

# Kekse Packen (biscuits)

Tante Khong organisiert einen Wettbewerb mit x Teilnehmern und möchte allen Teilnehmern eine **Tasche mit Keksen** geben. Es gibt k verschiedene Arten von Keksen, die von 0 bis k-1 durchnummeriert sind. Jeder Keks der Art i ( $0 \le i \le k-1$ ) hat eine **Schmackhaftigkeit** von  $2^i$ . Tante Khong hat a[i] (möglicherweise auch null) Kekse der Art i in ihrer Vorratskammer.

Jede von Tante Khongs Kekstaschen wird null oder mehr Kekse jeder Art enthalten. Insgesamt dürfen nicht mehr als a[i] Kekse der Art i eingepackt werden. Die Gesamtsumme der Schmackhaftigkeiten aller Keksen in einer Tasche wird **Gesamtgeschmack** der Tasche genannt.

Hilf Tante Khong, die Anzahl möglicher Werte von y zu finden, für die es möglich ist, x Taschen so mit Keksen zu packen, dass jede Tasche einen Gesamtgeschmack von y hat.

### Implementierungsdetails

Implementiere folgende Funktion:

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

- x: Anzahl Taschen, die gepackt werden sollen.
- ullet a: ein Array der Länge k. Für  $0 \leq i \leq k-1$  ist a[i] die Anzahl Kekse der Art i in der Vorratskammer.
- ullet Die Funktion soll die Anzahl möglicher Werte von y zurückgeben, für die es möglich ist, x Taschen so mit Keksen zu packen, dass jede Tasche einen Gesamtgeschmack von y hat.
- Die Funktion wird insgesamt q Mal aufgerufen (siehe Abschnitte Einschränkungen und Teilaufgaben für die möglichen Werte von q). Jeder dieser Aufrufe soll als eigener Fall behandelt werden.

## Beispiele

#### Beispiel 1

Nimm folgenden Aufruf an:

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

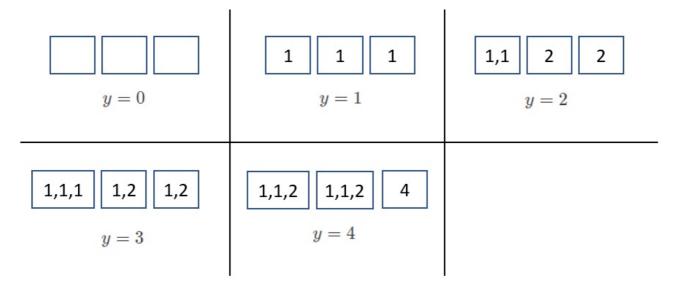
Dies bedeutet, dass die Tante 3 Taschen packen möchte und dass es 3 Arten von Keksen in der Vorratskammer gibt.

- 5 Kekse der Art 0, jeder Keks hat die Schmackhaftigkeit 1,
- 2 Kekse der Art 1, jeder Keks hat die Schmackhaftigkeit 2 und
- 1 Keks der Art 2, der Keks hat die Schmackhaftigkeit 4.

Die möglichen Werte für y sind [0,1,2,3,4]. Um beispielsweise 3 Taschen mit Gesamtgeschmack 3 zu packen, kann die Tante folgendermassen vorgehen:

- eine Tasche enthält drei Kekse der Art 0, und
- zwei Taschen enthalten jeweils einen Keks der Art 0 und einen der Art 1.

Da es 5 mögliche Werte für y gibt, soll die Funktion 5 zurückgeben.



#### Beispiel 2

Nimm folgenden Aufruf an:

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

Dies bedeutet, dass die Tante 2 Taschen packen möchte und dass es 3 Arten von Keksen in der Vorratskammer gibt:

- 2 Kekse der Art 0 mit Schmackhaftigkeit 1,
- 1 Keks der Art 1 mit Schmackhaftigkeit 2 und
- 2 Kekse der Art 2 mit Schmackhaftigkeit 4.

Die möglichen Werte für y sind [0,1,2,4,5,6]. Da es 6 mögliche Werte für y gibt, soll die Funktion den Wert 6 zurückgeben.

# Beschränkungen

- $1 \le k \le 60$
- $1 \le q \le 1000$
- $1 < x < 10^{18}$

- $0 \leq a[i] \leq 10^{18}$  (für alle  $0 \leq i \leq k-1$ )
- $\bullet$  Für alle Aufrufe von <code>count\_tastiness</code> ist die Summe der Schmackhaftigkeiten der Kekse in der Vorratskammer höchstens  $10^{18}$ .

# Teilaufgaben

- 1. (9 Punkte)  $q \le 10$  und für alle Aufrufe von <code>count\_tastiness</code> ist die Summe der Schmackhaftigkeiten der Kekse in der Vorratskammer höchstens 100~000.
- 2. (12 Punkte) x=1,  $q\leq 10$
- 3. (21 Punkte)  $x \le 10~000, q \le 10$
- 4. (35 Punkte) Der korrekte Rückgabewert für jeden Aufruf von  $count\_tastiness$  ist höchstens 200~000.
- 5. (23 Punkte) Keine zusätzlichen Beschränkungen.

## Beispiel-Grader

Der Beispiel-Grader liest die Eingabe im folgenden Format. Die erste Zeile enthält eine Ganzzahl q. Danach folgen q Paare von Zeilen. Jedes Paar beschreibt einen einzigen Fall im folgenden Format:

- Zeile 1: *k x*
- ullet Zeile 2: a[0] a[1]  $\dots$  a[k-1]

Der Beispiel-Grader schreibt die Ausgabe im folgenden Format:

• Zeile i ( $1 \le i \le q$ ): Rückgabewert von <code>count\_tastiness</code> für den i-ten Fall der Eingabe.