Patricia Tries

Yufei Tao
KAIST

April 16, 2013
We will continue the discussion of the **exact matching problem** on strings.

**Problem**

Let $S$ be a set of strings, each of which has a unique integer id. Given a query string $q$, a query reports:

- the id of $q$ if it exists in $S$
- nothing otherwise.
Let

- $A$ be the alphabet (i.e., every character of any string must come from $A$).
- $|s|$ be the length of a string $s$, i.e., the number of characters in $s$.
- $m = |S|$, i.e., the number of strings in $S$.
- $n = \text{the total length of the strings in } S$, i.e., $n = \sum_{s \in S} |s|$.
So far, all our tries use $O(n)$ space. In this lecture, we will improve the space consumption to $O(m)$, without affecting the query time.

This is achieved by a variant of tries called the Patricia trie.
Let \( S = \{\text{aaabb}, \text{aabaa}, \text{aabab}, \text{abbb}, \text{abbaa}, \text{abbbb}\} \). The trie of \( S \) is:

\[
\begin{array}{c}
\emptyset \\
a \\
a \\
\text{a} \\
\text{a} \\
\text{b} \\
\text{b} \\
\text{b} \\
\text{b} \\
\text{b} \\
\end{array}
\]

A trie can have many internal nodes that have only one child. A Patricia trie essentially eliminates all such nodes.
We will from now on denote the strings in $S$ as $s_1, s_2, \ldots, s_m$, respectively. We will consider that each $s_i$ is stored in an array of size $|s_i|$, such that $s_i[j]$ gives the $j$-th ($1 \leq j \leq |s_i|$) character of $s_i$. 
The longest common prefix (LCS) of a set $S$ of strings is a string $\sigma$ such that:

- $\sigma$ is a prefix of every string in $S$.
- There is no string $\sigma'$ such that $\sigma'$ is a prefix of every string in $S$, and $|\sigma'| > |\sigma|$.

For example, the LCS of $\{\text{aaabb}_\bot, \text{aab}_\bot, \text{aabaa}_\bot\}$ is $\text{aa}$, and the of LCS of $\{\text{aaabb}_\bot, \text{baa}_\bot\}$ is $\emptyset$. 
Given two strings $s_1, s_2$, we use $s_1 \cdot s_2$ to denote their concatenation.

**Definition (Extension Set)**

Let $S$ be a set of strings, and $\sigma$ the LCS of $S$. The extension set of $S$ is the set of characters $c$ such that $\sigma \cdot c$ is a prefix of at least one string in $S$.

For example, the extension set of $\{aaabb\perp, aab\perp, aabaa\perp\}$ is $\{a, b\}$. The extension set of $\{aaabb\perp, baa\perp\}$ is also $\{a, b\}$. 
The Patricia trie $T$ on $S$ is a tree where each node $u$ carries a positional index $PI(u)$, and a representative pointer $RP(u)$. $T$ can be recursively defined as follows:

1. If $|S| = 1$, then $T$ has only one node whose its PI is $|S|$, and its RP references $s \in S$.

2. Otherwise, let $\sigma$ be the LCS of $S$. The root of $T$ is a node $u$ with $PI(u) = |\sigma|$, and $RP(u)$ referencing $s$, where $s$ is an arbitrary string in $S$.

3. Let $E$ be the extension set of $S$. Then, $u$ has $|E|$ child nodes, one for each character $c$ in $E$. Specifically, the child node $v_c$ for $c$ is the root of a Patricia trie on the set of strings in $S$ with $\sigma \cdot c$ as a prefix.
Example: Let \( S = \{\text{aaabb}, \text{aabaa}, \text{aabab}, \text{abbb}, \text{abbba}, \text{abbbb}\} \). The Patricia trie of \( S \) is:

![Patricia Trie Diagram]
Lemma

A Patricia trie on \( m \) strings has at most \( 2m - 1 \) nodes.

It is clear that every string in \( S \) corresponds to a leaf in its Patricia trie. Let \( u \) be a node in the Patricia trie. We say that \( s \in S \) is in the subtree of \( u \) if the leaf corresponding to \( s \) is in the subtree of \( u \).

Lemma

Let \( u \) be a node in a Patricia trie. Let \( k = PL(u) \) and \( s \) the string referenced by \( RP(u) \). All the strings in the subtree of \( u \) have prefix \( s[1] \cdot s[2] \cdot ... \cdot s[k] \).
Think

How would you answer an exact matching query with \( q = \text{aabab} \). How about \( q = \text{abbab} \)?
Combining the Patricia trie with the balanced trie, we obtain:

**Theorem**

For the exact matching problem on strings, there is a structure that occupies $O(m)$ space, and answers a query with string $q$ in $O(\log m + |q|)$ time.

**Think**

How?
Let us now consider the **prefix matching problem** on strings:

**Problem**

Let $S$ be a set of strings, each of which has a unique integer id. Given a query string $q$, a query reports all the ids of the strings $s \in S$ such that $q$ is a prefix of $S$.

It is left as an exercise for you to design a structure that uses $O(m)$ space, and answers a query with string $q$ in $O(\log m + |q| + k)$ time, where $k$ is the number of ids reported.