



The Big Prize

O Grande Prêmio é um famoso concurso na TV. Você é o sortudo competidor que avançou à rodada final. Você está diante de uma sequência de n caixas, rotuladas de 0 a $n - 1$ da esquerda para a direita. Cada caixa contém um prêmio que não pode ser visto até que a caixa seja aberta. Existem $v \geq 2$ diferentes tipos de prêmios. Os tipos são numerados de 1 a v em ordem decrescente de valor.

O prêmio do tipo 1 é o mais caro: um diamante. Há exatamente um diamante nas caixas. O prêmio do tipo v é o mais barato: um pirulito. Para tornar o jogo mais excitante, o número de prêmios mais baratos é muito maior do que o número de prêmios mais caros. Mais especificamente, para todo t tal que $2 \leq t \leq v$ nós sabemos o seguinte: se existem k prêmios do tipo $t - 1$, então existem *estritamente mais* do que k^2 prêmios do tipo t .

Seu objetivo é ganhar o diamante. No final do jogo você terá que abrir uma caixa e você receberá o prêmio que ela contiver. Antes de ter que escolher a caixa a ser aberta você poderá fazer a Rambod, o apresentador do programa, algumas questões. Para cada questão, você escolhe alguma caixa i . Como resposta, Rambod lhe dará um vetor a contendo dois inteiros. Seu significado é o seguinte:

- Entre as caixas à esquerda da caixa i existem exatamente $a[0]$ caixas que contém um prêmio mais caro do que o da caixa i .
- Entre as caixas à direita da caixa i existem exatamente $a[1]$ caixas que contém um prêmio mais caro do que o da caixa i .

Por exemplo, suponha que $n = 8$. Para a sua questão, você escolhe a caixa $i = 2$. Como sua resposta, Rambod lhe diz que $a = [1, 2]$. O significado desta resposta é:

- Exatamente uma das caixas 0 e 1 contém um prêmio mais caro do que o da caixa 2.
- Exatamente duas das caixas 3, 4, ..., 7 contêm um prêmio mais caro do que o da caixa 2.

Sua tarefa é encontrar a caixa contendo o diamante fazendo um número pequeno de questões.

Detalhes de implementação

Você deve implementar a seguinte função:

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

- A função será chamada exatamente uma vez pelo corretor.

- n : o número de caixas.
- A função deve retornar o rótulo da caixa que contém o diamante, isto é, o único inteiro d ($0 \leq d \leq n - 1$) tal que a caixa d contém o prêmio do tipo 1.

A função anterior pode fazer chamadas à seguinte função:

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

- i : rótulo da caixa sobre qual você escolher questionar. O valor de i tem que estar entre 0 e $n - 1$, inclusive.
- Esta função retorna o vetor a com 2 elementos. Aqui, $a[0]$ é o número de prêmios mais caros nas caixas à esquerda da caixa i e $a[1]$ é o número de prêmios mais caros nas caixas à direita da caixa i .

Exemplo

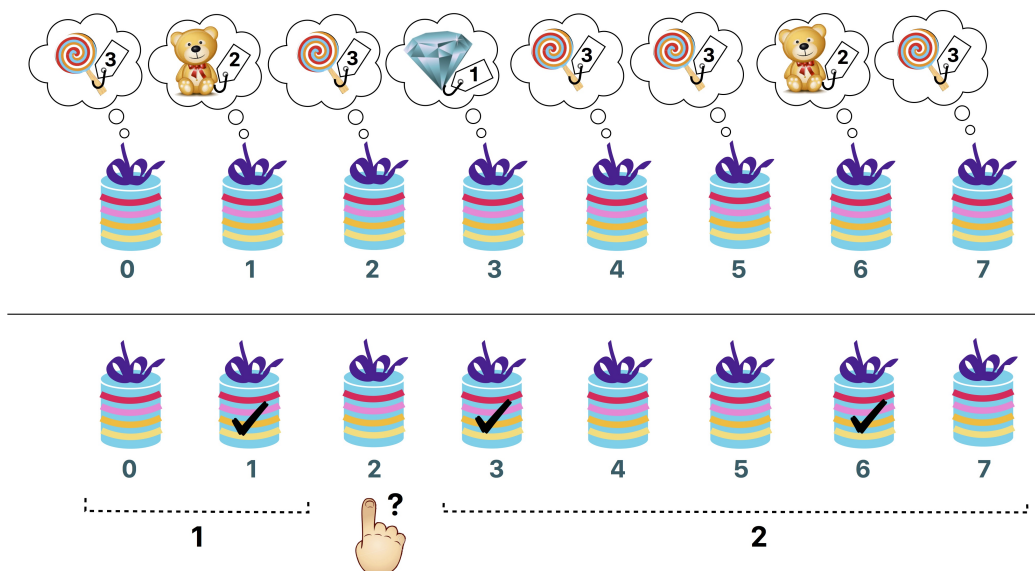
O corretor faz a seguinte chamada de função:

```
find_best(8)
```

Existem $n = 8$ caixas. Suponha que os tipos de prêmios sejam $[3, 2, 3, 1, 3, 3, 2, 3]$. Todas as possíveis chamadas à função `ask` e os correspondentes valores de retorno estão listados a seguir.

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

Neste exemplo, o diamante está na caixa 3. Portanto a função `find_best` deve retornar 3.



A figura anterior ilustra este exemplo. A parte de cima mostra os tipos dos prêmios em cada caixa. A parte de baixo ilustra a questão `ask(2)`. As caixas marcadas contêm prêmios mais caros do que o da caixa 2.

Restrições

- $3 \leq n \leq 200\,000$.
- O tipo do prêmio em cada caixa está entre 1 e v , inclusive.
- Há exatamente um prêmio do tipo 1.
- Para todo $2 \leq t \leq v$, se existem k prêmios do tipo $t - 1$, então existem *estritamente* mais do que k^2 prêmios do tipo t .

Sub-tarefas e pontuação

Em alguns casos de teste o comportamento do corretor é adaptativo. Isto significa que nestes casos de teste o corretor não tem uma sequência fixa de prêmios. Ao invés disso, as respostas dadas pelo corretor podem depender das questões perguntadas por sua solução.

É garantido que o corretor responde de tal maneira que depois de cada resposta existe pelo menos uma sequência de prêmios consistente com todas as respostas dadas até então.

1. (20 pontos) Existe exatamente 1 diamante e $n - 1$ pirulitos (assim, $v = 2$). Você pode chamar a função `ask` no máximo 10 000 vezes.
2. (80 pontos) Nenhuma restrição adicional.

Na sub-tarefa 2 você pode obter uma pontuação parcial. Seja q o número máximo de chamadas à função `ask` entre todos os casos de teste nesta sub-tarefa. Então, sua pontuação para esta sub-tarefa é calculada de acordo com a seguinte tabela:

Questões	Pontuação
$10\,000 < q$	0 (reportado no CMS como 'Wrong Answer')
$6000 < q \leq 10\,000$	70
$5000 < q \leq 6000$	$80 - (q - 5000)/100$
$q \leq 5000$	80

Corretor exemplo

O corretor exemplo não é adaptativo. Ao invés disso, ele apenas lê e usa um vetor fixo p de tipos de prêmios. Para todo $0 \leq b \leq n - 1$, o tipo do prêmio na caixa b é dado como $p[b]$. O corretor exemplo espera a entrada no seguinte formato:

- linha 1: n
- linha 2: $p[0] \ p[1] \ \dots \ p[n - 1]$

O corretor exemplo imprime uma única linha contendo o valor de retorno de `find_best` e o número de chamadas à função `ask`.