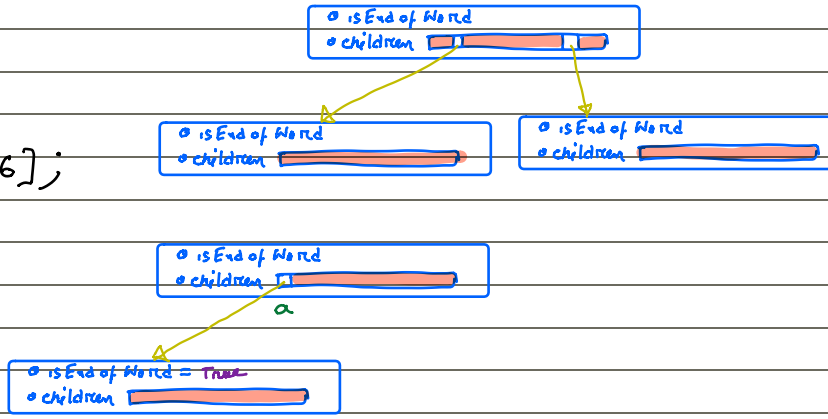
TRIE

Prefix Tree

insert(word) search(word) prefixSearch(prefix)

insert("apple") search("apple") prefixSearch("app")
True True.

```
struct TriNode {
    bool isEndofWord;
    TriNode * children[26];
};
```



E.g :- insert("a")

Longest Common Suffix

3093. Longest Common Suffix Queries

Hard | Topics | Companies | Hint

You are given two arrays of strings `wordsContainer` and `wordsQuery`.

For each `wordsQuery[i]`, you need to find a string from `wordsContainer` that has the **longest common suffix** with `wordsQuery[i]`. If there are two or more strings in `wordsContainer` that share the longest common suffix, find the string that is the **smallest** in length. If there are two or more such strings that have the **same** smallest length, find the one that occurred **earlier** in `wordsContainer`.

Return *an array of integers* `ans`, *where* `ans[i]` *is the index of the string in* `wordsContainer` *that has the* ***longest common suffix*** *with* `wordsQuery[i]`.

Input: `wordsContainer = ["abcd","bcd","xbcd"], wordsQuery = ["cd","bcd","xyz"]`

Output: `[1,1,1]`

Explanation:

Let's look at each `wordsQuery[i]` separately:

- For `wordsQuery[0] = "cd"`, strings from `wordsContainer` that share the longest common suffix `"cd"` are at indices 0, 1, and 2. Among these, the answer is the string at index 1 because it has the shortest length of 3.
- For `wordsQuery[1] = "bcd"`, strings from `wordsContainer` that share the longest common suffix `"bcd"` are at indices 0, 1, and 2. Among these, the answer is the string at index 1 because it has the shortest length of 3.
- For `wordsQuery[2] = "xyz"`, there is no string from `wordsContainer` that shares a common suffix. Hence the longest common suffix is `""`, that is shared with strings at index 0, 1, and 2. Among these, the answer is the string at index 1 because it has the shortest length of 3.

Example 2:

Input: `wordsContainer = ["abcdefgh","poiuygh","ghghgh"], wordsQuery = ["gh","acbfgh","acbfegh"]`

Output: `[2,0,2]`

Explanation:

Let's look at each `wordsQuery[i]` separately:

- For `wordsQuery[0] = "gh"`, strings from `wordsContainer` that share the longest common suffix `"gh"` are at indices 0, 1, and 2. Among these, the answer is the string at index 2 because it has the shortest length of 6.
- For `wordsQuery[1] = "acbfgh"`, only the string at index 0 shares the longest common suffix `"fgh"`. Hence it is the answer, even though the string at index 2 is shorter.
- For `wordsQuery[2] = "acbfegh"`, strings from `wordsContainer` that share the longest common suffix `"gh"` are at indices 0, 1, and 2. Among these, the answer is the string at index 2 because it has the shortest length of 6.

* at each node update the idx of smallest till now for the prefix till that character.

✓ Multiple things taken care of (just by inserting in this data structure)

i.e. smallest index; when abc, xbc

length = 3 & 3 but so node [c] [b] was there idx of smallest till now as 0 not 1

✓ if we had abc, xbc, bc then [c] [b] c & b node had value 2

Maximum XOR of 2 numbers in an Array

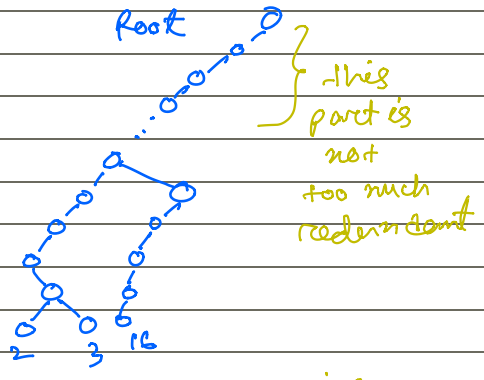
arr [3, 5, 2, 7]

3 - 0 1 1
5 - 1 0 1
2 - 0 1 0
7 - 1 1 1

0 1 1
1 0 1

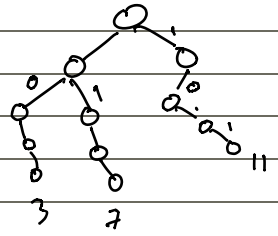
need to store whole 32 bit otherwise problem of not counting might occur.
& also in forward order

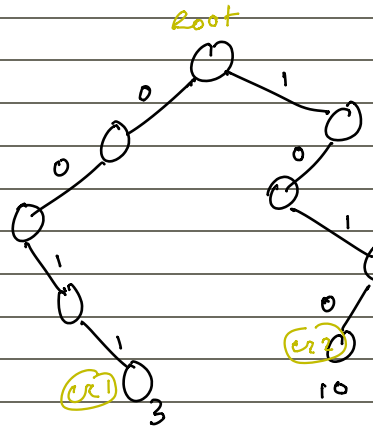
3 = 0000 ... 0011
2 = 0000 ... 0010
10 = 0000 ... 1010



3, 11, 7, 10

0 0 1 1
1 0 1 1
0 1 1 1





i = 3
3 = 0 0 1 1
p

temp = 1 → 1
temp = 10
temp = 100
temp = 1001

temp = 0

3 → 0 0 1 1
10 → 1 0 1 0
XOR 1 0 0 1

Replace Words :-

word container : ["cat", "bat"]

sentence : "a cattle was battling"

O/P : "A cat was bat"

✓ Approach : Using Hash (unordered Set).

✓ Approach : Using Trie

why Trie is better ;

map : {"cat", "bat"} words to be checked ⇒ a, cattle, was, battle.

cattle (i → i) } check map.find("c")
map.find("ca")
map.find("cat") ✓

was (i →) } check map.find("w") → # In Trie we can cut the time to traverse whole "was", could conclude that any word with 'w' isn't present in the wordContainer.
map.find("wa")
map.find("was") ✗

→ Pallindrome Pairs :-

!! Difficult AF !!