# Взлом! (hack)

Прошел уже час с начала контеста на Codeforces, когда вы заметили, что другой участник в вашей комнате решил задачу, используя `unordered_set`. Время для взлома!

Вы знаете, что `unordered_set` использует хеш-таблицу с $n$ корзинами, пронумерованными от $0$ до $n-1$. К сожалению, вы не знаете значение $n$ и хотите его восстановить.

Когда вы вставляете целое число $x$ в хеш-таблицу, оно помещается в корзину под номером $(x \bmod n)$ . Если в этой корзине уже находится $b$ элементов, то вставка вызовет $b$ хеш-коллизий.

Предоставляя интерактору $k$ различных целых чисел $x[0], x[1], \ldots, x[k-1]$, вы можете узнать общее количество хеш-коллизий, возникших при создании `unordered_set` из этих чисел. Однако передача интерактору $k$ чисел за один запрос будет стоить $k$.

Например, если $n = 5$, и вы передаете интерактору $x = [2, 15, 7, 27, 8, 30]$, то произойдет $4$ коллизии:

| Операция | Новые коллизии | Состояние корзин |
|---|---|---|
| *изначально* | $-$ | $[], [], [], [], []$ |
| вставка $x[0] = 2$ | $0$ | $[], [], [2], [], []$ |
| вставка $x[1] = 15$ | $0$ | $[15], [], [2], [], []$ |
| вставка $x[2] = 7$ | $1$ | $[15], [], [2, 7], [], []$ |
| вставка $x[3] = 27$ | $2$ | $[15], [], [2, 7, 27], [], []$ |
| вставка $x[4] = 8$ | $0$ | $[15], [], [2, 7, 27], [8], []$ |
| вставка $x[5] = 30$ | $1$ | $[15, 30], [], [2, 7, 27], [8], []$ |

Обратите внимание, что интерактор создает хеш-таблицу, вставляя элементы по порядку в изначально пустой `unordered_set`, и для каждого запроса создается новый пустой `unordered_set`. Другими словами, все запросы независимы.

Ваша задача — определить количество корзин $n$, используя суммарную стоимость не более $1\,000\,000$.

## Детали реализации

Вы должны реализовать следующую процедуру:

```
int hack()
```

- Эта процедура должна возвращать целое число — скрытое значение $n$.
- Для каждого теста грейдер может вызвать эту функцию более одного раза. Каждый вызов должен обрабатываться как совершенно новый сценарий.

Внутри этой процедуры вы можете вызывать следующую функцию:

```
long long collisions(std::vector<long long> x)
```

- $x$: массив различных чисел, где $1 \leq x[i] \leq 10^{18}$ для каждого $i$.
- Эта функция возвращает количество коллизий, возникших при вставке элементов $x$ в `unordered_set`.
- Эту функцию можно вызывать многократно. Сумма длин всех массивов $x$ в рамках одного вызова `hack()` не должна превышать $1\ 000\ 000$.

Примечание: **Поскольку функция `hack()` будет вызываться более одного раза, участникам следует обратить внимание на влияние оставшихся данных от предыдущего вызова на текущий вызов, особенно на состояние глобальных переменных.**

Ограничение в $1\ 000\ 000$ по стоимости применяется к каждому тестовому сценарию. В общем, если будет $t$ вызовов `hack()`, вы можете использовать общую стоимость не более $t \times 1\ 000\ 000$, при этом каждый отдельный вызов `hack()` должен использовать стоимость не более $1\ 000\ 000$.

Интерактор не адаптивен, то есть значения $n$ фиксированы до начала взаимодействия.

## Пример

Предположим, есть 2 тестовых сценария. Грейдер выполнит следующий вызов:

```
hack()
```

Допустим, внутри функции вы делаете следующие вызовы:

| Вызов | Возвращенное значение |
|---|---|
| `collisions([2, 15, 7, 27, 8, 30])` | $4$ |
| `collisions([1, 2, 3])` | $0$ |
| `collisions([10, 20, 30, 40, 50])` | $10$ |

После этого, если вы определили, что значение $n$ равно $5$, процедура `hack()` должна вернуть $5$.

Затем грейдер выполнит следующий вызов:

```
hack()
```

Допустим, внутри функции вы делаете следующие вызовы:

| Вызов | Возвращенное значение |
|---|---|
| `collisions([1, 3])` | $1$ |
| `collisions([2, 4])` | $1$ |

Единственное значение $n$, которое удовлетворяет этим запросам — это $2$. Следовательно, процедура `hack()` должна вернуть $2$.

## Ограничения

- $1 \le t \le 10$, где $t$ — количество тестовых сценариев.
- $2 \le n \le 10^9$
- $1 \le x[i] \le 10^{18}$ для каждого вызова `collisions()`.

## Подзадачи

1. (8 баллов) $n \le 500\ 000$
2. (17 баллов) $n \le 1\ 000\ 000$
3. (75 баллов) Без дополнительных ограничений.

В последней подзадаче можно получить частичный балл. Пусть $q$ — максимальная суммарная стоимость среди всех вызовов `hack()` по каждому тестовому сценарию подзадачи. Ваш балл за эту подзадачу рассчитывается по следующей таблице:

| Условие | Баллы |
|---|---|
| $1\ 000\ 000 < q$ | $0$ |
| $110\ 000 < q \leq 1\ 000\ 000$ | $75 \cdot \log_{50}\left(\frac{10^6}{x-90000}\right)$ |
| $q \leq 110\ 000$ | $75$ |

Если в любом из тестов вызовы функции `collisions()` не соответствуют описанным ограничениям в разделе Детали реализации или возвращенное число из `hack()` неверно, то ваш балл за эту подзадачу будет равен 0.

## Пример грейдера

Грейдер читает вход в следующем формате:

- строка $1$: $t$

Далее следуют $t$ строк, каждая из которых содержит значение $n$:

- строка $1$: $n$

Для каждого тестового сценария пусть $m$ — возвращаемое значение из `hack()`, а $c$ — общая стоимость всех запросов. Грейдер выведет ваш ответ в следующем формате:

- строка $1$: $m\ c$