Suffix Tree (Ukkonen's algorithm)

Wing

February 21, 2019

Table of contents

Trie

Trie

An ordered tree data structure used to store a dynamic set or map where the keys are usually strings

Suffix Trie

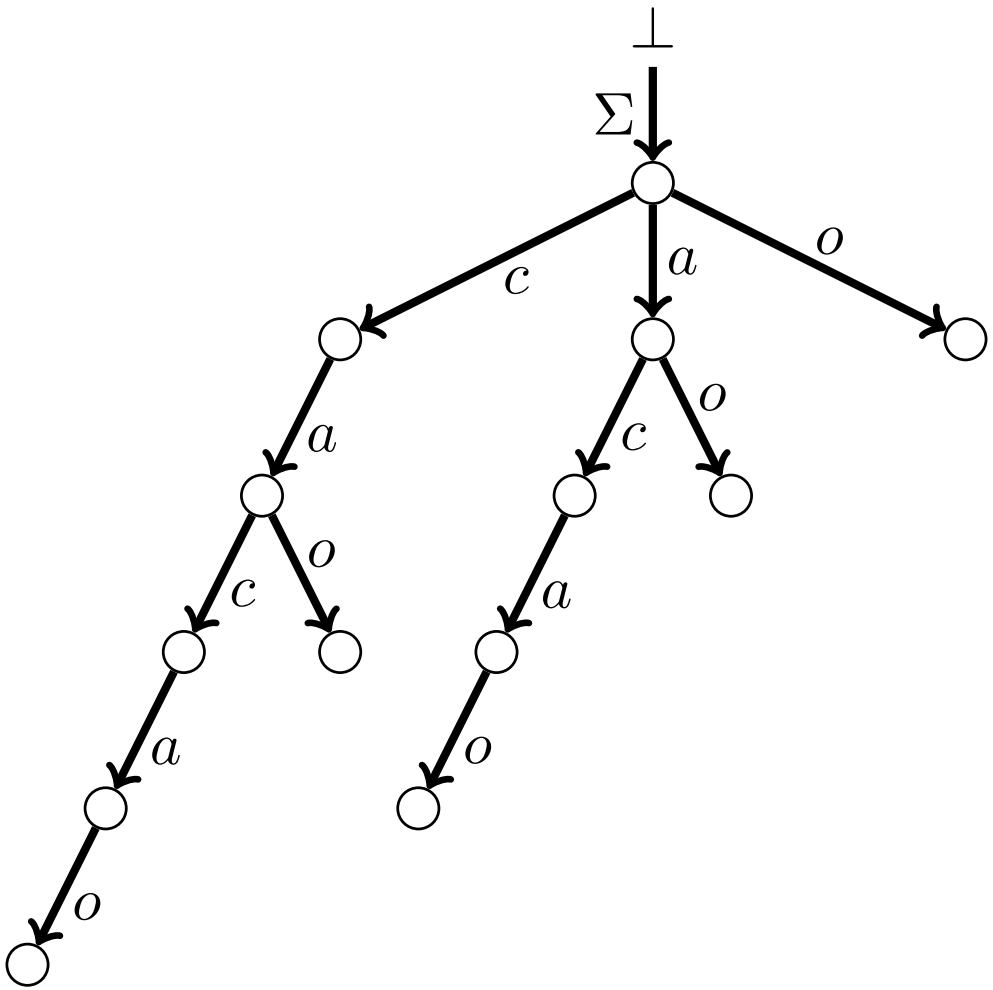

Figure: Suffix Trie for "$cacao$"

Suffix Trie

- Proposed by Esko Ukkonen (University of Helsinki, Finland)
- An algorithm easier to grasp than the those in the literature at that time
- On-line algorithm: Processes the string symbol by symbol from left to right, and always has the suffix tree for the scanned part of the string ready

Construction of Suffix Trie

String

Let $T = t_1 t_2 \cdots t_n$ be a string over alphabet Σ

Substring

Each string x s.t. $T = uxv$ for some (possibly empty) string u and v is a substring of T

Suffix

$T_i = t_i \cdots t_n$ where $1 \leq i \leq n + 1$

- $T_{n+1} = \epsilon$ is the *empty* suffix

Construction of Suffix Trie

Set of all suffixes of T

$\sigma(T)$

The suffix trie of T is a trie representing $\sigma(T)$

Construction of Suffix Trie

Suffix Trie

Denote suffix trie of T as $STrie(T) = (Q \cup \{\perp\}, root, F, g, f)$

Define such a trie as an augmented deterministic finite-state automaton which has a tree-shaped transition graph representing the trie for $\sigma(T)$

augmented with

- f : suffix function
- $\perp$: auxiliary state

Construction of Suffix Trie

Set Q of the states of $STrie(T)$

The set Q of the states of $STrie(T)$ can be put in a one-to-one correspondence with the substrings of T.

Denote $\bar{x}$ the state that corresponds to a substring x
Shorthand: $\bar{x} \sim x$

- $root \sim \epsilon$
- $\sigma(T) \sim$ set F of final states

Construction of Suffix Trie

Transition function g

$$\begin{cases} g(\bar{x}, a) = \bar{y} & \forall \bar{x}, \bar{y} \in Q \text{ s.t. } y = xa, \text{ where } a \in \Sigma \\ g(\perp, a) = root & \forall a \in \Sigma \end{cases}$$

Suffix function f

$\forall \bar{x} \in Q,$

$$\begin{cases} f(\bar{x}) = \bar{y} & \text{if } \bar{x} \neq root, \text{then } x = ay, a \in \Sigma \\ f(root) = \perp & \\ f(\perp) \text{ is undefined} & \end{cases}$$

$\perp \sim a^{-1} \; \forall a \in \Sigma$

$a^{-1}a = \epsilon$

Construction of Suffix Trie

Suffix Link

$f(r)$ is the suffix link of state r

Prefix

$T^i = t_1 \cdots t_i$ of T for $0 \leq i \leq n$

Construction of Suffix Trie

Key observation

How is $STrie(T^i)$ obtained from $STrie(T^{i-1})$?

The suffixes of T^i can be obtained by catenating t_i to the end of each suffix of T^{i-1} and by adding an empty suffix, i.e.

$$\sigma(T^i) = \sigma(T^{i-1})t_i \cup \{\epsilon\}$$

$STrie(T^{i-1})$ accepts $\sigma(T^{i-1})$, to make it accept $\sigma(T^i)$, examine F_{i-1} of $STrie(T^{i-1})$

- $r \in F_{i-1}$ doesn't have a t_i-transition $\Rightarrow$ add transition $r \rightarrow$ new state
- $r \in F_{i-1}$ has a t_i-transition $\Rightarrow$ follow the transition to the old state
- All such states plus $root$ will be F_i of $STrie(T^i)$

Construction of Suffix Trie

How to find states $r \in F_{i-1}$ that get new transitions?

From definition of the suffix function f,
$r \in F_{i-1} \Leftrightarrow r = f^j(\overline{t_1 \cdots t_{i-1}})$ for some $0 \leq j \leq i-1$

Boundary path

Boundary path of $STrie(T^{i-1})$:
Path starting from deepest state $\overline{t_1 \cdots t_{i-1}}$ of $STrie(T^{i-1})$,
following the suffix links and ending at $\perp$

$\therefore$ All states in F_{i-1} are on the boundary path of $STrie(T^{i-1})$

Construction of Suffix Trie

The boundary path is traversed.

If a state $\bar{z}$ on the boundary path does not have a transition on t_i yet, add a new state $\overline{zt_i}$ and a new transition $g(\bar{z}, t_i) = \overline{zt_i}$

To update f, new states $\overline{zt_i}$ are linked together with new suffix links starting from $\overline{t_1 \cdots t_i}$.
Obviously, this is the boundary path of $STrie(T^i)$

Construction of Suffix Trie

Observation

The traversal over F_{i-1} along the boundary path can be stopped immediately when the first state $\bar{z}$ is found s.t. state $\overline{zt_i}$ (and hence also transition $g(\bar{z}, t_i) = \overline{zt_i}$) already exists.

Let namely $\overline{zt_i}$ already be a state.

Then $STrie(T^{i-1})$ has to contain state $\overline{z't_i}$ and transition $g(z', t_i) = \overline{z't_i}\ \forall z' = f^j(\bar{z}), j \leq 1$.
In other words, if $\overline{zt_i}$ is a substring of T_{i-1} then every suffix of $\overline{zt_i}$ is a substring of T_{i-1}.

Such $\bar{z}$ must exist as $\perp$ is the last state on the boundary path that has the t_i-transition $\forall t_i$

Construction of Suffix Trie

Algorithm 1:

1 $r \leftarrow root$
2 **while** $g(r, t_i)$ *is undefined* **do**
3 | create new state r' and new transition $g(r, t_i) = r'$;
4 | **if** $r \neq top$ **then** create new suffix link $f(oldr') = r'$;
5 | $oldr' \leftarrow r'$;
6 | $r \leftarrow f(r)$;
7 create new suffix link $f(oldr') = r'$
8 $top \leftarrow g(top, t_i)$.

Construction of Suffix Trie

Running Algorithm 1 for $t_i = t_1, t_2, \cdots, t_n$ visits each $\bar{x} \in Q$ once.

Theorem 1

Suffix trie $STrie(T)$ can be constructed in time proportional to the size of $STrie(T)$ which, in the worst case, is $\mathcal{O}(|T|^2)$.

References

Original Paper:
https://www.cs.helsinki.fi/u/ukkonen/SuffixT1withFigs.pdf

Wikipedia for the definition of Trie