Circuiti digitali

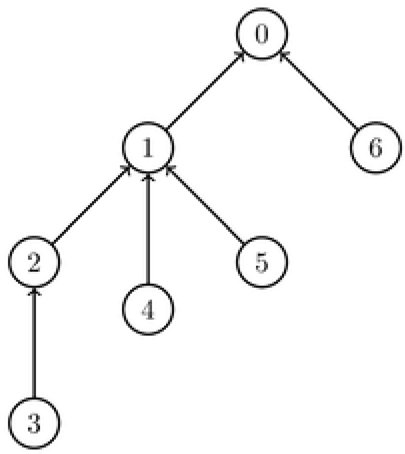
Hai un circuito composto da N **gate di soglia** (numerati da 0 a $N-1$) ed M **gate sorgente** (numerati da N a $N+M-1$).

Ogni gate i ($1 \le i < N+M$), eccetto per il gate 0, è un **input** per esattamente un gate di soglia $P[i] < N$. Inoltre, vale sempre che $P[i] < i$ e $P[0] = -1$. I gate sorgente non hanno input, mentre i gate di soglia hanno sempre uno o più input.

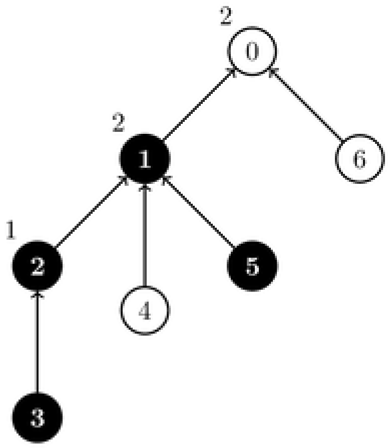
Ogni gate ha sempre uno **stato** binario 0 o 1. Lo stato iniziale dei gate sorgente è fornito in un array A di M interi, per cui lo stato del gate $N+j$ è $A[j]$ per $0 \le j < M$.

Dato un gate di soglia con c input, possiamo assegnargli un *valore di soglia* p ($1 \le p \le c$). In ogni momento, lo stato di un gate di soglia dipende dallo stato corrente dei suoi input e dal suo valore di soglia. Se lo stato di almeno p input è 1, il gate di soglia avrà stato 1; altrimenti avrà stato 0.

Per esempio, supponi ci siano $N=3$ gate di soglia e $M=4$ gate sorgente, con input collegati come in figura:



Supponi che i gate sorgente 3 e 5 abbiano stato 1, mentre i gate sorgente 4 e 6 abbiano stato 0. Assumi vengano assegnati i valori di soglia 1, 2 e 2 ai gate di soglia 2, 1 e 0 rispettivamente. In questo caso, i gate 2 e 1 avranno stato 1, mentre il gate 0 avrà stato 0, come mostrato nella seguente figura (dove gli stati 1 sono evidenziati in nero):



Lo stato dei gate sorgente verrà aggiornato Q volte. In ogni aggiornamento, dati L ed R con $N \leq L \leq R < N + M$, viene invertito lo stato di tutti i gate sorgente compresi tra L ed R inclusi, mentre tutti gli altri gate sorgente rimangono invariati.

Dopo ogni aggiornamento, devi contare quanti diversi assegnamenti di valori di soglia ai gate fanno sì che il gate 0 abbia stato 1, modulo $1\,000\,002\,022$. Due assegnamenti sono considerati diversi se almeno uno dei gate di soglia ha valore di soglia diverso nei due assegnamenti.

Nell'esempio descritto precedentemente, ci sono $2 \cdot 3 \cdot 1 = 6$ diversi assegnamenti di valori di soglia, dato che i gate 0, 1 e 2 hanno 2, 3 e 1 input rispettivamente. Di questi 6 assegnamenti, 2 fanno sì che il gate 0 abbia stato 1.

Dettagli di implementazione

Devi implementare la seguente funzione:

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

- N: il numero di gate di soglia.
- M: il numero di gate sorgente.
- P: un array di lunghezza $N + M$ che descrive gli input.
- A: un array di lunghezza M che descrive gli stati iniziali dei gate sorgente.
- Questa funzione è chiamata esattamente una volta, prima di ogni chiamata a `count_ways`.

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

- L, R: gli estremi dell'intervallo di gate sorgente che viene invertito.
- La funzione deve restituire il numero di assegnamenti di valori soglia, modulo $1\,000\,002\,022$, che fanno sì che il gate 0 abbia stato 1.
- Questa funzione è chiamata esattamente Q volte.

Caso di esempio

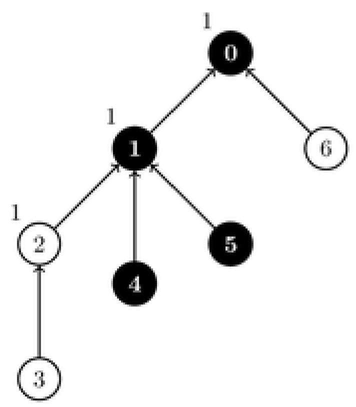
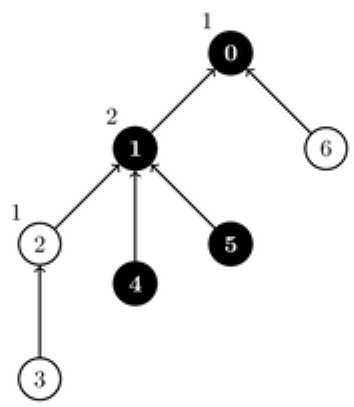
Considera la seguente sequenza di chiamate:

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

Questo caso corrisponde all'esempio descritto nel testo.

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

In questo aggiornamento vengono invertiti gli stati dei gate 3 e 4, per cui il gate 3 diventa 0 e il gate 4 diventa 1. Due modi per assegnare valori soglia che fanno sì che il gate 0 abbia stato 1 sono i seguenti:

Way 1	**Way** 2
	

In tutti gli altri assegnamenti il gate 0 ha stato 0, quindi la funzione deve restituire 2.

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

Questo inverte gli stati dei gate 4 e 5, portando tutti i gate sorgente ad avere stato 0. Quindi in ogni assegnamento di valori soglia il gate 0 ha stato 0, e la funzione deve restituire 0.

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

Questo cambia lo stato di tutti i gate sorgente a 1, per cui ora il gate 0 ha stato 1 in ogni assegnamento, e la funzione deve restituire 6.

Assunzioni

- $1 \leq N, M \leq 100\,000$.
- $1 \leq Q \leq 100\,000$.
- $P[0] = -1$.
- $0 \leq P[i] < i$ e $P[i] < N$ (per ogni $1 \leq i < N + M$).
- Ogni gate di soglia ha almeno un input (per ogni $0 \leq i < N$ esiste un $i < x < N + M$ tale che $P[x] = i$).
- $0 \leq A[j] \leq 1$ (per ogni $0 \leq j < M$).
- $N \leq L \leq R < N + M$.

Subtask

1. (2 punti) $N = 1$, $M \leq 1000$, $Q \leq 5$.
2. (7 punti) $N, M \leq 1000$, $Q \leq 5$, e ogni gate di soglia ha esattamente due input.
3. (9 punti) $N, M \leq 1000$, $Q \leq 5$.
4. (4 punti) $M = N + 1$, M è una potenza di 2, $P[i] = \lfloor \frac{i-1}{2} \rfloor$ (per $1 \leq i < N + M$), ed $L = R$.
5. (12 punti) $M = N + 1$, M è una potenza di 2, $P[i] = \lfloor \frac{i-1}{2} \rfloor$ (per $1 \leq i < N + M$).
6. (27 punti) Ogni gate di soglia ha esattamente due input.
7. (28 punti) $N, M \leq 5000$.
8. (11 punti) Nessuna limitazione aggiuntiva.

Grader di esempio

Il grader di esempio legge l'input secondo il seguente formato:

- riga 1: $N \; M \; Q$
- riga 2: $P[0] \;\; P[1] \;\; \ldots \;\; P[N + M - 1]$
- riga 3: $A[0] \;\; A[1] \;\; \ldots \;\; A[M - 1]$
- righe $4 + k$ ($0 \leq k < Q$): $L \; R$ per l'aggiornamento k

Il grader di esempio stampa l'output secondo il seguente formato:

- righe $1 + k$ ($0 \leq k < Q$): il valore restituito da `count_ways` per l'aggiornamento k