مدار دیجیتالی

مداری داریم که شامل $N+M$ **گیت** است که از 0 تا $N+M-1$ شماره‌گذاری شده‌اند. گیت‌های شماره 0 تا $N-1$ **گیت‌های آستانه‌ای**، و گیت‌های شماره N تا $N+M-1$ گیت‌های منبع هستند.

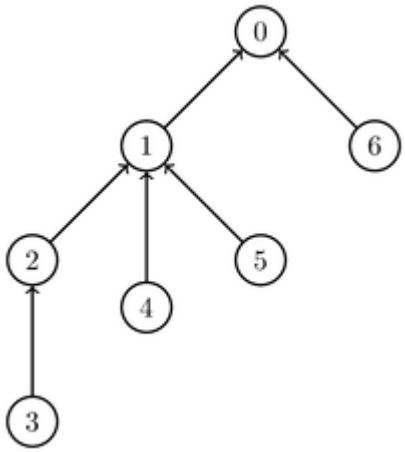
تمام گیت‌ها به جز گیت شماره 0 یک ورودی برای دقیقا یک گیت آستانه‌ای هستند. مشخصا، برای هر گیت شماره i به طوری که $1 \leq i \leq N+M-1$، گیت شماره i یک ورودی برای گیت آستانه‌ای شماره $P[i]$ است، به طوریکه $0 \leq P[i] \leq N-1$. همینطور توجه کنید که داریم $P[i] < i$. برای گیت شماره 0 در نظر می‌گیریم $P[0] = -1$. هر گیت آستانه‌ای یک یا چند ورودی دارد. گیت‌های منبع هیچ ورودی‌ای ندارند.

هر گیت دارای یک حالت (State) است که برابر با 0 یا 1 است. حالت اولیه گیت‌های منبع به صورت یک آرایه A به طول M به شما داده شده است. یعنی برای هر j به طوری که $0 \leq j \leq M-1$، حالت اولیه گیت شماره $N+j$ برابر با $A[j]$ است.

حالت هر گیت آستانه‌ای وابسته به حالت ورودی‌های آن است و به طوری که در ادامه توضیح داده می‌شود بدست می‌آید. برای هر گیت آستانه‌ای یک **پارامتر آستانه** تعیین می‌شود. پارامتری که به یک گیت آستانه‌ای با c ورودی نسبت داده می‌شود باید عددی بین 1 تا c باشد (شامل هر دو). سپس، حالت یک گیت آستانه‌ای با پارامتر آستانه‌ی p برابر با 1 خواهد بود اگر حداقل p تا از ورودی‌های آن حالت 1 را داشته باشند، در غیر این صورت حالت آن 0 خواهد بود.

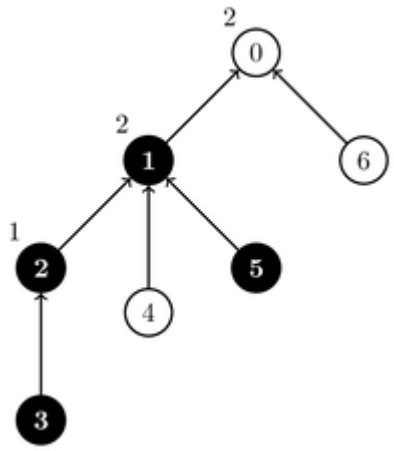
برای مثال، فرض کنید که $N=3$ گیت آستانه‌ای و $M=4$ گیت منبع داریم. ورودی‌های گیت 0 گیت‌های 1 و 6، ورودی‌های گیت 1 گیت‌های 2، 4 و 5، و تنها ورودی گیت 2 گیت 3 است.

این مثال در تصویر زیر قابل مشاهده است:



فرض کنید گیت‌های منبع 3 و 5 دارای حالت 1، در حالی که گیت‌های منبع 4 و 6 دارای حالت 0 هستند. در نظر بگیرید که پارامترهای آستانه‌ی 1، 2 و 2 به ترتیب به گیت‌های آستانه‌ی 2، 1 و 0 نسبت داده شده است. در این حالت، گیت 2 دارای حالت 1، گیت 1 دارای حالت 1 و گیت 0 دارای حالت 0 خواهد بود. این مقادیر نسبت داده شده‌ی پارامترهای

آستانه و حالت‌ها در تصویر زیر نشان داده شده است. گیت‌هایی که دارای حالت 1 هستند با رنگ سیاه نمایش داده شده‌اند.



حالت گیت‌های منبع Q بار آپدیت می‌شود. هر آپدیت با دو عدد صحیح L و R ($N \leq L \leq R \leq N + M - 1$) توصیف می‌شود و حالت تمام گیت‌های منبعی که عدد آن‌ها در بازه $[L, R]$ است (شامل هر دو) را برعکس می‌کند. یعنی برای هر i به طوری که $L \leq i \leq R$، اگر گیت منبع i حالت 0 را داشته باشد به 1 تغییر می‌کند و اگر حالت 1 را داشته باشد به 0 تغییر می‌کند. حالت جدید این گیت‌ها بدون تغییر باقی می‌مانند تا زمانی که دوباره توسط آپدیتی دیگر تغییر پیدا کنند.

وظیفه‌ی شما این است که بعد از هر آپدیت، بدست آورید چند روش مختلف برای مشخص کردن مقدار پارامترهای آستانه وجود دارد که در نهایت گیت شماره 0 در حالت 1 باشد. دو روش متفاوت محسوب می‌شوند اگر حداقل یک گیت آستانه‌ای وجود داشته باشد که پارامتری که به آن نسبت داده می‌شود در این دو روش متفاوت باشند. از آنجایی که تعداد کل روش‌ها می‌تواند زیاد باشد، شما باید مقدار باقیمانده‌ی این عدد بر $1\,000\,002\,022$ را بدست آورید.

توجه کنید که در مثالی که در بالا آمده است، ۶ روش مختلف برای اختصاص دادن پارامترهای آستانه وجود دارد، به دلیل آن که گیت‌های 0، 1 و 2 به ترتیب 2، 3 و 1 ورودی دارند. در 2 تا از این 6 روش گیت 0 در حالت 1 خواهد بود.

Implementation Details

Your task is to implement two procedures.

```
void init(int N, int M, int[] P, int[] A)
```

- N: the number of threshold gates.
- M: the number of source gates.
- P: an array of length $N + M$ describing the inputs to the threshold gates.
- A: an array of length M describing the initial states of the source gates.
- This procedure is called exactly once, before any calls to `count_ways`.

```
int count_ways(int L, int R)
```

- L, R: the boundaries of the range of source gates, whose states are toggled.
- This procedure should first perform the specified update, and then return the number of ways, modulo $1\,000\,002\,022$, of assigning parameters to the threshold gates, which result in gate 0 having state 1.
- This procedure is called exactly Q times.

Example

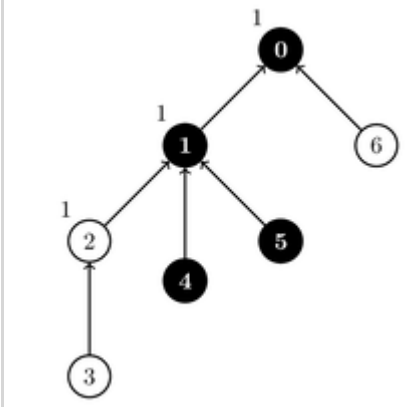
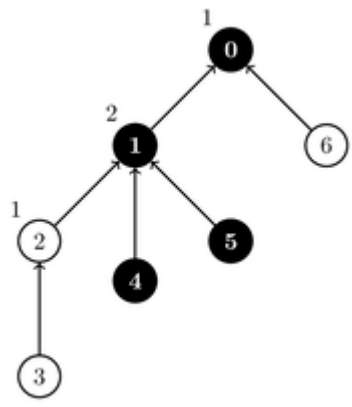
Consider the following sequence of calls:

```
init(3, 4, [-1, 0, 1, 2, 1, 1, 0], [1, 0, 1, 0])
```

This example is illustrated in the task description above.

```
count_ways(3, 4)
```

This toggles the states of gates 3 and 4, i.e. the state of gate 3 becomes 0, and the state of gate 4 becomes 1. Two ways of assigning the parameters which result in gate 0 having state 1 are illustrated in the pictures below.

Way 1	Way 2
	

In all other assignments of parameters, gate 0 has state 0. Thus, the procedure should return 2.

```
count_ways(4, 5)
```

This toggles the states of gates 4 and 5. As a result, all source gates have state 0, and for any assignment of parameters, gate 0 has state 0. Thus, the procedure should return 0.

```
count_ways(3, 6)
```

This changes the states of all source gates to 1. As a result, for any assignment of parameters, gate 0 has state 1. Thus, the procedure should return 6.

Constraints

- $1 \leq N, M \leq 100\,000$
- $1 \leq Q \leq 100\,000$
- $P[0] = -1$
- $0 \leq P[i] < i$ and $P[i] \leq N - 1$ (for each i such that $1 \leq i \leq N + M - 1$)
- Each threshold gate has at least one input (for each i such that $0 \leq i \leq N - 1$ there exists an index x such that $i < x \leq N + M - 1$ and $P[x] = i$).
- $0 \leq A[j] \leq 1$ (for each j such that $0 \leq j \leq M - 1$)
- $N \leq L \leq R \leq N + M - 1$

Subtasks

1. (2 points) $N = 1$, $M \leq 1000$, $Q \leq 5$
2. (7 points) $N, M \leq 1000$, $Q \leq 5$, each threshold gate has exactly two inputs.
3. (9 points) $N, M \leq 1000$, $Q \leq 5$
4. (4 points) $M = N + 1$, $M = 2^z$ (for some positive integer z), $P[i] = \lfloor \frac{i-1}{2} \rfloor$ (for each i such that $1 \leq i \leq N + M - 1$), $L = R$
5. (12 points) $M = N + 1$, $M = 2^z$ (for some positive integer z), $P[i] = \lfloor \frac{i-1}{2} \rfloor$ (for each i such that $1 \leq i \leq N + M - 1$)
6. (27 points) Each threshold gate has exactly two inputs.
7. (28 points) $N, M \leq 5000$
8. (11 points) No additional constraints.

Sample Grader

The sample grader reads the input in the following format:

- line 1: $N\ M\ Q$
- line 2: $P[0]\ P[1]\ \ldots\ P[N + M - 1]$
- line 3: $A[0]\ A[1]\ \ldots\ A[M - 1]$
- line $4 + k$ ($0 \leq k \leq Q - 1$): $L\ R$ for update k

The sample grader prints your answers in the following format:

- line $1 + k$ ($0 \leq k \leq Q - 1$): the return value of `count_ways` for update k