Eidgenössische
Technische Hochschule
Zürich

Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Federal Institute of Technology at Zurich

Department Informatik
Markus Püschel
Peter Widmayer
Thomas Tschager
Tobias Pröger
Tomáš Gavenčiak

20th October 2016

Datenstrukturen & Algorithmen Exercise Sheet P5 AS 16

Hand-in: Before Thursday, 27th October 2016 10:00 via the online judge (source code only).

Exercise P5.1 *Binary function value search.*

A function $f(x)$ is given as a part of the program template as `int f(int x)`. The function is defined for all $x \in \{0, 1, \dots x_{max}\}$ where $x_{max} = 20\,000\,000$ and it is increasing within this range, that is when $0 \leq i < j \leq x_{max}$ then $f(i) < f(j)$. All the values are integers.

On the input, you are given $n \geq 1$ integers a_0 to a_{n-1}. For every a_i, you should find x_i such that $a_i = f(x_i)$ and $0 \leq x_i \leq x_{max}$, or determine that no such x_i exists.

Input The input consists of two lines. The first line contains just the integer n. The second line contains n integers a_0 to a_{n-1} separated by spaces.

Output The output should contain n lines, one line for every a_i: either the value of x_i satisfying $a_i = f(x_i)$, or the string `NO` (upper case) if no such x_i exists.

Grading Your get 2 bonus points if your program works for all inputs. Your program should work as fast as a binary search for each a_i, that is to call function f only $\mathcal{O}(n \log x_{max})$ times. Specifically, avoid calling f for all the values $0, \dots x_{max}$.

Submit your `Main.java` at `https://judge.inf.ethz.ch/team/websubmit.php?cid=18985&problem=DA_P5.1`, enroll password is "`quicksort`".

Examples

Input:

```
6
12 6 13 6 0 2
```

Output:

```
4
3
NO
3
0
NO
```

Note: $f(0) = 0, f(1) = 1, f(2) = 4, f(3) = 6, f(4) = 12, f(5) = 15.$

Notes For this exercise, we provide a program template as an Eclipse project archive on the lecture website, which will load the input for you and contains the definition of f. The archive also contains more tests for you convenience – you can copy&paste the data into your running program.

We recommend solving this via a binary search for x in the range $0 \dots x_{max}$ for every given a. To see how to do this with an array, imagine that the values of f for $0 \dots x_{max}$ are an array of $x_{max} + 1$ elements, but you only really compute f for the values that you need. Both simple binary search with a while-loop and recursive binary search should work well.

Also, we do not recommend to try to analyze the function f or to compute its inverse directly – treat it as a blackbox. (You may of course try.)