vertex u to the root. Note that the subtree of a vertex x includes the vertex x itself.

You are presented with q updates. Each update is described by two integers, v and x, and it requires you to set $a_u := a_u \,\&\, x$ for each vertex u in the subtree of vertex v. After each update, you should output the power and superpower of the current tree.

As output values can be large, print them modulo $10^9 + 7$.

Input format

The first line of the input contains the integers n and q.

The second line of the input contains $n-1$ integers, namely $p_1, p_2, \ldots, p_{n-1}$, which determine the structure of the tree. For each $i \in \{1, \ldots, n-1\}$, p_i is the index of the parent of vertex i, and it holds that $0 \le p_i < i$.

The third line of the input contains n integers, namely $a_0, a_1, \ldots, a_{n-1}$. These are the values assigned to the vertices.

Each of the following q lines contains two integers, v ($0 \le v < n$) and x. These integers specify the individual updates.

Output format

Output $q+1$ lines. Each line should contain two integers separated by a space. In the first line, print the power and the superpower (modulo 10^9+7) of the initial tree. In the i-th line of the remaining q lines ($i \in \{1, \ldots, q\}$), print the power and the superpower (modulo 10^9+7) of the tree after the i-th update.

Input bounds

- $1 \leq n, q \leq 10^6$.
- $0 \leq a_i < 2^{60}$ for each $i \in \{0, \ldots, n-1\}$.
- $0 \leq x < 2^{60}$ for each update (v, x).

Scoring

For a given test case, your solution will receive 50% of the score if it correctly computes all power values but incorrectly computes at least one superpower value for that test case.

Likewise, 50% of the score for a given test case will be awarded to a solution that correctly computes all superpower values for that test case but incorrectly computes at least one power value.

Subtasks

1. (4 points) $n = 3$.
2. (7 points) $n, q \leq 700$.
3. (13 points) $n, q \leq 5000$.
4. (6 points) $n \leq 10^5$, $p_i = i-1$ (for each $i \in \{1, \ldots, n-1\}$), and $a_i, x < 2^{20}$ (for each $i \in \{0, \ldots, n-1\}$ and for each update (v, x)).
5. (7 points) $p_i = i-1$ (for each $i \in \{1, \ldots, n-1\}$).
6. (12 points) $a_i, x < 2^{20}$ (for each $i \in \{0, \ldots, n-1\}$ and for each update (v, x)).
7. (14 points) $n \leq 10^5$.
8. (11 points) $n \leq 5 \cdot 10^5$.
9. (26 points) No additional constraints.

Sample test case 1

Input

```
3 3
0 0
7 3 4
1 6
2 2
0 3
```

Output

```
196 61
169 50
81 14
25 6
```

Explanation

Initially, we have

$$f_0 = 7,\ f_1 = 7\&3 = 3,\ f_2 = 7\&4 = 4.$$

Therefore, the power of the tree is equal to

$$f_0 \cdot f_0 + f_0 \cdot f_1 + f_0 \cdot f_2 + f_1 \cdot f_0 + f_1 \cdot f_1 + f_1 \cdot f_2 + f_2 \cdot f_0 + f_2 \cdot f_1 + f_2 \cdot f_2 =$$

$$= 7 \cdot 7 + 7 \cdot 3 + 7 \cdot 4 + 3 \cdot 7 + 3 \cdot 3 + 3 \cdot 4 + 4 \cdot 7 + 4 \cdot 3 + 4 \cdot 4 = 196.$$

The superpower is equal to

$$f_0 \cdot f_1 + f_0 \cdot f_2 + f_1 \cdot f_2 = 7 \cdot 3 + 7 \cdot 4 + 3 \cdot 4 = 61.$$

After the first update:

$$a_0 = 7,\ a_1 = 3\&6 = 2,\ a_2 = 4;$$

$$f_0 = 7,\ f_1 = 2,\ f_2 = 4.$$

After the second update:

$$a_0 = 7,\ a_1 = 2,\ a_2 = 4\&2 = 0;$$

$$f_0 = 7,\ f_1 = 2,\ f_2 = 0.$$

After the third update:

$$a_0 = 7\&3 = 3,\ a_1 = 2\&3 = 2,\ a_2 = 0\&3 = 0;$$

$$f_0 = 3,\ f_1 = 2,\ f_2 = 0.$$

Sample test case 2

Input

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

Output

```
256 84
144 36
16 4
```

Explanation

Initially, we have

$$f_0 = 6,\ f_1 = 6\&5 = 4,\ f_2 = 6\&6 = 6,\ f_3 = 2\&5\&6 = 0.$$

After the first update:

$$a_0 = 6,\ a_1 = 5\&2 = 0,\ a_2 = 6,\ a_3 = 2\&2 = 2;$$

$$f_0 = 6,\ f_1 = 0,\ f_2 = 6,\ f_3 = 2\&0 = 0.$$

After the second update:

$$a_0 = 7,\ a_1 = 2,\ a_2 = 4\&2 = 0;$$

$$f_0 = 7,\ f_1 = 2,\ f_2 = 0.$$

Sample test case 3

Input

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

Output

```
900 367
784 311
576 223
256 83
```