Hack! (hack)

你参加了一场 Codeforces 的编程比赛，现在离比赛结束只有一个小时了。你发现同房间的另外一位选手通过使用 `unordered_set` 通过了一道题目。是时候把他的代码 hack 掉了！

你知道 `unordered_set` 是使用一个包含 n 个桶的哈希表实现的，其中桶的编号为 0 到 $n-1$。不过很可惜，你并不知道 n 具体是多少，所以你希望通过下面的操作将 n 还原出来。

当你将一个整数 x 插入哈希表时，它将会被插入到第 $(x \bmod n)$ 个桶中。如果在这次插入之前这个桶中已经有 b 个元素，这将会导致 b 次哈希冲突的产生。

每次操作中，你可以向交互库提交 k 个不同的整数 $x[0], x[1], \ldots, x[k-1]$ 进行查询，交互库会依次将这些整数插入到哈希表中，并返回创建包含这些数字的哈希表会引起的哈希冲突总次数。然而，向交互库提交一次包含 k 个不同的整数的查询需要 k 的花费。

例如，当 $n = 5$ 时，如果你向交互库提交数组 $x = [2, 15, 7, 27, 8, 30]$，将会引起总共 4 次哈希冲突，具体如下：

操作	新增哈希冲突次数	桶状态
初始状态	$-$	$[], [], [], [], []$
插入 $x[0] = 2$	0	$[], [], [2], [], []$
插入 $x[1] = 15$	0	$[15], [], [2], [], []$
插入 $x[2] = 7$	1	$[15], [], [2, 7], [], []$
插入 $x[3] = 27$	2	$[15], [], [2, 7, 27], [], []$
插入 $x[4] = 8$	0	$[15], [], [2, 7, 27], [8], []$
插入 $x[5] = 30$	1	$[15, 30], [], [2, 7, 27], [8], []$

请注意：交互库在每次你提交的时候都将 `unordered_set` 初始化为空集，然后把提交的数字依次插入来创建哈希表。也就是说，每一次的交互查询是相互独立的。

你的任务是使用不超过 $1\,000\,000$ 的花费求出桶的数量 n。

实现细节

你需要实现以下函数：

```
int hack()
```

- 该函数返回一个整数 n。
- 对于每个测试点，评测程序可能会调用该函数多于一次。每次调用都应该当做新的情况分别处理。

在这个函数中，你可能会调用以下交互函数：

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

- x: 一个包含不同整数的数组，其中对于所有 i，满足 $1 \le x[i] \le 10^{18}$。
- 该函数返回一个整数，表示将数组 x 中所有元素依次插入哈希表引起的哈希冲突总次数。
- 该函数可以多次调用。在一次 `hack()` 的调用中，多次调用的数组 x 的总长度不能超过 $1\ 000\ 000$。

注意：由于 **`hack()`** 函数调用可能会发生多次，选手需要注意之前调用的残余数据对于后续调用的影响，尤其是全局变量的状态。

$1\ 000\ 000$ 的花费限制应用于每一组测试数据。即，如果 `hack()` 函数被调用了 t 次，你可以使用不超过 $t \times 1\ 000\ 000$ 的总花费，并且每次独立调用 `hack()` 时的花费不能超过 $1\ 000\ 000$。

n 的值在交互函数调用前已经固定。

例子

假设有两组测试用例，评测程序将首先调用 `hack()` 函数：

```
hack()
```

在这个函数中，你可以进行以下调用：

函数调用	返回值
`collisions([2, 15, 7, 27, 8, 30])`	4
`collisions([1, 2, 3])`	0
`collisions([10, 20, 30, 40, 50])`	10

如果你还原出 $n = 5$， 那么函数 `hack()` 返回 5。

接下来，评测程序将再一次调用 `hack()` 函数：

```
hack()
```

在这个函数中，你可以进行以下调用：

函数调用	返回值
`collisions([1, 3])`	1
`collisions([2, 4])`	1

你从上述调用中还原出唯一满足的 n 是 2， 那么函数 `hack()` 返回 2。

约束条件

- $1 \leq t \leq 10$，其中 t 为每组测试点中的测试用例数量。
- $2 \leq n \leq 10^9$
- 对于每次 `collisions()` 调用，$1 \leq x[i] \leq 10^{18}$

子任务

1. (8 分) $n \leq 500\ 000$
2. (17 分) $n \leq 1\ 000\ 000$
3. (75 分) 没有额外的约束条件。

在最后一个子任务中，你可以获得部分分。令 q 为该子任务下所有测试用例 `hack()` 函数中的最大总花费。该子任务的部分分计算如下：

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

在任意测试用例中，如果对 `collisions()` 函数调用不满足实现细节中的约束条件，或者 `hack()` 函数调用的返回值错误，该子任务的分数为 0。

评测程序示例

评测程序示例按以下格式读取输入：

- 第 1 行：t

对于接下来 t 组数据的每一组：

- 第 1 行：n

对于每组测试用例，令 m 为函数 `hack()` 的返回值，c 为所有查询的总花费。评测程序示例按以下格式打印你的答案：

- 第 1 行：$m\ c$