The Big Prize

Stai partecipando a *The Big Prize*, famoso gioco a premi della TV iraniana, e sei ora in piedi di fronte ad una fila di n scatole numerate da 0 a $n-1$ da sinistra a destra. Ogni scatola contiene un premio che non è visibile fino a quando la scatola non viene aperta.

Esistono $v \geq 2$ diversi *tipi* di premi. I tipi sono numerati da 1 a v in ordine di valore **decrescente**: il premio di tipo 1 è il più prezioso (un diamante) mentre il premio di tipo v è il meno prezioso (una caramella). C'è sempre **esattamente un diamante** tra tutte le scatole.

Nelle scatole ci sono molti più premi scarsi che importanti: per ogni t tale che $2 \leq t \leq v$, se ci sono k premi di tipo $t-1$, allora ci sono *strettamente più* di k^2 premi di tipo t.

Alla fine del gioco potrai aprire una delle scatole ricevendo così il premio in essa contenuto, e ovviamente tu vuoi vincere il diamante. Prima di fare la tua scelta, però, puoi fare alcune domande a Rambod (il conduttore del gioco). Per ogni domanda potrai scegliere una certa scatola i, e Rambod ti risponderà con un array a contenente due numeri interi, tali per cui:

- Tra tutte le scatole a sinistra della scatola i ci sono esattamente $a[0]$ scatole contenenti un premio *strettamente più* prezioso di quello nella scatola i.
- Tra tutte le scatole a destra della scatola i ci sono esattamente $a[1]$ scatole contenenti un premio *strettamente più* prezioso di quello nella scatola i.

Supponiamo ad esempio che $n = 8$ e che come domanda tu scelga $i = 2$. Se la risposta di Rambod fosse $a = [1, 2]$, significherebbe che:

- Esattamente **una** tra le scatole 0 e 1 contiene un premio più prezioso di quello nella scatola 2.
- Esattamente **due** tra le scatole $3 \ldots 7$ contengono anch'esse un premio più prezioso di quello.

Il tuo obiettivo è trovare la scatola contenente il diamante facendo poche domande.

Dettagli di implementazione

Devi implementare la seguente funzione:

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

- n: il numero di scatole.
- Questa funzione viene chiamata esattamente una volta dal grader.
- La funzione deve restituire l'indice della scatola che contiene il diamante, ovvero, il numero intero d ($0 \leq d \leq n-1$) tale che la scatola d contiene un premio di tipo 1.

La funzione può effettuare chiamate alla seguente funzione ausiliaria definita nel grader:

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

- i: l'indice della scatola della quale stai chiedendo informazioni a Rambod, che deve essere compreso tra 0 e $n - 1$ estremi inclusi.
- La funzione restituisce un array a di 2 elementi tale per cui $a[0]$ è il numero di premi più preziosi nelle scatole a sinistra della scatola i, mentre $a[1]$ è il numero di premi più preziosi nelle scatole a destra della scatola i.

Esempio

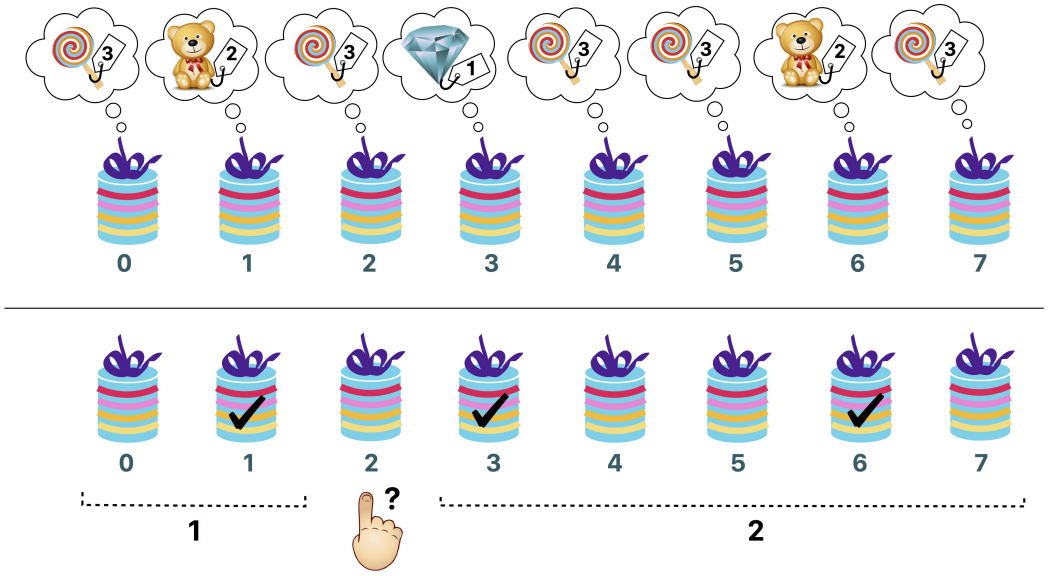
Il grader effettua la seguente chiamata a funzione:

```
find_best(8)
```

Ci sono $n = 8$ scatole contenenti premi di tipi $[3, 2, 3, 1, 3, 3, 2, 3]$. Qui sotto sono mostrate tutte le possibili chiamate alla funzione ausiliaria `ask` ed i relativi valori di ritorno.

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

In questo esempio, il diamante si trova nella scatola 3 quindi `find_best` deve restituire 3.



La figura qui sopra illustra l'esempio: la parte superiore mostra i tipi dei premi in ciascuna scatola, e la parte inferiore illustra la domanda `ask(2)` fatta a Rambod. Le scatole marcate con il simbolo ✔ contengono premi più preziosi di quello nella scatola 2.

Assunzioni

- $3 \leq n \leq 200\,000$.
- Il tipo dei premi in ciascuna scatola è compreso tra 1 e v estremi inclusi.
- C'è **esattamente un** premio di tipo 1.
- Per ogni $2 \leq t \leq v$, se ci sono k premi di tipo $t-1$ allora ci sono *strettamente più* di k^2 premi di tipo t.

Assegnazione del punteggio

In alcuni testcase il comportamento del grader è adattivo, cioè il grader non ha una sequenza fissa di premi. Quindi, le risposte date dal grader potrebbero dipendere dalle domande effettuate dalla tua soluzione. Viene però garantito che il grader risponda in un modo tale che ci sia sempre almeno una sequenza di premi consistente con tutte le risposte date fino a quel momento.

1. **(20 points)** Ci sono esattamente 1 diamante e $n-1$ caramelle (ovvero $v = 2$), e puoi chiamare la funzione ausiliaria `ask` al massimo $10\,000$ volte.
2. **(80 points)** Nessuna limitazione specifica.

Nel subtask 2 è possibile ottenere punteggi parziali. Se q è il massimo numero di chiamate alla funzione ausiliaria `ask` tra tutti i testcase in questo subtask, il tuo punteggio nel subtask è calcolato secondo la seguente tabella:

Domande	Punteggio
$10\,000 < q$	0 (indicato sul server di correzione come 'Wrong Answer')
$6000 < q \leq 10\,000$	70
$5000 < q \leq 6000$	$80 - (q - 5000)/100$
$q \leq 5000$	80

Grader di prova

Il grader di esempio non è adattivo: semplicemente, legge un array fisso p di tipi di premi e lo usa. Per ogni $0 \leq b \leq n-1$, il tipo di premio nella scatola b è dato da $p[b]$. Il grader di prova legge l'input nel seguente formato:

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

Il grader di prova stampa una sola riga contenente il valore restituito da `find_best` ed il numero di chiamate alla funzione ausiliaria `ask`.