May 17, 2021

Assignment 5
Algorithms for Sequence Analysis, Summer 2021

Algorithmic Bioinformatics · Prof. Dr. Sven Rahmann

Hand in date: Monday, May 24, before 20:00

Exercise 1: Number of distinct substrings (4 Theory)
Give a linear-time algorithm to compute the total number of distinct substrings of a string s. Think in terms of $s\$$ and the enhanced suffix array. For example, `baaa` has 7 distinct substrings: `a, b, aa, ba, aaa, baa, baaa`.

Solution:
Consider a string $s\$$ with $n = |s\$|$. The maximal number of distinct substrings (if all characters are different) is $U := 1 + 2 + \cdots + n$ substrings of lengths $n, n-1, \ldots, 1$, respectively. By Gauss, $U = n(n+1)/2$. Common substrings correspond to common prefixes of suffixes. If $\texttt{lcp}[k] = \ell$, there are ℓ identical substrings to the lexicographically previous suffix (lengths $1, \ldots, \ell$). Therefore, we must subtract each lcp value from U (without the -1 sentinels), so we have $n(n+1)/2 - \sum_{r=1}^{n-1} \texttt{lcp}[r]$ distinct substrings, including those with sentinel (i.e., complete suffixes). Removing those n suffixes with sentinel, the solution becomes

$$\frac{(n-1) \cdot n}{2} - \sum_{r=1}^{n-1} \texttt{lcp}[r]\,.$$

Again, note that n includes the sentinel.
In the example above, we consider $s\$ = \texttt{baaa\$}$ of length 5, the lcp values between sorted suffixes (`$, a$, aa$, aaa$, baaa$`) are $(0, 1, 2, 0)$, and the correct answer is $10 - 3 = 7$.

Exercise 2: Supermaximal repeats (4 Theory)
Give a linear-time algorithm to find all supermaximal repeats of a string s, using its enhanced suffix array and the definitions below. To develop the algorithm, characterize supermaximal repeats in terms of intervals in the enhanced suffix array.

A triple (p, p', m) of two positions $p < p'$ and a length $m \geq 1$ is called

- a *repeated pair* if and only if $s[p : p + m] = s[p' : p' + m] =: w$
  (using Python half-open indexing, i.e., the right boundary is not included),
- *right-maximal* if $s[p + m] \neq s[p' + m]$,
- *left-maximal* if $p = 0$ or $s[p - 1] \neq s[p' - 1]$,
- *maximal* if it is both left-maximal and right-maximal.

Then the repeated substring w is called a [left-/right-]maximal *repeat* of length m at p, p'. A *supermaximal repeat* is a maximal repeat that is not a substring of any other repeat.

Example: $s =$ `mambamba$`.
Maximal repeats with corresponding maximal repeated pairs:
`a`: $(1, 7, 1)$, `amba`: $(1, 4, 4)$, `m`: $(0, 2, 1), (0, 5, 1)$.
Supermaximal repeats: `amba` only; the other maximal repeats are substrings of `amba`.

Note that there can be several maximal repeated pairs for the same string; there can be also other (non-maximal) repeated pairs for the string; it suffixes that there is at least one maximal repeated pair for a string to call the string a maximal repeat – the repeated pair $(2, 5, 1)$ is not maximal for $w =$ `m`, but $w =$ `m` is still a maximal repeat because other maximal repeated pairs exist for it, such as $(0, 2, 1)$, see above.

It may help to think about this problem in terms of the suffix tree first (which inner nodes are supermaximal repeats?) and then translate your insights to suffix array intervals.

Solution:
Supermaximal repeats are inner nodes in the suffix tree that only have leaves as children whose preceding characters are all different (left-maximality). For example, node `amba` has two children, leaves 1 and 4, and the preceding characters at positions 0 and 3 are different (`m` and `b`).
This corresponds to an interval $[L, R]$ of ranks in the suffix array with $R \geq L + 1$, such that the lcp values at $L + 1, \ldots, R$ are constant and form a local maximum with the surrounding ones, so lcp at L and $R + 1$ must be smaller. In addition, it must hold that $s[\texttt{pos}[r] - 1]$ are all different for $r = L, \ldots, R$. All of this can be checked by a linear scan over the enhanced suffix array (assuming constant alphabet).