CAPSTONE PROJECT

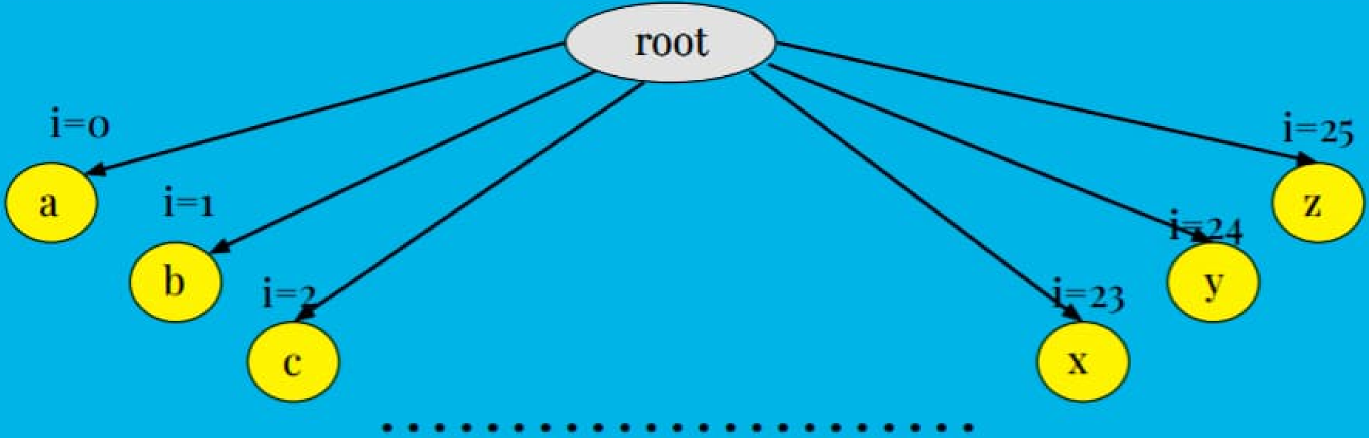
-THE CODE CRUSADERS

GROUP MEMBERS-
VARSHIL JAYESHBHAI PATEL-202301010
KATHAN PARIKH-202301133
ARYAN PATEL-202301005
KRISH PATEL-202301078

PROBLEM

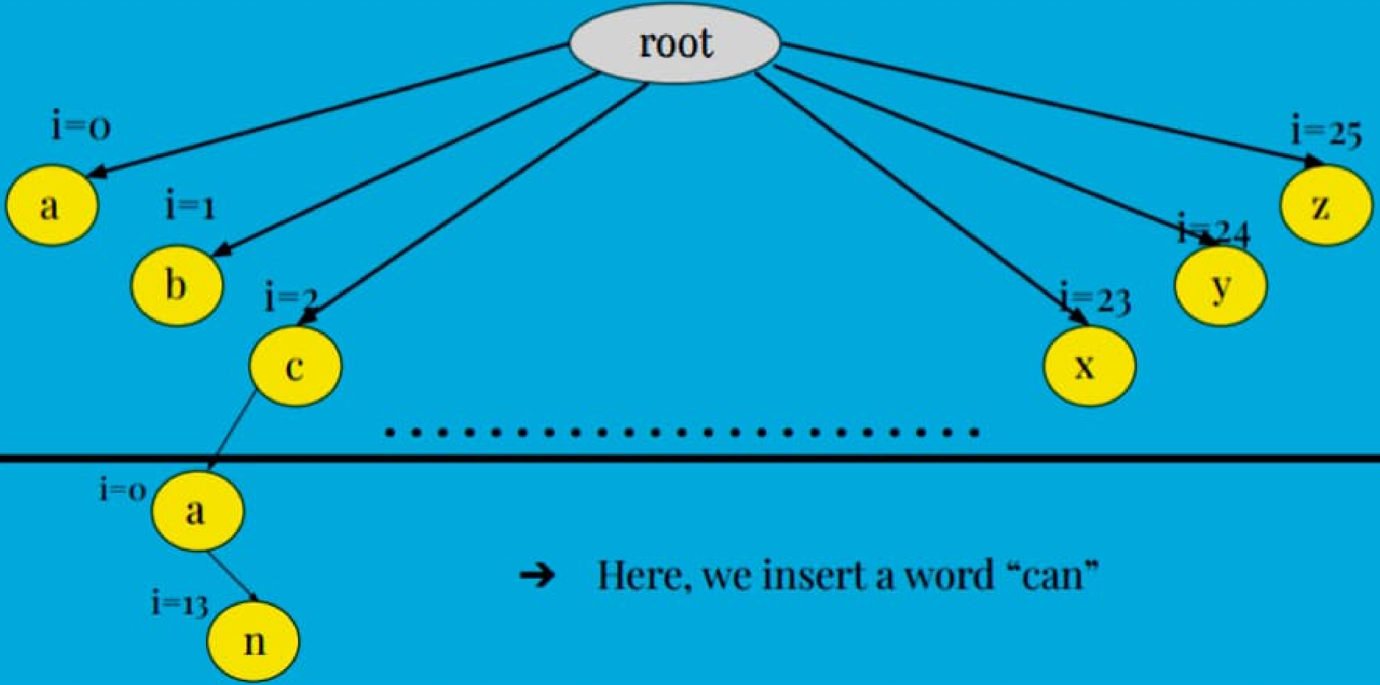
P8. Spell Checker:
Develop a spell checker application that checks the spelling of words in a given text document, utilizing data structures like hash tables or trie for efficient dictionary storage and lookup. After highlighting the misspelled words, with the user's approval, replace them with the correct spellings.

Algorithm



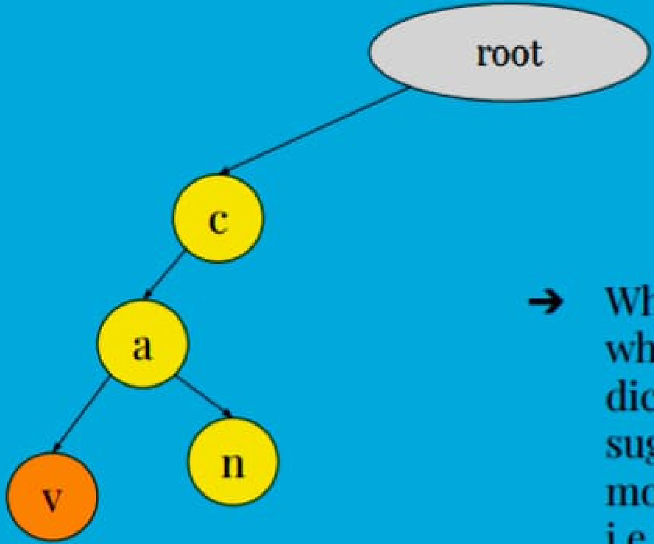
This shows the creation of root which has 26 children i.e. 'a' to 'z'

Insertion



➔ Here, we insert a word "can"

Suggestions



- ➔ When we try to find a word "carv" which is not present in the dictionary, it would give suggestions of name which are most likely to be similar to "cav" i.e. the word "can" which we inserted before.

Dictionary Inserting

```
Declare function load_dictionary(filename)

while(file >> word)

push word in file

return dictionary
```

Main Function

```
In main function
load dictionary into a trie
open input file
read each word from the input file
    If(trie.search(word))
    append it to coloured_file and wrong_file
            else
    append it to wrong_word and coloured_file wit[illegible] color
Correct paragraph using corrected_word and using [illegible]
Print corrected paragraph
```

DATA STRUCTURE CHOSEN

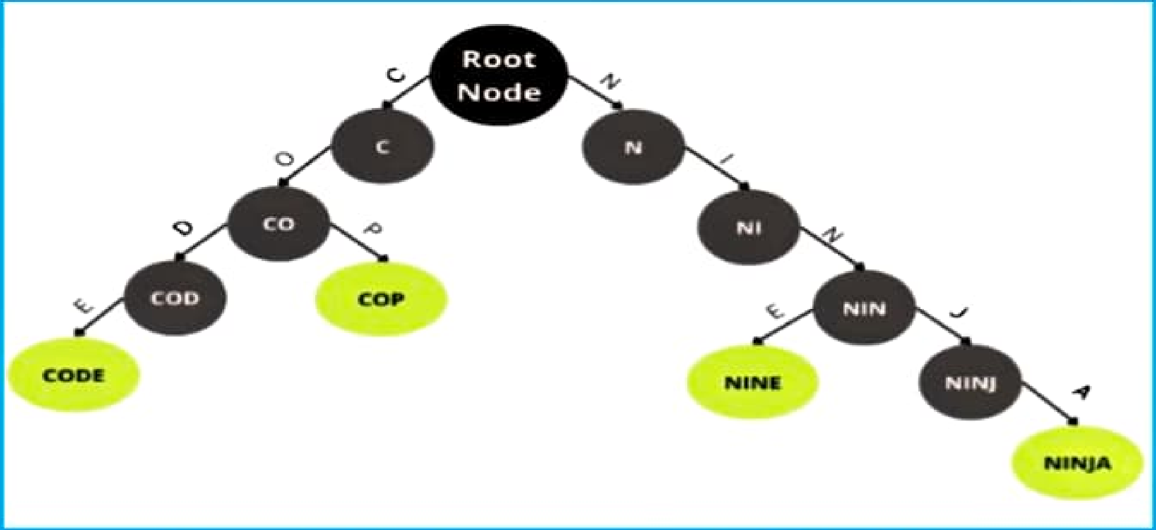
The data structure we have chosen for given capstone project is Trie.

WHY TRIE?

For this project we had two options either to go with hashmaps or to go with Trie, so we decided to go with trie due to the reasons listed on the next slide.

1) Efficient for operations involving common prefix

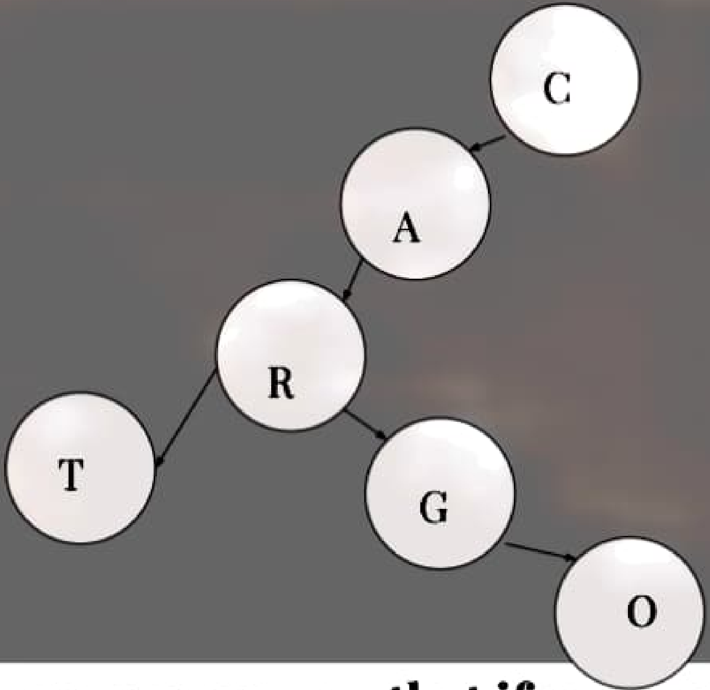
In hashtable we have to make assign new space to all words we enter, no matter how much common things they have in them, but in trie this is not the case, although hashtables provide better time complexity but the advantage in space complexity trie provides is not comparable to hashtables.



- **The above picture depicts the reasoning of efficient for operations involving common prefix, here we see that while entering the word CODE we also entered another word COP, which would not have been possible with hashtables, because we would have to create whole new space for new word.**

2) Tries are an efficient choice in giving suggestions of incorrect word

Tries are efficient in giving suggestions because while checking the word we will reach till a place from where the word becomes wrong, so from that place we can easily give suggestions to the user.



Like here you can see that if someone has entered word "carv", and we know it is wrong, so here can give suggestion of "cart" and "cargo". (This diagram is just for illustration purpose, the actual trie we made has 26 children node for each node)

3)Tries are also memory efficient when dealing with large dictionary

In trie, as explained previously we can reuse the entered word if the entered word is mostly common with other word, but in hashtables it is not possible, and also hashtables store entire word, but in trie we store characters and form word using it.

Time-Complexity

- **Trie Construction & loading a dictionary**:

  The time complexity of constructing the Trie and loading a dictionary is ***O(n.m)*** where 'n' is the size of the dictionary and 'm' is the average length of the words in the dictionary.

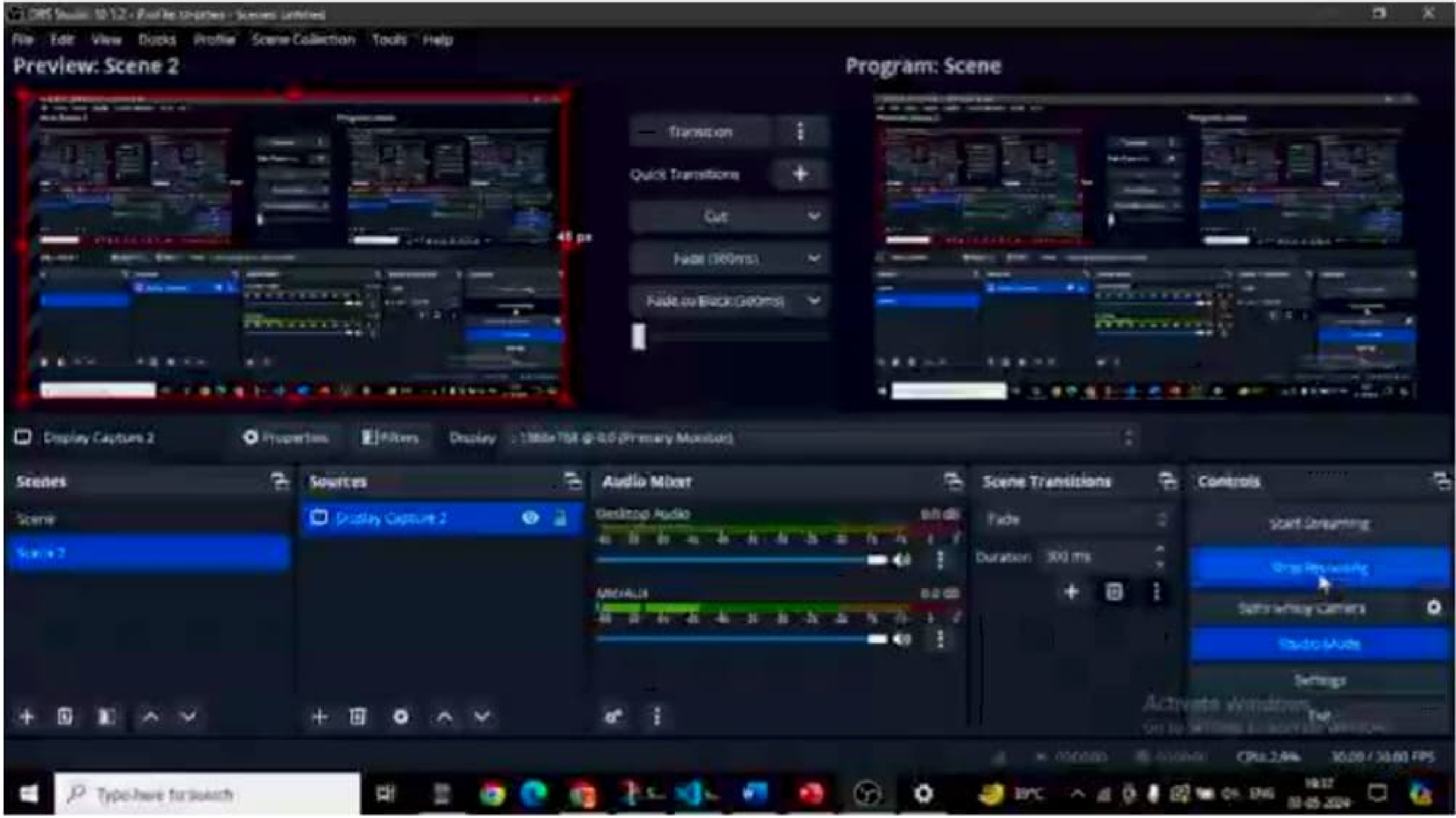
- **Searching in a Trie :**

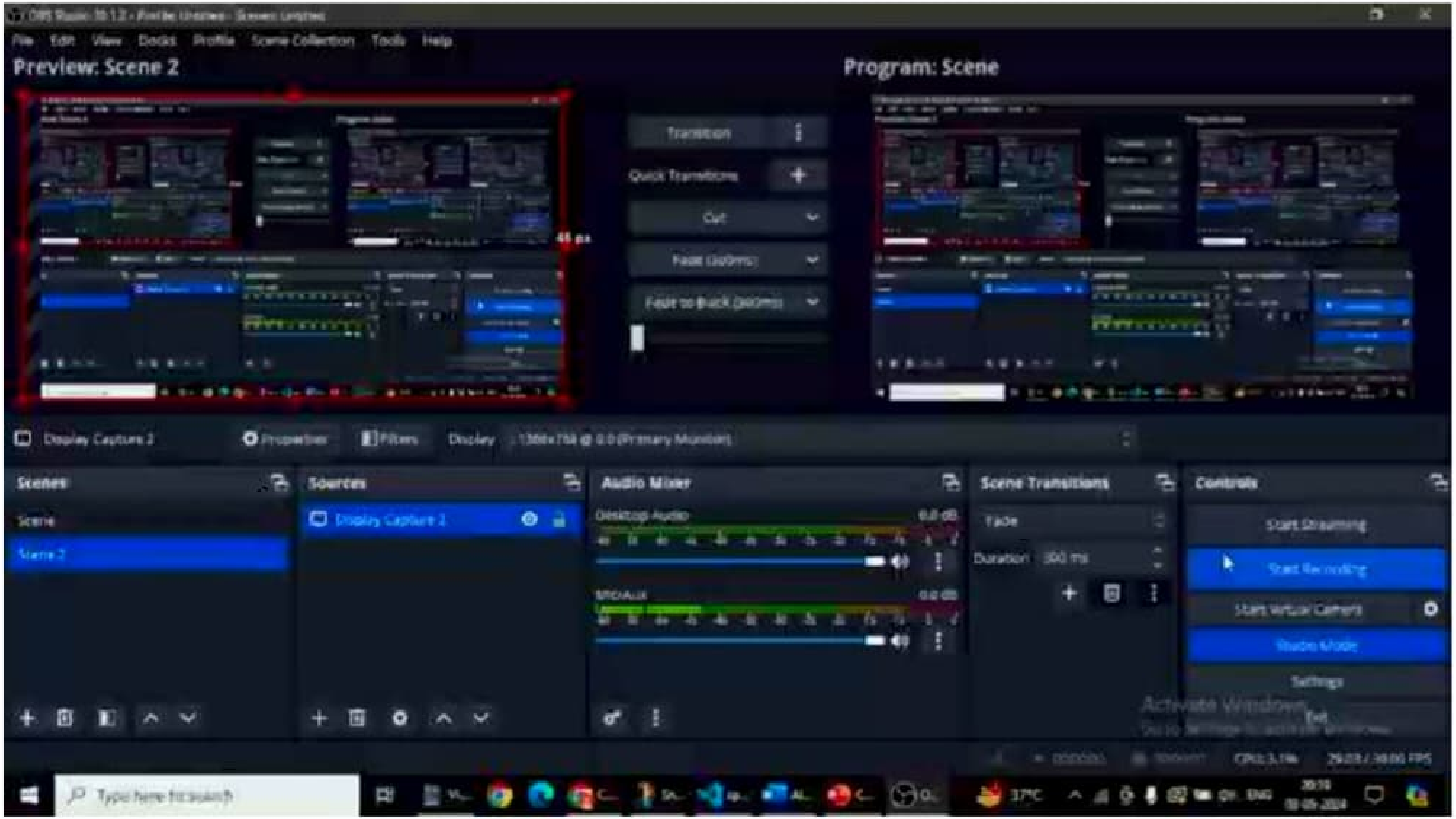
  Here as you would traverse from the root of Trie to leaf node it would make it's time complexity ***O(m)*** where 'm' would be word length which is to be searched.

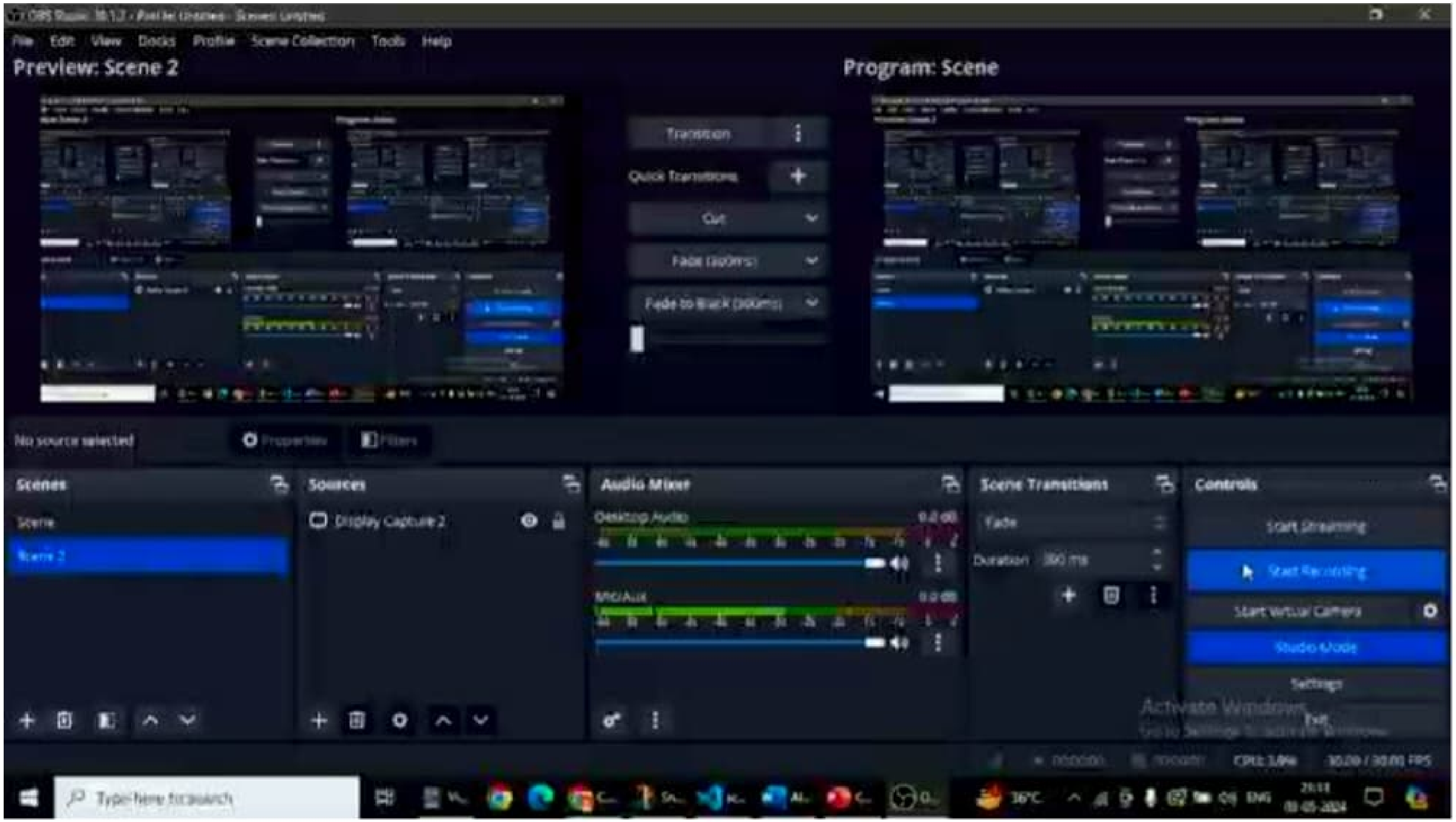
Space-Complexity

The Space complexity of constructing the Trie and loading a dictionary is ***O(n)***.

NOW LET'S HAVE A CODE WALKOVER


File Edit View Docks Profile Scene Collection Tools Help
Preview: Scene 2
Program: Scene
Transition
Quick Transitions
Cut
Fade (300ms)
Fade to Black (300ms)
Display Capture 2
Properties
Filters
Display
Scenes
Scene
Scene 2
Sources
Display Capture 2
Audio Mixer
Desktop Audio
Mic/Aux
Scene Transitions
Fade
Duration 300 ms
Controls
Start Streaming
Stop Recording
Start Virtual Camera
Studio Mode
Settings
Exit
Activate Windows
Type here to search


File Edit View Docks Profile Scene Collection Tools Help
Preview: Scene 2
Program: Scene
Transition
Quick Transitions
Cut
Fade (300ms)
Fade to Black (300ms)
Display Capture 2
Properties
Filters
Display
Scenes
Scene
Scene 2
Sources
Display Capture 2
Audio Mixer
Desktop Audio
0.0 dB
Mic/Aux
0.0 dB
Scene Transitions
Fade
Duration 300 ms
Controls
Start Streaming
Start Recording
Start Virtual Camera
Studio Mode
Settings
Exit
Activate Windows
Type here to search
37°C


File Edit View Docks Profile Scene Collection Tools Help
Preview: Scene 2
Program: Scene
Transition
Quick Transitions
Cut
Fade (300ms)
Fade to Black (300ms)
No source selected
Properties
Filters
Scenes
Scene
Scene 2
Sources
Display Capture 2
Audio Mixer
Desktop Audio
0.0 dB
Mic/Aux
0.0 dB
Scene Transitions
Fade
Duration 300 ms
Controls
Start Streaming
Start Recording
Start Virtual Camera
Studio Mode
Settings
Exit
Activate Windows
Go to Settings to activate Windows.
00:00:00
00:00:00
CPU: 1.8%
30.00 / 30.00 FPS
Type here to search
ENG
03-05-2024

THANK YOU