Smallest Good Base

For an integer n, we call k>=2 a ***good base*** of n, if all digits of n base k are 1.

Now given a string representing n, you should return the smallest good base of n in string format.

Example 1:

```
Input: "13"
Output: "3"
Explanation: 13 base 3 is 111.
```

Example 2:

```
Input: "4681"
Output: "8"
Explanation: 4681 base 8 is 11111.
```

Example 3:

```
Input: "1000000000000000000"
Output: "999999999999999999"
Explanation: 1000000000000000000 base 999999999999999999 is 11.
```

Note:

1. The range of n is [3, 10^18].
2. The string representing n is always valid and will not have leading zeros.

Solution 1

```
class Solution(object):
    def smallestGoodBase(self, N):
        n = int(N);

        for k in xrange(int(math.log(n, 2)), 1 , -1):
            a = int(n ** k ** -1)                   # kth-root of n
            if (1 - a ** (k + 1)) // (1 - a) == n: # [a^0 + a^1 + ... + a^k] == n
                return str(a)

        return str(n - 1)
```

written by hausch original link here

Solution 2

First things first. Let's see the math behind it.

From given information, we can say one thing- Numbers will be of form-

$n = k^m + k^{(m-1)} + \ldots + k + 1$
$\Rightarrow n-1 = k^m + k^{(m-1)} + \ldots + k$
$\Rightarrow n-1 = k\,(k^{(m-1)} + k^{(m-2)} + \ldots + k + 1)$ ...... **[1]**

Also, from $n = k^m + k^{(m-1)} + \ldots + k + 1$, we can say,
$n-k^m = k^{(m-1)} + k^{(m-2)} + \ldots + k + 1$ ...... **[2]**

from [1] and [2],

$n-1 = k\,(n - k^m)$
$\Rightarrow k^{(m+1)} = nk - n + 1$

if you shuffle sides you will end up getting following form,

$\mathbf{(k^{(m+1)} - 1)/(k - 1) = n}$

Also from [1] note that, $(n - 1)$ must be divisible by k.**[3]**

[EDIT] -->

With inputs from @StefanPochmann we can also say, from binomial thorem, $\mathbf{n = k^m + \ldots + 1 < (k+1)^m}$, therefore, $k+1 >$ m-th root of $n > k$. Thus ⌊**m-th root of n**⌋ is the only candidate that needs to be tested.**[4]**

<--

So our number should satisfy this equation where **k** will be our base and **m** will be (number of 1s - 1)

This brings us to the search problem where we need to find k and m.

Linear search from 1 to n does not work. it gives us TLE. So it leaves us with performing some optimization on search space.

From **[4]** we know that the only candidate that needs to be tested is,⌊**m-th root of n**⌋

We also know that the smallest base is 2 so we can find our m must be between 2 and $\log_2 n$ else m is (n-1) **[5]**

That brings me to the code:
[EDIT] -- >

```
import math
class Solution(object):
    def smallestGoodBase(self, n):
        """
        :type n: str
        :rtype: str
        """
        n = int(n)
        max_m = int(math.log(n,2))
        for m in range(max_m,1,-1):
            k = int(n**m**-1)
            if (k**(m+1)-1)//(k-1) == n:
                return str(k)

        return str(n-1)
```

<—-

written by harshaneel original link here

Solution 3

sum base r == 1111... means sum = 1+r+r^2+r^3+...+r^p
I tried two different search.

1. for r = 2-10, increase p until sum>= n
2. for p=2,3,4..., calculate r = sum^(1/p). then try a few numbers around r to see if the sum fits the above equation. In fact, I only tried r and got accepted by OJ.

```
    public String smallestGoodBase(String n) {
        long nl = 0, cur = 1;
        for (int i=n.length()-1;i>=0;i--){
            nl+=(n.charAt(i)-'0')*cur;
            cur*=10;
        }
        for (long i=2;i<10;i++){
            long s = 0;
            cur = 1;
            for (int j=0;j<nl;j++){
                s+=cur;
                cur*=i;
                if (s == nl) return Long.toString(i);
                if (s > nl) break;
            }
        }
        long res = nl-1;
        for (int i=2;i<1000;i++){
         int r = (int)Math.pow(nl,1.0/i);
         if (r<5) break;
         if (helper(r,i,nl)&&res>r)
          res = r;
        }
        return Long.toString(res);
    }
    boolean helper(int r, int i, long nl){
     long res = 0;
     long cur = 1;
     for(int j=0;j<=i;j++){
      res+=cur;
      cur*=r;
      if (cur>1000000000)
       cur%=1000000000;
     }
     if (res%1000000000 == nl%1000000000) return true;
     else return false;
    }
```

written by jinsheng original link here