Solution explanation:

Using the unoptimized code to solve this problem we see that the difference between the first and second number is how many 1's occur, the difference between the second and third number is how many 2's occur etc.

e.g.

for 14

sum: 14 + 7 + 4 + 3 + 2 + 2 + 2 + 1 + 1 + 1 + 1 + 1 + 1 + 1 (pattern)

7-4 2's 14-7 1's

result: 41

so if n is the number of days and i is the day we are checking then

```
(n//i)-n//(i+1)
```

Should be the amount of times i has occurred in this pattern

Using this technique, we can add the amount of repeating numbers through an operation (number of times occurred) * (number).

If the result of the comparison between the two numbers is greater than 1, we know we have more than 1 (repeating) numbers.

This technique basically uses the front of the pattern to predict how many repeating numbers will be at the end of the pattern.

When the numbers start to repeat in this pattern, we do not have any more repeating numbers at the end of the pattern.

i.e this technique cannot be used anymore.

After we have exhausted this technique, we can start adding all of the non-repeating numbers using the normal technique except we will only need to do this for numbers up to where the repeating numbers have started.

The reason why we swap to a separate loop to deal with non-repeating numbers is because we can make the calculations use less divisions in this loop, because divisions are quite resource intensive this is quite attractive.

This technique will have reduced the amount of loops done from the number input to the amount of unique numbers in the pattern

In the case of 10^12 this has made the algorithm use 2120764 loops instead of 10^12 loops this is 471528 times faster