USACO January Gold 2

Andrew Wang

January 2021

1 Problem Statement

After sequencing the genomes of his cows, Farmer John has moved onto genomic editing! As we know, a genome can be represented by a string consisting of As, Cs, Gs, and Ts. The maximum length of a genome under consideration by Farmer John is 10^5. Farmer John starts with a single genome and edits it by performing the following steps:

1. Split the genome between every two consecutive equal characters.
2. Reverse each of the resulting substrings.
3. Concatenate the reversed substrings together in the same order.

For example, if FJ started with the genome AGGCTTT then he would perform the following steps:

1. Split between the consecutive Gs and Ts to get AG — GCT — T — T.
2. Reverse each substring to get GA — TCG — T — T.
3. Concatenate the substrings together to get GATCGTT.

Unfortunately, after editing the genome, Farmer John's computer crashed and he lost the sequence of the genome he originally started with. Furthermore, some parts of the edited genome have been damaged, meaning that they have been replaced by question marks.

Given the sequence of the edited genome, help FJ determine the number of possibilities for the original genome, modulo $10^9 + 7$.

Input: A non-empty string where each character is one of A, G, C, T, or ?.

Output: The number of possible original genomes modulo $10^9 + 7$.

2 Key observations

1. DP Problem based on recurrence relations
2. After dividing the final string to the substrings before it was concatenated:
   - For any two adjacent substrings, the first letter of the first substring must be equivalent to the last letter of the second substring
   - No substring can have adjacent letters that are equivalent
3. O(N) based on constraints

3 O(N^2) Solution

A good place to start with DP is defining how we transition between states. The most logical transition would be to build the string character by character. Whenever we add a character we either:

1. Start a new substring.
2. Add on to the last substring.

If we start a new substring, the last substring must be complete meaning it satisfies the two restrictions listed as observations. To check if a substring is copmlete, we have to

1. Check if the last character is equivalent to the first of the previous substring.
2. Check there are no adjacent characters that are the same.

To satisfy the second restriction, we can do a preliminary dp to find the number of ways to fill in the question marks so that the substring beginning with letters l and ending with m $s[j...k]$(starting at index j and ending at index k) have no two adjacent characters that are the same. From this, we can check for the index of where the previous substring ends at and compare the first character of the previous substring to the last character of our current substring we are trying to complete while also checking their are no adjacent characters that are the same in the substring we are completing.

4 Intended O(N) Solution

To build on our previous solution, we only need to know the first letter of the previous substring along with the first and last characters of our last character to satisfy our first restriction to complete a substring. Thus, let $dp[k][n][l][m]$ be the number of ways at index k with n being the first character of the second-to-last substring, l being the first character of the last substring, and m being the last character of our last substring. To take care of the second restriction, we simply don't add on to the last substring if m is equivalent to the character we are about to add. Therfore our transitions become:

1. $dp[k-1][n][l][m] \rightarrow dp[k][n][l][m']$ if $n \neq m'$.
2. $dp[k-1][n][l][m] \rightarrow dp[k][l][m'][m']$ if $n = m$.

Finally, our answer is the sum of all $dp[N][m][l][m]$ to ensure our last substring is complete.