Den försvunna diamanten

Den försvunna diamanten är en berömd TV-show (inspirerad av det kända brädspelet). Du är den lyckliga deltagaren som har tagit sig till den sista rundan. Du står framför en rad med n lådor, numrerade från 0 till $n-1$ från vänster till höger. Varje låda innehåller ett pris som man inte kan se förrän lådan har öppnats. Det finns $v \geq 2$ olika typer av priser. Typerna är numrerade från 1 till v i ordning av minskande värde.

Priset av typ 1 är det dyraste: en diamant. Det finns exakt en diamant i lådorna. Priserna av typ v är de billigaste: en slickepott. För att spelet ska bli spännande finns det mycket fler billiga än dyra priser. Mer specifikt, för alla t sådana att $2 \leq t \leq v$ gäller att om det finns k priser av typ $t-1$ så finns det strikt fler än k^2 priser av typ t.

Ditt mål är att vinna diamanten. Vid slutet av spelet måste du öppna en låda och får då priset i lådan du öppnat. Innan du måste välja låda får du ställa några frågor till programledaren, Rambo. För varje fråga väljer du någon låda i. Rambo kommer då att svara med en array a som innehåller två heltal. Heltalens betydelse är:

- Bland lådorna till vänster om låda i finns det exakt $a[0]$ lådor som innehåller ett dyrare pris än det som finns i låda i.
- Bland lådorna till höger om låda i finns det exakt $a[1]$ lådor som innehåller ett dyrare pris än det som finns i låda i.

Som exempel, anta att $n = 8$. Som din första fråga väljer du låda $i = 2$. Som svar säger Rambo att $a = [1, 2]$. Detta betyder att:

- Exakt en av lådorna 0 och 1 innehåller ett pris som är dyrare än det i låda 2.
- Exakt två av lådorna $3, 4, \cdots, 7$ innehåller ett pris som är dyrare än det i låda 2.

Din uppgift är att hitta lådan som innehåller diamanten, genom att ställa ett litet antal frågor.

Implementation details

You should implement the following procedure:

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

- n: number of boxes.
- This procedure should return the label of the box which contains the diamond, i.e., the unique integer d ($0 \leq d \leq n-1$) such that box d contains a prize of type 1.

The above procedure can make calls to the following procedure:

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

- i: label of the box that you choose to ask about. The value of i must be between 0 and $n-1$, inclusive.
- This procedure returns the array a with 2 elements. Here, $a[0]$ is the number of more expensive prizes in the boxes to the left of box i and $a[1]$ is the number of more expensive prizes in the boxes to the right of box i.

Example

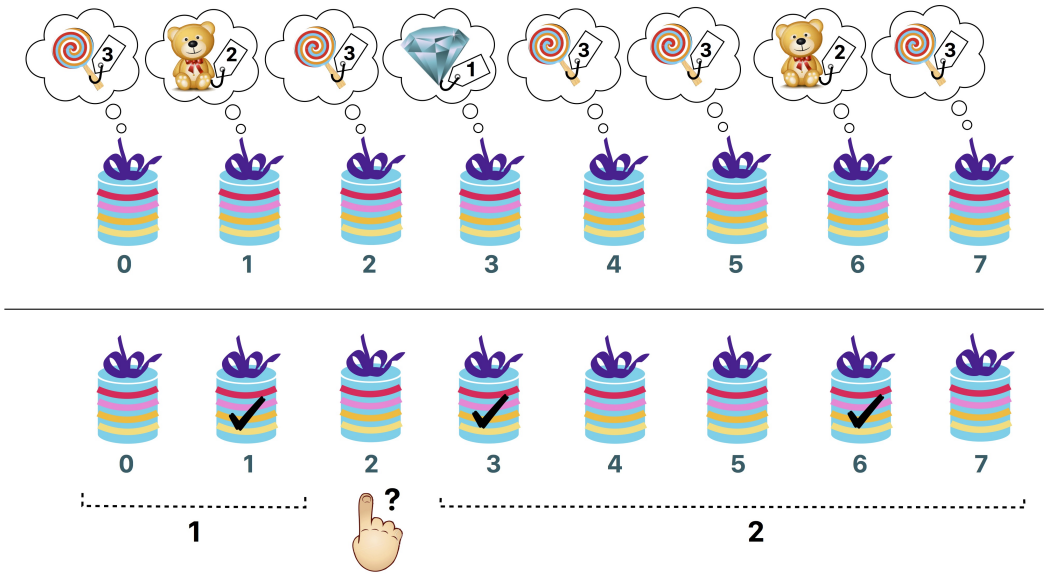
The grader makes the following procedure call:

```
find_best(8)
```

There are $n = 8$ boxes. Suppose the prize types are $[3, 2, 3, 1, 3, 3, 2, 3]$. All possible calls to the procedure `ask` and the corresponding return values are listed below.

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

In this example, the diamond is in box 3. So the procedure `find_best` should return 3.



The above figure illustrates this example. The upper part shows the values/types of the prizes in each box. The lower part illustrates the query `ask(2)`. The marked boxes contain more expensive prizes than the one in box 2.

Constraints

- $3 \leq n \leq 200\,000$.
- The type of the prize in each box is between 1 and v, inclusive.
- There is exactly one prize of type 1.
- For all $2 \leq t \leq v$, if there are k prizes of type $t-1$, there are strictly more than k^2 prizes of type t.

Subtasks and scoring

In some test cases the behavior of the grader is adaptive. This means that in these test cases the grader does not have a fixed sequence of prizes. Instead, the answers given by the grader may depend on the questions asked by your solution. It is guaranteed that the grader answers in such a way that after each answer there is at least one sequence of prizes consistent with all the answers given so far.

1. (20 points) There is exactly 1 diamond and $n-1$ lollipops (hence, $v = 2$). You can call the procedure `ask` at most $10\,000$ times.
2. (80 points) No additional constraints.

In subtask 2 you can obtain a partial score. Let q be the maximum number of calls to the procedure `ask` among all test cases in this subtask. Then, your score for this subtask is calculated according to the following table:

Questions	Score
$10\,000 < q$	0 (reported in CMS as 'Wrong Answer')
$6000 < q \leq 10\,000$	70
$5000 < q \leq 6000$	$80 - (q - 5000)/100$
$q \leq 5000$	80

Sample grader

The sample grader is not adaptive. Instead, it just reads and uses a fixed array p of prize types. For all $0 \leq b \leq n-1$, the type of the prize in box b is given as $p[b]$. The sample grader expects input in the following format:

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

The sample grader prints a single line containing the return value of `find_best` and the number of calls to the procedure `ask`.