Task: CYK
Cyclic shifts

Olimpiada Informatyczna

XXVI OI, Stage II, Day two. Source file `cyk.*` Available memory: 64 MB. *14.02.2019*

Looking after a small child is hard work, especially if the child's idea of fun is playing with sequences of numbers. Bytalene has an increasing sequence consisting of n integers $a_1 < a_2 < \ldots < a_n$. You are invited to a game of asking queries in order to reveal the length of the sequence, i.e., the number n. Each query is of the following form:

You give Bytalene an integer x. The girl shifts the sequence cyclically by x elements, i.e., performs a sequence of x moves, each one moving the current first element to the end of the sequence. Having done that, Bytalene tells you the current value of the first element of the sequence.

To make your task more challenging, right before the game Bytalene chooses a certain x_0 herself and shift the sequence cyclically by x_0 elements. Thus, the resulting sequence that you start querying may not be increasing, but there certainly exists a cyclic shift that makes it so.

Will you be able to determine the sequence length in at most 100 queries?

Communication

Your program should use a library which allows querying Bytalene as well as reporting your final answer. To use the library, write the following in your program header:

- **C++:** `#include "cyklib.hpp"`
- **Python:** `from cyklib import circular_shift, give_answer`

The library provides the following two functions:

- `circular_shift`(x)
  Use this function to query Bytalene. The argument x ($0 \leq x \leq 10^9$) is the number of times Bytalene is to move the current first element to the end of the sequence. The function returns the value of the first element of the sequence after those operations.
  - **C++:** `long long circular_shift(int x);`
  - **Python:** `def circular_shift(x)`
- `give_answer`(n)
  This function should be called only once, at the termination of your program. It reports to Bytaelene that n is your guess of the sequence length.
  - **C++:** `void give_answer(int n);`
  - **Python:** `def give_answer(n)`

Your program **cannot** read any data, neither from standard input nor any files. Likewise, it **cannot** write to any files nor the standard output. It can write to the standard diagnostic output (`stderr`) – remember though that this takes (precious) time.

Your program should terminate immediately after calling the function `give_answer`.

Sample execution of a program

Consider an increasing sequence $(0, 5, 15, 17, 18)$, shifted cyclically by $x_0 = 2$ by Bytalene. The initial sequence is thus $(15, 17, 18, 0, 5)$.

Call	Sequence (a_i) after the call	Returned value
`circular_shift(2)`	$(18, 0, 5, 15, 17)$	18
`circular_shift(1)`	$(0, 5, 15, 17, 18)$	0
`circular_shift(8)`	$(17, 18, 0, 5, 15)$	17
`circular_shift(0)`	$(17, 18, 0, 5, 15)$	17
`circular_shift(8)`	$(5, 15, 17, 18, 0)$	5
`circular_shift(1)`	$(15, 17, 17, 0, 5)$	15
`give_answer(5)`	–	–

Before the first query, the sequence is $(15, 17, 18, 0, 5)$. The operation `circular_shift(2)` moves the current first element to the end of the sequence twice. After the first such move, the sequence is $(17, 18, 0, 5, 15)$, and after the second it is $(18, 0, 5, 15, 17)$. The function call returns the first element of the sequence after the moves are performed, i.e., 18.

Grading

In all of the tests, Bytalene's secret sequence satisfies the following conditions:

- $1 \le n \le 500\,000$;
- $0 \le a_i \le 10^{18}$ for all $1 \le i \le n$.

In each test, the initial sequence (a_i) (after Bytalene's shift) is fixed and does not change between queries. For example, the above sample test would be passed by calling the function `give_answer(5)` and terminating afterwards without any queries. In general, it is possible (and allowed) to truly guess the sequence length, i.e., reporting an answer without being certain that it is correct.

The set of tests consists of the following subsets, each with its own additional conditions. Within each subset, there may be several unit tests.

Subset	Condition	Score
1	$n \le 10$	9
2	$n \le 2000$	33
3	$x_0 = 0$, i.e., the initial sequence (a_i) is increasing	22
4	no further conditions	36

Experiments

The directory `dlazaw` contains the following files, which allow testing if your program is formally correct:

- **C++:** a library `cyklib.hpp` and an incorrect sample program `cyk.cpp`
- **Python:** a library `cyklib.py` and an incorrect sample program `cyk.py`

Note that these libraries are merely for the sake of checking correctness of interaction and differ from those which will eventually evaluate your solution. While in the `dlazaw` directory, you can compile and run `cyk.cpp` or `cyk.py`. To compile in C++, type:

- `g++ -O3 -static cyk.cpp -std=c++11 -o cyk.e`

The resulting program reads from input the number n and the initial cyclic shift x_0 (first line) as well as the increasing sequence (a_i) prior to shift by x_0 (second line). The same format is used by the sample grading test files ("ocen") in the `in` directory.

Remember that the provided sample library does not check if the input data is in the correct format, nor whether aforementioned conditions are satisfied. To run your program on the first sample grading test, use one of the following commands:

- **C++:** `./cyk.e < in/cyk1ocen.in`
- **Python:** `python3 cyk.py < in/cyk1ocen.in`

Sample grading tests:

`1ocen:` sequence of digits, no shift;

`2ocen:` $n = 500$, $x_0 = 249$, successive even numbers starting from 0;

`3ocen:` $n = 500\,000$, $x_0 = 0$, successive odd numbers starting from 1.