Todays Content:

- Trie data structure
- Check if given word exists

(Q) Given N strings & Q queries, for each query check if given query is in given N strings

Note: All characters are 'a' - 'z' & len(String) <= l (smaller)

Words:	Queries
damp	data
dark	draw
data	drew
drake	dump
drawn	drawed
drew	
dried	
drunk	
draw	
tree	
tried	
trump	
tea	

Idea1: For every query iterate & compare with all N input strings

TC: Q * {N * {l}} SC: O(1)

Note: to compare 2 strings of len: l : O(l)

Idea2: → Insert all Input words in hashset
→ For every query search in hashset

TC: N*O(L) + Q*O(L) SC: N*O(L)

Note1: To insert/search/delete a string
↳ of len l in hashset/map TC: O(l)
↳ : To store a single strings of len l in hashset/map SC: O(l)

Idea3: → Insert all strings in Trie
→ Search all strings in Trie

TC: N*h + Q*h SC: less than N*h ?

Trie: → It's also a type of tree
↳ We insert char by char

google docs: cricket * word is wrong
↳ All correct words are stored: (Tries)

Trie: →

```
Class Node{
    Node c[26]
    bool isEnd
    Node() {
        isEnd = false
        for(i = 0; i < 26; i++) {
            c[i] = null
        }
    }
}
```

a b c ... z

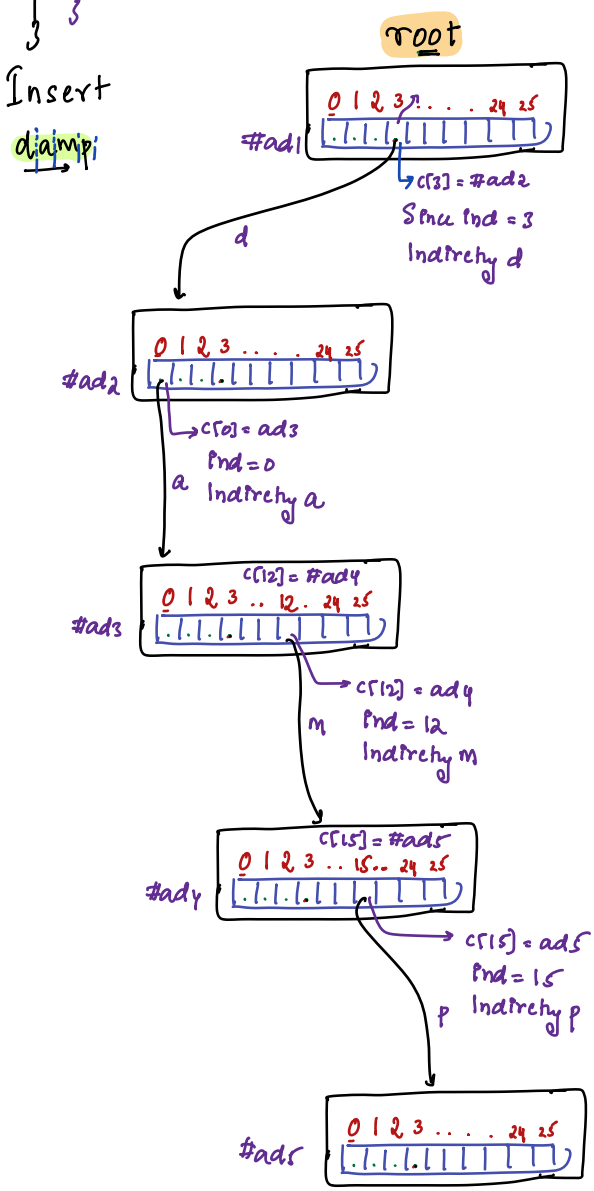
c[0] c[1] c[2] ..... c[25]

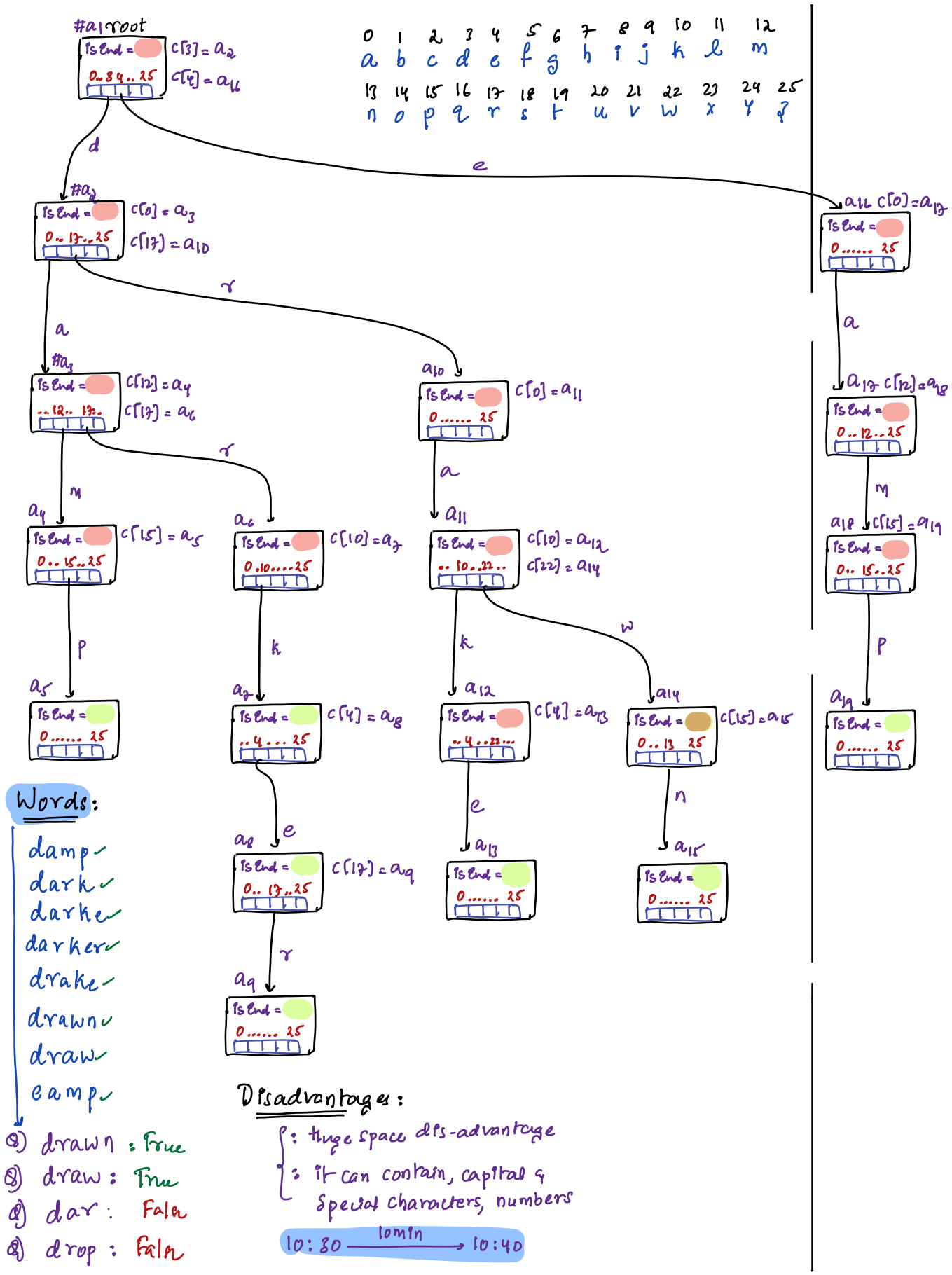
All of them are object reference

// Tells if a word ends at that object

Insert

diamipi





Words:

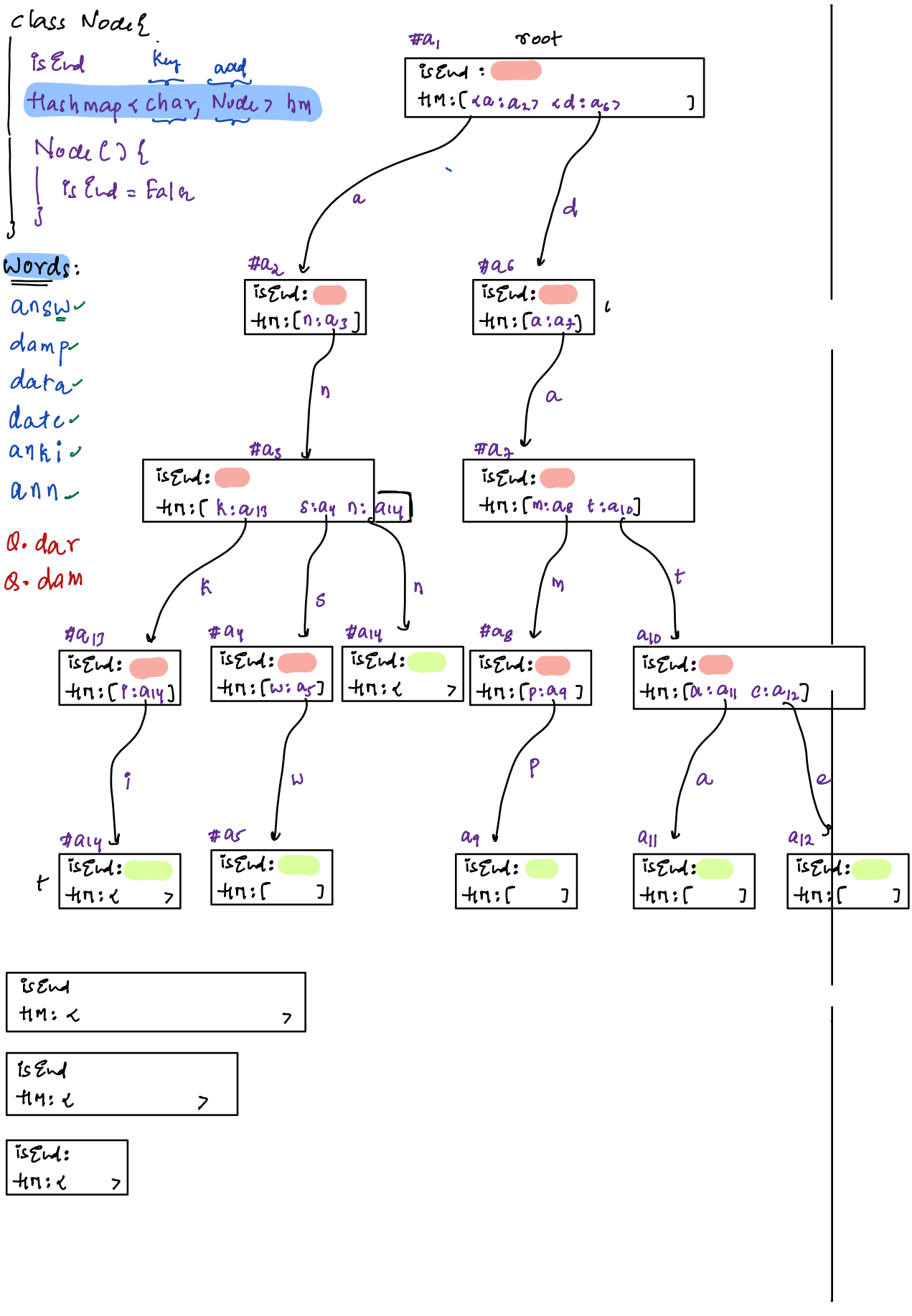
damp ✓
dark ✓
darke ✓
darker ✓
drake ✓
drawn ✓
draw ✓
camp ✓

Q) drawn : True
Q) draw : True
Q) dar : False
Q) drop : False

Disadvantages:

- : huge space dis-advantage
- : it can contain, capital & special characters, numbers

10:30 —10min→ 10:40


class Node{
isEnd
key
addr
Hashmap<char, Node> hm
Node(){
isEnd = False
}
Words:
answ
damp
data
date
anki
ann
Q. dar
Q. dam
#a1
root
isEnd :
HM:[<a:a2> <d:a6>]
a
d
#a2
isEnd:
HM:[n:a3]
#a6
isEnd:
HM:[a:a7]
n
a
#a3
isEnd:
HM:[k:a13 s:a4 n:a14]
#a7
isEnd:
HM:[m:a8 t:a10]
k
s
n
m
t
#a13
isEnd:
HM:[i:a14]
#a4
isEnd:
HM:[w:a5]
#a14
isEnd:
HM:<>
#a8
isEnd:
HM:[p:a9]
a10
isEnd:
HM:[a:a11 e:a12]
i
w
p
a
e
#a14
t
isEnd:
HM:<>
#a5
isEnd:
HM:[]
a9
isEnd:
HM:[]
a11
isEnd:
HM:[]
a12
isEnd:
HM:[]
isEnd
HM:<>
isEnd
HM:<>
isEnd:
HM:<>

Class Structure:

```
Class Node {
    bool isEnd
    hashmap<char, node> hm
    Node() {
        isEnd = false
        hm = new hashmap<char, node>();
    }
}

main() {
    Node root = new Node()
    int N
    Read N
    i = 1; i <= N; i++) {
        String word;
        Read word;
        insert(root, word)
    }
    int Q
    Read Q
    while(Q--) {
        String word;
        Read word;
        if(search(root, word)){
            print("present")
        }
        else{
            print("not present")
        }
    }
}
```

```
void insert(Node root, string data) {
    Node t = root
    int l = data.size()
    i=0; i<l; i++) { TC: l * O(1)
        // char we want to insert data[i]
        char ch = data[i]
        // search ch in hashmap of t node = t.hm
        if (t.hm.search(ch) == True) {      hashmap
            t = t.hm[ch] // get add update t
        }
        else {
            Node nn = new Node()
            t.hm.insert(ch, nn)
            t = nn
        }
    }
    t.isEnd = True
}
```

```
bool search(Node root, string data) {
    Node t = root
    int l = data.size()
    i=0; i<l; i++) { TC: l * O(1)
        // char we want to search data[i]
        char ch = data[i]
        // search ch in hashmap of t node = t.hm
        if (t.hm.search(ch) == True) {
            t = t.hm[ch] // get add update t
        }
        else {
            return false
        }
    }
    return t.isEnd;
}
```

Q) Given N Input Strings & Q queries, for each query check if given query is prefix of any input strings

// Substring starting at index = 0

Constraints : $1 \leq$ len(string) $\leq |l|$ (next class)

Input Strings (N)	Queries Q
anaconda	anaco
dress	fry
eaten	roade
friends	algor
roades	sour
anaco	dress
algorithms	
sound	