Trie:

apple
apps
apxl
bac
bat

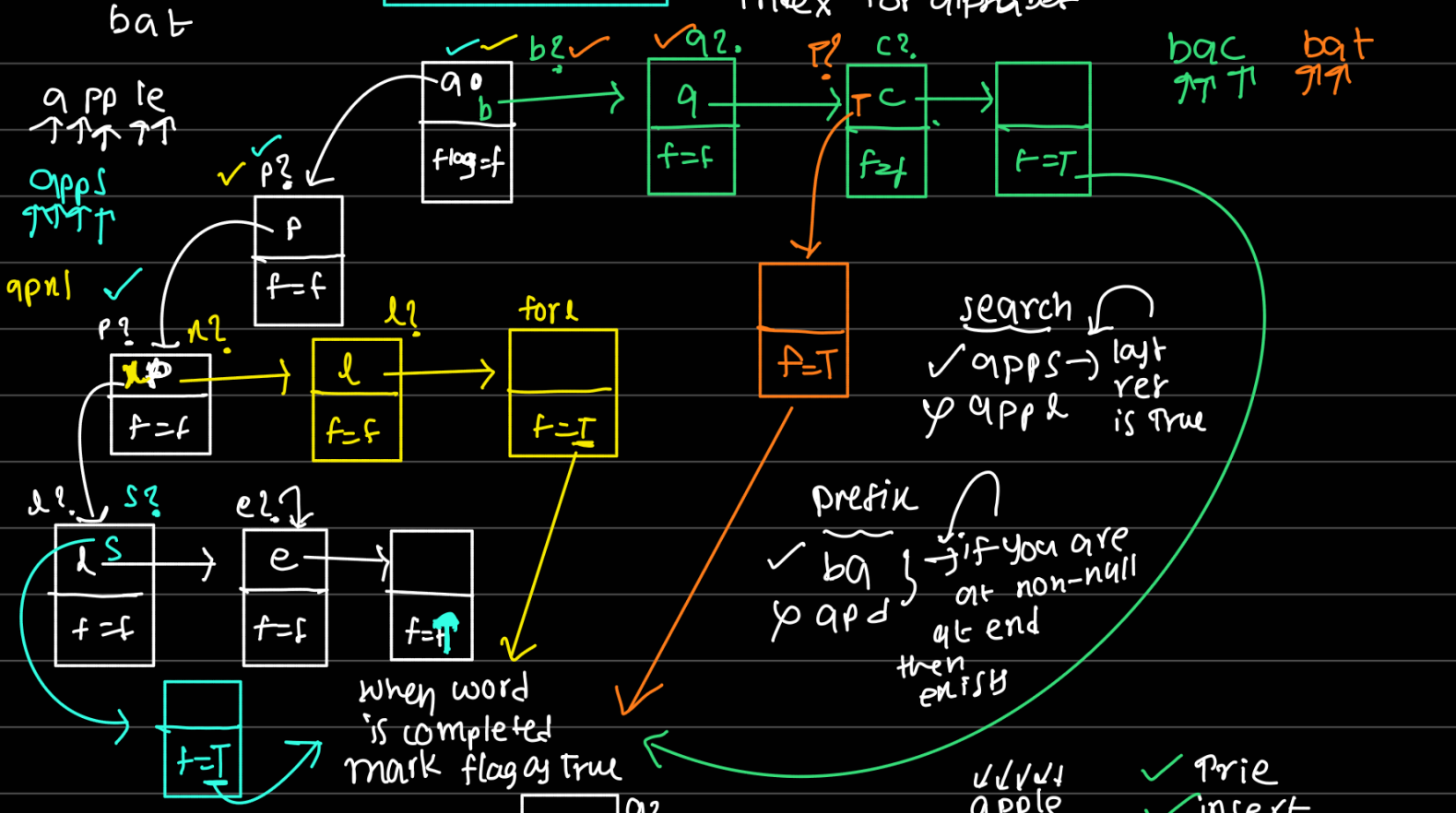
Each Node

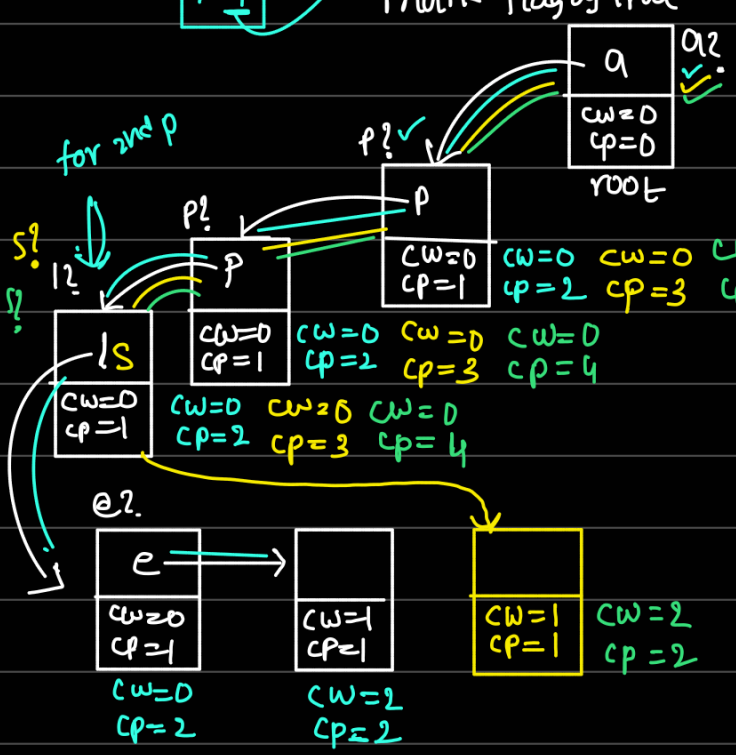
```
trie{
  trie* a[26];
  bool flag;
}
```

store at corresponding index for alphabet

Insert → search if its there in curr node no then create a new trie node store the reference in curr node

Suffix exist





apple
apple
apps
apps

✓ Trie
✓ insert

count words equal to ("word")

Count words starting with

count words equal to word
apps cw
abc

count words starting with
ap
appl cp

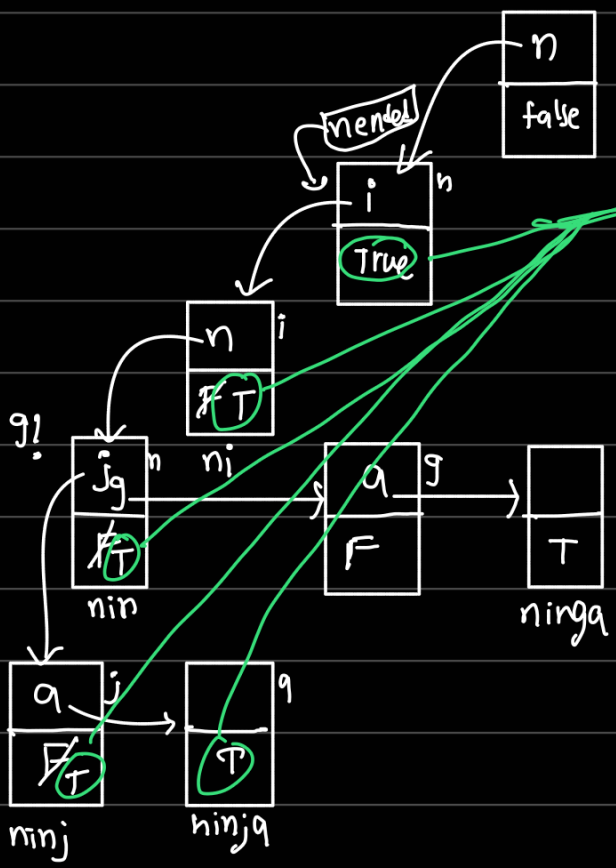
```
trie{
  links[26];
  cw=0
  cp=0
}
```

count how many words ends here

count prefix how many times it appeared

13 complete string is when all its substring exists as words
like for ninja => n, ni, nin, ninj, ninja
→ n, ninja, ninj, ni, nin, ninga => insert all these
return largest prefix

```
Trie{
  a[26];
  bool flag;
}
```



→ This is what we have to check for every word
all substrings flag=T
or not

14. Count Distinct substrings using trie.

Brute force

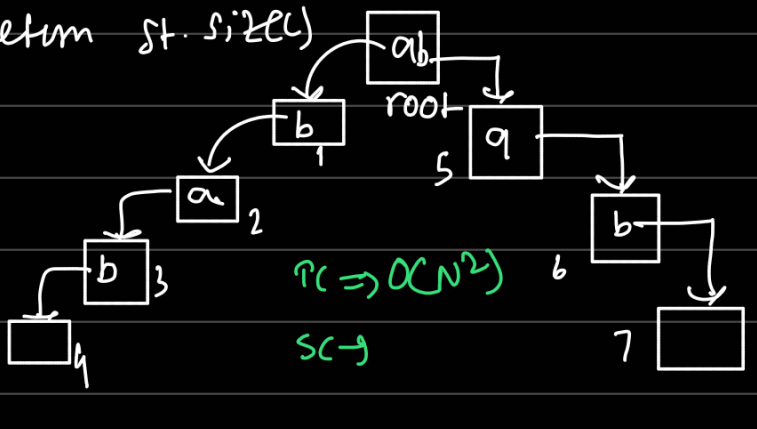
```
set<string> allss;
for(i=0 → i=n-1) → str="";
    for(j=i to n){
        str = str + a[j];
        allss.push(str);
    }
return st.size()
```

TC
$O(N^2 \log N)$
SC
N^2

add → $O(\log N)$

abab

a
ab
aba
abab
b
ba
bab
a b
ab



TC => $O(N^2)$
SC =>

abab bab ab b

```
trie{
  arr[26];
}
```

(L5) Bit prequisites for tries

9 → 000 . . . . . 1001 (bits 31 30 . . . 3 2 1 0) — 32/64 bits

XOR

1^0, 0^1 } 1 ⇒ odd no. of 1's ⇒ 1

1^1, 0^0 } 0 even no. of 1's ⇒ 0

→ check if a bit is set or not — 3rd bit ⇒ 1 ?

→ Turn on 2nd bit from right — num >> 3 & 1 ⇒ 1 ; 0 → 0

9 ⇒ 000 . . . 1001

(num & (1<<2))

or (of course)

num | 100

num | 2^2 = num | 4 ⇒ answer (or)

L6 Maximum XOR. Given an array of numbers & x

⇒ Insert all numbers into trie (binary bits)

⇒ Take x & find max no. from array where (no ^ x) ↑↑

6 8 → p, 2 8 2, ^ of pair, max

x = 8 [9 8 7 5 4]

x ^ a[i] = ↑↑

8 ^ a[i] = ↑↑

9 → 01001

8 → 01000

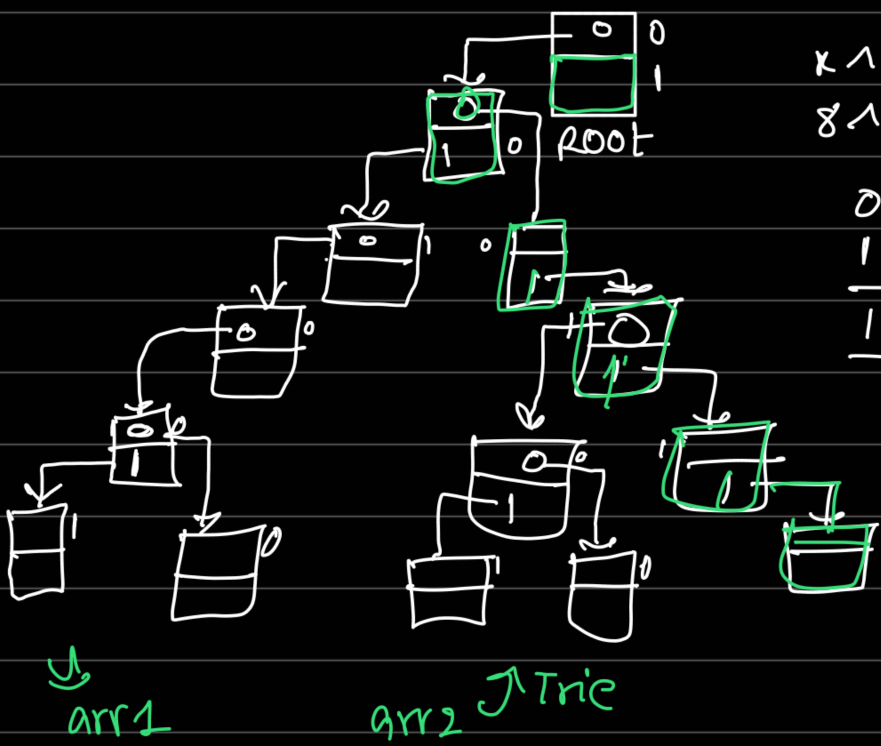
  0 1000
  1 0111

  11111 ⇒ 31

7 → 00111

5 → 00101

4 → 00100



8 x → 0 1000
0 0111
------
0 1111 ⇒ 15

arr1[0] ^ (Trie)
1
2
⋮
[n-1] ^ Trie
} max ⇒ ✓

extension: problem

L7 Maximum XOR Queries

Given an array/list ARR ⇒ N-non-neg integers

list QUERIES → M queries

ith query is a list/array of 2 non-neg integers 'xi' 'Ai'

i.e 'QUERIES[i]' = ['xi', 'Ai']

↳ answer = maximum bitwise xor of 'xi' & any integer from ARR ≤ 'Ai'

arr[i] → [ ]

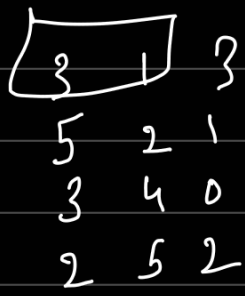
arr[i] ↑ x

(xi, ai) ⇒ <=Ai

```
for(i=0 → q-1){
    xi = queries[i][0]
    ai = queries[i][1]
    maxxor = -1
    for (j=0; j<n; j++){
        if (arr[j] <= a[i]
            maxxor = max(maxxor, arr[j]^xi)
    }
}
```

trie ← Arr & arr ^ trie ≤ ai

(1 2 3 4 5)
(1 3 2 5 4)



3 9 0
5 2 1
2 5 2
3 1 3

```cpp
vector<int> maxXorQueries(vector<int> &arr, vector<vector<int>> &queries){
    sort(arr.begin(), arr.end());
    vector<pair<int, pair<int,int>>> oQ;
    int q = queries.size();
    for(int i = 0;i<q;i++) {
        oQ.push_back({queries[1], {queries[0], i}});
    }
    sort(oQ.begin(), oQ.end());
    vector<int> ans(q, 0);
    int ind = 0;
    int n = arr.size();
    Trie trie;
    for(int i = 0;i<q;i++) {
        int ai = oQ[i].first;
        int xi = oQ[i].second.first;
        int qInd = oQ[i].second.second;
        while(ind < n && arr[ind] <= a[i]) {
            trie.insert(arr[ind]);
            ind++;
        }
        if(ind == 0) ans[qInd] = -1;
        else ansQ[ind] = trie.findMax(xi);
    }
    return ans;
}
```