

Scatole di biscotti (biscuits)

La zia Khong sta organizzando una competizione con x partecipanti e vuole dare a ciascuno una scatola di biscotti. Esistono k varietà di biscotti numerate da 0 a $k - 1$, l' i -esima delle quali ha un valore di bontà di 2^i e la zia ha a disposizione $a[i]$ (anche zero) biscotti di quel tipo.

Ognuna delle scatole conterrà zero o più biscotti di ciascun tipo e la somma dei valori di bontà viene chiamata *bontà totale* della scatola. La zia Khong non vuole fare preferenze, quindi tutte le x scatole devono avere la stessa bontà totale.

Aiuta la zia Khong a trovare quante bontà totali y sono possibili con i biscotti a sua disposizione senza fare preferenze.

Note di implementazione

Devi implementare la seguente funzione:

```
int64 count_tastiness(int64 x, int64[] a)
```

- x : il numero di scatole da riempire.
- a : un array di lunghezza k , ove $a[i]$ ($0 \leq i \leq k - 1$) è il numero di biscotti disponibili di tipo i .
- La funzione deve restituire il numero di valori distinti y , tali per cui con i biscotti disponibili è possibile riempire x scatole tutte di bontà totale y .
- Questa funzione viene chiamata un totale di q volte (vedi la sezione *Assunzioni* e *Subtasks* per i possibili valori di q). Ognuna di queste chiamate è indipendente dalle precedenti.

Esempi

Esempio 1

Considera la seguente chiamata:

```
count_tastiness(3, [5, 2, 1])
```

La zia vuole preparare 3 scatole e ci sono 3 tipi di biscotti:

- 5 biscotti del tipo 0, ognuno con bontà 1,
- 2 biscotti del tipo 1, ognuno con bontà 2,
- 1 biscotto del tipo 2 con bontà 4.

I possibili valori di y sono $[0, 1, 2, 3, 4]$. Per esempio, per fare 3 scatole di bontà totale 3 la zia può preparare:

- una scatola con 3 biscotti di tipo 0, e
- due scatole con un biscotto di tipo 0 e un biscotto di tipo 1.

Visto che ci sono 5 possibili valori di y , la funzione deve restituire 5.

<div> <div></div> <div></div> <div></div> </div> $y = 0$	<div> <div>1</div> <div>1</div> <div>1</div> </div> $y = 1$	<div> <div>1,1</div> <div>2</div> <div>2</div> </div> $y = 2$
<div> <div>1,1,1</div> <div>1,2</div> <div>1,2</div> </div> $y = 3$	<div> <div>1,1,2</div> <div>1,1,2</div> <div>4</div> </div> $y = 4$	

Esempio 2

Considera la seguente chiamata:

```
count_tastiness(2, [2, 1, 2])
```

La zia vuole preparare 2 scatole e ci sono 3 tipi di biscotti:

- 2 biscotti del tipo 0, ognuno con bontà 1,
- 1 biscotto del tipo 1 con bontà 2,
- 2 biscotti del tipo 2, ognuno con bontà 4.

I possibili valori di y sono $[0, 1, 2, 4, 5, 6]$. Dato che ci sono 6 possibilità per y , la funzione deve restituire 6.

Assunzioni

- $1 \leq k \leq 60$
- $1 \leq q \leq 1000$
- $1 \leq x \leq 10^{18}$
- $0 \leq a[i] \leq 10^{18}$ (per ogni $0 \leq i \leq k - 1$)
- Per ogni chiamata a `count_tastiness`, la somma dei valori di bontà di tutti i biscotti disponibili non supera 10^{18} .

Subtask

1. (9 punti) $q \leq 10$, e per ogni chiamata a `count_tastiness`, la somma dei valori di bontà di tutti i biscotti non supera 100 000.
2. (12 punti) $x = 1, q \leq 10$
3. (21 punti) $x \leq 10\,000, q \leq 10$
4. (35 punti) La soluzione corretta di ogni chiamata non supera 200 000.
5. (23 punti) Nessuna limitazione aggiuntiva.

Grader di esempio

Il grader di esempio legge l'input nel seguente formato. La prima riga contiene l'intero q , seguito da q coppie di righe ognuna delle quali descrive un singolo scenario nel seguente formato:

- riga 1: $k \ x$
- riga 2: $a[0] \ a[1] \ \dots \ a[k-1]$

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

- riga i ($1 \leq i \leq q$): il valore restituito da `count_tastiness` per l' i -esimo scenario in input.