om sai Ram.

RCS-502

Design and Analysis of Algorithm

UNIT- II.

1. Tries

2. Skip lists

String processing has a variety of real world applications such as:

- Search Engines
- Genome Analysis
- Datā analytics.

Therefore strings are essentially thē most important & common topics for programming problems.

TRIES : are an extremely special and useful datastructure that are based on the prefix of a string. The word is derived from ReTrieval.

Prefix : of a string is any n letters $n \leq |S|$ that can be considered beginning strictly from the starting of thē string. For eg. thē word ababaac has following prefixes:

a
ab
aba
abab
ababa
ababaa
ababaac

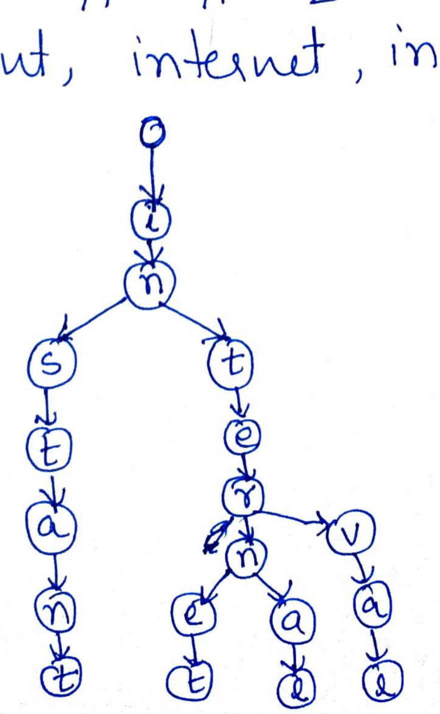
A trie is a tree representing a collection of strings with one node per common prefix.

Smallest Tree such that:

- Each edge is labeled with a character c∈Σ
- A node has at most one outgoing edge labeled c, for c ∈ Σ
- Each key is "spelled out" along some path starting at the root.

A Trie is a special data structure used to store strings that can be visualized like a graph. Each node consists of atmost 26 children. strings are stored in a top to bottom manner on the basis of their prefix in a tree.
All prefixes of length 1 are stored at level 1,
" " " " 2 " " " " 2 ...

foreg. instant, internet, internal, interval



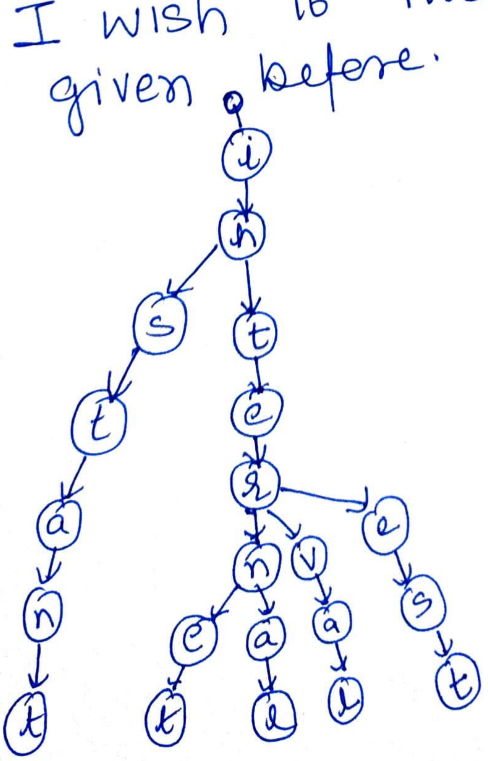
Inserting a string into a trie:

```
For each char in string s
    • if child node belonging to current char
      is null
        ↳ Then make a new child node
    • make this child node the current node.
```

Check whether a word exists in a dictionary of words or not:

```
for every char in string s
    • if child node is null
        return false.
• return true.
```

eg. (i) If I wish to insert "interest" into the trie given before.



for each char in interest

. i	→ not null
n	→ not null
t	– not null
e	– not null
r	→ not null
e	↳ null, new node
s	→ null, new node
t	→ null, new node.

(ii) If I wish to search for a string "ink" in the above trie

→ i – found
n – found
k – no child node → return false.

Analysis.

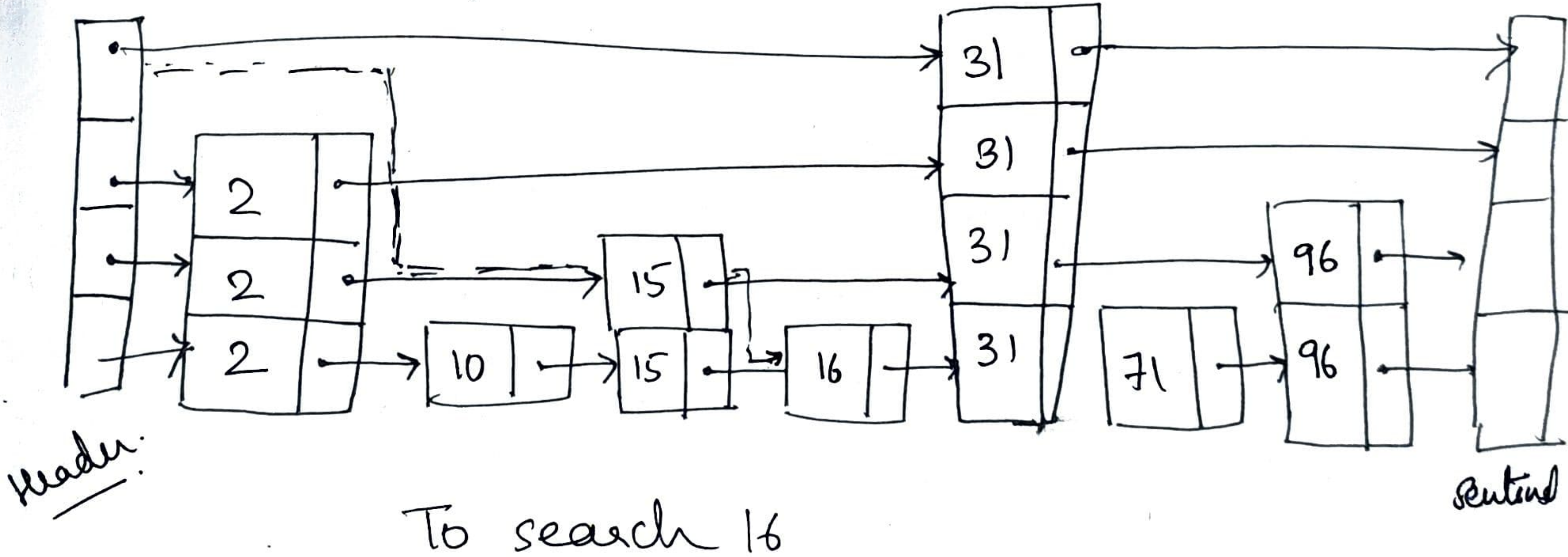
A standard trie uses $O(n)$ space and supports searches, insertions & deletions in time $O(dm)$, where,

n = total size of the string in S
m = size of the string parameter of the operation
d = size of the alphabet.

Skip Lists

Is a probabilistic data structure that is built upon the general idea of a linked list. The skip list uses probability to build subsequent ~~lists~~ layers of linked list upon an original linked list. Each layer of linked list contains fewer elements but no new elements.

- * Linked lists are very useful data structures as ~~it~~ is very easy to insert & delete elements - in constant $O(1)$ time.
- * search is costly in linked lists - $O(n)$
- * Skip lists fix this problem by reducing the search time to $O(\lg n)$.



To search 16

Perfect skip lists

- keys in sorted order
- O(logn) levels
- Each higher level contains ½ the elements of the level below it.
- Header & Sentinel nodes are in every level

* Called skip lists because higher levels lists let you skip over many items.

* To find an item, we scan ~~the~~ along the shortest list until we would pass the desired item.

At that point, we drop down to a slightly more complete list at one level lower.