Problem D. todd_and_steven

This contest is open for practice. You can try every problem as many times as you like, though we won't keep track of which problems you solve. Read the Quick-Start Guide to get started.

small 1 points *2 minute timeout*	The contest is finished.
large 30 points *10 minute timeout*	The contest is finished.

Problem

Todd and Steven

By now, it is a programming interview cliché: How do you sort a very large sequence of unique integers in increasing order? Finally, we are ready to reveal the correct answer:

1. Give all of the odd integers from the sequence to one assistant named Todd, and ask Todd to sort those integers in increasing order.
2. Give all of the even integers from the sequence to another assistant named Steven, and ask Steven to sort those integers in increasing order.
3. Merge Todd's sequence with Steven's sequence to produce the final sorted sequence.

Todd and Steven have already performed steps 1 and 2 on a certain sequence, and now we would like you to perform step 3. Since the resulting sorted sequence can be very long, we only ask you to output a hash of it as proof that you found it. Let X_j denote the j-th element (counting starting from 0) of the merged sequence. Please find the sum, over all j, of (X_j) XOR j. (Here XOR refers to the bitwise operation between the two integers, represented in both C++ and Java by the operator ^.) Since the output can be a really big number, we only ask you to output the remainder of dividing the result by the prime 10^9+7 (1000000007).

Input

The input library is called "todd_and_steven"; see the sample inputs below for examples in your language. It defines two methods:

- **GetToddLength()**:
  - Takes no argument.
  - Returns a 64-bit integer: the number of values in Todd's sorted sequence.
  - Expect each call to take 0.05 microseconds.
- **GetToddValue(i)**:
  - Takes a 64-bit integer argument in the range 0 ≤ i < `GetToddLength()`.
  - Returns a 64-bit integer: the i-th value of Todd's sorted sequence.
  - Expect each call to take 0.05 microseconds.
- **GetStevenLength()**:
  - Takes no argument.
  - Returns a 64-bit integer: the number of values in Steven's sorted sequence.
  - Expect each call to take 0.05 microseconds.
- **GetStevenValue(i)**:
  - Takes a 64-bit integer argument in the range 0 ≤ i < `GetStevenLength()`.
  - Returns a 64-bit integer: the i-th value of Steven's sorted sequence.
  - Expect each call to take 0.05 microseconds.

Output

Output one line with a single 64-bit integer: the sum described in the problem statement, modulo the prime 10^9+7 (1000000007).

Limits

Time limit: 4 seconds.
Memory limit per node: 128 MB.
Maximum number of messages a single node can send: 1000.
Maximum total size of messages a single node can send: 8 MB.
$1 \le$ `GetToddValue(i)` $\le 5 \times 10^9$, for all i.
`GetToddValue(i)` < `GetToddValue(i + 1)`, for all i. (Todd's sequence is sorted in increasing order.)
`GetToddValue(i) % 2 = 1`, for all i. (All elements in Todd's sequence are odd.)

$1 \le$ `GetStevenValue(i)` $\le 5 \times 10^9$, for all i.
`GetStevenValue(i)` < `GetStevenValue(i + 1)` for all i. (Steven's sequence is sorted in increasing order.)
`GetStevenValue(i) % 2 = 0`, for all i. (All elements in Steven's sequence are even.)

Small dataset

Number of nodes: 10.
$1 \le$ `GetToddLength()` $\le 10^6$.
$1 \le$ `GetStevenLength()` $\le 10^6$.

Large dataset

Number of nodes: 100.
$1 \le$ `GetToddLength()` $\le 10^9$.
$1 \le$ `GetStevenLength()` $\le 10^9$.

Sample

Input	Output
See input files below.	For sample input 1: 8 For sample input 2: 200 For sample input 3: 111

Sample input libraries:
Sample input for test 1: todd_and_steven.h [CPP] todd_and_steven.java [Java]
Sample input for test 2: todd_and_steven.h [CPP] todd_and_steven.java [Java]
Sample input for test 3: todd_and_steven.h [CPP] todd_and_steven.java [Java]