DAY-10

Trie

→ a tree like structure where each node represents a single character of a given string.

NOTE: * Unlike binary tree, a node can take more than two children.

→ search complexities can be brought to to optimal limit with "Trie".

=> BST takes $M \log N$ time M - max string len.
N - No. of key.

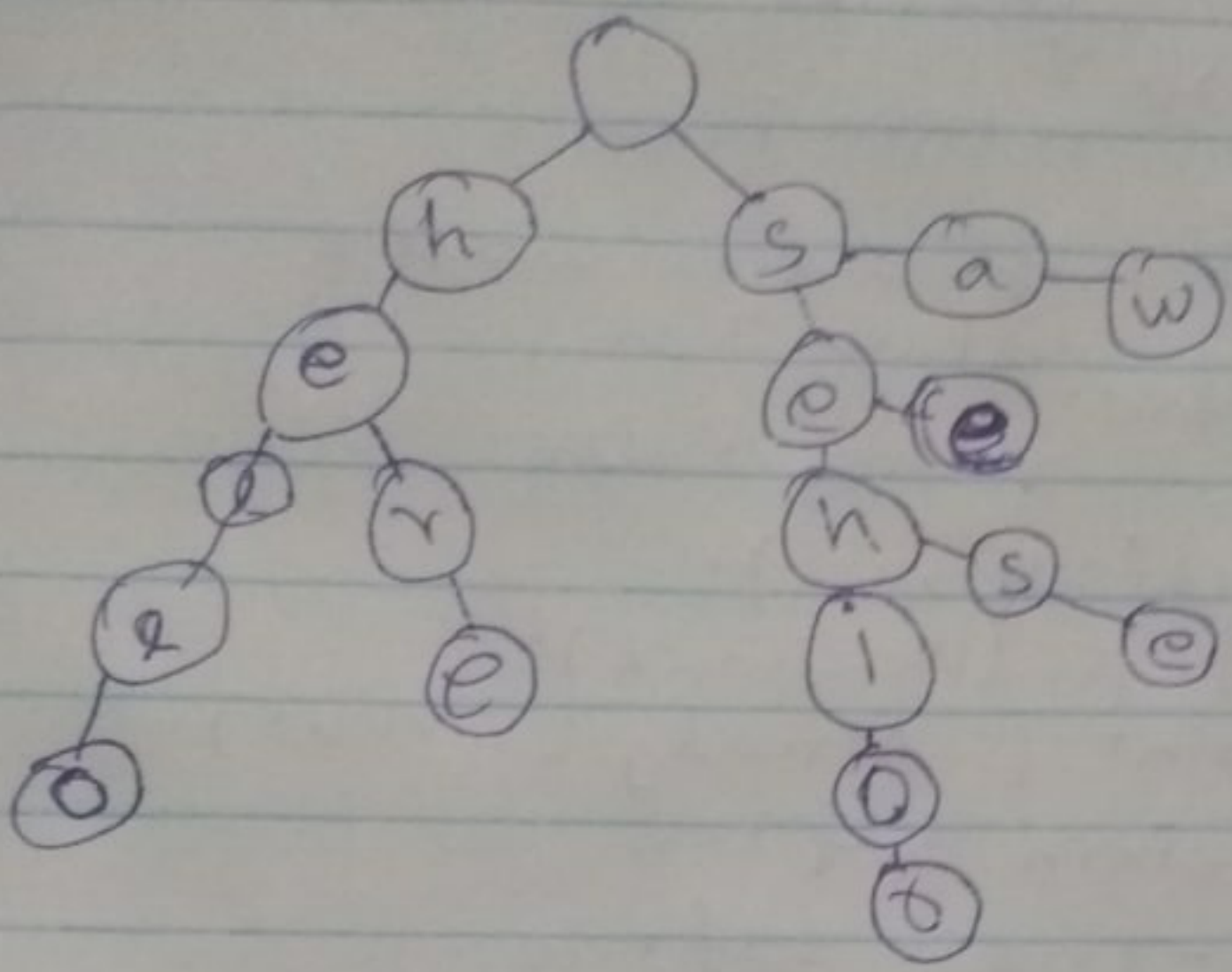
=> Trie — $O(M)$.

but, more storage needed.

If you ever wondered how you could type without spelling mistake,

Well, it is because of "Trie".

Implementation:




```
class Trie():
    def __init__(self):
        self._end = '*'
        self.trie = dict()
    def __repr__(self):
        return repr(self.trie)

    def ~~make Trie~~
    def makeTrie(self, *words):
        trie = dict()
        for word in words:
            temp_dict = trie
            for letter in word:
                temp_dict = temp_dict.setdefault(letter,
                                                 {})
            temp_dict[self._end] = self._end
        return trie
```

```
def findWord(self,word):
    sub_trie = self.trie
    for letter in word:
        if letter in sub_trie:
            sub_trie = sub_trie[letter]
        else:
            return False.
    else:
        if self._end in subtrie:
            return True
        else:
            return False

def addWord(self,word):
    if self.findWord(word):
        print("Already Irukku")
        return self.trie
    temp_trie = self.trie
    for letter in word:
        if letter in temp_trie:
            temp_trie = temp_trie[letter]
        else:
            temp_trie = temp_trie.setdefault(letter,{})
    temp_trie[self._end] = self._end
    return temp_trie
```