# Epidemiology

<https://dmoj.ca/problem/ccc20j2>

We might be tempted to use a simple while loop to solve the problem:

P = int(input())

N = int(input())

R = int(input())

sick = N

days = 0

while sick <= P:

days += 1

sick += N \* (R \*\* days)

print(days)

However, while this code yields correct answers, it also yields timeout errors. What’s wrong?

It turns out the DMOJ problem is set to a very low time limit. Some languages (such as PyPy) are fast enough to finish this loop in a second, while others require a more analytic solution.

Specifically, our loop is too slow for cases where there is a high population and a slow spread. Say you have one sick person who infects one other person every day, with a population of 10 million (P = 10000000, N = 1, R = 1). In Python our computers can do about 5 million things per second, so running our loop 10 million times, adding only 1 sick person at a time, is too slow.

Instead, we need to use math to figure out the number of days it'll take without looping.

The math we need is a geometric sequence; read up on it here if needed: <https://www.mathsisfun.com/algebra/sequences-sums-geometric.html>

Here’s a formula for finding the nth term of a given geometric sequence:

i.e., the nth term is equal to the first term (a) times the common ratio (r) to the power of n – 1.

In Epidemiology, we’re given the nth term (P), the first term (N), and the common ratio (R). When it asks us to find the number of days to exceed P, that represents *n* in the formula.

So we rearrange the formula to isolate *n*. Here’s that rearrangement in two steps:

For logarithms, remember that in Python, we have access to the math module, which contains the method math.log10.

In Python, then, this would look like:

import math

P = int(input())

N = int(input())

R = int(input())

days = math.log10(P / N) / math.log10(R)

print(days)

However, this will not quite pass DMOJ because of some special cases.

1. If R is 1, as in the second example given in the problem statement (P = 10, N = 2, R = 1), then notice that its exponent will be 0, thus dividing by 0 on the days= line.  
   So let’s create a special case for when R is 1.  
     
   The math here can be worked out mentally:

if R == 1:

days = (P + 1 - N) / N

print(math.ceil(days)) # ceiling means round up

1. If the entire population already starts out sick, we can predict that we exceed P on day 1. (Rounding difficulties using log with tiny numbers require us to make this a special case.)

elif N == P:

print(1)

1. Our formula for finding the nth term may yield decimals. For example, for the first case, days is about 4.1. However, the problem requires us to output integer days. So we ask: if we round down to 4, will that still be greater than P? Or do we need to round up to 5? Note that we can’t just round up to be safe because we have to output the earliest day that the number of sick exceeds P.  
     
   For this purpose, we use a geometric sequence formula that yields the sum for a given n:

We use this to check whether rounding down (flooring) is greater than P. If so, we output it. But if the floor is too low, we have to use the ceiling as the earliest day sick passes P.

days = math.log10(P / N) / math.log10(R)

floor = math.floor(days)

ceil = math.ceil(days)

tn\_low = N \* ((1 - R \*\* (floor + 1)) / (1 - R))

if tn\_low > P:

print(floor)

else:

print(ceil)

Here is the full solution that should pass all DMOJ cases without wrong answers or timeouts:

import math

P = int(input())

N = int(input())

R = int(input())

if R == 1:

days = (P + 1 - N) / N

print(math.ceil(days)) # ceiling means round up

elif N >= P:

print(1)

else:

days = math.log10(P / N) / math.log10(R)

floor = math.floor(days)

ceil = math.ceil(days)

tn\_low = N \* ((1 - R \*\* (floor + 1)) / (1 - R))

if tn\_low > P:

print(floor)

else:

print(ceil)

You might wonder: “Is that not a little complicated?!” Remember how I said there are two doorways into to computer science, language or math, and that mine was language? If you are a math person, feel free to suggest a more elegant solution — this took me 3 hours to figure out :')