Blind Trie

Paolo Ferragina

Dipartimento di Informatica, Università di Pisa, Italy

Let $\mathcal{S}_\pi$ be a string set and take PT_π as a simplified trie in which each arc label is replaced by only its first character. See Figure 1 for an illustrative example. We assume that PT_π is stored in internal memory and the string set $\mathcal{S}_\pi$ is stored on disk.

The goal of PT_π is to help finding the lexicographic position of the searched pattern P in the ordered set $\mathcal{S}_\pi$. This search is a little bit more complicated than the one in classical tries, because of the presence of only one character per arc label, and in fact consists of two stages:

- Trace a downward path in PT_π to locate a leaf l which points to an *interesting* string of $\mathcal{S}_\pi$. This string does not necessarily identify P's position in $\mathcal{S}_\pi$ (which is our goal), but it provides enough information to find that position in the second stage (see Figure 1). The retrieval of the interesting leaf l is done by traversing PT_π from the root and comparing the characters of P with the single characters which label the traversed arcs until a leaf is reached or no further branching is possible (in this case, choose l to be any descendant leaf from the last traversed node).
- Compare the string pointed by l with P in order to determine their longest common prefix. A useful property holds: *the leaf l stores one of the strings in $\mathcal{S}_\pi$ that share the* **longest** *common prefix with P.* The length ℓ of this common prefix and the mismatch character $P[\ell + 1]$ are used in two ways: first to determine the shallowest ancestor of l spelling out a string longer than ℓ; and then, to select the leaf descending from that ancestor which identifies the lexicographic position of P in $\mathcal{S}_\pi$.

An illustrative example of a search in a Patricia tree is shown in Figure 1 for a pattern $P =$ "$GCACGCAC''$. The leaf l found after the first stage is the second one from the right. In the second stage, the algorithm first computes $\ell = 2$ and $P[\ell + 1] = A$; then, it proceeds along the leftmost path descending from the node u, since the 3rd character on the arc leading to u (i.e. the mismatch G) is grater than the corresponding pattern character A. The position reached by this two-stage process is indicated in Figure 1, and results the correct lexicographic position of P among $\mathcal{S}_\pi$'s strings.

We remark here that PT_π requires space linear in the *number* of strings of $\mathcal{S}_\pi$, therefore the space usage is independent of their *total length*. Consequently, the number of strings in $\mathcal{S}_\pi$ can be properly chosen in order to be able to fit PT_π in a cache. An additional nice property of PT_π is that it allows to find the lexicographic position of P in $\mathcal{S}_\pi$ by fully comparing P with just *one of the strings in $\mathcal{S}_\pi$*, thus taking $O(\frac{p}{B} + 1)$ disk accesses.

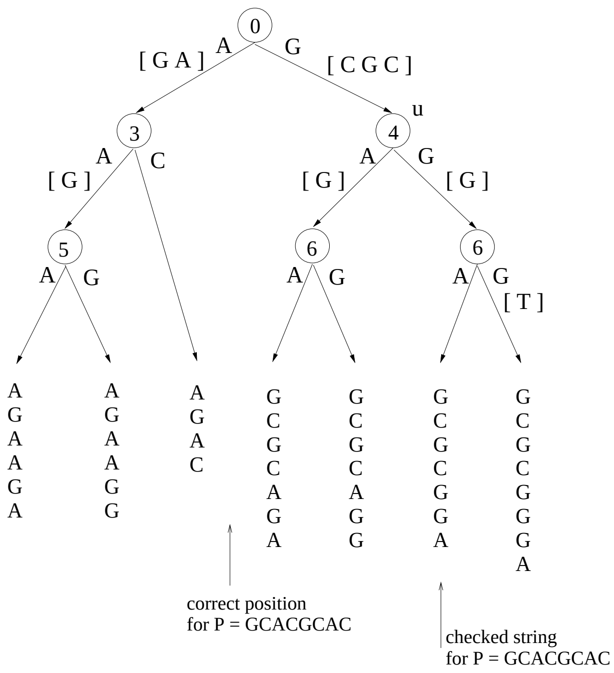

Fig. 1. *An example of Patricia tree built on a set of $k = 7$ DNA strings drawn from the alphabet $\Sigma = \{A, G, C, T\}$. Each leaf points to one of the k strings; each internal node u (they are at most $k - 1$) is labeled with one integer $len(u)$ which denotes the length of the common prefix shared by all the strings pointed by the leaves descending from u; each arc (they are at most $2k - 1$) is labeled with only one character (called* branching character*). The characters between square-brackets are not explicitly stored, and denote the other characters labeling a trie arc.*