Układ cyfrowy

Mamy układ złożony z $N + M$ **bramek** ponumerowanych od 0 do $N + M - 1$. Bramki od 0 do $N - 1$ są **bramkami progowymi**, a bramki od N do $N + M - 1$ są **bramkami źródłowymi**.

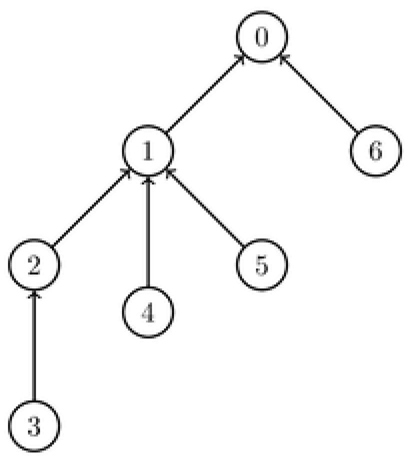
Wyjście każdej bramki, poza bramką 0, jest podpięte do **wejścia** dokładnie jednej bramki progowej. Dokładniej, dla każdego i spełniającego $1 \leq i \leq N + M - 1$, wyjście bramki i jest wejściem bramki $P[i]$, gdzie $0 \leq P[i] \leq N - 1$. Co istotne, zachodzi $P[i] < i$. Dodatkowo, przyjmujemy $P[0] = -1$. Każda bramka progowa ma jedno lub więcej wejść. Bramki źródłowe nie mają żadnych wejść.

Każda bramka ma **stan**, który może być równy 0 lub 1. Początkowe stany wszystkich bramek źródłowych są zadane tablicą A złożoną z M liczb całkowitych. Innymi słowy, dla każdego j spełniającego $0 \leq j \leq M - 1$, początkowy stan bramki źródłowej $N + j$ to $A[j]$.

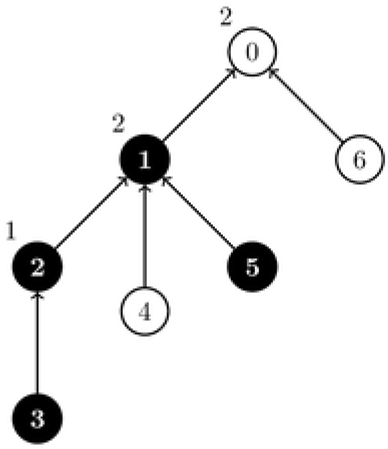
Stan każdej bramki progowej zależy od stanów podpiętych do niej bramek w następujący sposób. Najpierw każdej bramce progowej jest przypisywany **parametr**. Parametr przypisany bramce progowej, do której podpiętych jest c bramek, musi być liczbą całkowitą od 1 do c (włącznie). Następnie, stan bramki progowej z parametrem p jest ustawiany na 1, gdy co najmniej p z podpiętych do niej bramek ma stan 1, oraz 0 w przeciwnym przypadku.

Na przykład, rozważmy $N = 3$ bramek progowych i $M = 4$ bramek źródłowych. Do bramki 0 są podpięte bramki 1 i 6, do bramki 1 są podpięte bramki 2, 4 i 5, a do bramki 2 jest podpięta tylko bramka 3.

Ten przykład jest zilustrowany na poniższym obrazku.



Załóżmy, że stany bramek źródłowych 3 i 5 to 1, a stany bramek źródłowych 4 i 6 to 0. Załóżmy, że przypisaliśmy parametry 1, 2 i 2 do (odpowiednio) bramek progowych 2, 1 i 0. Wtedy stan bramki 2 to 1, stan bramki 1 to 1, a stan bramki 0 to 0. Ten wybór parametrów i odpowiadające mu stany są zilustrowane na poniższym obrazku. Bramki, które są w stanie 1, są zaznaczone na czarno.



Stany bramek źródłowych będą aktualizowane Q razy. Każda aktualizacja jest opisana przez dwie liczby całkowite L i R ($N \le L \le R \le N + M - 1$), które oznaczają, że należy zmienić stany wszystkich bramek źródłowych od L do R (włącznie). Innymi słowy, dla każdego i spełniającego $L \le i \le R$, bramka źródłowa i zmienia swój stan na 1, jeśli jej aktualny stan to 0, oraz na 0, gdy jej aktualny stan to 1. Nowy stan pozostaje niezmieniony aż do następnej dotyczącej go aktualizacji (o ile taka będzie).

Twoim zadaniem jest obliczenie, po każdej aktualizacji, ile różnych przypisań parametrów do bramek progowych sprawia, że bramka 0 ma stan 1. Dwa przypisania są uważane za różne, gdy istnieje co najmniej jedna bramka progowa, której stany w obu przypisaniach są różne. Ponieważ ta liczba może być bardzo duża, powinieneś obliczyć ją modulo $1\ 000\ 002\ 022$.

W powyższym przykładzie istnieje 6 różnych przypisań parametrów do bramek progowych, ponieważ liczba bramek podpiętych do bramek 0, 1 i 2 to odpowiednio 2, 3 i 1. W 2 z tych 6 przypisań stan bramki 0 to 1.

Szczegóły implementacji

Twoim zadaniem jest zaimplementowanie poniższej procedury i funkcji.

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

- N: liczba bramek progowych.
- M: liczba bramek źródłowych.
- P: tablica długości $N + M$ opisująca, gdzie podpięte są wyjścia kolejnych bramek.
- A: tablica długości M opisująca początkowe stany bramek wejściowych.

- Ta procedura będzie wywołana dokładnie raz, przed jakimkolwiek wywołaniem `count_ways`.

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

- L, R: końce przedziału bramek źródłowych, których stany są zmieniane.
- Funkcja powinna najpierw wykonać aktualizację, a następnie wyznaczyć liczbę, modulo $1\,000\,002\,022$, przypisań parametrów do bramek progowych powodujących, że stan bramki 0 to 1. Liczba ta powinna być wynikiem działania funkcji.
- Ta funkcja będzie wywołana dokładnie Q razy.

Przykład

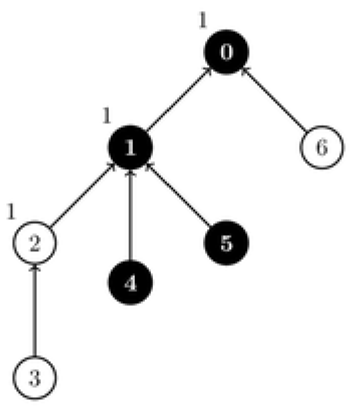
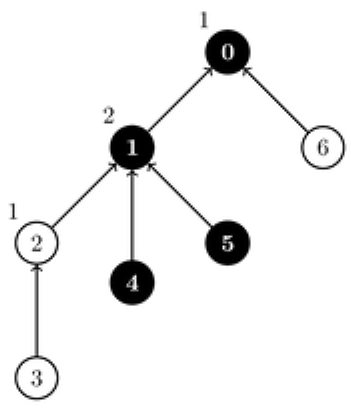
Rozważmy następującą sekwencję wywołań:

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

Ten przykład jest zilustrowany powyżej w treści zadania.

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

To wywołanie zmienia stany bramek 3 i 4, to znaczy stan bramki 3 zmienia się na 0, a stan bramki 4 zmienia się na 1. Dwa sposoby przypisania parametrów, dla których stan bramki 0 to 1, są zilustrowane na poniższych obrazkach.

Sposób 1	**Sposób** 2
	

Dla wszystkich innych przypisań parametrów, stan bramki 0 to 0. W związku z tym wynikiem działania funkcji powinno być 2.

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

To wywołanie zmienia stany bramek 4 oraz 5. Powoduje to, że stany wszystkich bramek źródłowych to 0, oraz dla dowolnego przypisania parametrów stan bramki 0 to 0. W związku z tym wynikiem działania funkcji powinno być 0.

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

To wywołanie zmienia stany wszystkich bramek źródłowych na 1. Powoduje to, że dla dla dowolnego przypisania parametrów stan bramki 0 to 1. W związku z tym wynikiem działania funkcji powinno być 6.

Ograniczenia

- $1 \le N, M \le 100\,000$
- $1 \le Q \le 100\,000$
- $P[0] = -1$
- $0 \le P[i] < i$ i $P[i] \le N - 1$ (dla każdego i spełniającego $1 \le i \le N + M - 1$)
- Do każdej bramki progowej jest podpięta przynajmniej jedna bramka (dla każdego i spełniającego $0 \le i \le N - 1$ istnieje indeks x spełniający $i < x \le N + M - 1$ i $P[x] = i$).
- $0 \le A[j] \le 1$ (dla każdego j spełniającego $0 \le j \le M - 1$)
- $N \le L \le R \le N + M - 1$

Podzadania

1. (2 punkty) $N = 1$, $M \le 1000$, $Q \le 5$
2. (7 punktów) $N, M \le 1000$, $Q \le 5$, każda bramka progowa ma dokładnie dwa wejścia.
3. (9 punktów) $N, M \le 1000$, $Q \le 5$
4. (4 punkty) $M = N + 1$, $M = 2^z$ (dla pewnej dodatniej liczby całkowitej z), $P[i] = \lfloor \frac{i-1}{2} \rfloor$ (dla każdego i spełniającego $1 \le i \le N + M - 1$), $L = R$
5. (12 punktów) $M = N + 1$, $M = 2^z$ (dla pewnej dodatniej liczby całkowitej z), $P[i] = \lfloor \frac{i-1}{2} \rfloor$ (dla każdego i spełniającego $1 \le i \le N + M - 1$)
6. (27 punktów) Każda bramka progowa ma dokładnie dwa wejścia.
7. (28 punktów) $N, M \le 5000$
8. (11 punktów) Brak dodatkowych ograniczeń.

Przykładowy program oceniający

Przykładowy program oceniający wczytuje dane wejściowe w następującym formacie:

- wiersz 1: $N\ M\ Q$
- wiersz 2: $P[0]\ P[1]\ \ldots\ P[N + M - 1]$
- wiersz 3: $A[0]\ A[1]\ \ldots\ A[M - 1]$
- wiersz $4 + k$ ($0 \le k \le Q - 1$): $L\ R$ dla aktualizacji k

Przykładowy program oceniający wypisuje Twoje odpowiedzi w następującym formacie:

- wiersz $1 + k$ ($0 \leq k \leq Q - 1$): wynik działania `count_ways` dla aktualizacji k