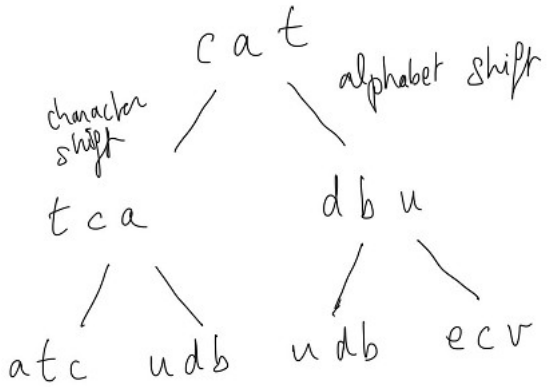
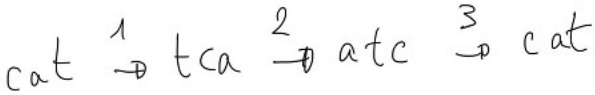
To make the text shorter, I will call
- the operation of shifting each character of the word to the left = character shift (of the word)
- the operation of transforming each character of the word to the next character of the alphabet = alphabet shift (of the word)

At first, we may think about using an overkill solution starting from the current word: That for each one of them, we either character shift or alphabet shift it, without even knowing when to stop the search. That means we will have an explosive complexity of a power of 2.

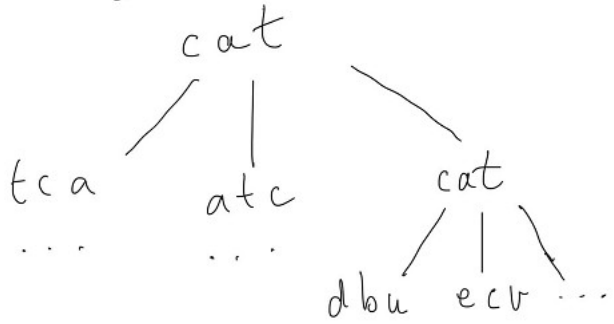


However, the problem is easier than that: Given a word of N characters, if we character shift the word N times, we will get back to the same word, so the Nth time is equivalent to no character shift at all.



Likewise, if we alphabet shift it 26 times, we will get back to the same word. This limitation of the number of operations puts us to either of 2 scenarios (both are correct):

Scenario 1: For each of the N character shifts, we do 26 alphabet shifts.



Scenario 2: For each of the 26 alphabet shifts, we do N character shifts.

In both scenarios, that means simply using **brute force** thanks to a double loop.