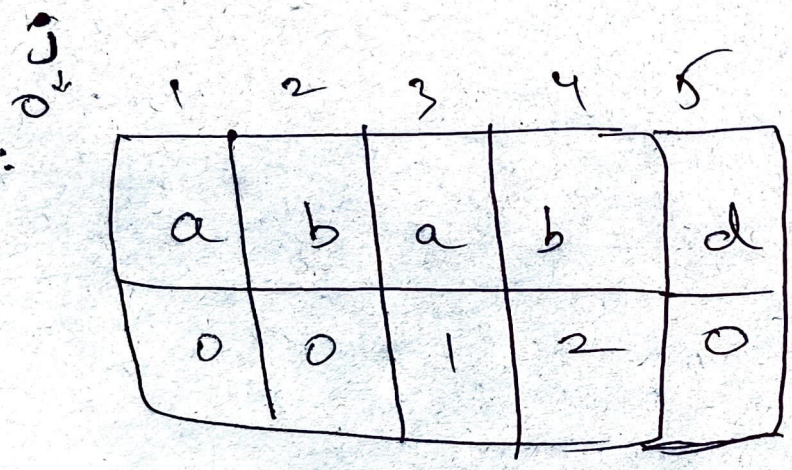
KMP Algorithm.

String: a b a b c a b c a b a b a b d.
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15



KMP Algorithm.

String: a b a b c a b c a b a b a b d

i: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

j	0	1	2	3	4	5
pattern:		a	b	a	b	d
index:	0	0	0	1	2	0

⇒ Compare i with j+1 if equal then move i an j

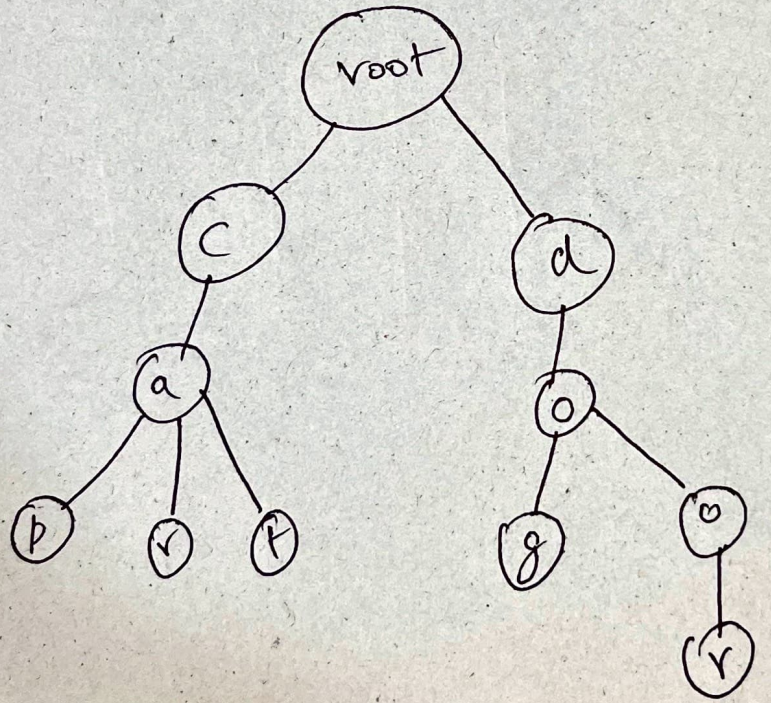
⇒ if there is a mismatch then move j to index.

⇒ if j is on 0 move i -> i+1

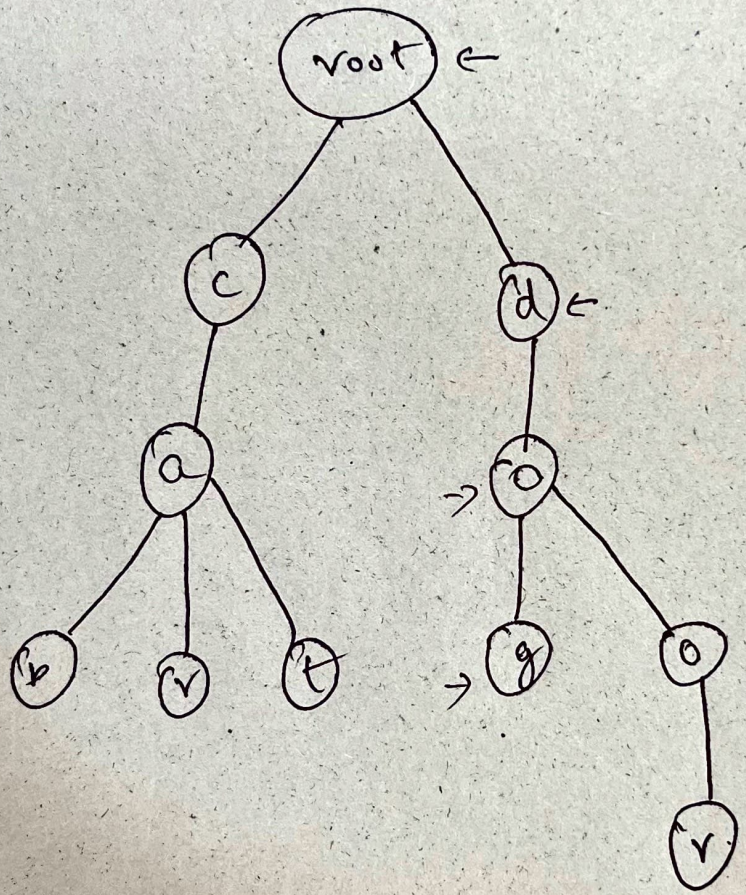
Tries

→ Tree is an ordered tree to maintain a set of strings.

Text: cat, cap, car, dog, door.



pattern: dog.



Time complexity:

O(length of pattern)

```
Struct trinode.
{
  char data;
  Trinode *child[26];
  int we
}
```

wordend;

Max of 26 children corresponding 26 alphabets for every node in the Trie.

cat

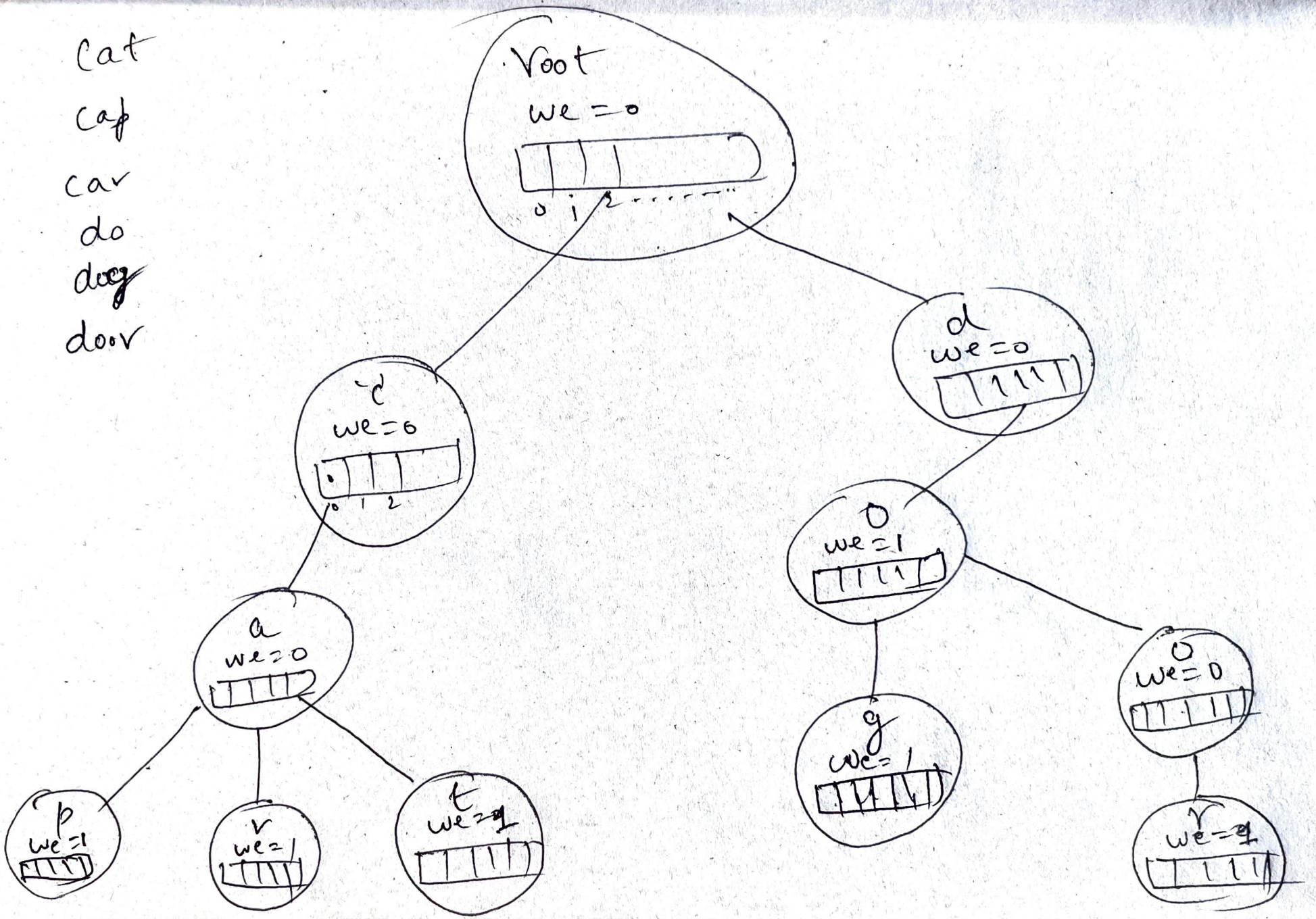
cap

car

do

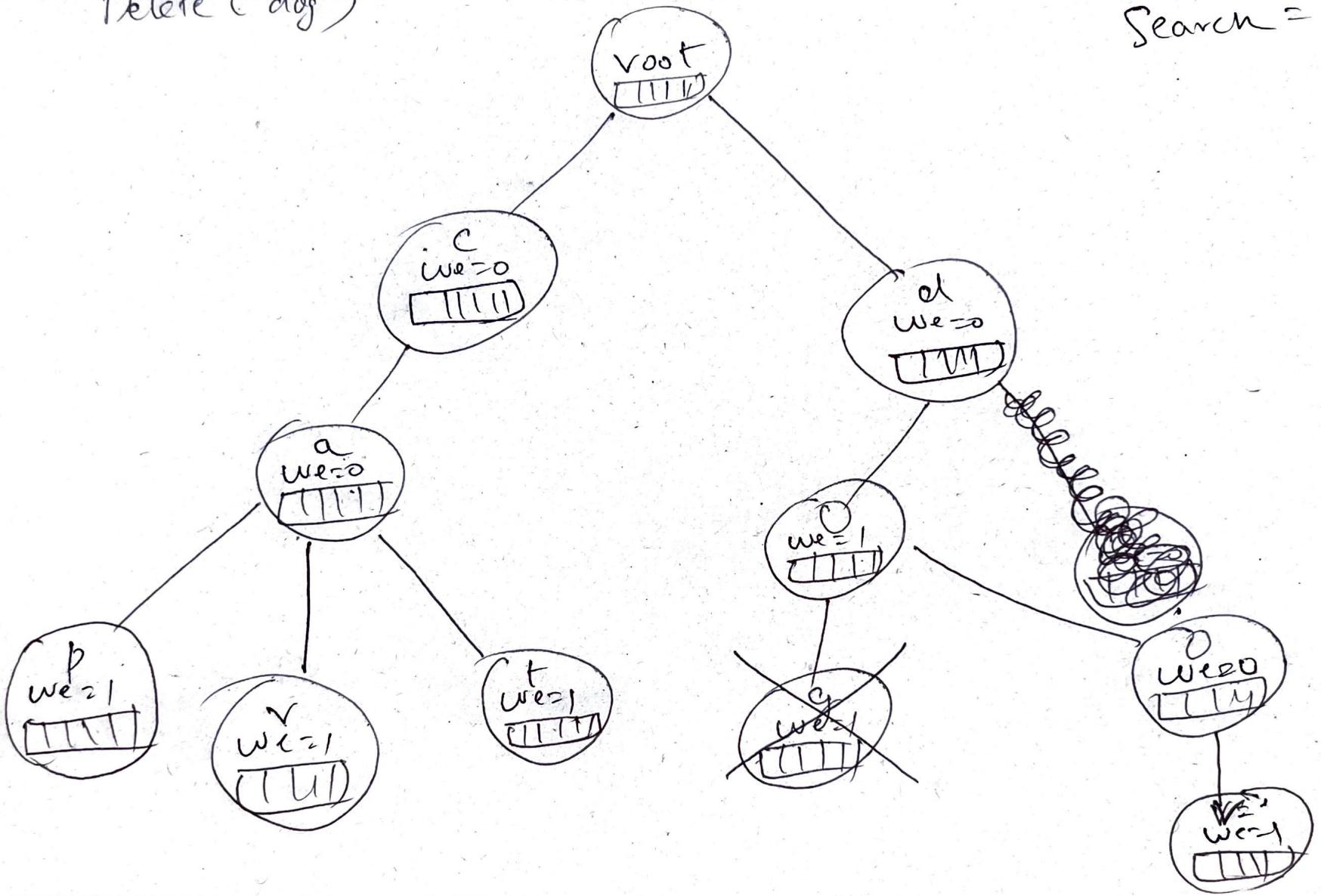
dog

door

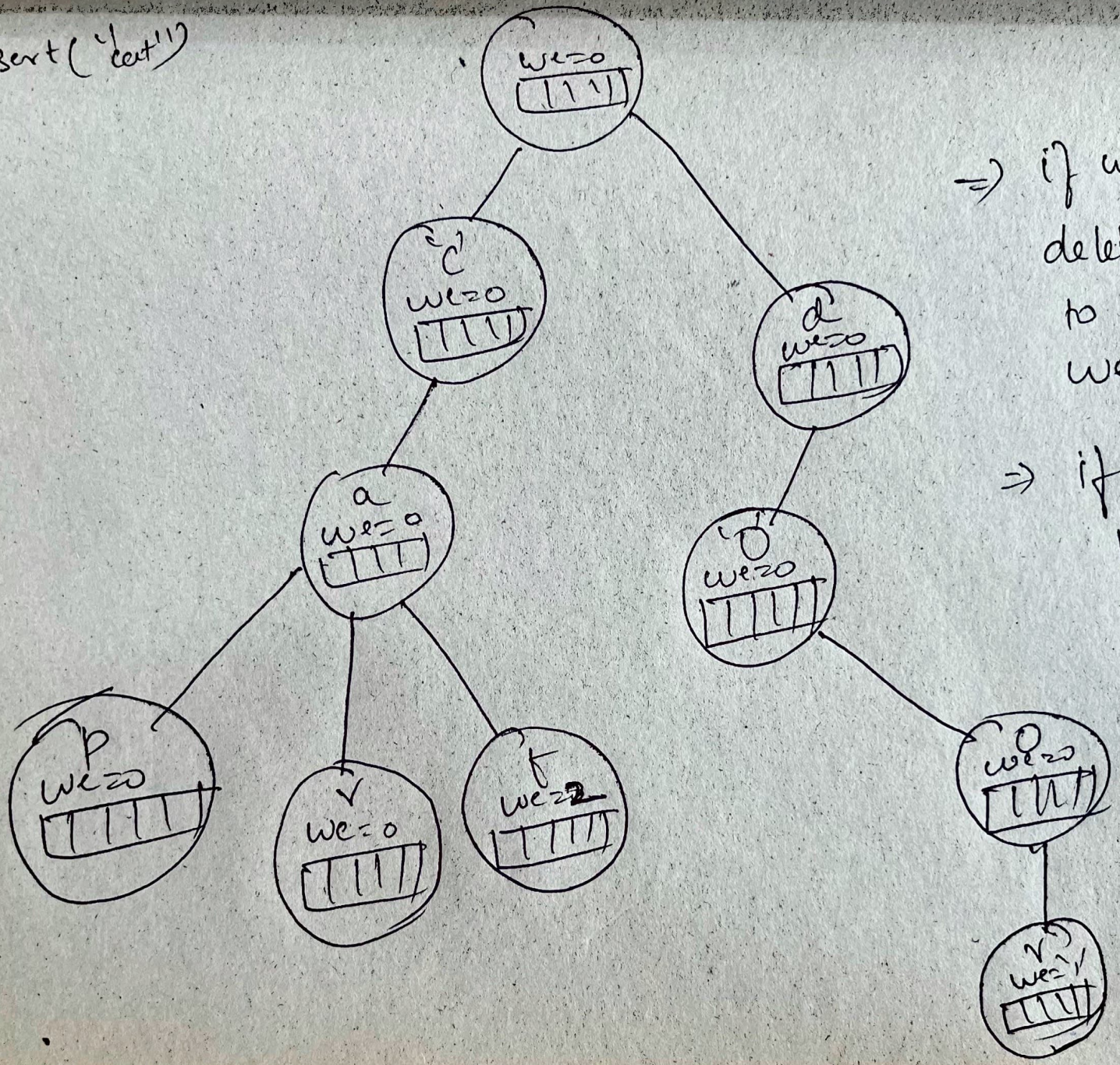


Delete ("dog")

Search = O(m)
↓
length of pattern.



Insert("cat")

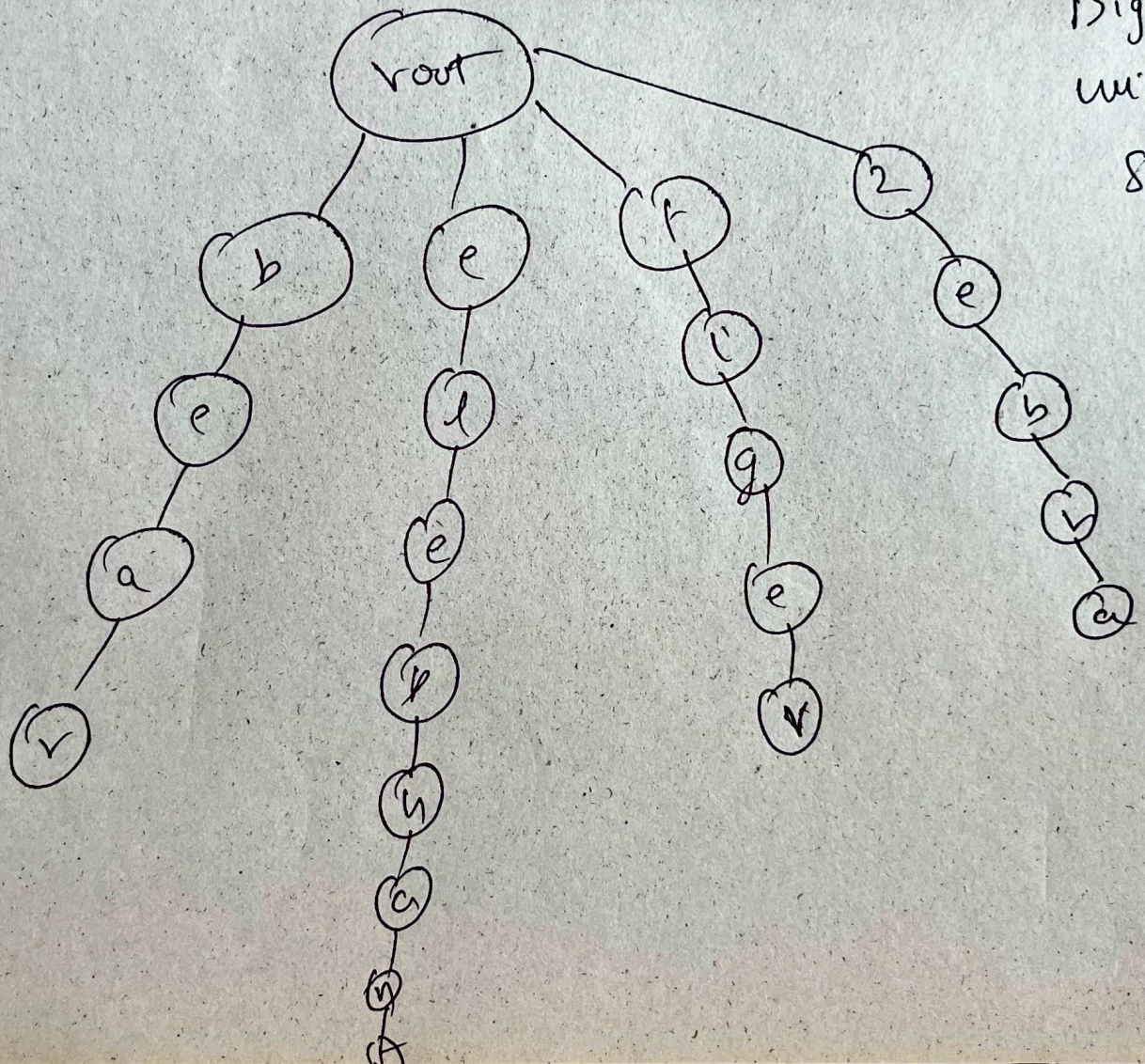


⇒ if we want to delete "dog" we have to decrement the we by 1

⇒ if the node is last then we can delete it.

T: O(m)

Text: horse, elephant, tiger, zebra, bear



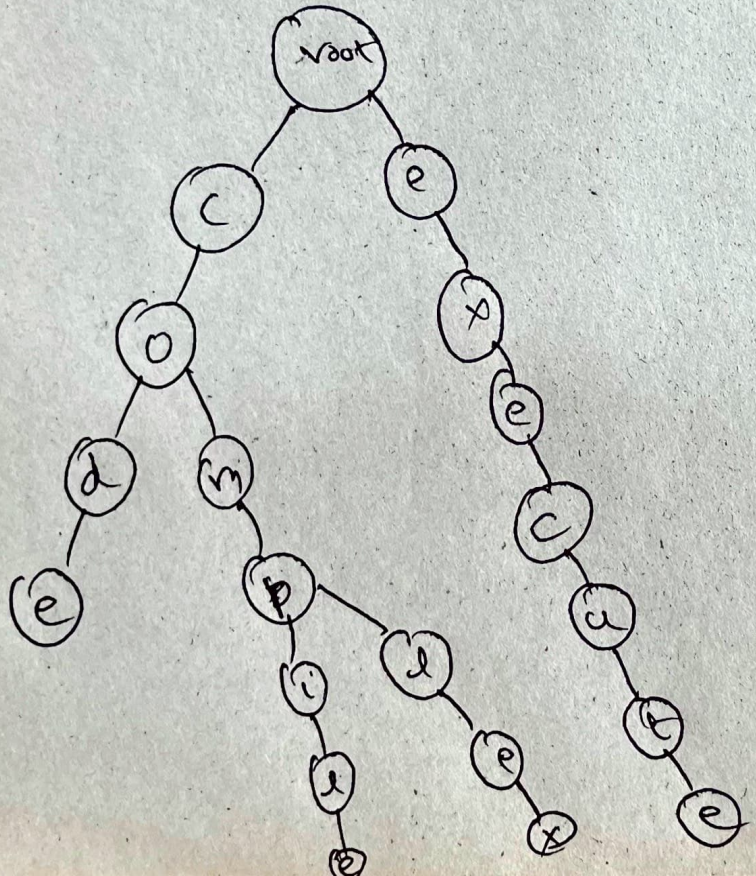
Biggest drawback with tries is space complexity

$O(n)$

worst case when no overlapping

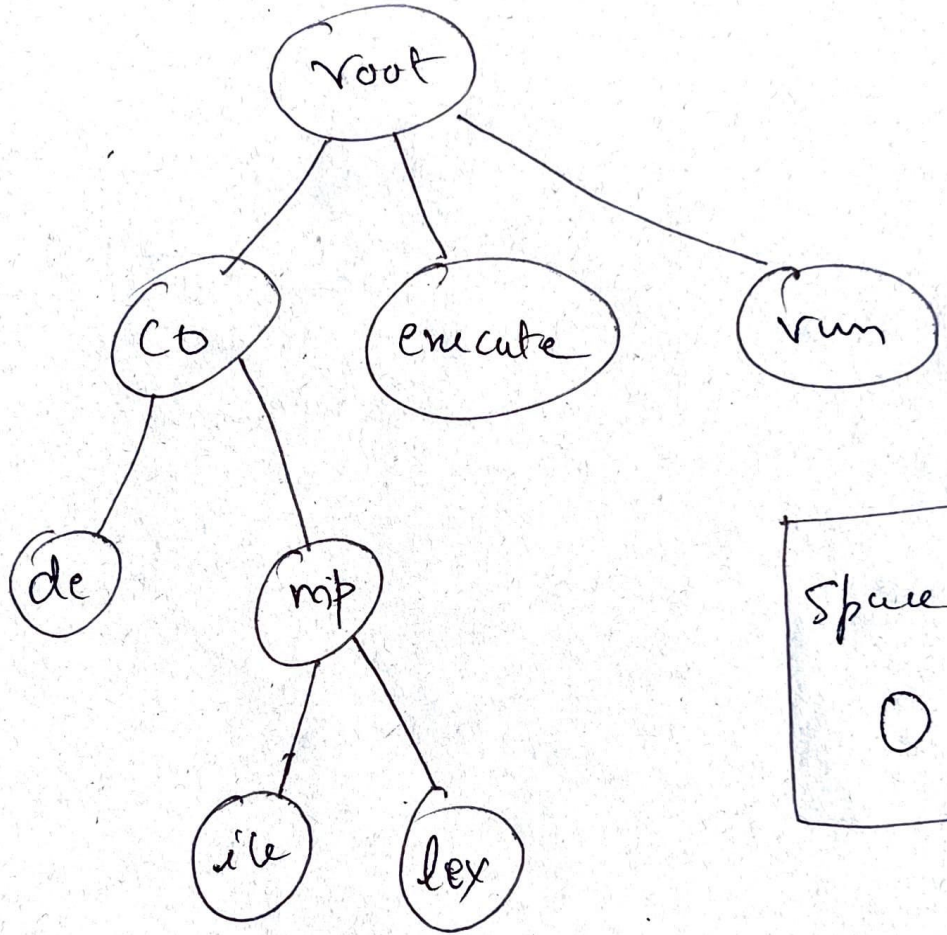
Compressed Tries

"Compressed is also a standard trie where each and every node has atleast two child"



Text: code
: compile
complex
execute
run.

Compressed Trie from Standard Trie.



Space Complexity

O (no of words)