Hack! (hack)

在 Codeforces 競賽進行了一個小時之後，你注意到你房間的另一名選手利用了 `unordered_set` 解決了一道問題。是時候該 hack 一下了！

你知道 `unordered_set` 利用了一個具有 n 格位置的雜湊表實作而得。這些格子的編號從 0 至 $n-1$。雖然你不知道 n 值為何，但你希望找出這個值。

當你加入一個整數 x 到這個雜湊表，它會被放到第 $(x \bmod n)$ 格中。如果在這個加入操作之前，這格子中就已經有了 b 個數字，那麼這個操作會導致 b 次碰撞。

現在有一位互動精靈，她支援以下的詢問操作。每一次詢問操作，可以給她 k 個相異整數 $x[0], x[1], \ldots, x[k-1]$，她會告訴你，如果她產生一個上述的雜湊表，將這 k 個數字依序加入雜湊表，總共會產生多少碰撞。進行一次這樣的詢問操作，代價為 k。

這邊舉個例子。如果 $n = 5$，而且你在一次詢問操作給了互動精靈 $x = [2, 15, 7, 27, 8, 30]$，那麼總共會造成 4 次碰撞。

操作	新造成的碰撞數	格子狀態
初始	$-$	$[], [], [], [], []$
加入 $x[0] = 2$	0	$[], [], [2], [], []$
加入 $x[1] = 15$	0	$[15], [], [2], [], []$
加入 $x[2] = 7$	1	$[15], [], [2, 7], [], []$
加入 $x[3] = 27$	2	$[15], [], [2, 7, 27], [], []$
加入 $x[4] = 8$	0	$[15], [], [2, 7, 27], [8], []$
加入 $x[5] = 30$	1	$[15, 30], [], [2, 7, 27], [8], []$

注意！互動精靈在面對每一次詢問操作時，都是重新建造出一個新的雜湊表，再把要加入的數字們依序加入。換句話說，所有詢問操作是不相干的。

你的任務是：找出這個 n 格雜湊表的 n 值是多少，只能花費至多 $1\ 000\ 000$ 的代價。

實作細節 (Implementation details)

你需要實做以下函式：

```
int hack()
```

- 這個函式的回傳值是 – 想找出的 n 值。
- 對每一筆測試資料，評分程式可能會呼叫上述函式超過一次。每一次呼叫，都是一次全新的狀態。

在上面這函式中，你或許會呼叫下面這函式：

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

- x: 一個具有相異數字的陣列。每一個 i 都滿足 $1 \leq x[i] \leq 10^{18}$。
- 這函式的回傳值是將 x 中的數字們依序加入 `unordered_set` 會導致的碰撞次數。
- 這函式可以被呼叫不只一次。在一次 `hack()` 的執行中，這個函式的所有呼叫，所使用 x 長度加總起來，不能超過 $1\ 000\ 000$。

注意！既然 **`hack()`** 函式會被呼叫超過一次，參賽選手們需要留意在前幾次呼叫時殘存的資料是否會對之後的呼叫造成影響，特別是一些全域變數的存取。

對於每一筆測試資料，所花費的代價上限為 $1\ 000\ 000$。如果 `hack()` 呼叫了 t 次，那麼總花費代價不超過 $t \times 1\ 000\ 000$，而且 `hack()` 每一次呼叫所花費的代價不超過 $1\ 000\ 000$。

互動精靈不會根據與你的函式互動而改變她的行為，也就是說，n 值在互動開始之前，就已經固定了。

範例 (Example)

假如你有兩筆測資，評分程式會呼叫下面的函式：

```
hack()
```

假設在這次 `hack()` 函式呼叫中，你做了下面的呼叫：

呼叫	回傳值
`collisions([2, 15, 7, 27, 8, 30])`	4
`collisions([1, 2, 3])`	0
`collisions([10, 20, 30, 40, 50])`	10

完成上面的呼叫後，如果你發現了 n 值是 5，那麼 `hack()` 函式應該回傳 5。

接著，評分程式會進行另一次的呼叫：

```
hack()
```

假設在這次 `hack()` 函式呼叫中，你做了下面的呼叫：

呼叫	回傳值
`collisions([1, 3])`	1
`collisions([2, 4])`	1

這時，唯一可以滿足上面這些詢問操作的 n 值只有 2。所以，`hack()` 函式應該回傳 2。

限制 (Constraints)

- $1 \leq t \leq 10$, 其中 t 是測資的筆數。
- $2 \leq n \leq 10^9$
- $1 \leq x[i] \leq 10^{18}$ 在每一次的 `collisions()` 呼叫中。

子任務 (Subtasks)

1. (8 points) $n \leq 500\ 000$
2. (17 points) $n \leq 1\ 000\ 000$
3. (75 points) 無額外限制。

在最後一個子任務中，你可以得到部分分數。你所實作的 `hack()` 函式，在這個子任務中的每一筆測資，所花費的代價們取一個最大值。令這個最大值為 q。你在這個子任務的評分將由下面這表格計算：

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

如果，在任何一筆測試資料，在呼叫 `collisions()` 函式時，沒有遵守實作的規範，或者 `hack()` 回傳值有錯，那麼你在該子任務的得分將為 0。

範例評分程式 (Sample Grader)

範例評分程式以下面的格式讀入輸入：

- line 1: t

接著有 t 行，每一行有一個 n 值：

- line 1: n

對於每一筆測資，令 m 為 `hack()` 的回傳值，令 c 全部詢問操作的總代價。範例評分程式會以下面的格式將你的回答輸出。

- line 1: m c