Coding Practice With Kick Start 2020 - Coding Practice with Kick Start

Analysis: Even Digits

To make our discussion easier, let us define a **beautiful** integer as an integer which has only even digits in its decimal representation.

Test Set 1

One useful observation for solving this problem is that we either only want to press the minus button or only want to press the plus button. It is useless to have both types in the same sequence, since they cancel each other out.

Therefore, we want to either keep pressing the minus button, or keep pressing the plus button, until we have a beautiful integer. Let M be the minimum number of minus button presses before reaching a beautiful integer, and let P be the minimum number of plus button presses before reaching a beautiful integer. Then the answer is the smaller of M and P.

Note that for this problem, the answer for an input **N** is at most **N**, since we can just press the minus button **N** times to reach the beautiful integer `0`. Therefore, since **N** ≤ 10^5, we can solve Test Set 1 using complete search.

We can loop over a value i in the range [0, N], and, for each i, check whether **N**+i or **N**-i is a beautiful integer. If that is the case, then we break the loop and return i as the answer.

Test Set 2

Iterating over the range [0, **N**] would be too slow for this test set. Therefore, we need another approach.

To find the value of M, we must find the largest beautiful integer that is still no greater than **N**. Let us call this integer X. Similarly, to find the value of P, we can find the smallest beautiful integer that is still no smaller than **N**. Let us call this integer Y.

We can find X by decreasing the first odd digit (counting from left) in **N** by one and then replacing all digits to the right of that odd digit with the largest even digit (i.e. `8`). For example, if **N**= `4436271`, then X=`4428888`. This can create a leading 0 when the leftmost digit of **N** is 1, in which case we can simply drop the leading 0. If there are no odd digits in **N**, then **N** is already a beautiful integer, thus X=**N**.

By constructing X this way, it is guaranteed that there will be no beautiful integer between X and **N**, since the next beautiful integer after X would be larger than **N** at the first odd digit. For the example above, the next beautiful integer after X is `4440000`, which is larger than **N**.

Similarly, we can find Y by increasing the first odd digit in **N** by one and replacing all digits to the right of that odd digit with the smallest even digit (i.e. `0`). If there are no odd digits in **N**, then **N** is a beautiful integer, thus Y=**N**. However, when finding Y, we must watch out for some tricky cases. If the first odd digit is `9`, then we must replace the digit directly to the left of that odd digit as well with the next even digit. For example, if **N**=`86912`, then Y=`88000`.

Another tricky case is when the digit directly to the left of the first odd digit is `8`. In this case, we must replace the digit directly to the left of that `8` digit as well, and keep doing this until we have a non-`8` digit. For example, if **N**=`6488962`, then Y=`6600000`. Finally, if all digits to the left continue to be `8` until we reach the leftmost digit, or if the first digit of **N** is a 9, then we must add the smallest non-zero even digit (i.e. `2`) as a new digit on the left. For example, if **N**=`88892` or **N**=`91112`, then Y=`200000`.

Once we know X and Y, we also know M and P, and we can take the smallest of those values, just as we did before.