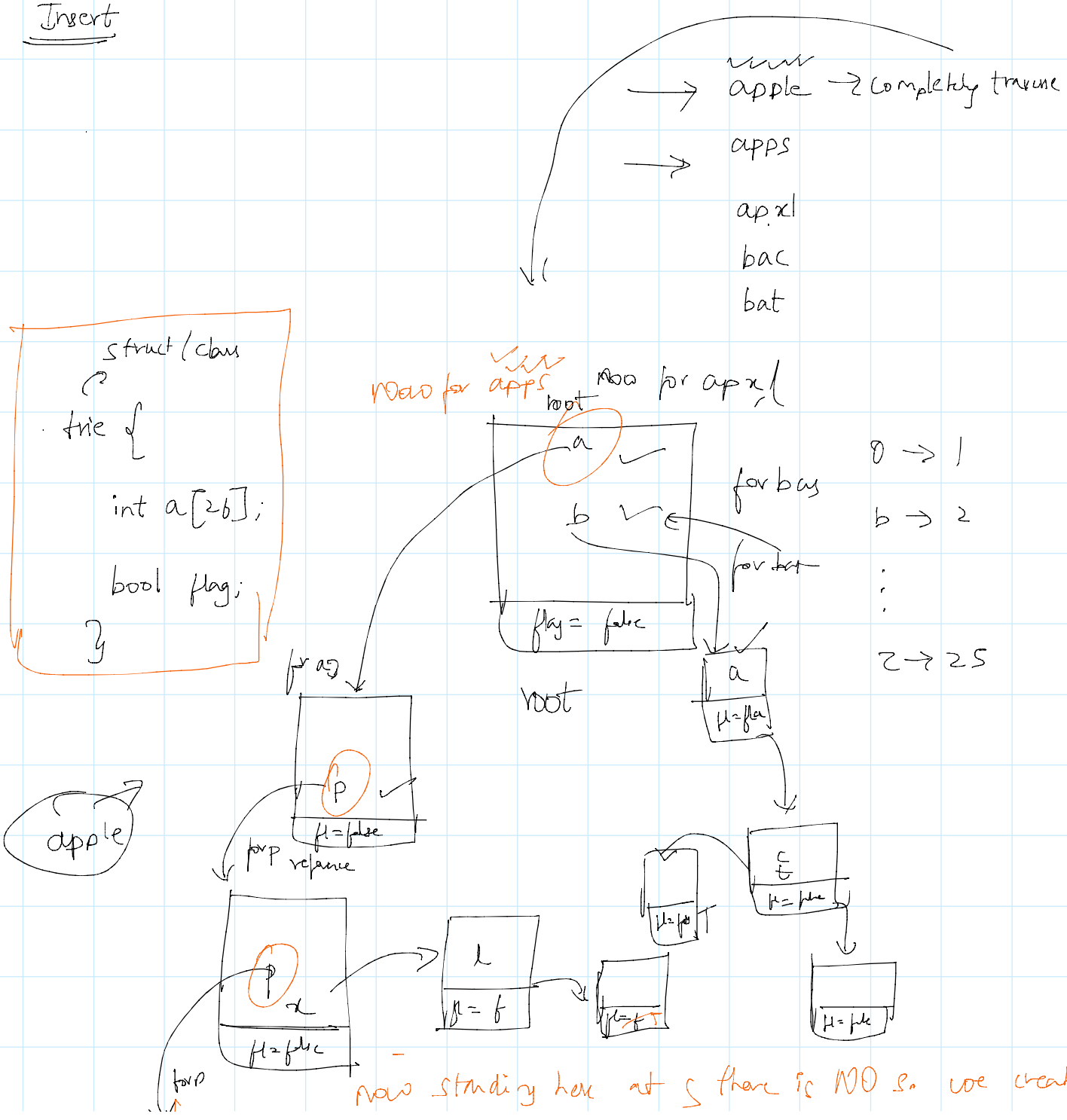
Implement Trie

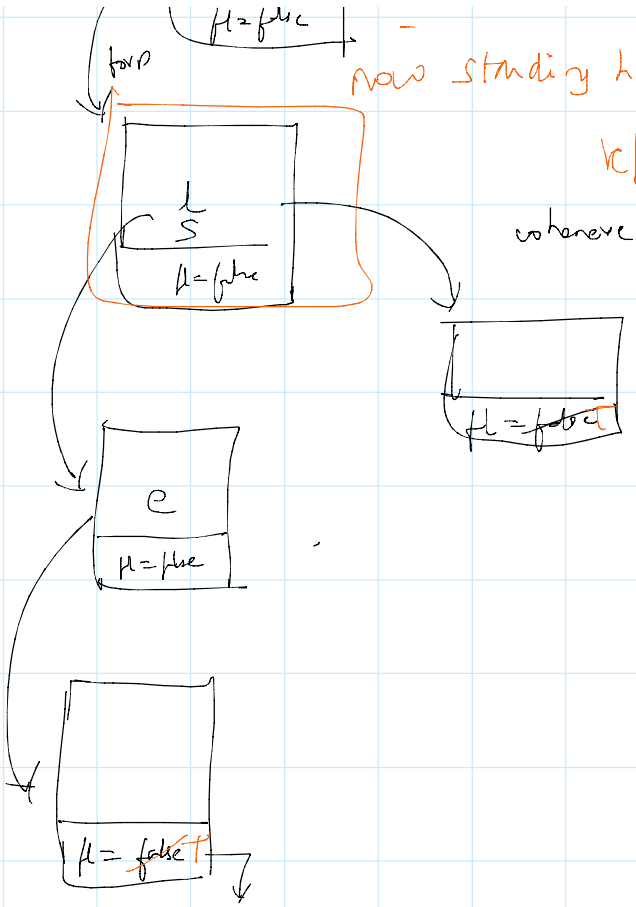
14 February 2022 11:26 AM

It's used whenever you have something like insert a word, search a word, word exist, start with word (prefix).

So we use Trie.

Insert





now standing here at S there is NO S. we create a reference -

wherever coord is complete mark the last word as true

This true significant that e is present.

Search

→ apps (start from root and keep searching and the last coord should be marked as True, if not true then its false)

→ appl (In this case the last word (l) is false so answer is false)

move start with

→ br ⇒ at the end if you are standing at not-null (true) position then true

→ apd ⇒ no d, it will point at null then false

```
class Node {
    Node links[] = new Node[26];
    boolean flag = false;

    public Node() {

    }

    boolean containsKey(char ch) {
        return (links[ch - 'a'] != null);
    }
    Node get(char ch) {
        return links[ch-'a'];
    }
    void put(char ch, Node node) {
        links[ch-'a'] = node;
    }
    void setEnd() {
        flag = true;
    }
    boolean isEnd() {
        return flag;
    }
};
```

```
public class Trie {
    private static Node root;

    //Initialize your data structure here

    Trie() {
        root = new Node();
    }


    //Inserts a word into the trie

    public static void insert(String word) {
        Node node = root;
        for(int i = 0;i<word.length();i++) {
            if(!node.containsKey(word.charAt(i))) {
                node.put(word.charAt(i), new Node());
            }
            node = node.get(word.charAt(i));
        }
        node.setEnd();
    }
```

```
    //Returns if the word is in the trie

    public static boolean search(String word) {
        Node node = root;
        for(int i = 0;i<word.length();i++) {
            if(!node.containsKey(word.charAt(i))) {
                return false;
            }
            node = node.get(word.charAt(i));
        }
        if(node.isEnd()) {
            return true;
        }
        return false;
    }


    //Returns if there is any word in the trie that starts with the given prefix

    public static boolean startsWith(String prefix) {
        Node node = root;
        for(int i = 0;i<prefix.length();i++) {
            if(!node.containsKey(prefix.charAt(i))) {
                return false;
            }
            node = node.get(prefix.charAt(i));
        }
        return true;
    }
}
```