Suffix Trees

November 7, 2022

Aditya Patil (2021CSB1062) ,
Alankrit Kadian (2021CSB1065) ,
Harsh Raj Srivastava (2021CSB1091)

Instructor:
Dr. Anil Shukla

Teaching Assistant:
Akanksha

Summary: We have first performed and analysed the construction of the suffix tree in linear time using the *Ukkonen's algorithm*. We also implemented our own thought application of suffix trees in which we also incorporated *BK-Tree* and *Bloom Filter*. In a encoded string we search a substring match and suggest the correct string if user input is wrong using BK-Tree from a unique set of keys defined by bloom filter and use suffix tree to finds all the occurences of the substring.

---

1. Introduction

A suffix tree is a data structure that exposes the internal structure of a string. Suffix trees are useful in various string processing and computational problems. They provide linear-time solutions to many complex string problems, after just O(m), or linear, preprocessing time of a given string. Applications of suffix trees include exact matching problem, substring problem, pattern searching, longest repeating substring, building linear time suffix array, longest common substring, longest palindromic substring, etc.

2. Basic Introductions to Suffix Trees

A suffix tree τ for an m-character string S is a rooted directed tree with exactly m leaves indexed 1 to m. For any leaf i, the concatenation of the edge-labels on the path from the root to leaf i exactly spells out the suffix of S that starts at position i, i.e. S[i...m]. Each internal node, other than the root, has at least two children and each edge is labeled with a non-empty substring of S. No two edges out of the same node can have edge-labels beginning with the same character.
The *label of a path* from the root that ends at a node is the concatenation in order, of the substrings labeling the edges of that path. The *path label* of a node is the label of the path from the root to that node.
For example, let us take a look at the suffix tree for string S = "xabxac" which have six suffixes, namely "xabxac", "abxac", "bxac", "xac, "ac" and "c".

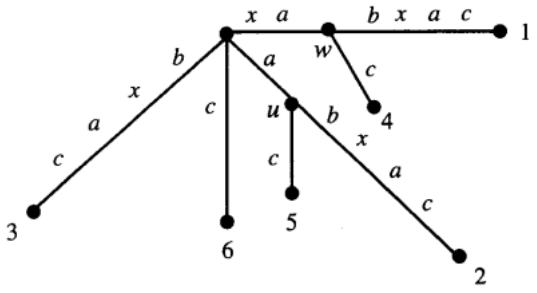

Figure 1: Suffix Tree for string S = "xabxac"

As shown in Figure 1, the suffix tree for string S has one root node, two internal nodes and six leaf edges representing each of its six suffixes.
The above definition of a suffix tree for S does not gaurantee that the suffix tree for any string S actually exists. The problem is that if one suffix of S matches the prefix of another suffix of S, then no suffix tree obeying the above definition is possible. To avoid this problem, we have to assume that the last character of S appears nowhere else in S, so that no suffix of the resulting string can be a prefix of any other suffix. To achieve this in practice, we can add a character, like '$', to the end of S that is not present in S already. This is called the *termination character*.

3. Suffix Tree Construction

There are various algorithms to build a suffix tree for a string S. First, we will present a straightforward algorithm which takes $O(m^2)$ time to build the suffix tree for a string S of length m, and then we will see another algorithm which, with some observations, tricks and implementation speedups, will build the suffix tree for us in linear i.e. $O(m)$ time.

3.1. A naive algorithm to build a suffix tree

This naive method first enters a single edge for suffix S[1...m]$(the entire string) into the tree, and then it successively enters suffix S[i...m]$ into the growing tree, for $2 \leq i \leq m$. Let τ_i denote the intermediate tree that encodes all the suffixes from 1 to i.
Tree τ_1 consists of a single edge between the root of the tree and a leaf labeled 1 and the edge is labeled with the string S$. Tree τ_{i+1} is constructed from τ_i as follows:
Starting at the root of τ_i, find the longest path by successively comparing and matching characters from the root whose label matches a prefix of S[i+1...m]$. The matching path will be unique as no two edges out of a node can have labels that begin with the same character. The matching path either ends at a node, say w, or it ends in the middle of an edge. If it is in the middle of an edge, say (u,v), then it breaks the edge (u,v) into edges by inserting in a new node, say w, just after the last character on that edge that mismatched. The new edge (u,w) is labeled with the part of (u,v) label that matched with S[i+1...m], and the new edge (w,v) is labeled with the remaining part of the (u,v) label. A new edge (w,i+1) running from w to a new leaf labeled i+1 is created, and it is labeled with the unmatched part of suffix S[i+1...m]$. The suffix tree now contains a unique path from root to leaf i+1 with path label S[i+1...m]$.
As mentioned earlier, the naive method takes $O(m^2)$ time to build a suffix tree for the string S of length m.

3.2. Ukkonen's linear-time suffix tree algorithm

Esko Ukkonen devised a linear-time algorithm for constructing a suffix tree that, with its "on-line" property and the simplicity of its description, proof, and time analysis, may be the conceptually easiest linear-time construction algorithm.
While constructing suffix trees using Ukkonen's algorithm, we will see 'implicit suffix tree' in some intermediate steps depending on characters in string S.
An *implicit suffix tree* for string S is a tree obtained from the suffix tree for S$ by removing every copy of the termination character $ from the edge labels of the tree, then removing any edge that has no label, and then removing any node that does not have two or more children. Even though an implicit suffix tree may not have a leaf for each suffix, it does encode all the suffixes of S, as each suffix is spelled out by the characters on some path from the root of the implicit suffix tree.

3.3. Algorithms

Pseudo-algorithms can be written in LaTeX with the `algorithm` and `algorithmic` packages. An example is shown in Algorithm 1.

Algorithm 1 Name of the Algorithm

```
Initial instructions
for for − condition do
    Some instructions
    if if − condition then
        Some other instructions
    end if
end for
while while − condition do
    Some further instructions
end while
Final instructions
```

4. Some further useful suggestions

Theorems have to be formatted as follows:

Theorem 4.1. *Write here your theorem.*

Proof. If useful you can report here the proof.

Propositions have to be formatted as follows:

Proposition 4.1. *Write here your proposition.*

How to insert itemized lists:

- first item;
- second item.

How to write numbered lists:

1. first item;
2. second item.

5. Conclusions

A final section containing the main conclusions of your research/study and possible future developments of your work have to be inserted in the section "Conclusions".

6. Bibliography and citations

Your thesis must contain a suitable Bibliography which lists all the sources consulted on developing the work. The list of references is placed at the end of the manuscript after the chapter containing the conclusions. It is suggested to use the BibTeX package and save the bibliographic references in the file `bibliography.bib`. This is indeed a database containing all the information about the references. To cite in your manuscript, use the `\cite{}` command as follows:
Here is how you cite bibliography entries: [2], or multiple ones at once: [3, 4].
The bibliography and list of references are generated automatically by running BibTeX [1].

Acknowledgements

Just before References, here you might want to acknowledge someone.

References

[1] CTAN. BiBTeX documentation.

[2] Donald E. Knuth. Computer programming as an art. *Commun. ACM*, pages 667–673, 1974.

[3] Donald E. Knuth. Two notes on notation. *Amer. Math. Monthly*, 99:403–422, 1992.

[4] Leslie Lamport. *LaTeX: A Document Preparation System*. Pearson Education India, 1994.

A. Appendix A

If you need to include an appendix to support the research in your thesis, you can place it at the end of the manuscript. An appendix contains supplementary material (figures, tables, data, codes, mathematical proofs, surveys, ...) which supplement the main results contained in the previous sections.

B. Appendix B

It may be necessary to include another appendix to better organize the presentation of supplementary material.