Wall

Jian-Jia sta costruendo un muro sovrapponendo mattoni della stessa dimensione. Questo muro consiste di n colonne di mattoni, che sono numerate da 0 a $n-1$ da sinistra a destra. Le colonne possono avere differenti altezze, e l'altezza di una colonna corrisponde al numero di mattoni di cui è composta.

Jian-Jia costruisce il muro come segue. All'inizio non ci sono mattoni in nessuna colonna. Successivamente, Jian-Jia effettua k fasi di aggiunta *(add)* o rimozione *(remove)* di mattoni. Il processo di costruzione termina quando tutte le k fasi sono state effettuate. In ogni fase, Jian-Jia riceve un intervallo di colonne consecutive e un'altezza h, ed esegue la procedura seguente:

- In una fase di *add*, Jian-Jia aggiunge mattoni alle colonne dell'intervallo che hanno meno di h mattoni, portandole ad avere esattamente h mattoni. Alle colonne con h o più mattoni non succede nulla.
- In una fase di *remove*, Jian-Jia rimuove mattoni alle colonne dell'intervallo che hanno più di h mattoni, portandole ad avere esattamente h mattoni. Alle colonne con h o meno mattoni non succede nulla.

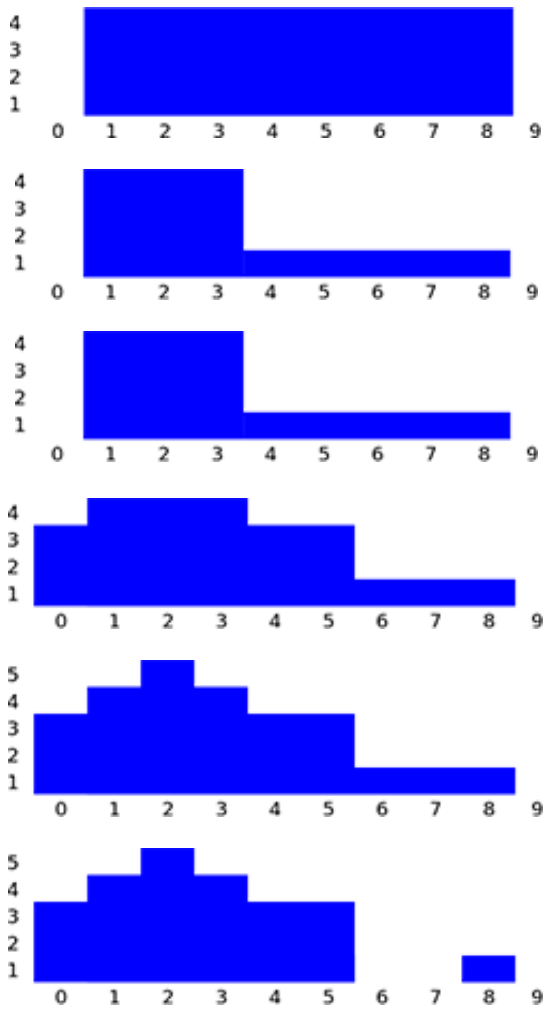
Il tuo compito è determinare la forma finale del muro.

Esempio

Supponiamo che ci siano 10 colonne di mattoni e 6 fasi di costruzione. Tutti gli intervalli nella tabella seguente sono da considerarsi con estremi inclusi. Le rappresentazioni del muro dopo ogni fase sono riportate sotto.

fase	tipo	intervallo	altezza
0	add	colonne da 1 a 8	4
1	remove	colonne da 4 a 9	1
2	remove	colonne da 3 a 6	5
3	add	colonne da 0 a 5	3
4	add	colonna 2	5
5	remove	colonne da 6 a 7	0

Dato che tutte le colonne sono inizialmente vuote, al termine della fase 0 ogni colonna da 1 a 8 avrà esattamente 4 mattoni, mentre le colonne 0 e 9 rimangono vuote. Durante la fase 1, vengono rimossi mattoni dalle colonne da 4 a 8 finché ciascuna di esse non arriva a contenere un unico mattone, mentre la colonna 9 rimane ancora vuota e le colonne da 0 a 3 (che sono all'esterno dell'intervallo) rimangono inalterate. La fase 2 non provoca cambiamenti dato che le colonne da 3 a 6 non hanno più di 5 mattoni. Dopo la fase 3 il numero di mattoni nelle colonne 0, 4, e 5 incrementa a 3. La fase 4 porta a 5 il numero di mattoni nella colonna 2. La fase 5 rimuove tutti i mattoni dalle colonne 6 e 7.



Descrizione del problema

Data la descrizione delle k fasi, devi calcolare il numero di mattoni in ogni colonna dopo che tutte le fasi sono terminate, implementando la funzione `buildWall` secondo le specifiche seguenti.

- `buildWall(n, k, op, left, right, height, finalHeight)`
  - `n`: numero di colonne nel muro.
  - `k`: numero di fasi.
  - `op`: array di lunghezza k; `op[i]` è il tipo della fase i, dove 1 indica una fase di *add* e 2 una fase di *remove*, per $0 \le i \le k-1$.
  - `left` and `right`: array di lunghezza k; l'intervallo delle colonne nella fase i inizia dalla colonna `left[i]` e termina con la colonna `right[i]` (includendo entrambi gli estremi `left[i]` $\le$ `right[i]`), per $0 \le i \le k-1$.
  - `height`: array di lunghezza k; `height[i]` è il parametro *altezza* della fase i, per $0 \le i \le k-1$.
  - `finalHeight`: array di lunghezza n; devi restituire il risultato calcolato memorizzando il numero finale di mattoni della colonna i in `finalHeight[i]`, per $0 \le i \le n-1$.

Subtask

Per tutti i subtask il parametro di altezza di ogni fase è sempre un intero non negativo minore o uguale a $100\,000$.

subtask	punti	n	k	note
1	8	$1 \le n \le 10\,000$	$1 \le k \le 5\,000$	nessun limite aggiuntivo
2	24	$1 \le n \le 100\,000$	$1 \le k \le 500\,000$	tutte le fasi di *add* precedono tutte le fasi di *remove*
3	29	$1 \le n \le 100\,000$	$1 \le k \le 500\,000$	nessun limite aggiuntivo
4	39	$1 \le n \le 2\,000\,000$	$1 \le k \le 500\,000$	nessun limite aggiuntivo

Dettagli di implementazione

Devi sottoporre esattamente un file, di nome `wall.c`, `wall.cpp` o `wall.pas`. Il file deve implementare la procedura descritta sopra utilizzando l'intestazione seguente. Per i programmi in C/C++, devi anche includere il file header `wall.h`.

Linguaggio C/C++

```
void buildWall(int n, int k, int op[], int left[], int right[],
int height[], int finalHeight[]);
```

Linguaggio Pascal

```
procedure buildWall(n, k : longint; op, left, right, height :
array of longint; var finalHeight : array of longint);
```

Grader di esempio

Il grader di esempio legge l'input nel formato seguente:

- riga 1: `n`, `k`.
- righe $2 + i$ ($0 \le i \le k - 1$): `op[i]`, `left[i]`, `right[i]`, `height[i]`.