SEARCH ENGINE

Name : Christy Biju
MIS : 112003031
Division : 1

Data Structures used :

- Trie :- - s : Search function
  -a [Prefix] : Autocomplete feature

- Splay Trees :- -r : Recently Searched Question
  -fm : Most frequently searched questions
  -fl : Least Searched questions
  -h : Displaying the History
  -d : Clear History

Why Trie Data Structure ?

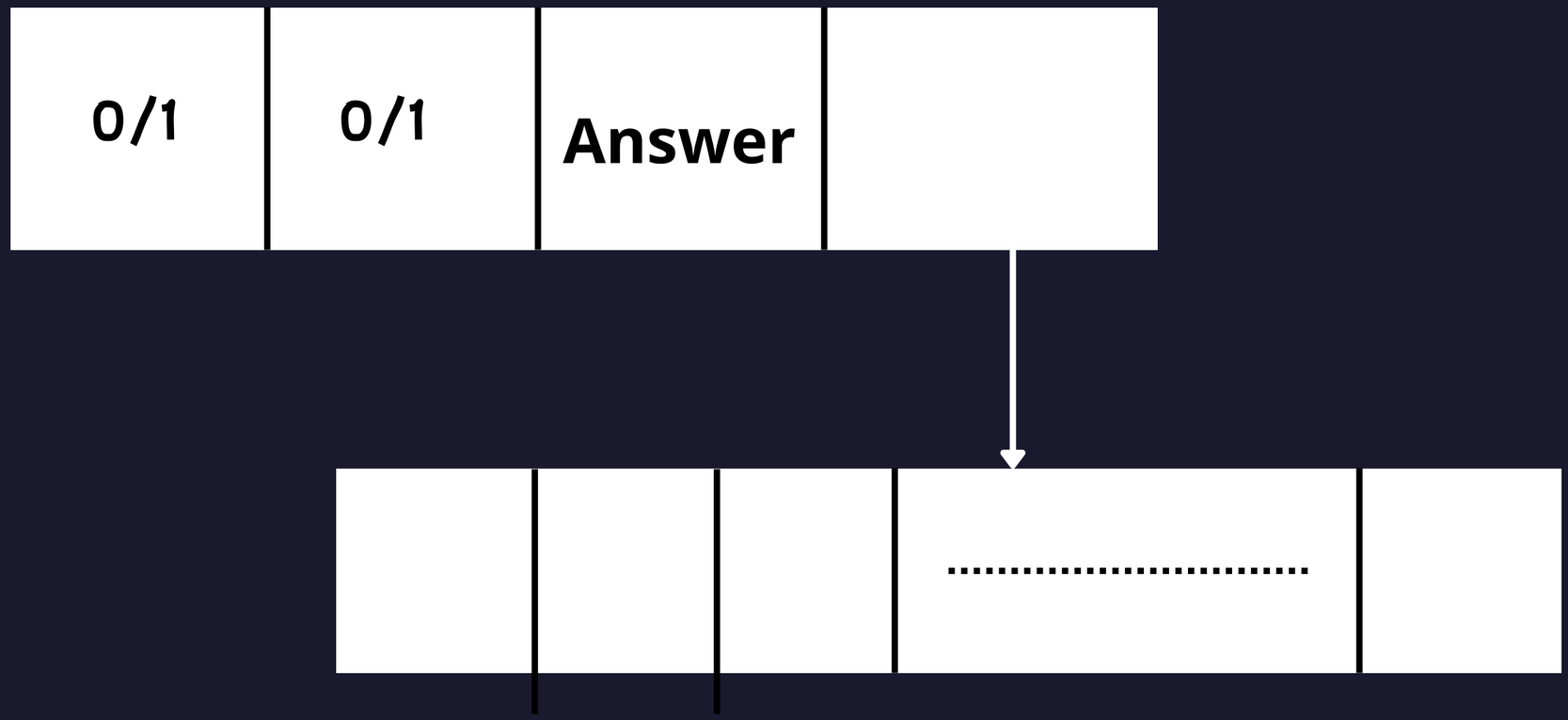
- Trie is a tree that stores strings .
- Insertion and searching takes O(L) time where L is the length of the string.
- Trie is preferred over hash as the autocomplete feature can be easily implemented using trie(prefix tree). Also, we do not need to compute any hash function for searching.
- Trie is more flexible compared to hash as it supports prefix searches and also alphabetical traversal.

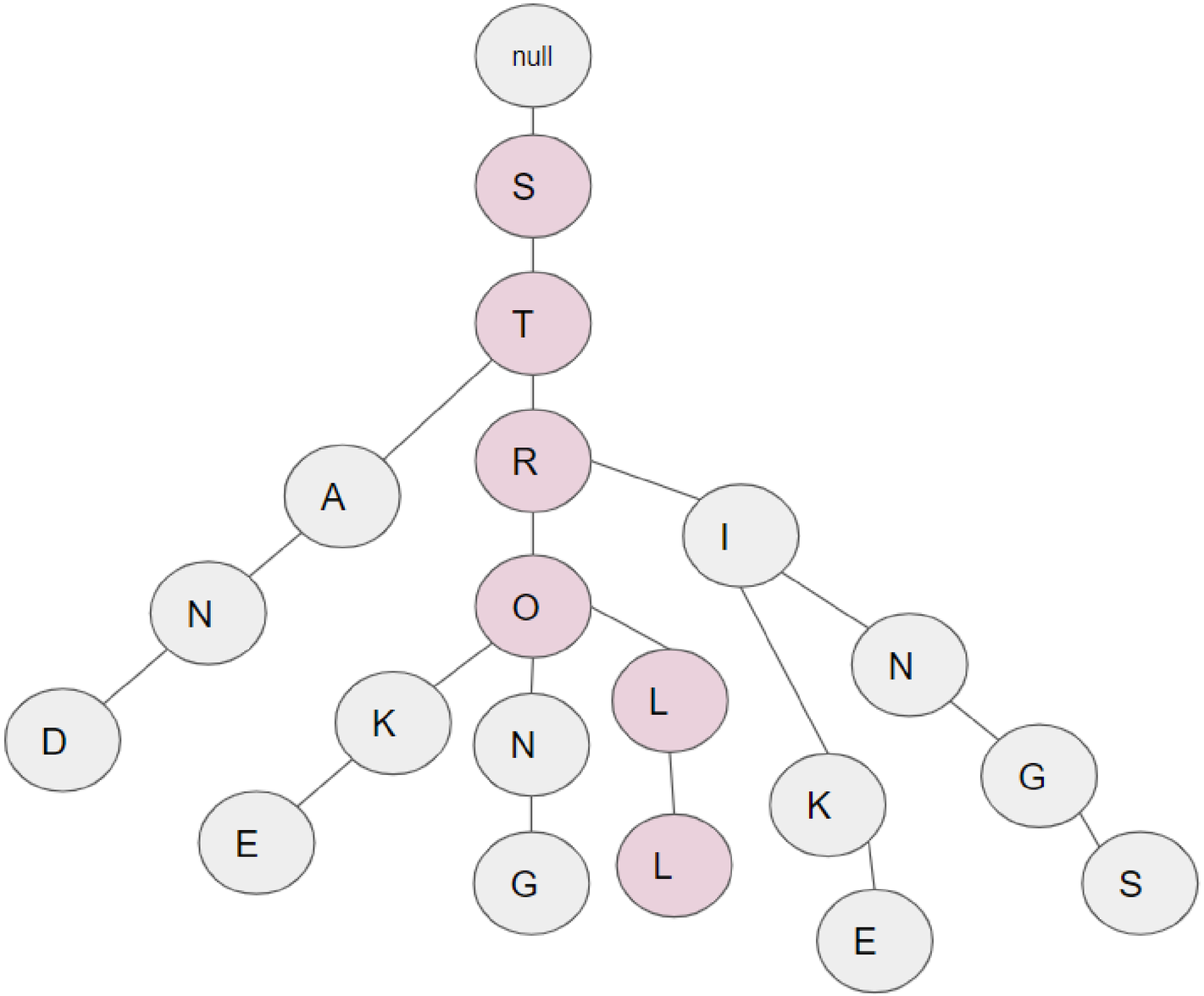
Structure of Node

Space (End of word)

End of question

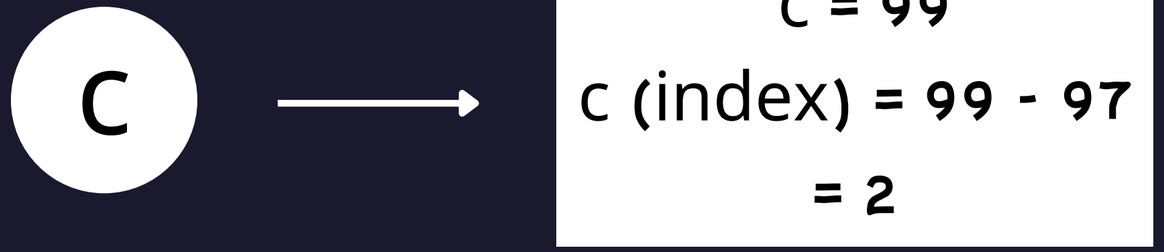
Array of 26 pointers

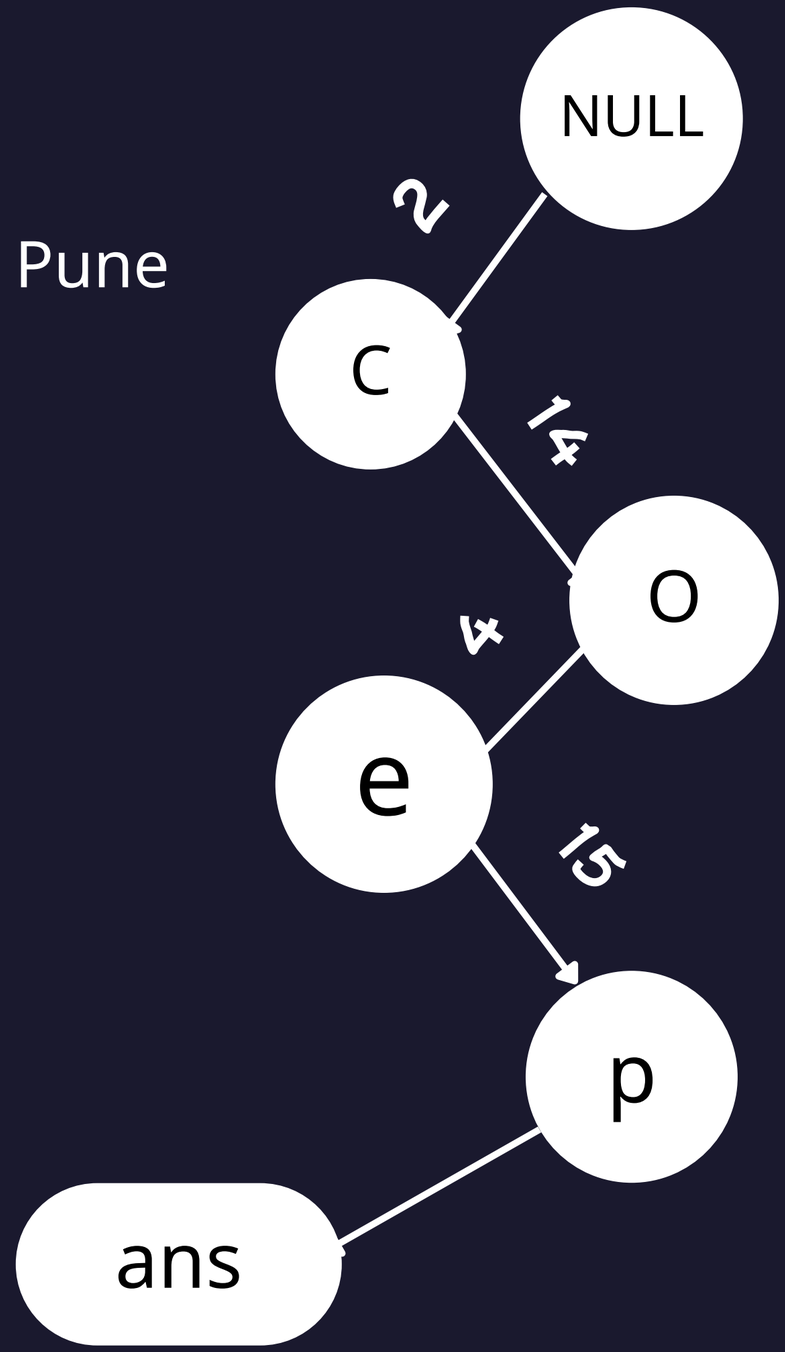



null
S
T
A
R
I
N
O
N
D
K
N
L
G
K
G
E
L
E
S

coep College of Engineering Pune

question : coep ans : College of Engineering Pune

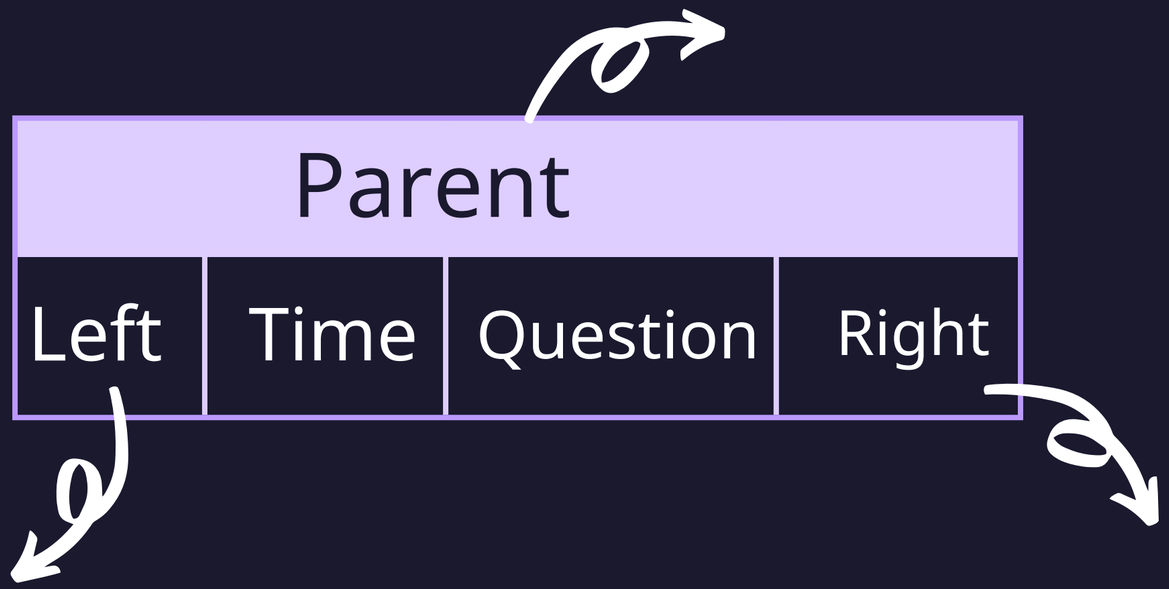




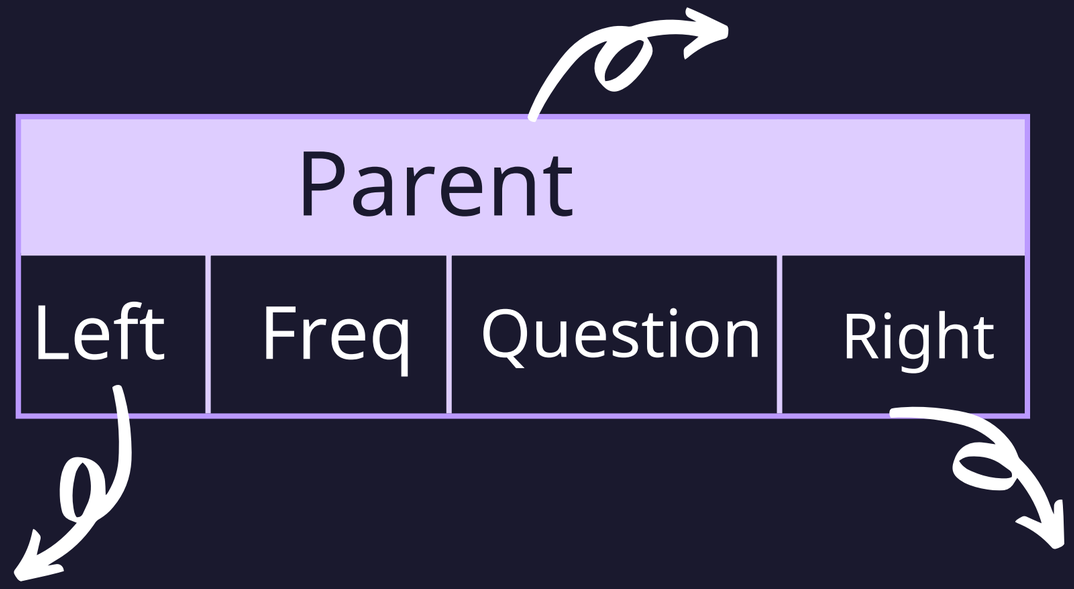
Why Splay Trees ?

- A splay tree is a self-balancing tree
- It requires less space as no balance information is required.
- The main idea of splay tree is to bring the recently accessed item to root of the tree, this makes the recently searched item to be accessible in O(1) time if accessed again.
- It provides better performance as the frequently accessed nodes will move nearer to the root node, due to which the elements can be accessed quickly in splay trees.

1) Recent searched question,
2) Displaying History,
3) Clearing History



1) Frequently asked questions,
2) Least asked questions



Time Complexity Analysis:

Trie

- Insertion :
  Avg case : O(L) ; L = Length of string
- Searching :
  Best case : O(1)
  Avg case : O(L)
- Autocomplete (Prefix Search) :
  Avg Case : O(n), n is number of nodes included (Prefix and branches)

Splay Tree

- Insertion :
  Avg Case : O(log n) ; n = no. of nodes
- Recently/frequently asked question :
  Avg Case : O(1)