Kick Start 2021 - Round H

Analysis: Painter

Given the three primary colors: *Red*, *Yellow*, and *Blue*, we need to determine the minimum number of strokes required to paint a 1-dimensional painting $\mathbf{P}$ of length $\mathbf{N}$. In 1 stroke, you can choose a color either *Red*, *Yellow*, or *Blue*, and two integers l and r, such that $1 \le l \le r \le \mathbf{N}$, and apply the chosen color to all squares $\mathbf{P_j}$ such that $l \le j \le r$.

Note as the colors: *Red*, *Yellow*, and *Blue* are independent of each other and the proportion and order of each color in the combination does not matter, the minimum number of strokes to complete the paining will be the minimum number of *Red* strokes $+$ minimum number of *Blue* strokes $+$ minimum number of *Yellow* strokes.

Let $F(x)$ be the minimum number of strokes needed of primary color x. Notice minimum number of strokes of x is same as number of continuous blocks of squares that will need the primary color x to form the required color. To find $F(x)$ we will start from the leftmost end of the squares. Whenever we encounter a square either with the color x or with some color y which is a combination of x and other primary colors, we paint the squares with the color x in 1 stroke until we reach the rightmost end, or some color y which is neither x nor a combination of x and any other primary color. We continue this until we have encountered all the squares. The number of strokes used is equal to $F(x)$.

Test Set 1

We are given that $\mathbf{P_i}$ will be one of $\{Y, B, G\}$. The final answer will be $F(Yellow) + F(Blue)$.

Complexity : $O(\mathbf{N})$ *per test case*

Test Set 2

We are given that $\mathbf{P_i}$ will be one of $\{U, R, Y, B, O, P, G, A\}$. The final answer will be $F(Red) + F(Yellow) + F(Blue)$.

Complexity : $O(\mathbf{N})$ *per test case*

Sample Code(C++)

```
int minimum_number_of_strokes_for_primary_color_x(string S, int N, char x, map<char, string> color_to_primar
  int min_strokes = 0;
  int last_stroke_position = INT_MIN;
  for(int i = 0; i < N; i++) {
    if (S[i] == 'U') {
      continue;
    }
    if(S[i] == x || color_to_primary_colors[S[i]].find(x) != string::npos) {
      if (last_stroke_position != i - 1) {
        min_strokes++;
      }
      last_stroke_position = i;
    }
  }
  return min_strokes;
}
```