Problem Tutorial: "Cat"

We'll consider a solution using suffix array, but solutions using other suffix structures are possible as well.

Let $s = a+b$. We need to find the number of distinct substrings of s with at least one occurrence containing characters at positions $|a|$ and $|a|+1$.

Let $n = |s|$. Let's build the suffix array $p_1, p_2, \ldots, p_n$ of s and let $l_i = LCP(s_{p_i..n}, s_{p_{i+1}..n})$. If we just needed to count distinct substrings of s, that number would be $\binom{n+1}{2} - l_1 - l_2 - \ldots - l_{n-1}$.

Let's consider suffixes in order $p_1, p_2, \ldots, p_n$. For each i, first, some prefixes of suffix p_{i-1} can be marked as they will never appear again. Then, suffix p_i brings substrings $s_{p_i..p_i+l_{i-1}}, s_{p_i..p_i+l_{i-1}+1}, \ldots, s_{p_i..n}$ into play. If $p_i \le |a|$, for $|b|$ longest prefixes of $s_{p_i..n}$, we also know now that they have an occurrence covering positions $|a|$ and $|a|+1$.

It's enough to maintain some data structure that simulates an array with the following queries:

- set 0 or 1 to all values in some range;
- find the sum of value in some range.

A usual segment tree will do. (It's also possible to use the structure of queries and go with `std::set` or something similar.)