Editorial: 1209. 1, 10, 100, 1000…

Problem link: https://acm.timus.ru/problem.aspx?space=1&num=1209

Let's consider an infinite sequence of digits constructed of ascending powers of 10 written one after another. Here is the beginning of the sequence: 110100100010000… You are to find out what digit is located at the definite position of the sequence.

Input

There is the only integer N in the first line ($1 \le N \le 65535$). The i-th of N left lines contains the integer K_i — the number of position in the sequence ($1 \le K_i \le 2^{31} - 1$).

Output

You are to output N digits 0 or 1 separated with a space. More precisely, the i-th digit of output is to be equal to the K_i-th digit of described above sequence.

Sample

input	output
4 3 14 7 6	0 0 1 0

Mohamed Anis Mani’s solution

	1	1	0	1	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	..
k	1	2		4			7				11					16						22							..

He figure out that the ones’ positions follow this pattern:

$$U_m = \begin{cases} U_0 = 1 \\ U_{m-1} + m \end{cases}$$

$U_0 = 1$ (position of the 1st 1)

$U_1 = 1 + 1 = 2$ (position of the 2nd 1)

$U_2 = 2 + 2 = 4$ (position of the 3rd 1)

$U_3 = 4 + 3 = 7$ (position of the 4th 1)

….

All the 1s are placed in a position :

$$k = (1+2+3+4 ... + m) + 1$$
$$k = \frac{m(m+1)}{2} + 1$$

now for a given k , we check if k is in the logic suite or not, by computing m , and return the result as follow:

$$answer = \begin{cases} 1, if\ m \in \mathbb{N} \\ 0, otherwise \end{cases}$$

To compute m is easy, it is a quadratic polynomial equation:

$$k = \frac{m(m+1)}{2} + 1$$
$$2k = m^2 + m + 2$$
$$=> m^2 + m + 2 * (1 - k) = 0$$
$$\Delta = 1 - 8(1-k) = 1 + 8(k-1) = 8k - 7$$
$$m = \frac{-1 + \sqrt{(8k-7)}}{2}$$

example:

$k=16$, $x=16$ => at position 16 we have 1

$k=21$, $x=5.84428877$ ==> at position 21 we have 0

Python Code: O(n) – Accepted

```
def answer(n):
    ni = ((8*n-7)**0.5 - 1) / 2
    return 1 if (ni == int(ni)) else 0

n = int(input())
for i in range(n):
    print(answer(int(input())))
```

C++ code: O(n) – Accepted

```
#include <bits/stdc++.h>

int main() {
    unsigned long long n;
    std::cin >> n;

    for (unsigned long long i = 0 ; i < n ; ++i) {
        unsigned long long k;
        std::cin >> k;

        double ki = (sqrt(8*k-7) - 1) / 2;
        int bit = (ki == (unsigned long long)ki) ? 1 : 0;
        std::cout << bit << ' ';
    }


    return 0;
}
```

My solution

I figure out that every range of k has a specific sequence as shown in the following table:

	1	1	0	1	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	..
k	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	..

The idea is to:

1. consider the sequence at the $k^{\text{i-th}}$ as a binary representation, as shown in the table below

k	**Sub-sequence**	**Consider Sub-sequence as binary and convert it to decimal**
1	$10^0 = 1$	1
2	$10^1 = 10$	2
3	$10^1 = 10$	2
4	$10^2 = 100$	4
5	$10^2 = 100$	4
6	$10^2 = 100$	4
7	$10^3 = 1000$	8
8	$10^3 = 1000$	8
9	$10^3 = 1000$	8
10	$10^3 = 1000$	8
11	$10^4 = 10000$	16
12	$10^7 = 10000$	16
13	$10^4 = 10000$	16
14	$10^4 = 10000$	16
15	$10^4 = 10000$	16
16	$10^5 = 100000$	32

2. Compute the $k^{\text{i-th}}$ sequence using this suite:

a(k)	1	2	2	4	4	4	8	8	8	8	16	16	16	16	16	32	32	32	32	32	32	64	64	64	64	64	64	64	..
k	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	

using https://oeis.org/search?q=1%2C2%2C2%2C4%2C4%2C4%2C8%2C8%2C8%2C8%2C16%2C16%2C16%2C16%2C16&sort=&language=english&go=Search

The forumla is: $a(k)=2^{[\sqrt{(2k+2)}-0.5]}$

3. for example: if the original $k=7$ (given in the input):

$$a(6)=2^{[\sqrt{(2*6+2)}-.5]}$$
$$=2^{[3.742640687]}$$
$$=2^{3}$$
$$=8$$
$$(8)_{10}=(1000)_{2}$$

4. Now, I have to detect the new position of k in binary representation 1000

Position in binary	*3*	*2*	*1*	*0*
a(k) in binary	1	0	0	0
k	7	8	9	10
0-based indexing k	*6*	*7*	*8*	*9*

Let's see the sequence of positions:

Positions in binary	0	1	0	2	1	0	3	2	1	0	4	3	2	1	0	5	4	3	2	1	0
original	1	1	0	1	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0	0	0
a(k)	1	2	2	4	4	4	8	8	8	8	16	16	16	16	16	32	32	32	32	32	32
k	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

To compute that, I use https://oeis.org/search?q=1+0+2+1+0+3+2+1+0+4+3+2+1+0&sort=&language=english&go=Search

0, **1, 0, 2, 1, 0, 3, 2, 1, 0, 4, 3, 2, 1, 0**, 5, 4, 3, 2, 1, 0, 6, 5, 4, 3, 2, 1, 0, 7, 6, 5, 4, 3, 2, 1, 0, 8, 7, 6, 5, 4, 3, 2, 1, 0, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3 (list; table; graph; refs; listen; history; text; internal format)

formula: $a(k)=(1/2)*(t^{2}+t-2*k-2)$, where $t=floor(\sqrt{(2*k+1)}+1/2)$

example: $a(6)$ (remember the original k is 7)

$$t = floor\left(\sqrt{(2*6+1)}+1/2\right)$$
$$= floor(4.105551275)$$
$$= 4$$
$$a(6) = (1/2)*(4^2+4-2*6-2)$$
$$= (1/2)*(6)$$
$$= 3$$

3 is the position of bit to return which is 1.

3	2	1	0
1	0	0	0
8	8	8	8
6 (input k = 7)	7	8	9

My C++ code: O(n log n) – C++ code generate an WA error due to the data type limts.

```
#include <bits/stdc++.h>

int main() {
    long long n;
    std::cin >> n;

    for (long long i = 0 ; i < n ; ++i) {
        long long k;
        std::cin >> k;

        // Decrement k to fit with the 0-based indexing sequences of oeis.org
        k--;

        // Compute the ki-th sequence
        // https://oeis.org/search?
q=1%2C2%2C2%2C4%2C4%2C4%2C8%2C8%2C8%2C8%2C16%2C16%2C16%2C16%2C16&sort=&language=english&go=Sea
rch
        long long t = floor(pow (2*k+2, .5)-.5);
        unsigned long long a_k = 1 << t;

        // Compute the position in the binary represenation
        // https://oeis.org/search?
q=1+0+2+1+0+3+2+1+0+4+3+2+1+0&sort=&language=english&go=Search
        t = floor(pow(2*k+1, .5)+.5);
        long long pos =  ( t*t + t - (k << 1) - 2) >> 1;

        // Return the bit in position 'pos' from a(k)
        int bit = (a_k >> pos & 1);

        std::cout << bit << ' ';
    }

    return 0;
}
```

Python code: O(n) – Accepted

```
from math import pow, floor

n = int(input())

for i in range(n):
    k = int(input())

    # Decrement k to fit with the 0-based indexing sequences of oeis.org
    k -= 1

    # Compute the ki-th sequence
    # https://oeis.org/search?
q=1%2C2%2C2%2C4%2C4%2C4%2C8%2C8%2C8%2C8%2C16%2C16%2C16%2C16%2C16&sort=&language=english&go=Search
    t = floor(pow (2*k+2, .5)-.5)
    a_k = 1 << t

    # Compute the position in the binary represenation
    # https://oeis.org/search?q=1+0+2+1+0+3+2+1+0+4+3+2+1+0&sort=&language=english&go=Search
    t = floor(pow(2*k+1, .5)+.5)
    pos =  ( t*t + t - (k << 1) - 2) >> 1

    # Return the bit in position 'pos' from a(k)
    bit = (a_k >> pos & 1)

    print(bit, end=' ')
```