CS213/293 Data Structure and Algorithms 2025

Lecture 11: Trie: storing *string* $\rightarrow$ *Values*

Instructor: Ashutosh Gupta

IITB India

Compile date: October 3, 2025

If keys are strings!

The problem of storing maps boils down to storing keys in an organized manner.

- For unordered keys, we used hash tables
- For ordered keys, we used red-black trees.
- Let us suppose. Our keys are strings.

Strings have more structure, *e.g.*, substring relation.

Can we exploit the structure?

Exercise 11.1

Can we define total order over strings?

Applications of string keys

- Web search
- All occurrences of a text
- Routing table (Keys are IP addresses)

Topic 11.1

Trie

Trie

Let letters of strings be from an alphabet Σ.

Definition 11.1

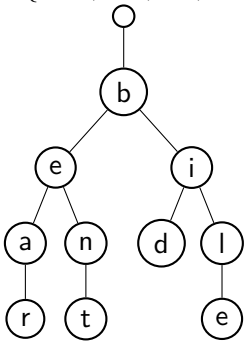
A trie is an ordered tree such that each node except the root is labeled with a letter in Σ and has at most $|\Sigma|$ children with distinct labels.

A trie may store a set of words.

A word stored in a trie is a path from the root to a leaf.

Example 11.1

The following trie stores words $\{bear, bile, bid, bent\}$.

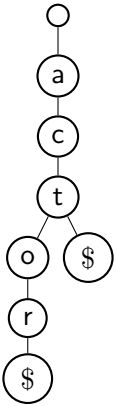


End marker

Sometimes a word is a prefix of another word. We need to add end markers in our trie. In our slides, We will use $.

Example 11.2

Consider set of words $\{act, actor\}$



Running times

- Storage $O(W)$, W is the total sum of lengths of words
- Find, insert, and delete will take $O(|\Sigma|m)$, where m is the length of the input word
  - At each node, we need to search among children for the node with the next letter.

Exercise 11.2

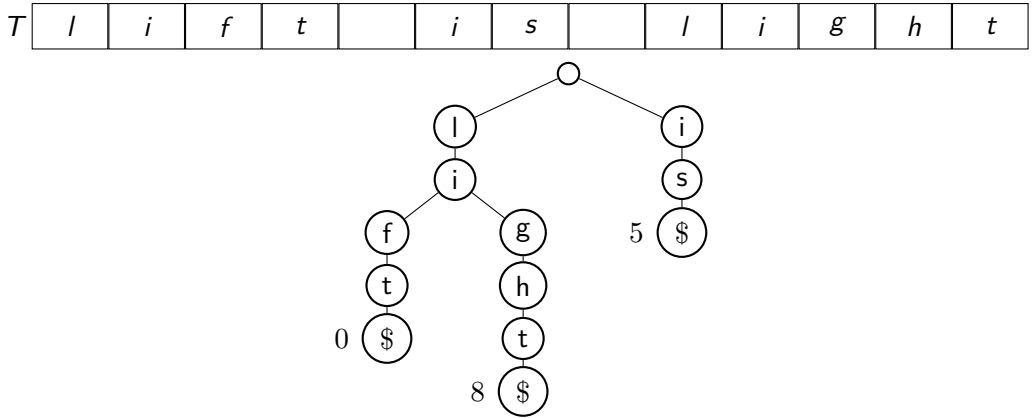
a. Can we do the above operations in $O(m)$ while allowing extra space?
b. Can we do the above operations in $O(m \log |\Sigma|)$ while not using so much extra space?

Application: word search

We use trie to store the positions of all words of a text. The leaves of trie point at

- the first occurrence position of the word or
- a list of all occurrence positions.

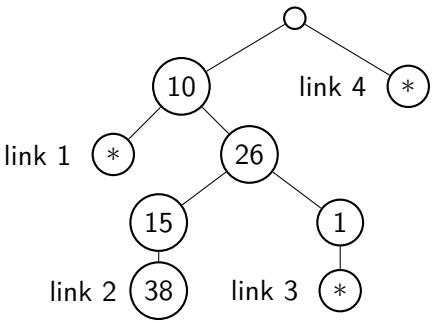
Example 11.3



Application: routing table

Example 11.4

An internet router contains a routing table that maps IP addresses to links attached to the router.



We match the IP address of a packet with the trie. The link with the longest match receives the packet.

Exercise 11.3

Which link will receive the packets for following IP address?

- 21.10.1.6
- 10.26.10.6
- 10.26.1.6
- 10.26.15.9

Topic 11.2

Compressed trie

Compressed trie

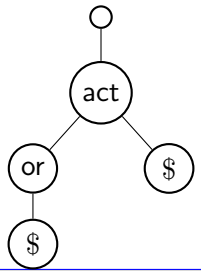
Definition 11.2

A compressed trie is a trie with nodes that do not have single children and nodes are labeled with substrings of words.

Obtained from standard trie by compressing chain of redundant nodes.

Example 11.5

In the following compressed trie, we store words $\{actor, act\}$.



The number of nodes in the compressed trie

Commentary: We did not prove exactly the following theorem. But, we can easily modify our earlier proof.

Theorem 11.1 (Recall)

If each internal node has at least two children, then the number of internal nodes is less than the number of leaves.

Each leaf represents a word.

Therefore, the number of internal nodes in the compressed trie is bounded by the number of words stored in the trie.

Does compression save the space?

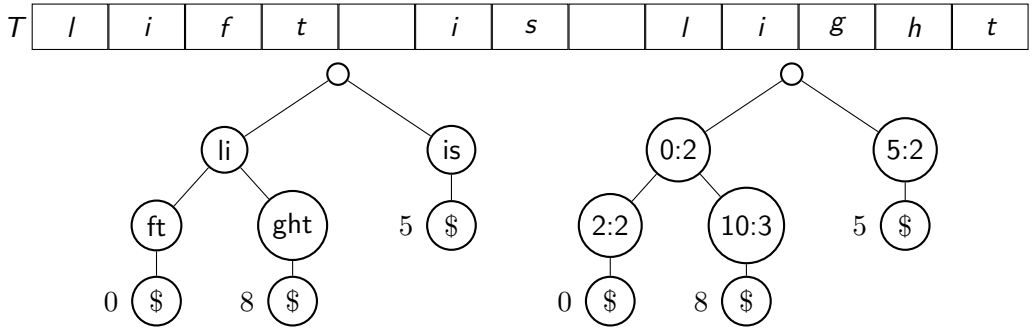
Typical usage of trie: fast search of words on a large text

The text must have been stored separately from the trie.

We need not store the strings on the nodes.

All we need to point at the position of the stored text and the length of the substring.

Example 11.6



Insertion and deletion on trie

Exercise 11.4

Give an algorithm for insertion and deletion in a compressed trie.

Topic 11.3

Suffix tree

Pattern search problem: Another perspective

Typical setting: We search in a (mostly) stable text T using many patterns several times.

Example 11.7

- A text editor, where text changes slowly and searches are performed regularly.
- Searching in well-known large sequences like genomes.

Can we construct a data structure from the text that allows fast search?

Suffix tree

Definition 11.3

A suffix tree of a text T is a trie built for suffixes of T.

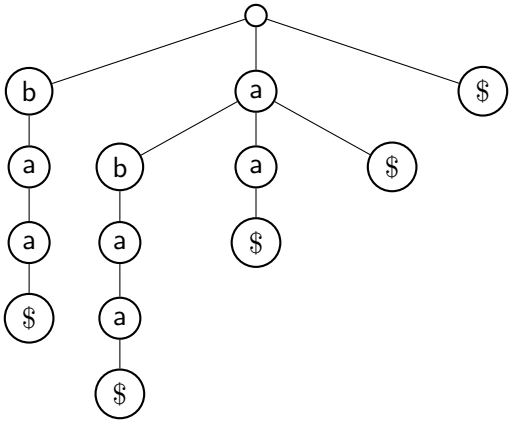
Exercise 11.5

How many leaves are possible for a suffix tree?

Example: suffix tree

Example 11.8

The following is the suffix tree of "abaa".



Usage of suffix tree

- Check if pattern P occurs in T.
- Check if P is a suffix of T.
- Count the number of occurrences of P in T.

Exercise 11.6

Given suffix tree of a text T, find the longest string that repeats in T. (midsem 2024)

Commentary: Answer: deepest node that has at least two children.

Suffix links

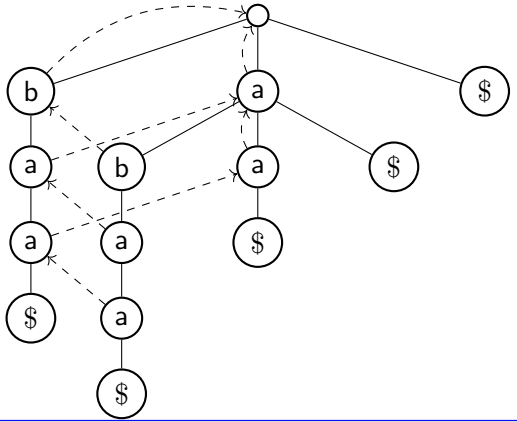
We can solve more interesting problems if we add more structure to our suffix tree.

Definition 11.4

In each node $x\alpha$, we add a pointer suffix link that points to node α.

Example 11.9

The following is the suffix tree of "abaa" with suffix links.



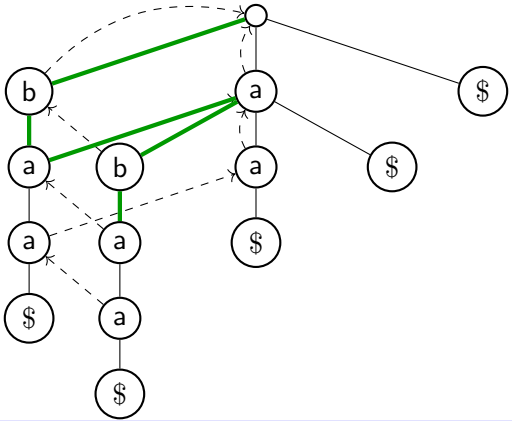
Usage of suffix links

Find the longest sub-string of T and P.

- Walk down the suffix tree following P.
- At a dead end, save the current depth and follow the suffix link from the current node.
- After exhausting P, return the longest substring found.

Example 11.10

The following is the suffix tree of T = "abaa". Let us find the longest sub-string of "baba" and T.



Topic 11.4

Constructing suffix tree

Suffix tree construction

If we have the suffix tree for $T[0:i-1]$, we construct the suffix tree for $T[0:i]$.

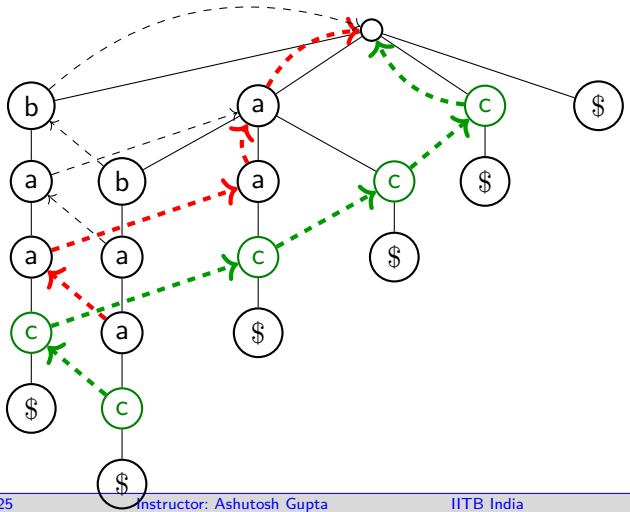
In the order of the suffix links, Insert $T[i]$ at the end of each path of the tree,

Exercise 11.7

a. What is the complexity of the above algorithm?
b. What will be the complexity of suffix tree construction without suffix links?

Example 11.11

Let us add c in the suffix tree of "abaa".



Ukkonen's algorithm

Suffix links give us $O(n^2)$ construction, can we do better?

Yes.

Ukkonen's algorithm uses more programming tricks to achieve $O(n)$. We will not cover the algorithm in this course.

Topic 11.5

Tutorial problems

Exercise: suffix tree

Exercise 11.8

Compute the suffix tree for *abracadabra*\$. Compress degree 1 nodes. Use substrings as edge labels. Put a square around nodes where a word ends. Use it to locate the occurrences of *abr*.

Exercise: worst-case suffix tree

Exercise 11.9

Review the argument that for a given text T, consisting of k words, the ordinary trie occupies space which is a constant multiple of $|T|$. How is it that the suffix tree for a text T is of size $O(|T|^2)$? Give a worst-case example.

Commentary: Milind notes

Topic 11.6

Problems

True or False

Exercise 11.10

Mark the following statements True / False and also provide justification.

1. A node in a trie has at most two children.
2. A trie for n words has n^2 leaves.
3. KMP is more efficient than trie when there are a large number of searches on a fixed text.
4. Each node in a trie represents a unique word.

End of Lecture 11