Le Gros Prix (The Big Prize)

Le Gros Prix est un jeu télévisé célèbre. Vous avez la chance d'être le candidat sélectionné pour la finale. Vous vous trouvez devant une rangée de n boîtes, numérotées de 0 à $n-1$ de gauche à droite. Chaque boîte contient un prix qui est invisible tant que la boîte n'a pas été ouverte. Il y a $v \geq 2$ *types* de prix distincts. Les types sont numérotés de 1 à v dans l'ordre *décroissant* de leur valeur.

Le prix de type 1 est celui qui a la plus grande valeur : un diamant. Il y a exactement un diamant parmi les boîtes. Le prix de type v est celui dont la valeur est la plus faible : une sucette. Pour que le jeu soit plus amusant, il y a beaucoup plus de prix de faible valeur que de prix de grande valeur. Plus précisément, pour tout t tel que $2 \leq t \leq v$, vous savez que : s'il y a k prix de type $t-1$, il y a *strictement plus* de k^2 prix de type t.

Votre but est de gagner le diamant. À la fin du jeu, vous devrez ouvrir une des boîtes, et vous recevrez le prix qu'elle contient. Avant de choisir la boîte à ouvrir, vous pouvez poser quelques questions à Rambod, le présentateur du jeu. Pour chaque question, vous choisissez une boîte i. La réponse de Rambod sera un tableau a contenant 2 entiers. Leur signification est la suivante :

- Parmi les boîtes qui se trouvent à gauche de la boîte i, exactement $a[0]$ boîtes contiennent un prix de valeur strictement supérieure à celui de la boîte i.
- Parmi les boîtes qui se trouvent à droite de la boîte i, exactement $a[1]$ boîtes contiennent un prix de valeur strictement supérieure à celui de la boîte i.

Par exemple, supposons que $n = 8$. Vous choisissez la boîte $i = 2$ pour votre question. La réponse de Rambod est $a = [1, 2]$. La signification de cette réponse est la suivante :

- Exactement une des boîtes 0 et 1 contient un prix de valeur supérieure à celui de la boîte 2.
- Exactement deux des boîtes $3, 4, \ldots, 7$ contiennent un prix de valeur supérieure à celui de la boîte 2.

Vous devez trouver la boîte qui contient le diamant en posant un nombre limité de questions.

Détails d'implémentation

Vous devez implémenter la fonction suivante :

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

- Cette fonction est appelée exactement une fois par l'évaluateur.

- n : nombre de boîtes.
- La fonction doit renvoyer le numéro de la boîte qui contient le diamant, c'est-à-dire l'unique entier d $(0 \leq d \leq n-1)$ tel que la boîte d contient un prix de type 1.

Cette fonction peut effectuer des appels à la fonction suivante :

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

- i : numéro de la boîte pour laquelle vous posez une question. La valeur de i doit être entre 0 et $n-1$ inclus.
- Cette fonction renvoie un tableau a de 2 éléments, où $a[0]$ est le nombre de prix de valeur supérieure parmi les boîtes à gauche de la boîte i, et $a[1]$ est le nombre de prix de valeur supérieure parmi les boîtes à droite de la boîte i.

Exemple

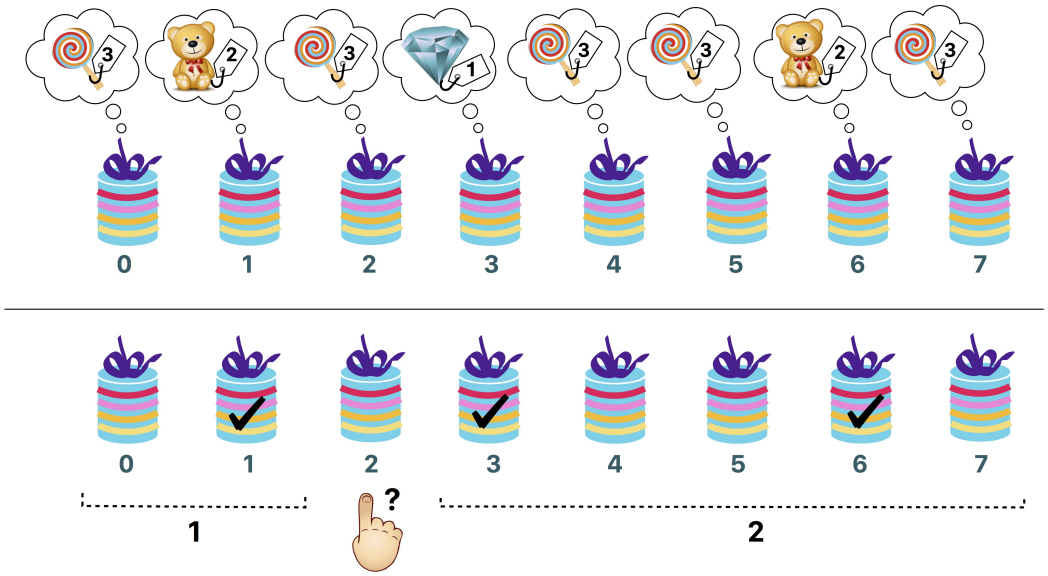
L'évaluateur effectue l'appel de fonction suivant :

```
find_best(8)
```

Il y a $n = 8$ boîtes. Supposons que les types de prix sont $[3, 2, 3, 1, 3, 3, 2, 3]$. Tous les appels possibles à la fonction `ask`, et les valeurs de retour correspondantes, sont listés ci-dessous.

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

Sur cet exemple, le diamant se trouve dans la boîte 3. La fonction `find_best` devrait donc renvoyer 3.



La figure ci-dessus illustre cet exemple. La partie supérieure donne les types des prix dans chaque boîte. La partie inférieure illustre la requête `ask(2)`. Les boîtes avec une marque contiennent des prix de plus grande valeur que celui de la boîte 2.

Contraintes

- $3 \leq n \leq 200\,000$.
- Tous les types des prix sont entre 1 et v inclus.
- Il y a exactement un prix de type 1.
- Pour tout $2 \leq t \leq v$, s'il y a k prix de type $t-1$, il y a *strictement plus* de k^2 prix de type t.

Sous-tâches et score

Dans certains cas, le comportement de l'évaluateur est adaptatif. Cela signifie que dans ces cas-là, les prix de chaque boîte ne sont pas fixés à l'avance dans l'évaluateur. Au lieu de cela, les réponses données par l'évaluateur peuvent dépendre des questions posées par votre programme. Il est garanti que l'évaluateur répond de manière à ce qu'après chaque réponse, il existe au moins une attribution des prix cohérente avec toutes les réponses données jusqu'alors.

1. (20 points) Il y a exactement 1 diamant et $n-1$ sucettes (et donc $v=2$). Vous pouvez appeler la fonction `ask` au plus $10\,000$ fois.
2. (80 points) Aucune contrainte supplémentaire.

Pour la sous-tâche 2, vous pouvez obtenir un score partiel. Soit q le nombre maximal d'appels à la fonction `ask` parmi tous les tests de cette sous-tâche. Votre score pour cette sous-tâche est alors calculé de la manière suivante :

Questions	Score
$10\,000 < q$	0 (sera considéré comme 'Wrong Answer' par CMS)
$6000 < q \leq 10\,000$	70
$5000 < q \leq 6000$	$80 - (q - 5000)/100$
$q \leq 5000$	80

Évaluateur d'exemple

L'évaluateur d'exemple n'est pas adaptatif. Il utilise simplement un tableau fixé p de types de prix donné en entrée. Pour tout $0 \leq b \leq n - 1$, le type du prix se trouvant dans la boîte numéro b est $p[b]$. L'évaluateur d'exemple attend une entrée du format suivant :

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

L'évaluateur d'exemple affiche une unique ligne qui contient la valeur de retour de `find_best` et le nombre d'appels à la fonction `ask`.