

18/03/2024

Page No.	
Date	

Day 21 of DSA

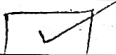
Tasks

Check
Box

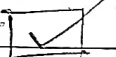
Trie



Implement basic trie, data structure



Trie: operations : Insertion, deletion, searching



Practice Implementation of Trie



Solve problems related to trie DS

* Trie :-

- Tree like data structure
- Used to store dynamic set of strings
- Efficient retrieval of keys & their prefixes
- Efficiently search for strings

Structure :-

- Each node of a Trie represents a single character of the string.
- The root node represents empty string.
- Each node may have multiple children.

* Operations :-

- Insertion
- Deletion
- Searching

* Insertion :-

- ① Start from the root Node.
- ② For each character in the string to be inserted:
 - a. Check if the character is already present as a child of the current node.
 - If yes, move to the child node corresponding to that character.
 - If no, create a new node for the character and add it as a child of the current node.
- ③ Once all characters are inserted, mark the last node as the end of the string.

Insert("apple")

Root

|

a

|

p

|

p

|

|

|

e

* Searching :

- ① Start from root Node
- ② For each character in the string to be searched:
 - a. Check if the character is present as a child of the current node.
 - If yes, move to the child node corresponding to that character.
 - If no, the string is not present in the Trie.
- ③ After traversing all characters
 - Check if the last node is marked as the end of a string.
 - If yes, the string is present in the trie otherwise, it's not.

* Deletion :-

① : Perform a search for the string to be deleted

② IF string is Found.

a. Mark the last node as not the end of a string.

b. IF the last node has no other children delete the node

c. Repeat step b recursively, until a node with other children is reached

③ IF the string is not found, it's not present in the Trie.