

# Distribuyendo Dulces

La tía Khong está preparando  $n$  cajas de dulces para los estudiantes de una escuela cercana. Las cajas están numeradas de  $0$  a  $n - 1$  e inicialmente están vacías. La caja  $i$  ( $0 \leq i \leq n - 1$ ) puede contener  $c[i]$  dulces.

A la tía Khong le toma  $q$  días preparar las cajas. En el día  $j$  ( $0 \leq j \leq q - 1$ ), ella realiza una acción especificada por los enteros  $l[j]$ ,  $r[j]$  y  $v[j]$  donde  $0 \leq l[j] \leq r[j] \leq n - 1$  y  $v[j] \neq 0$ . Para cada caja  $k$ , tal que  $l[j] \leq k \leq r[j]$ :

- Si  $v[j] > 0$ , la tía Khong agrega dulces a la caja  $k$ , uno por uno, hasta que haya agregado exactamente  $v[j]$  dulces o la caja se llene. En otras palabras, si la caja tiene  $p$  dulces antes de la acción, ahora tendrá  $\min(c[k], p + v[j])$  dulces después de la acción.
- Si  $v[j] < 0$ , la tía Khong remueve dulces de la caja  $k$ , uno por uno, hasta que haya removido exactamente  $-v[j]$  dulces o la caja se encuentre vacía. En otras palabras, si la caja tiene  $p$  dulces antes de la acción, ahora tendrá  $\max(0, p + v[j])$  después de la acción.

Tu tarea es determinar el número de dulces en cada caja después de los  $q$  días.

## Detalles de implementación

Debes implementar el siguiente procedimiento:

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

- $c$ : un arreglo de longitud  $n$ . Para  $0 \leq i \leq n - 1$ ,  $c[i]$  denota la capacidad de la caja  $i$ .
- $l$ ,  $r$  y  $v$ : tres arreglos de longitud  $q$ . En el día  $j$ , tal que  $0 \leq j \leq q - 1$ , la tía Khong realiza la acción especificada por los enteros  $l[j]$ ,  $r[j]$  y  $v[j]$ , como se describe anteriormente.
- Este procedimiento debe regresar un arreglo de longitud  $n$ , denotado como el arreglo  $s$ . Para  $0 \leq i \leq n - 1$ ,  $s[i]$  debe ser la cantidad de dulces en la caja  $i$  después de  $q$  días.

## Ejemplos

### Ejemplo 1

Considera la siguiente llamada:

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

Esto significa que la caja 0 tiene capacidad de 10 dulces, caja 1 tiene capacidad de 15 dulces, y la caja 2 tiene capacidad de 13 dulces.

Al final del día 0, la caja 0 tiene  $\min(c[0], 0 + v[0]) = 10$  dulces, la caja 1 tiene  $\min(c[1], 0 + v[0]) = 15$  dulces y la caja 2 tiene  $\min(c[2], 0 + v[0]) = 13$  dulces.

Al final del día 1, la caja 0 tiene  $\max(0, 10 + v[1]) = 0$  dulces, caja 1 tiene  $\max(0, 15 + v[1]) = 4$  dulces. Como  $2 > r[1]$ , no hay cambios de cantidad de dulces en la caja 2. El número de dulces al final de cada día está descrito a continuación:

Día	Caja 0	Caja 1	Caja 2
0	10	15	13
1	0	4	13

Por lo tanto, el procedimiento debe regresar  $[0, 4, 13]$ .

## Restricciones

- $1 \leq n \leq 200\,000$
- $1 \leq q \leq 200\,000$
- $1 \leq c[i] \leq 10^9$  (para todo  $0 \leq i \leq n - 1$ )
- $0 \leq l[j] \leq r[j] \leq n - 1$  (para todo  $0 \leq j \leq q - 1$ )
- $-10^9 \leq v[j] \leq 10^9, v[j] \neq 0$  (para todo  $0 \leq j \leq q - 1$ )

## Subtasks

1. (3 puntos)  $n, q \leq 2000$
2. (8 puntos)  $v[j] > 0$  (para todo  $0 \leq j \leq q - 1$ )
3. (27 puntos)  $c[0] = c[1] = \dots = c[n - 1]$
4. (29 puntos)  $l[j] = 0$  y  $r[j] = n - 1$  (para todo  $0 \leq j \leq q - 1$ )
5. (33 puntos) Sin restricciones adicionales.

## Evaluador de ejemplo

El evaluador de ejemplo lee la entrada en el siguiente formato:

- línea 1:  $n$
- línea 2:  $c[0] \ c[1] \ \dots \ c[n - 1]$
- línea 3:  $q$
- línea  $4 + j$  ( $0 \leq j \leq q - 1$ ):  $l[j] \ r[j] \ v[j]$

El evaluador de ejemplo imprime tu respuesta en el siguiente formato:

- línea 1:  $s[0] \ s[1] \ \dots \ s[n - 1]$