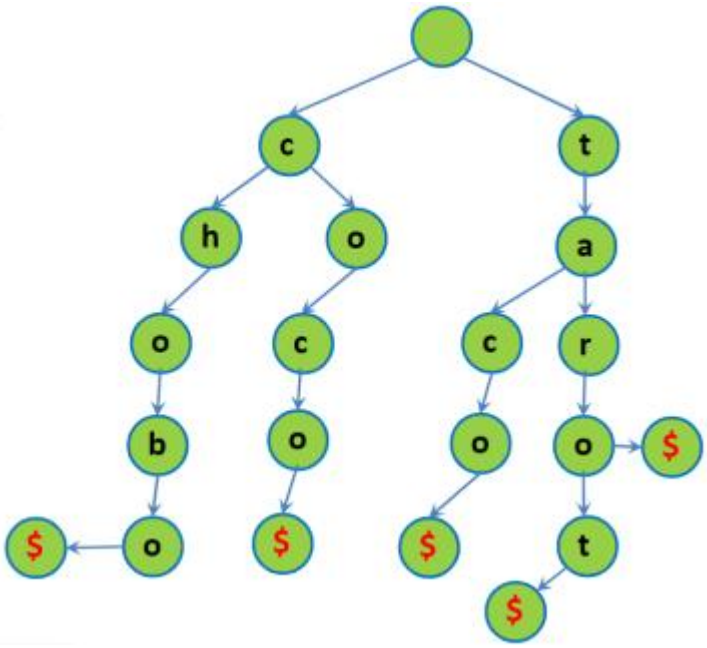
Trie

Fundamentals

- A data structure to store strings efficiently
- Organize strings according to characters
- Height of the trie will be the length of the longest string



- A terminal character is always added to denote the end of a string
- Allows us to efficiently search whether a string exists or not in the trie
- The string “tao” does not exist in the trie and we can determine that quickly using a trie

Complexity

Search -> Worst case: O(M), where M is the length of the search string

Best case: O(1) when the first character isn’t found within the trie

Sorting -> O(MN) where M is the length of the string and N is the number of words

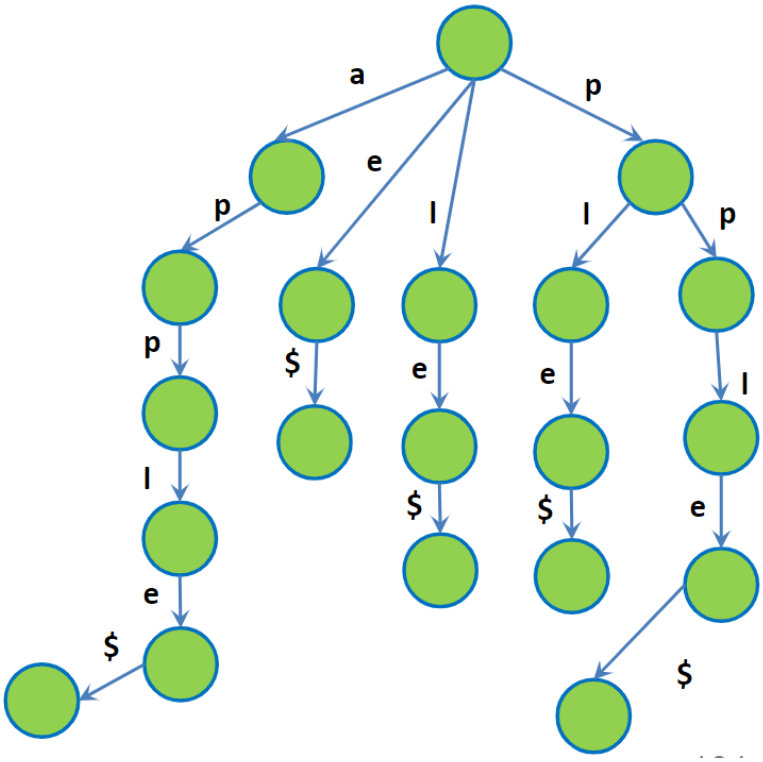
Suffix Trie

What are suffixes?

- Sequence of characters that is appended to the end of a string
- The suffixes of the string “GLHF” is:
  1) “GLHF”
  2) “LHF”
  3) “HF”
  4) “F”

Fundamentals

- It is a trie that stores all suffixes of a string
- Many applications such as finding substring, longest repeated substring etc…



Complexity

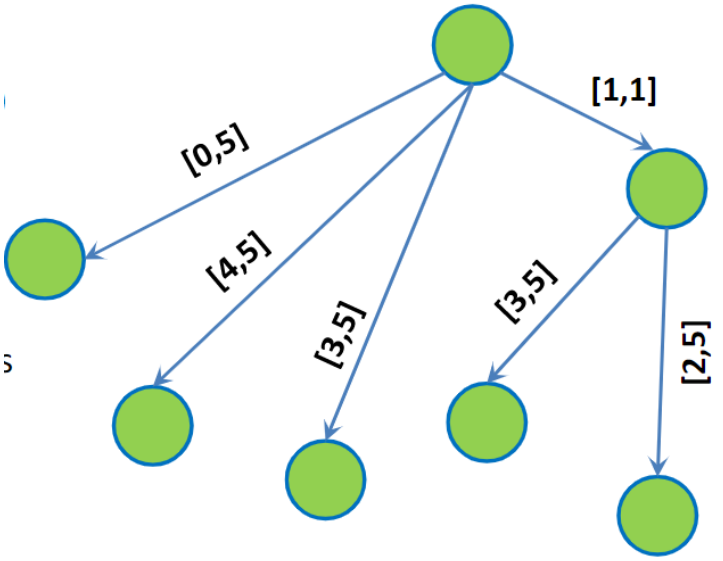
Time Complexity -> $O(N^2)$, where N is the length of the string

Space Complexity -> $O(N^2)$, where N is the length of the string

Suffix Tree

Fundamentals

- It is basically a compressed suffix trie
- Good to learn this because it will be extremely crucial in FIT3155 (Ukkonen's Algorithm)



a	p	p	l	e	$
0	1	2	3	4	5

Complexity

Time Complexity -> $O(N^2)$, where N is the length of the string

Space Complexity -> O(N), where N is the length of the string

Sample paper question

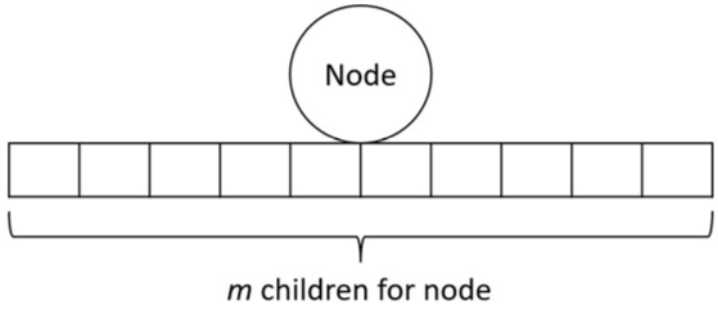
Retrieval Data Structures for Strings

Question 17

Assume that you have an alphabet size of ***M*** **unique characters** and let ***S*** **be a string of length** ***N***.

Thus, you **cannot assume the alphabet size to be** ***O(1)*** as discussed in the lecture. A node implemented with this condition, using an **array of size** ***M*** **for the children** is illustrated below.

2 Marks



For nodes implemented in this way, what is the **worst-case space complexity** in terms of *M* and *N* for string *S* of:

A suffix trie.

- Θ(N log M)
- Θ(N M^2)
- Θ(NM)
- Θ(N)
- Θ(N log N)
- Θ(N+M)
- Θ(N^2 M)
- Θ(M)

A suffix tree, with edges using the [start,end] or [start,length] representation.

- Θ(N log M)
- Θ(N M^2)
- Θ(NM)
- Θ(N)
- Θ(N log N)
- Θ(N+M)
- Θ(N^2 M)
- Θ(M)

Space complexity for suffix trie will be $O(N^2M)$.

Space complexity for suffix tree will be O(NM).