

# Süßigkeiten verteilen

Tante Khong bereitet  $n$  Boxen mit Süßigkeiten für Schüler einer nahegelegenen Schule vor. Die Boxen sind von  $0$  bis  $n - 1$  durchnummeriert und sind anfangs leer. In die Box  $i$  ( $0 \leq i \leq n - 1$ ) passen  $c[i]$  Süßigkeiten hinein; wir sagen, dass Box  $i$  Kapazität  $c[i]$  hat.

Tante Khong verbringt  $q$  Tage mit der Vorbereitung der Boxen. Am Tag  $j$  ( $0 \leq j \leq q - 1$ ) führt sie eine Aktion aus, die durch drei Ganzzahlen  $l[j]$ ,  $r[j]$  und  $v[j]$  beschrieben wird, wobei  $0 \leq l[j] \leq r[j] \leq n - 1$  und  $v[j] \neq 0$ . Für jede Box  $k$  mit  $l[j] \leq k \leq r[j]$ :

- Wenn  $v[j] > 0$ , fügt Tante Khong nacheinander Süßigkeiten zur Box  $k$  hinzu, bis sie entweder genau  $v[j]$  Süßigkeiten hinzugefügt hat oder die Box voll ist. Mit anderen Worten: Wenn die Box vorher  $p$  Süßigkeiten enthielt, wird sie nachher  $\min(c[k], p + v[j])$  Süßigkeiten enthalten.
- Wenn  $v[j] < 0$ , nimmt Tante Khong Süßigkeiten nacheinander aus der Box  $k$  heraus, bis sie entweder genau  $-v[j]$  Süßigkeiten aus der Box genommen hat oder diese leer ist. Mit anderen Worten: Wenn die Box vorher  $p$  Süßigkeiten enthielt, wird sie nachher  $\max(0, p + v[j])$  Süßigkeiten enthalten.

Bestimme die Anzahl an Süßigkeiten in jeder Box nach  $q$  Tagen.

## Implementierung

Du sollst folgende Funktion implementieren:

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

- $c$ : Ein Array der Länge  $n$ . Für  $0 \leq i \leq n - 1$ , gibt  $c[i]$  die Kapazität der Box  $i$  an.
- $l$ ,  $r$  and  $v$ : Drei Arrays der Länge  $q$ . Am Tag  $j$  ( $0 \leq j \leq q - 1$ ) führt Tante Khong eine Aktion aus, die durch die Ganzzahlen  $l[j]$ ,  $r[j]$  und  $v[j]$  beschrieben wird (siehe oben).
- Diese Funktion soll ein Array der Länge  $n$  zurückgeben. Für dieses Array  $s$  gilt: Für  $0 \leq i \leq n - 1$  soll  $s[i]$  die Anzahl der Süßigkeiten in der Box  $i$  nach  $q$  Tagen sein.

## Beispiele

### Beispiel 1

Betrachte folgenden Aufruf:

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

Das bedeutet, dass Box 0 eine Kapazität von 10 Süßigkeiten hat, Box 1 von 15 Süßigkeiten und 2 von 13 Süßigkeiten.

Am Ende von Tag 0 hat Box 0  $\min(c[0], 0 + v[0]) = 10$  Süßigkeiten, Box 1  $\min(c[1], 0 + v[0]) = 15$  Süßigkeiten und Box 2  $\min(c[2], 0 + v[0]) = 13$  Süßigkeiten.

Am Ende von Tag 1 hat Box 0  $\max(0, 10 + v[1]) = 0$  Süßigkeiten und Box 1  $\max(0, 15 + v[1]) = 4$  Süßigkeiten. Da  $2 > r[1]$ , ändert sich die Anzahl der Süßigkeiten in Box 2 nicht. Die Anzahl der Süßigkeiten am Ende jedes Tages ist unten zusammengefasst:

Tag	Box 0	Box 1	Box 2
0	10	15	13
1	0	4	13

Daher soll die Funktion  $[0, 4, 13]$  zurückgeben.

## Beschränkungen

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

## Subtasks

1. (3 Punkte)  $n, q \leq 2000$
2. (8 Punkte)  $v[j] > 0$  (für alle  $0 \leq j \leq q - 1$ )
3. (27 Punkte)  $c[0] = c[1] = \dots = c[n - 1]$
4. (29 Punkte)  $l[j] = 0$  und  $r[j] = n - 1$  (für alle  $0 \leq j \leq q - 1$ )
5. (33 Punkte) Keine zusätzlichen Beschränkungen.

## Beispiel-Grader

Der Beispiel-Grader liest die Eingabe in folgendem Format:

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

Der Beispiel-Grader gibt das Ergebnis in folgendem Format aus:

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