Der Grosse Preis

Der Grosse Preis ist eine berühmte Gameshow. Du bist der glückliche Teilnehmer, der in die Finalrunde gekommen ist. Du stehst vor einer Reihe von n Schachteln, die von links nach rechts mit 0 bis $n-1$ beschriftet sind. Jede Schachtel enthält einen Preis, den man nicht sehen kann, bevor die Schachtel geöffnet wird. Es gibt $v \geq 2$ verschiedene *Typen* von Preisen. Die Typen sind von 1 bis v in *absteigender* Reihenfolge ihrer Werte nummeriert.

Der Preis vom Typ 1 ist der wertvollste: ein Diamant. Es gibt genau einen Diamanten in den Schachteln. Der Preis vom Typ v ist der billigste: ein Lollipop. Um das Spiel aufregender zu gestalten, gibt es viel mehr billige als wertvolle Preise. Genauer gesagt wissen wir Folgendes für alle t mit $2 \leq t \leq v$: Wenn es k Preise vom Typ $t-1$ gibt, dann ist die Anzahl der Preise vom Typ t *strikt grösser* ($>$) als k^2.

Dein Ziel ist es, den Diamanten zu gewinnen. Am Ende des Spiels musst du eine Schachtel öffnen und erhältst den Preis, den sie enthält. Bevor du eine Schachtel auswählst, darfst du dem Moderator der Show, Rambod, ein paar Fragen stellen. Für jede Frage wählst du eine Schachtel i. Als Antwort gibt dir Rambod ein Array a, das zwei ganze Zahlen enthält. Sie bedeuten:

- Unter den Schachteln links der Schachtel i gibt es genau $a[0]$ Schachteln, die einen wertvolleren Preis enthalten als die Schachtel i.
- Unter den Schachteln rechts der Schachtel i gibt es genau $a[1]$ Schachteln, die einen wertvolleren Preis enthalten als die Schachtel i.

Als Beispiel nehmen wir an, dass $n = 8$ ist. Für deine Frage wählst du die Schachtel $i = 2$. Als Antwort sagt dir Rambod, dass $a = [1, 2]$ ist. Diese Antwort bedeutet:

- Genau eine der Schachteln 0 und 1 enthält einen Preis, der wertvoller ist als der in Schachtel 2.
- Genau zwei der Schachteln $3, 4, \ldots, 7$ enthalten einen Preis, der wertvoller ist als der in Schachtel 2.

Deine Aufgabe ist es, die Schachtel mit dem Diamanten zu finden, indem du eine geringe Anzahl von Fragen stellst.

Implementierungsdetails

Du sollst folgende Funktion implementieren:

```
int find_best(int n)
```

- Diese Funktion wird vom Grader genau einmal aufgerufen.
- n: Anzahl der Schachteln.
- Diese Funktion soll die Nummer der Schachtel mit dem Diamanten liefern, also die eindeutige ganze Zahl d ($0 \leq d \leq n - 1$) jener Schachtel, die den Preis vom Typ 1 enthält.

Die Funktion `find_best` kann folgende Funktion aufrufen:

```
int[] ask(int i)
```

- i: Nummer der Schachtel, die du zum Fragen ausgewählt hast. Der Wert von i muss im Bereich von 0 bis $n - 1$ inklusive sein.
- Der Rückgabewert ist ein Array a mit 2 Elementen: $a[0]$ ist die Anzahl von wertvolleren Preisen in den Schachteln links der Schachtel i und $a[1]$ ist die Anzahl von wertvolleren Preisen in den Schachteln rechts der Schachtel i.

Beispiel

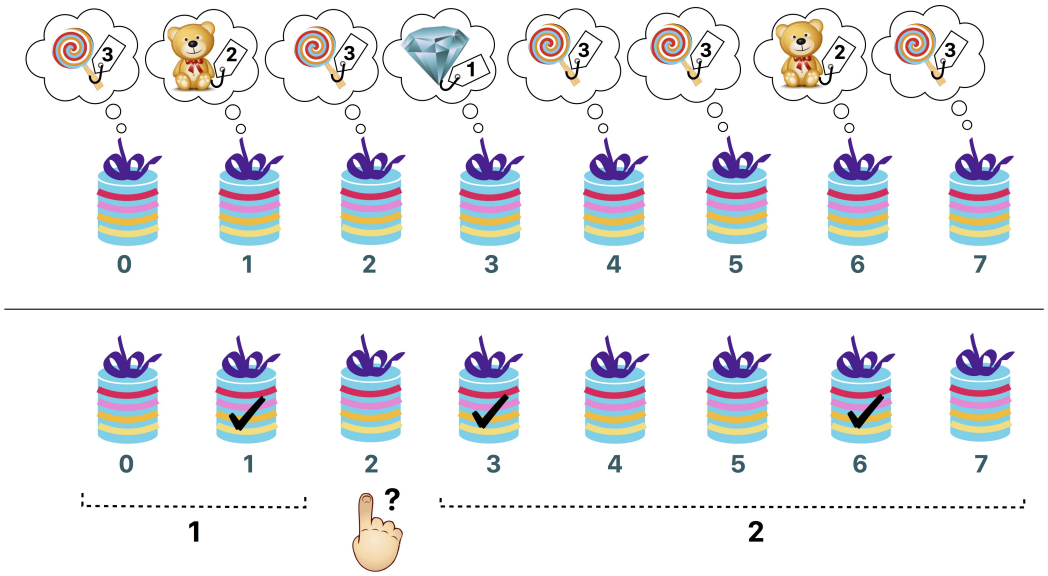
Der Grader macht den folgenden Funktionsaufruf:

```
find_best(8)
```

Es gibt $n = 8$ Schachteln. Angenommen, die Preistypen sind $[3, 2, 3, 1, 3, 3, 2, 3]$, dann liefert die Funktion `ask` folgende Rückgabewerte:

- `ask(0)` gibt $[0, 3]$ zurück
- `ask(1)` gibt $[0, 1]$ zurück
- `ask(2)` gibt $[1, 2]$ zurück
- `ask(3)` gibt $[0, 0]$ zurück
- `ask(4)` gibt $[2, 1]$ zurück
- `ask(5)` gibt $[2, 1]$ zurück
- `ask(6)` gibt $[1, 0]$ zurück
- `ask(7)` gibt $[3, 0]$ zurück

In diesem Beispiel ist der Diamant in Schachtel 3. Die Funktion `find_best` soll also 3 zurückgeben.



Die obige Abbildung illustriert dieses Beispiel. Der obere Abschnitt zeigt die Typen der Preise in jeder Schachtel. Der untere Abschnitt stellt die Anfrage `ask(2)` dar. Die markierten Schachteln enthalten wertvollere Preise als Schachtel 2.

Beschränkungen

- $3 \leq n \leq 200\,000$.
- Der Preistyp in jeder Schachtel liegt im Bereich von 1 bis v inklusive.
- Es gibt genau einen Preis vom Typ 1.
- Für alle t mit $2 \leq t \leq v$ gilt: wenn es k Preise vom Typ $t-1$ gibt, gibt es *mehr* als k^2 Preise vom Typ t.

Subtasks und Bewertung

Der Grader ist bei manchen Testfällen adaptiv. Das heisst, dass der Grader in diesen Fällen keine festgelegte Abfolge von Preisen hat. Stattdessen kann die Antwort des Graders von den von deiner Lösung gestellten Fragen abhängen. Es ist aber garantiert, dass nach jeder Antwort des Graders mindestens eine Preisabfolge mit allen bisher gegebenen Antworten konsistent ist.

1. (20 Punkte) Es gibt genau 1 Diamanten und $n-1$ Lollipops. Es gilt also $v = 2$. Du darfst die Funktion `ask` höchstens $10\,000$ Mal aufrufen.
2. (80 Punkte) Keine weiteren Beschränkungen.

In Subtask 2 kannst du Teilpunkte bekommen. Sei q die maximale Anzahl an Aufrufen der Funktion `ask`, die unter allen Testfällen vorkommt. Dann wird deine Punktzahl für diesen Subtask wie folgt berechnet:

Aufrufe	Punktzahl
$10\,000 < q$	0 (wird im CMS als "Wrong Answer" angezeigt)
$6000 < q \leq 10\,000$	70
$5000 < q \leq 6000$	$80 - (q - 5000)/100$
$q \leq 5000$	80

Beispiel-Grader

Der Beispiel-Grader ist nicht adaptiv. Stattdessen liest er ein Array p von Preisen ein und nutzt dieses. Für alle $0 \leq b \leq n - 1$ ist der Preistyp in Schachtel b durch $p[b]$ gegeben. Der Beispiel-Grader erwartet die Eingabe in folgendem Format:

- Zeile 1: n
- Zeile 2: $p[0]$ $p[1]$ $\ldots$ $p[n-1]$

Der Beispiel-Grader gibt eine einzige Zeile aus, die den Rückgabewert von `find_best` und die Anzahl der Aufrufe der Funktion `ask` enthält.