

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 von jeder Art enthalten. Die Gesamtzahl der Kekse der Art i in allen Packungen darf a[i] nicht überschreiten. Die Gesamtsumme aller Schmackhaftigkeiten von allen Keksen wird **Gesamtgeschmack** der Tasche genannt.

Hilf Tante Khong herauszufinden, wie viele verschiedene Werte von y existieren, sodass es möglich ist, x Taschen mit Keksen zu packen, wobei jede Kekspackung den Gesamtgeschmack y hat.

Implementierungsdetails

Du sollst folgende Funktion implementieren:

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

- x: die Anzahl von Taschen, die gepackt werden sollen.
- ullet a: ein Array der Länge k. Für $0 \leq i \leq k-1$ ist a[i] die Anzahl der Kekse der Art i in der Vorratskammer.
- Die Funktion soll die mögliche Anzahl von verschiedenen Werten von y zurückgeben, sodass die Tante x Taschen mit Keksen packen kann, wo jede den Gesamtgeschmack y hat.
- ullet Die Funktion wird insgesamt q mal aufgerufen (siehe Einschränkungen und Teilaufgaben-Abschnitt 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 will und dass es 3 Arten von Keksen in der

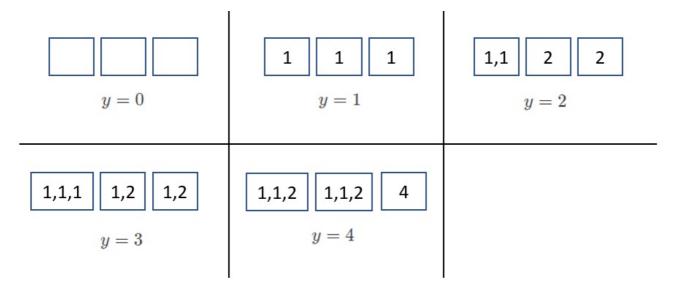
Vorratskammer gibt.

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

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

Beschränkungen

• $1 \le k \le 60$

- $1 \le q \le 1000$
- $1 \le x \le 10^{18}$
- ullet $0 \leq a[i] \leq 10^{18}$ (für alle $0 \leq i \leq k-1$)
- ullet Für alle Aufrufe von <code>count_tastiness</code> übersteigt die Summe der Schmackhaftigkeiten der Kekse in der Vorratskammer den Wert 10^{18} nicht.

Teilaufgaben

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

Beispiel-Grader

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

- ullet Zeile 1: k x
- Zeile 2: a[0] a[1] ... a[k-1]

Der Beispiel-Grader schreibt die Ausgabe im folgenden Format:

• Zeile i ($1 \le i \le q$): Rückgabewert von count tastiness für den i-ten Fall der Eingabe.