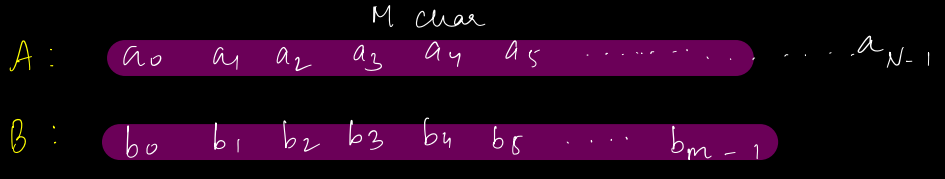
Prefix String : Substrings starting with index 0.

A : anaconda B : ana



TC : O(len(B))

Q) Given N words & Q queries. $'a' <= w[i] <= 'z'$: 26

Calculate no of Prefix Strings of given words = given query

$1 <= len(words) <= M$

$1 <= len(query) <= M$

N = 20

data	store	Q = 7
draw	string	st : 6
drew	list	li : 3
dark	linked	dr : 5
algorithm	link	da : 3
stack	stamp	dat : 1
structure	sound	dark : 1
struct	drunk	do : 0
drake	dried	
damp	almond	

Idea1:

For every query, iterate on all N strings, check if query is prefix or not

TC: $Q * [N * M]$ (N words)

Q: Queries

To check whether query is prefix of word

Idea2:

KMP | Rabin Karp *

Idea3: Sort all words in lexicographical order

dictionary.

0 algorithm
1 almond
2 damp → P_1
3 dank
4 data
5 drake → P_1
6 draw
7 drew
8 dried
9 drunk → P_2
10 link
11 linked
12 list
13 sound → ~~P_1~~
14 stack → ~~P_1~~
15 stamp
16 store
17 string → P_1
18 struct
19 structure → P_2

$Cut = P_2 - P_1 + 1$

$P_2 - P_1 + 1 : 3$

Queries: → No of words have prefix dr

dr : 5

str ✓ : 3

car : 0

~~sts~~ : 0

$[\frac{M}{\log N}]$

Idea: Sort N words of M length

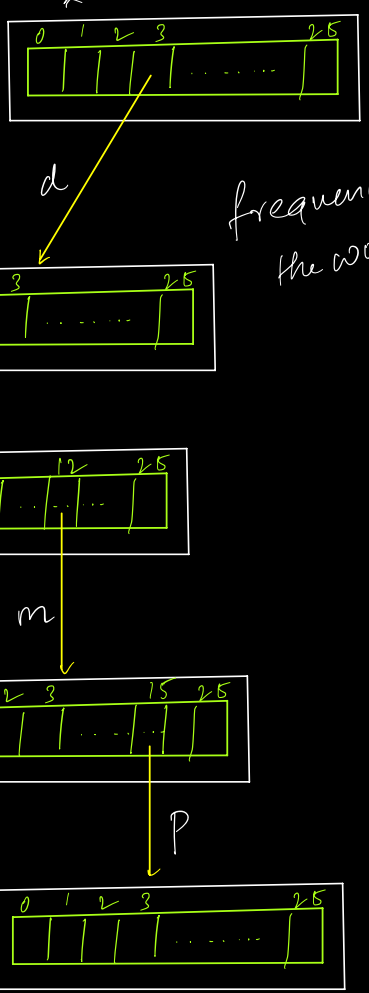
for every query apply Binary Search

$$TC: \underbrace{(N \log N)}_{N \text{ words}} * \underbrace{M}_{\text{Comparing 2 strings in merge fun}} + Q * M * \log_2 N$$

```
p1 = -1
for(int x = 0; x < M; x++) {
    i = 0, j = N-1
    while( i < j) {
        m = (i+j)/2
        if(s[m][x] == q[x]) {
            j = m-1
            p1 = m
        }
        else if(s[m][x] > q[x]) {
            j = m-1
        }
        else {
            i = m+1
        }
    }
}
```

Idea :

N-arrcay Tree :- → N childrens : 26 children



damp

dark

data

drake

draw

drew

dried

drunk

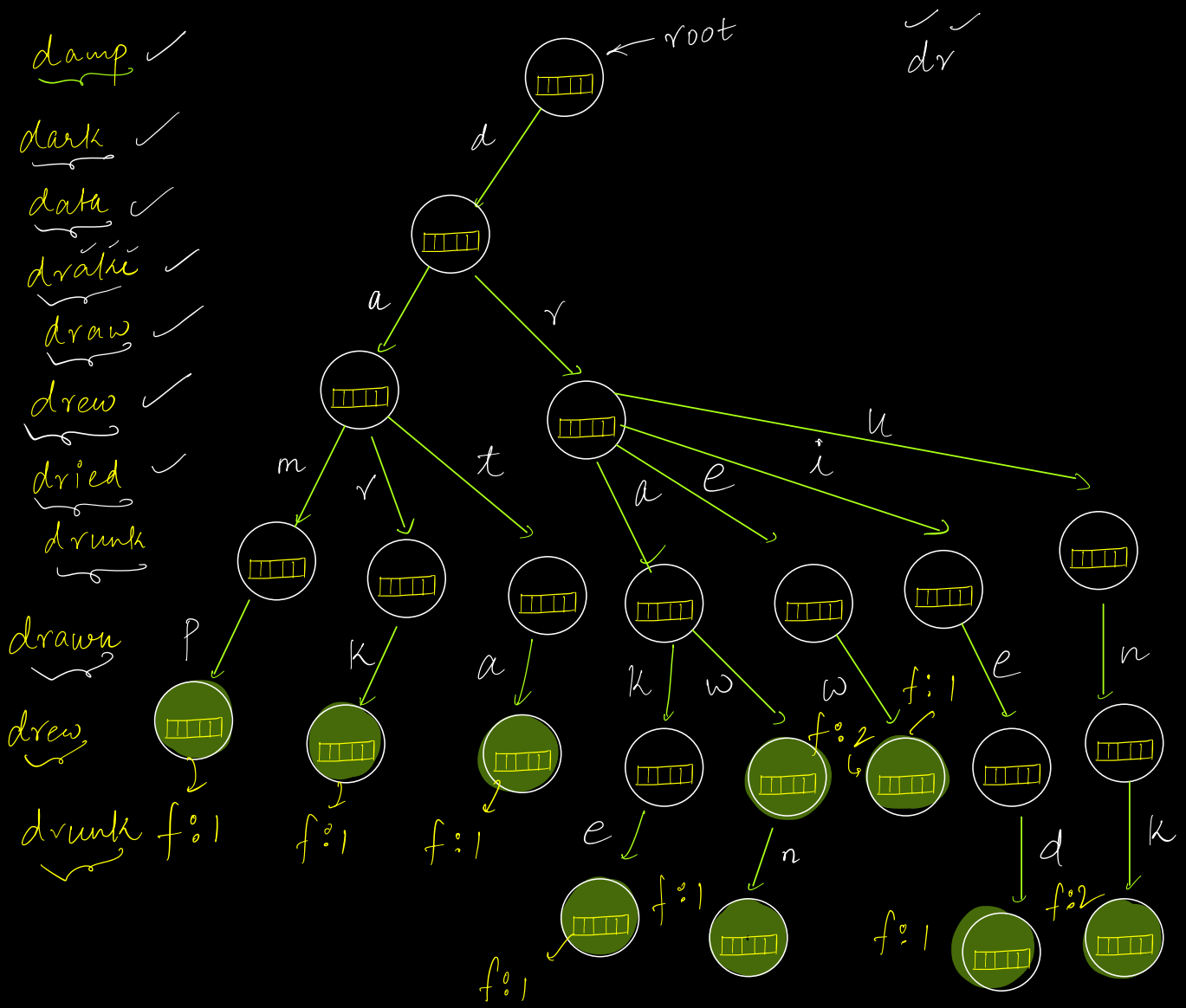
```
Class Node {
    Node c[26]
    // we initially all
    Children as NULL
    bool isEnd;
    int count;
}
```

frequency of the word

indicating whether node is ending there or not

0 — 'a'
1 — 'b'
2 — 'c'
3 — 'd'
4
.
.
.
25 — 'z'

N-ary Tree ⇒ Trie



Q) dr

How many words have prefix dr *

↳ Traverse from root Node to dr

- Calculate no of Leaf Nodes present in that Subtree?

Reason: Non leaf Nodes can also be end of a word

isEnd in Node Structure : While inserting nodes, make last node of word = T

Traverse from root node till dr : *

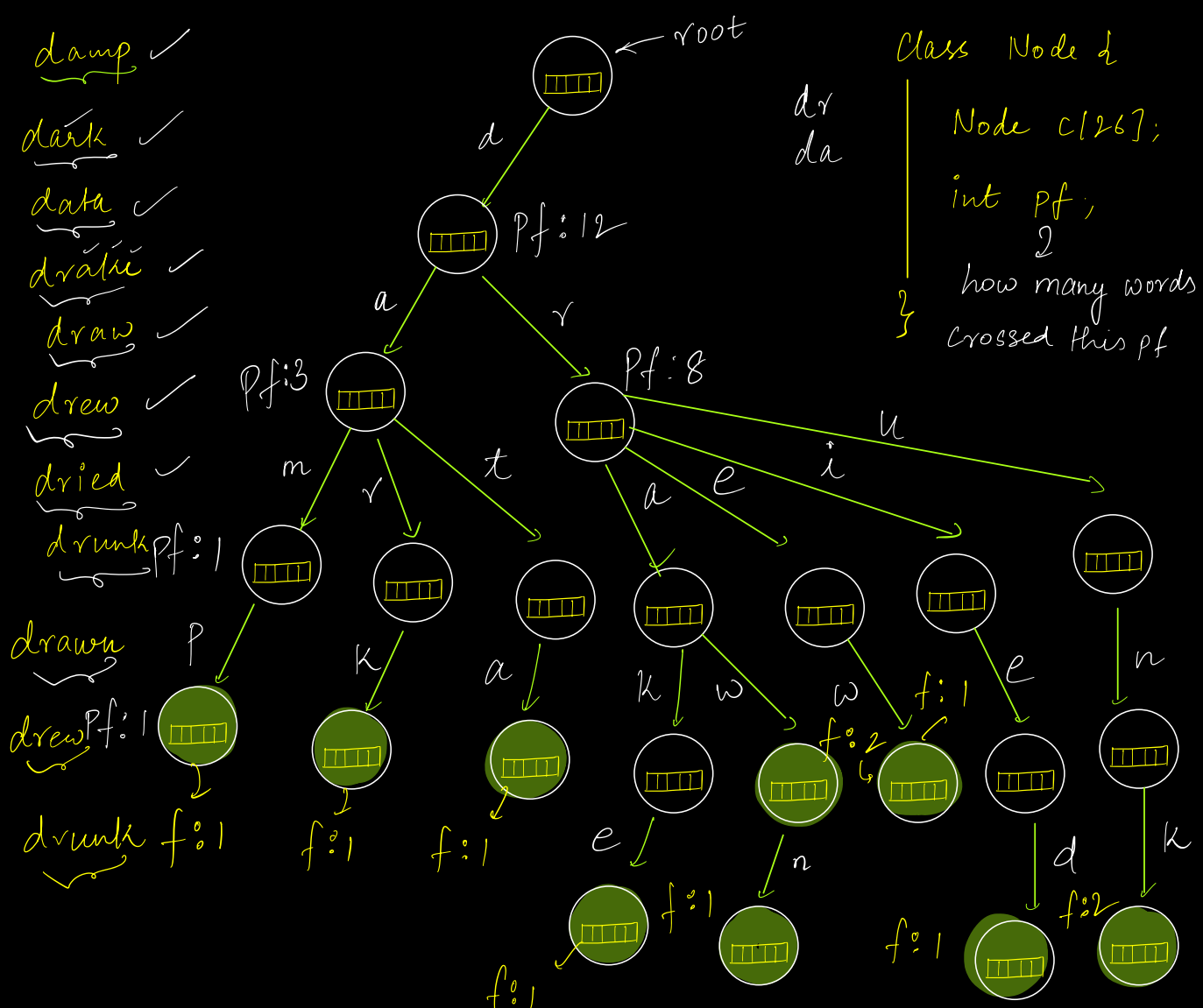
→ Calculate no of nodes in subtree for which isEnd = T.

Reason: Due to duplicate words

Count in Node Structure : ✓

Traverse from root Node to dr.

↳ Calculate freq of all Nodes.



dr

Pf: How many words are passing through current Node

Traverse from root to dr

→ get pf of current Node

```
class Node {
    Hashmap<char, Node> // Node C[26];
    int pf
}
```

O(N*M)

```
void insert(Node root, &string w){
    Node curr = root;
    for(int i=0; i<w.length(); i++){
        char ch = w[i];
        if (ch is not in curr.hm){
            Node t = new Node()
            curr.hm.insert(ch, t)   (1)
        }
        curr.c[w[i]-'a'] = t        (11)
        curr = curr.hm[ch];         (2)
        curr.pf++
    }
}
```

curr = curr.c[w[i]-'a'] (22)

Rough Pseudocode

```
Read N
loop N
    Read word
    insert(root, word)
Read Q
loop Q
    Read word
    Print(query(root, word))
```

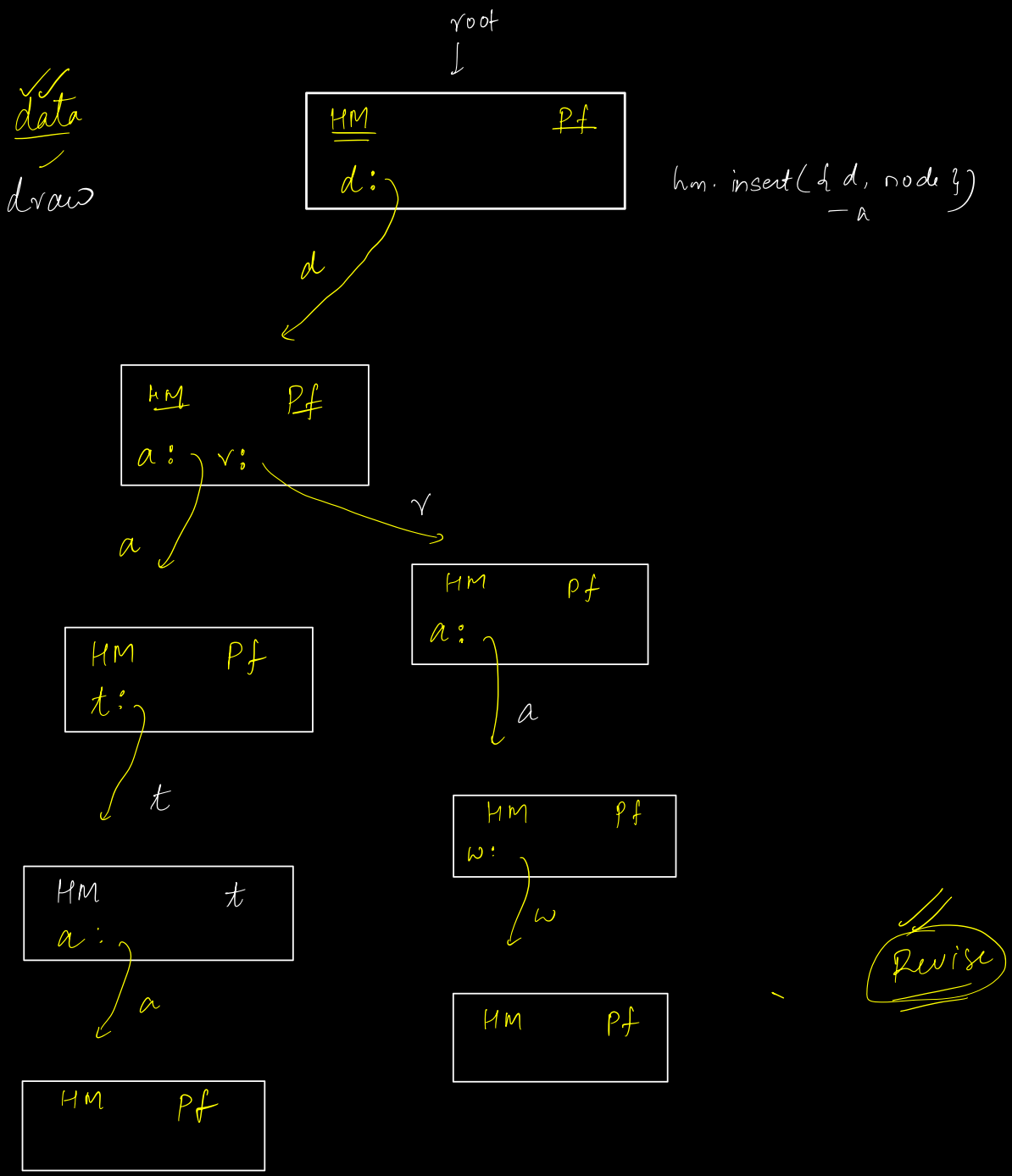
Q * M

TC: O(M)

N*M + Q*M

=

```
int query(@Node roof, &string w){
    Node curr = roof;
    for(i=0; i<w.length; i++){
        char ch = w[i]
        if(ch is not in curr.hm)
            return 0
        curr = curr.hm[ch]
    }
    return curr.pf
}
```


root
data
draw
HM
Pf
d:
hm.insert(d, node)
d
a:
v:
a
v
t:
t
w:
w
Revise