Hack! (hack)

Sau một giờ trôi qua trong cuộc thi Codeforces, bạn nhận thấy một thí sinh khác trong phòng của bạn đã giải được một bài toán bằng cách sử dụng `unordered_set`. Đã đến lúc hack!

Bạn biết rằng `unordered_set` sử dụng bảng băm với n nhóm, được đánh số từ 0 đến $n-1$. Thật không may, bạn không biết giá trị của n và muốn tìm nó.

Khi bạn chèn một số nguyên x vào bảng băm, nó được chèn vào nhóm thứ $(x \bmod n)$. Nếu có b phần tử trong nhóm này trước khi chèn, điều này sẽ gây ra b xung đột.

Bằng cách cung cấp k số nguyên $x[0], x[1], \ldots, x[k-1]$ phân biệt cho trình tương tác, bạn có thể tìm ra tổng số xung đột băm đã xảy ra trong khi tạo một `unordered_set` chứa các số trên. Tuy nhiên, việc cung cấp cho trình tương tác này k số nguyên trong một truy vấn sẽ phải chịu chi phí là k.

Ví dụ: nếu $n = 5$, việc cung cấp cho trình tương tác $x = [2, 15, 7, 27, 8, 30]$ sẽ gây ra tổng cộng 4 xung đột:

Thao tác	Số xung đột	Các nhóm
Ban đầu	$-$	$[], [], [], [], []$
Chèn $x[0] = 2$	0	$[], [], [2], [], []$
Chèn $x[1] = 15$	0	$[15], [], [2], [], []$
Chèn $x[2] = 7$	1	$[15], [], [2, 7], [], []$
Chèn $x[3] = 27$	2	$[15], [], [2, 7, 27], [], []$
Chèn $x[4] = 8$	0	$[15], [], [2, 7, 27], [8], []$
Chèn $x[5] = 30$	1	$[15, 30], [], [2, 7, 27], [8], []$

Lưu ý rằng trình tương tác tạo bảng băm bằng cách chèn các phần tử theo thứ tự vào một `unordered_set` rỗng ban đầu và một `unordered_set` rỗng mới sẽ được tạo cho mỗi truy vấn. Nói cách khác, tất cả các truy vấn là độc lập.

Nhiệm vụ của bạn là tìm ra số nhóm n với tổng chi phí tối đa là $1\,000\,000$.

Chi tiết cài đặt

Bạn cần cài đặt hàm sau đây:

```
int hack()
```

- Hàm này cần trả về một số nguyên – giá trị n không cho biết.
- Với mỗi test, trình chấm có thể gọi hàm nhiều hơn một lần. Mỗi lời gọi cần được xử lý như là một kịch bản thử nghiệm mới riêng biệt.

Trong hàm trên, bạn có thể gọi hàm sau:

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

- x: một mảng các số nguyên phân biệt, trong đó $1 \leq x[i] \leq 10^{18}$ với mỗi i.
- Hàm này trả về số lần xung đột được tạo ra bằng cách chèn các phần tử của x vào `unordered_set`.
- Hàm này có thể được gọi nhiều lần. Tổng độ dài của x trên tất cả các lời gọi hàm này trong một lời gọi đến `hack()` không được vượt quá $1\ 000\ 000$.

Lưu ý: **Vì hàm `hack()` sẽ được gọi nhiều lần, nên thí sinh cần chú ý đến tác động của dữ liệu còn lại của lần gọi trước lên lần gọi hiện tại, đặc biệt là trạng thái được lưu trữ trong các biến toàn cục.**

Giới hạn chi phí $1\ 000\ 000$ áp dụng cho mỗi kịch bản thử nghiệm. Nhìn chung, nếu có t lời gọi đến `hack()`, bạn có thể sử dụng tổng chi phí không quá $t \times 1\ 000\ 000$, với mỗi lời gọi riêng lẻ đến `hack()` sử dụng chi phí không quá $1\ 000\ 000$.

Trình tương tác ở đây là không thích ứng, nghĩa là các giá trị của n được cố định trước khi bắt đầu tương tác.

Ví dụ

Giả sử có 2 kịch bản thử nghiệm. Trình chấm sẽ đưa ra lời gọi sau:

```
hack()
```

Giả sử, bên trong hàm, bạn thực hiện các lời gọi sau:

Lời gọi	Giá trị trả về
`collisions([2, 15, 7, 27, 8, 30])`	4
`collisions([1, 2, 3])`	0
`collisions([10, 20, 30, 40, 50])`	10

Sau đó, nếu bạn tìm thấy giá trị của n là 5, hàm `hack()` cần trả về 5.

Sau đó, trình chấm sẽ thực hiện một lời gọi khác:

```
hack()
```

Giả sử, bên trong hàm, bạn thực hiện các lời gọi sau:

Lời gọi	Giá trị trả về
`collisions([1, 3])`	1
`collisions([2, 4])`	1

Số n duy nhất thỏa mãn các truy vấn là 2. Vì vậy, hàm `hack()` cần trả về 2.

Ràng buộc

- $1 \leq t \leq 10$, trong đó t là số lượng kịch bản thử nghiệm.
- $2 \leq n \leq 10^9$
- $1 \leq x[i] \leq 10^{18}$ cho mỗi lần gọi đến `collisions()`.

Subtask

1. (8 điểm) $n \leq 500\ 000$
2. (17 điểm) $n \leq 1\ 000\ 000$
3. (75 điểm) Không có ràng buộc nào thêm.

Trong subtask cuối cùng, bạn có thể nhận được một phần điểm số. Giả sử q là tổng chi phí lớn nhất trong số tất cả các lần gọi `hack()` trên mọi trường hợp thử nghiệm của subtask. Điểm của bạn cho subtask này được tính theo bảng sau:

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

Nếu trong bất kỳ test nào, các lệnh gọi đến hàm `collisions()` không tuân thủ các ràng buộc được mô tả trong **Chi tiết cài đặt** hoặc số trả về bởi `hack()` không chính xác, thì điểm cho giải pháp của bạn cho subtask đó sẽ là 0.

Trình chấm mẫu

Trình chấm mẫu đọc đầu vào theo định dạng sau:

- Dòng 1: t

Sau đó, t dòng tiếp theo, mỗi dòng chứa một giá trị:

- Dòng 1: n

Đối với mỗi kịch bản thử nghiệm, gọi m là giá trị trả về của `hack()` và c là tổng chi phí của tất cả các truy vấn. Trình chấm mẫu ghi câu trả lời của bạn theo định dạng sau:

- Dòng 1: m c