~~Dial Ride Problem~~ .

Trie Data structure :-

- Introduction
- Algo - How to insert , ~~&~~ search and delete in Trie
- Application

Algo :- root represent empty string (" ")

- Trie is Collection of trie Node
- 2 main Components

(1) map where the key is character & value is trie node &

- used to establish the parent child relationship

(2) Indicating that character represented , representing the trie node is a end of word (or) not.

```
        Trie Node
key   {
 ①      Map <char, TrieNode>
value                            chil
 ②       boolean endofword
      }
```

→ Introduction :

There are many Algo & data structure to index and search strings inside a text, some of them are included in the standard libraries, but not all of them, the trie DS is good example of one that until now not included in the standard libraries.

→ Trie / digital tree / radix tree / prefix tree (∵ searched by prefixes)

→ Trie is a kind of search tree - (an ordered tree DS i.e) used to store dynamic set (or) associated array where are keys are usually strings.

set → store certain values / no order / no repeated value
Associated array - Collection of <key, value> pairs. in which is unique.

middle syllable of retrieval.
(or) infix

③ Algo

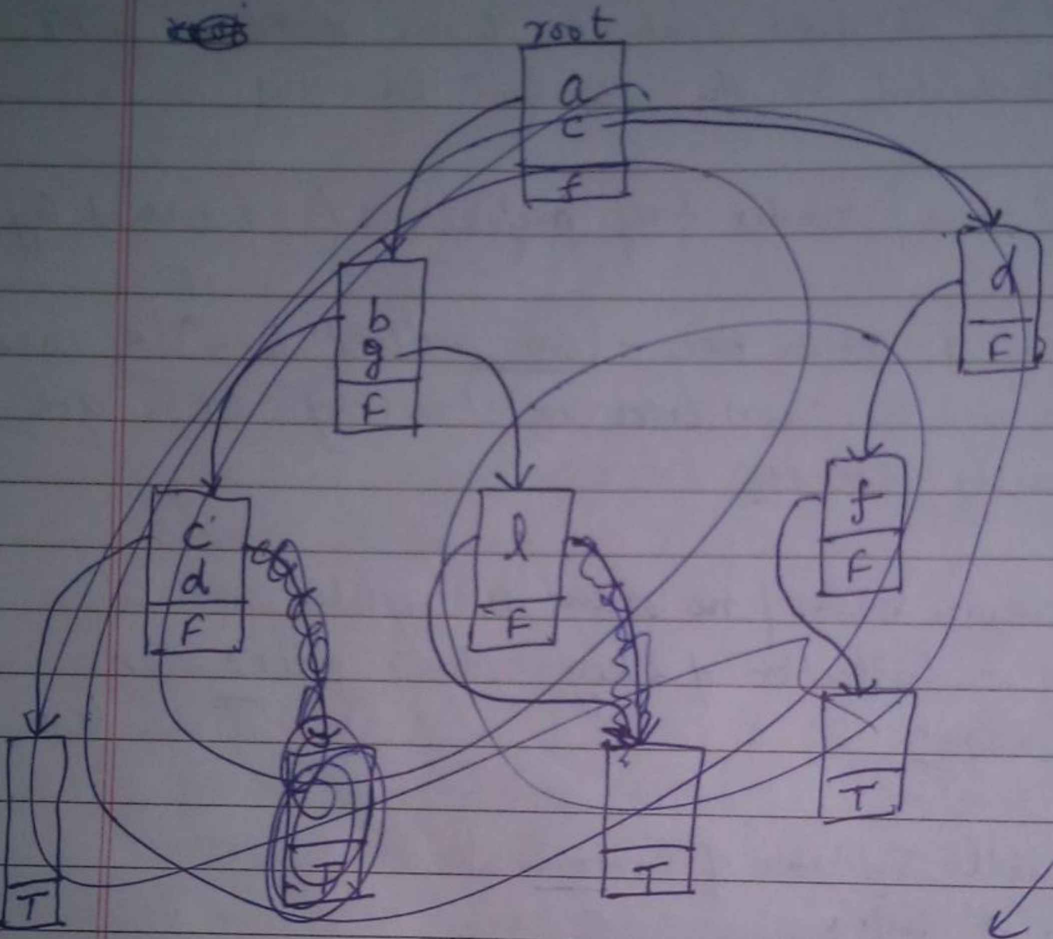
③ ~~Algo~~ :-

root

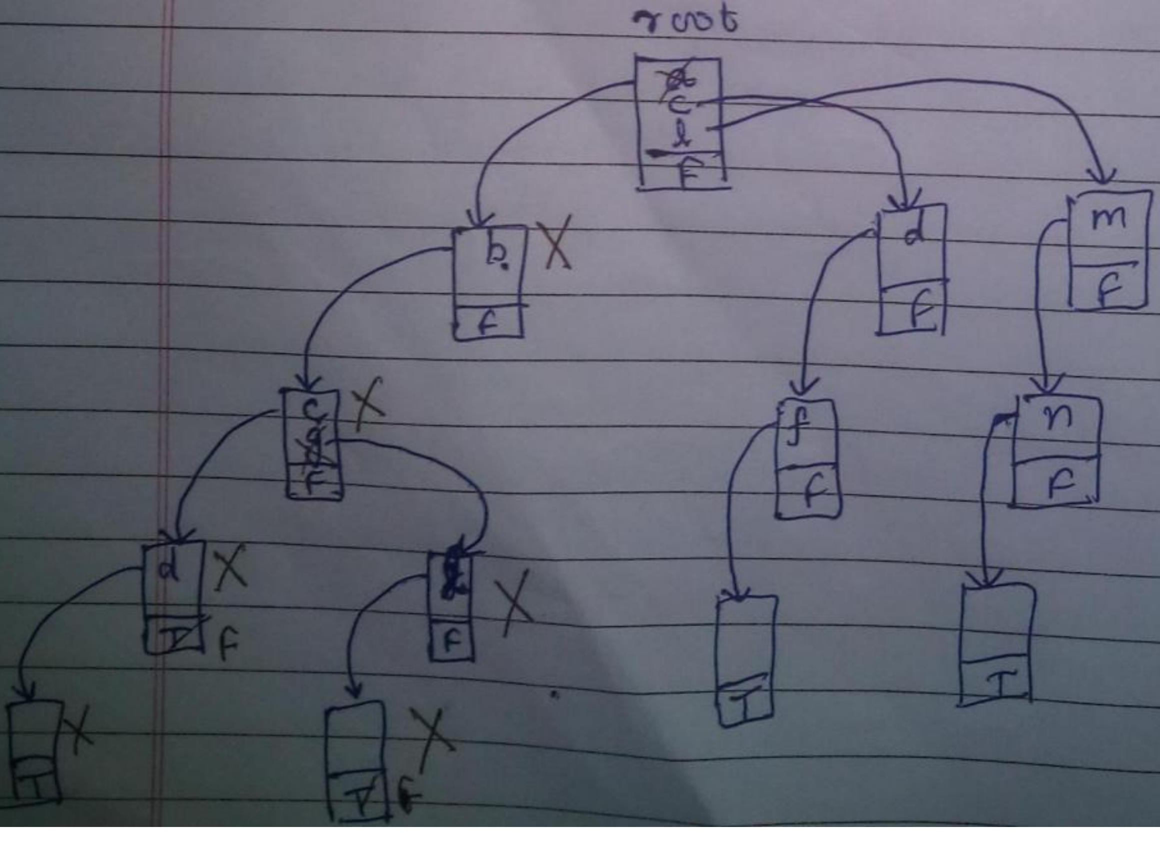
{} → empty map

F → boolean end of word is false

abc
abgl
cdf
abcd
lmn



Insertion



⇒ O(l×n)

l = avg. length of word

n = total # words

over hash Table

→ hash table can find words in dictionary that match exactly with the single word we are finding but.

at least one word start with ab —

Search: 2 kind of searches

trie trie allows us to find words that have a single character different, a prefix in common, a character missing etc

prefixes based: ab, lo → return T, F

whole word: : lmn, ab, cdf, gh.
T F F F

↳ $O(l)$ l = length of word

deletion: words

a b c
a b g l
a b c d

$\Rightarrow O(l \times n)$

⑤ Applications :-

① Replacement for other data structures

(+) Advantages over BST → no node in tree stores the key associated with that node.

- over hash

Adv

i) looking up data in a trie is faster in worst case. $O(m)$ time (m = length of search string) compared to imperfect hash table

ii) There is no collision of different keys in trie.

iii) No Need to provide hash function or change hash functions as more keys are added to a trie.

iv) Trie can provide alphabetical ordering of entries by key

Dis

- Tries can be slower in ~~worst~~ some case than hash Table for looking up data especially if the data is directly accessed on a hard disk or some other secondary storage device

- some keys such as floating point numbers, can lead

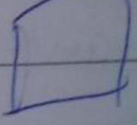
- some tries can requires more m/m then hash Table, as mem may be allocated for each character in ten search string. rather than a single chunk of mem for the whole entry ; as in most hash Tables.

② Dictionary Representation.

→ Used to storing a predictive text (or) autocomplete dictionary such found in Mobile phones

- also Used in implementing approximate matching algorithms

→ Used in The Boggle Word Game.



O(1)

If we are look up a word then hashTable is faster
If we find common prefixes, ordered retrieval, use trie.