Sacchetti di caramelle

L'architetto Timothy sta preparando n sacchetti di caramelle per gli studenti della scuola che ha appena costruito. I sacchetti sono numerati da 0 a $n-1$ e sono inizialmente vuoti. Il sacchetto i riesce a contenere fino a $c[i]$ caramelle.

Timothy impiega q giorni per preparare i sacchetti. Al giorno j ($0 \leq j \leq q-1$) esegue un'azione secondo i tre interi $\ell[j]$, $r[j]$ e $v[j]$, dove $0 \leq \ell[j] \leq r[j] \leq n-1$ e $v[i] \neq 0$. Per ogni sacchetto k tale che $\ell[j] \leq k \leq r[j]$:

- se $v[j] > 0$, aggiunge fino a $v[j]$ caramelle al sacchetto k, fermandosi se il sacchetto si riempie. In altre parole, se il sacchetto conteneva p caramelle, dopo l'azione conterrà $\min(c[k], p + v[j])$ caramelle.
- se $v[j] < 0$, rimuove fino a $-v[j]$ caramelle dal sacchetto k, fermandosi se il sacchetto si svuota. In altre parole, se il sacchetto conteneva p caramelle, dopo l'azione conterrà $\max(0, p + v[j])$ caramelle.

Determina il numero di caramelle in ciascun sacchetto dopo q giorni.

Note di implementazione

Devi implementare la seguente funzione:

```
int[] distribute_candies(int[] c, int[] l, int[] r, int[] v)
```

- c: array di lunghezza n, dove $c[i]$ è la capacità del sacchetto i ($0 \leq i \leq n-1$).
- ℓ, r e v: array di lunghezza q, dove $\ell[j]$, $r[j]$ e $v[j]$ descrivono l'azione effettuata nel giorno j ($0 \leq j \leq q-1$).
- Questa funzione deve restituire un array s di lunghezza n, dove $s[i]$ ($0 \leq i \leq n-1$) è il numero di caramelle nel sacchetto i dopo q giorni.

Esempio

Considera la seguente chiamata:

```
distribute_candies([10, 15, 13], [0, 0], [2, 1], [20, -11])
```

In questo caso, il sacchetto 0 ha una capacità di 10 caramelle, il sacchetto 1 ha una capacità di 15 caramelle, e il sacchetto 2 ha una capacità di 13 caramelle.

Alla fine del giorno 0, il sacchetto 0 contiene $\min(c[0], 0 + v[0]) = 10$ caramelle, il sacchetto 1 contiene $\min(c[1], 0 + v[0]) = 15$ caramelle e il sacchetto 2 contiene $\min(c[2], 0 + v[0]) = 13$ caramelle.

Alla fine del giorno 1, il sacchetto 0 contiene $\max(0, 10 + v[1]) = 0$ caramelle e il sacchetto 1 contiene $\max(0, 15 + v[1]) = 4$ caramelle. Dato che $2 > r[1]$, non ci sono cambiamenti al numero di caramelle nel sacchetto 2. Il numero di caramelle alla fine di ogni giorno è riassunto come segue:

Giorno	Sacchetto 0	Sacchetto 1	Sacchetto 2
0	10	15	13
1	0	4	13

Quindi, la funzione deve restituire $[0, 4, 13]$.

Assunzioni

- $1 \leq n \leq 200\,000$.
- $1 \leq q \leq 200\,000$.
- $1 \leq c[i] \leq 10^9$ (per ogni $0 \leq i \leq n - 1$).
- $0 \leq \ell[j] \leq r[j] \leq n - 1$ (per ogni $0 \leq j \leq q - 1$).
- $-10^9 \leq v[j] \leq 10^9, v[j] \neq 0$ (per ogni $0 \leq j \leq q - 1$).

Subtask

1. (3 punti) $n, q \leq 2000$.
2. (8 punti) $v[j] > 0$ (per ogni $0 \leq j \leq q - 1$).
3. (27 punti) $c[0] = c[1] = \ldots = c[n - 1]$.
4. (29 punti) $\ell[j] = 0$ and $r[j] = n - 1$ (per ogni $0 \leq j \leq q - 1$).
5. (33 punti) Nessuna limitazione aggiuntiva.

Grader di esempio

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

- riga 1: n
- riga 2: $c[0]$ $c[1]$ $\ldots$ $c[n - 1]$
- riga 3: q
- righe $4 + j$ ($0 \leq j \leq q - 1$): $\ell[j]$ $r[j]$ $v[j]$

Il grader di esempio stampa l'output nel seguente formato:

- riga 1: $s[0]$ $s[1]$ $\ldots$ $s[n - 1]$