REPORT

Suffix Tree

Definition:
Suffix Tree for an m-character string has
m leaves numbered 1 to m, edge-label vs node-label with each internal node has at least two children .The label of the leaf j is S[j..m].No two edges out of the same node can have edge-labels beginning with the same character.

Initially, building suffix trees appeared in order to solve the so-called substring problem. This problem can be stated as follows
“The substring problem: Pre-process text T so that the computation string matching problem is solved in time proportional to m, the length of pattern P “.

During the process of gathering information for the implementation of the suffix trees
there were two main algorithms majorly used
1. Naive algorithm
2. Ukkonen’s algorithm

Naive’s algorithm
Given a string S of length m, enter a single edge for suffix S[l ..m]$ (the entire string) into the tree, then successively enter suffix S[i..m]$ into the growing tree, for i increasing from 2 to m.
Let Ni denote the intermediate tree that encodes all the suffixes from 1 to i.
So Ni+1 is constructed from Ni as follows:
Start at the root of Ni
Find the longest path from the root which matches a prefix of S[i+1..m]$
Match ends either at the node (say w) or in the middle of an edge [say (u, v)].
If it is in the middle of an edge (u, v), break the edge (u, v) into two edges by inserting a new node w just after the last character on the edge that matched a character in S[i+l..m] and just before the first character on the edge that mismatched. The new edge (u, w) is labelled with the part of the (u, v) label that matched with S[i+1..m], and the new edge (w, v) is labelled with the remaining part of the (u, v) label.
Create a new edge (w, i+1) from w to a new leaf labelled i+1 and it labels the new edge with the unmatched part of suffix S[i+1..m]
Complexity to build the tree for string of length m is O(m2) .

Ukkonen’s algorithm

There are $i + 1 \leq m$ extensions in phase $i + 1$
In a single extension, the algorithm walks up at most one edge, traverses one suffix link,walks down some number of nodes, applies the extension rules and may add a suffix link.

The up-walk decreases the current node-depth by at most one.

Each suffix link traversal decreases the node-depth by at most another one.
Each down-walk moves to a node of greater depth.
Over the entire phase the node-depth is decremented at most 2m times.
No node can have depth greater than m, so the total increment to current node depth (down walks) is bounded by 3m over the entire phase.

Complexity is O(n)

Analysis

Implementation:

web resource:https://github.com/kasramvd/SuffixTree/blob/master/SuffixTree/suffixtree.py

Suffixtree1.py
Building of the tree is done by using Ukkonne's algorithm where an object of type Suffix tree is created
y=SuffixTree(word)
y.build_suffix_tree()

word is the string that is passed to the class SuffixTree to the object y.
In the next line the building of suffix tree is created .

check substring:

A different class is created to check the substring of a given string.Therefore during the object creation a = CheckSubString(y,'Lion', findall=True)

y->tree
lion->string to be searched
findall->all the strings otherwise longest substring.

Relevance.py

find the maximum number of occurrences of a given substring and rank accordingly the tales depending on how much the story is relevant to the given string.

we have a dictionary created which accepts the a tale with the total number og occurrences of the word and arranged accordingly so that the tales are ranked accordingly.

Sample Outputs:

word=’help’
sample of 50 tales are taken:

```
not present
**************************************************
not present
----------------end of file--------------------------
not present
**************************************************
not present
----------------end of file--------------------------
not present
**************************************************
not present
----------------end of file--------------------------
not present
**************************************************
not present
----------------end of file--------------------------
not present
**************************************************
not present
----------------end of file--------------------------
not present
**************************************************
not present
----------------end of file--------------------------
not present
**************************************************
not present
----------------end of file--------------------------
not present
**************************************************
not present
----------------end of file--------------------------
not present
```

```
**************************************************
not present
----------------end of file---------------------------
not present
**************************************************
not present
----------------end of file---------------------------
not present
**************************************************
not present
----------------end of file---------------------------
[417, 253, 533, 456, 550]
5
Title : Hercules and the Wagoner

occurrences in sentences :

help, until you have done your best to help yourself, or
depend upon it you will henceforth pray in vain

help him

help is the best help

help yourself, or
depend upon it you will henceforth pray in vain

help

**************************************************
first occurrence at 253 and sentence : help him

----------------end of file---------------------------
not present
**************************************************
not present
----------------end of file---------------------------
not present
**************************************************
not present
----------------end of file---------------------------
not present
**************************************************
```

not present
----------end of file----------
not present
****
not present
----------end of file----------
not present
****
not present
----------end of file----------
not present
****
not present
----------end of file----------
not present
****
not present
----------end of file----------
not present
****
not present
----------end of file----------
not present
****
not present
----------end of file----------
not present
****
not present
----------end of file----------
not present
****
not present
----------end of file----------
not present

****
not present
----------end of file----------
not present
****
not present
----------end of file----------
not present
****
not present
----------end of file----------
[546]
1
Title : The Tortoise and the Eagle

occurrences in sentences :

 shell to pieces

****
first occurrence at 546 and sentence : shell to pieces

----------end of file----------
not present
****
not present
----------end of file----------
not present
****
not present
----------end of file----------
not present
****
not present
----------end of file----------
not present
****
not present
----------end of file----------
not present
****
not present
----------end of file----------

[689]
1
Title : The Fox and the Goat

occurrences in sentences :

help you out afterwards

****
first occurrence at 689 and sentence : help you out afterwards

----------end of file----------
[345]
1
Title : The Bear and the Two Travelers

occurrences in sentences :

e held
his breath, and feigned the appearance of death as much as he
could

****
first occurrence at 345 and sentence : e held
his breath, and feigned the appearance of death as much as he
could

----------end of file----------
not present
****
not present
----------end of file----------
not present
****
not present
----------end of file----------
not present
****
not present
----------end of file----------
not present
****
not present

----------end of file----------
not present
****
not present
----------end of file----------
not present
****
not present
----------end of file----------
not present
****
not present
----------end of file----------
not present
****
not present
----------end of file----------
[159]
1
Title : The Lioness

occurrences in sentences :

helps at a birth

****
first occurrence at 159 and sentence : helps at a birth

----------end of file----------
not present
****
not present
----------end of file----------
not present
****
not present
----------end of file----------
not present
****
not present
----------end of file----------
not present
****

```
not present
----------------end of file--------------------------
not present
**************************************************
not present
----------------end of file--------------------------
```

```
***************       is the separation between question 1 and
question2
--end of file--       is the separation between different tales
```

Relevance.py

word:’mouse’
sample output:1

```
11 Title : The Fisherman Piping

10 Title : The Wolf and the Crane

6 Title : The Father and His Sons

2 Title : The Bat and the Weasels

20 Title : The Farmer and the Stork

16 Title : The Mole and His Mother

13 Title : The Ants and the Grasshopper

12 Title : Hercules and the Wagoner

1 Title : The Wolf and the Lamb
```

word=’Lion’
sample output:2

```
4 Title : The Lion and the Mouse
```

```
9 Title : The Kingdom of the Lion

20 Title : The Farmer and the Stork

17 Title : The Herdsman and the Lost Bull

16 Title : The Mole and His Mother

14 Title : The Traveler and His Dog

10 Title : The Wolf and the Crane

7 Title : The Boy Hunting Locusts

5 Title : The Charcoal-Burner and the Fuller
```

word=’man’
sample output:3

```
46 Title : The Boasting Traveler

20 Title : The Farmer and the Stork

17 Title : The Herdsman and the Lost Bull

14 Title : The Traveler and His Dog

12 Title : Hercules and the Wagoner

11 Title : The Fisherman Piping

10 Title : The Wolf and the Crane
```

```
44 Title : The Ass and the Lapdog

40 Title : The Goat and the Goatherd

39 Title : The Raven and the Swan

37 Title : The Oxen and the Axle-Trees

35 Title : The Fox and the Goat

34 Title : The Dog in the Manger

3 Title : The Ass and the Grasshopper

1 Title : The Wolf and the Lamb

29 Title : The Tortoise and the Eagle

27 Title : The Mountain in Labor

26 Title : The Swallow and the Crow

25 Title : The Bear and the Fox
```

Complexity:

find all pattern:
tree creation of length of n : O(n)
traversal for substring check takes O(m)
if there is any suffix of the given substring take O(z)
therefore: O(m+z)

first occurrence: O(m)

relevance:50*O(m+z)

for more practical experimentation refer to the ReadMe which is attached along the code and the report.