Tuesday, September 20, 2022 9:00 PM

[like, mango, man, go
i, sam, sung, samsung]

Given a string **A** and a dictionary of n words **B**, find out if A can be segmented into a space-separated sequence of dictionary words.

Note: From the dictionary **B** each word can be taken any number of times and in any order.

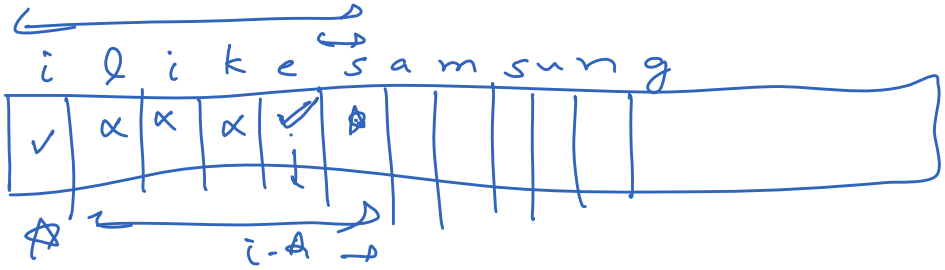
Example 1:

Input:
n = 12
B = { "i", "like", "sam",
"sung", "samsung", "mobile",
"ice","cream", "icecream",
"man", "go", "mango" }
A = "ilike"

ilikesamsung
i-like-sam-sung

9:01-9:11

i
like
sam
mango
sung



Design a k-stack in an array

int

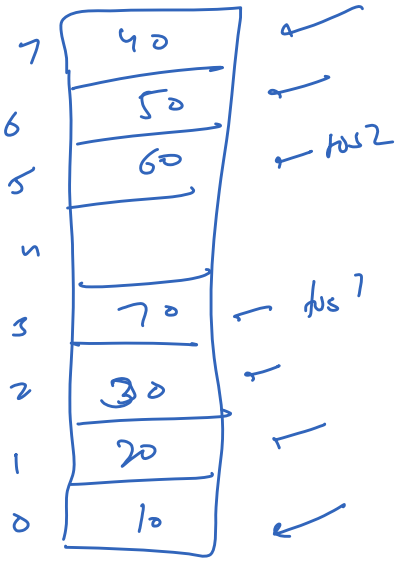
9:18-9:24

Push 10, 0
Push 20, 1
Push 30, 2
Push 40, 1
Push 50, 1

top 0
top 2
top 1
← tos2

Pop

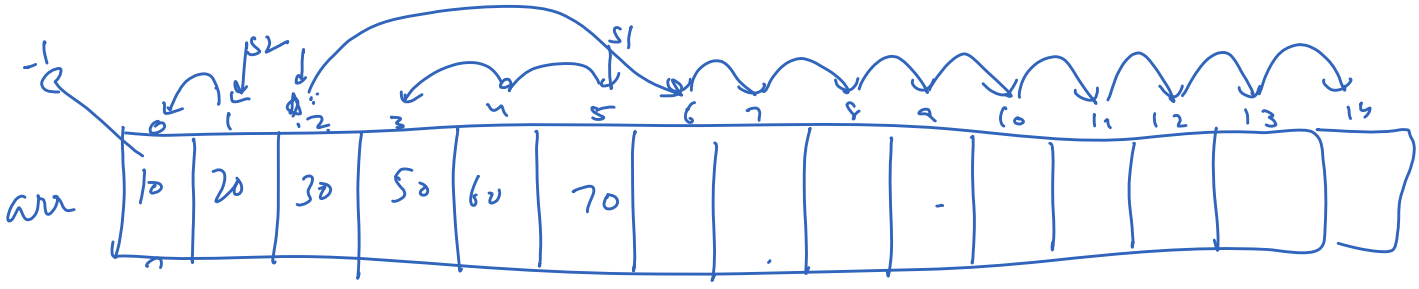
Push 10, 1

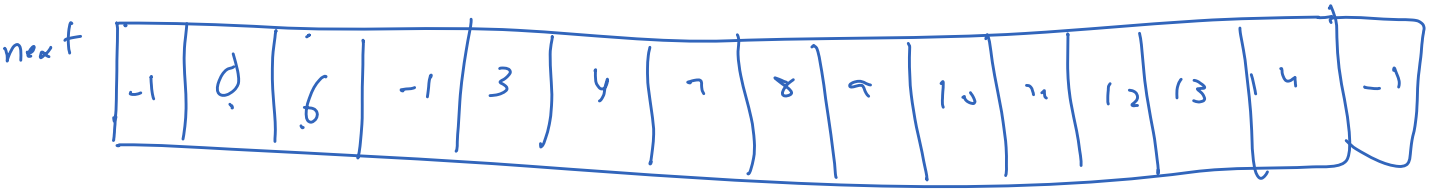


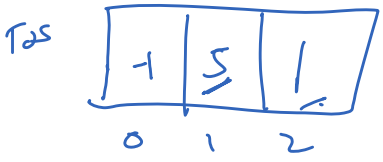
Pop 1

Push 10
Push 20
Push 30
Push 40
Push 50

Push 36
Push 26
Push 60







free = 0x36

Push 10, 2
— 20, 2
— 30, 2

Push 50, 1
— 60, 1
— 70, 1

```
int curr = free;

// remove from free
free = next[free];

// add to current
next[curr] = top[m];
top[m] = curr;

// receive value
arr[curr] = x;
```

Pop 2

10:35 - 10:48

art
arts
and
ant

```
class Trie {
    class Node {
        boolean eow = false;
        Node[] children = neew Node[26];
    }

    Node root;
    public Trie() {
```

ant

act

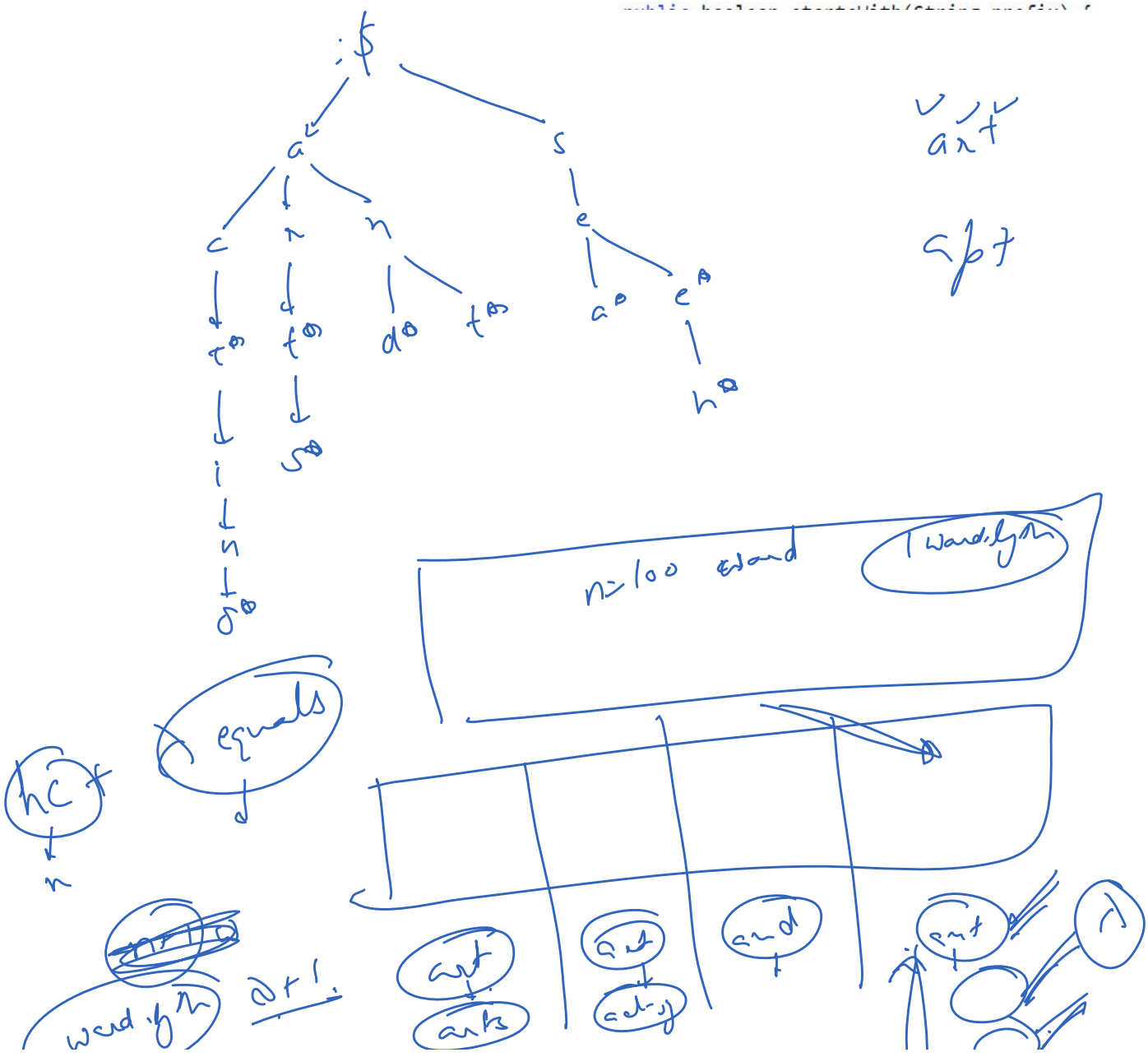
acl

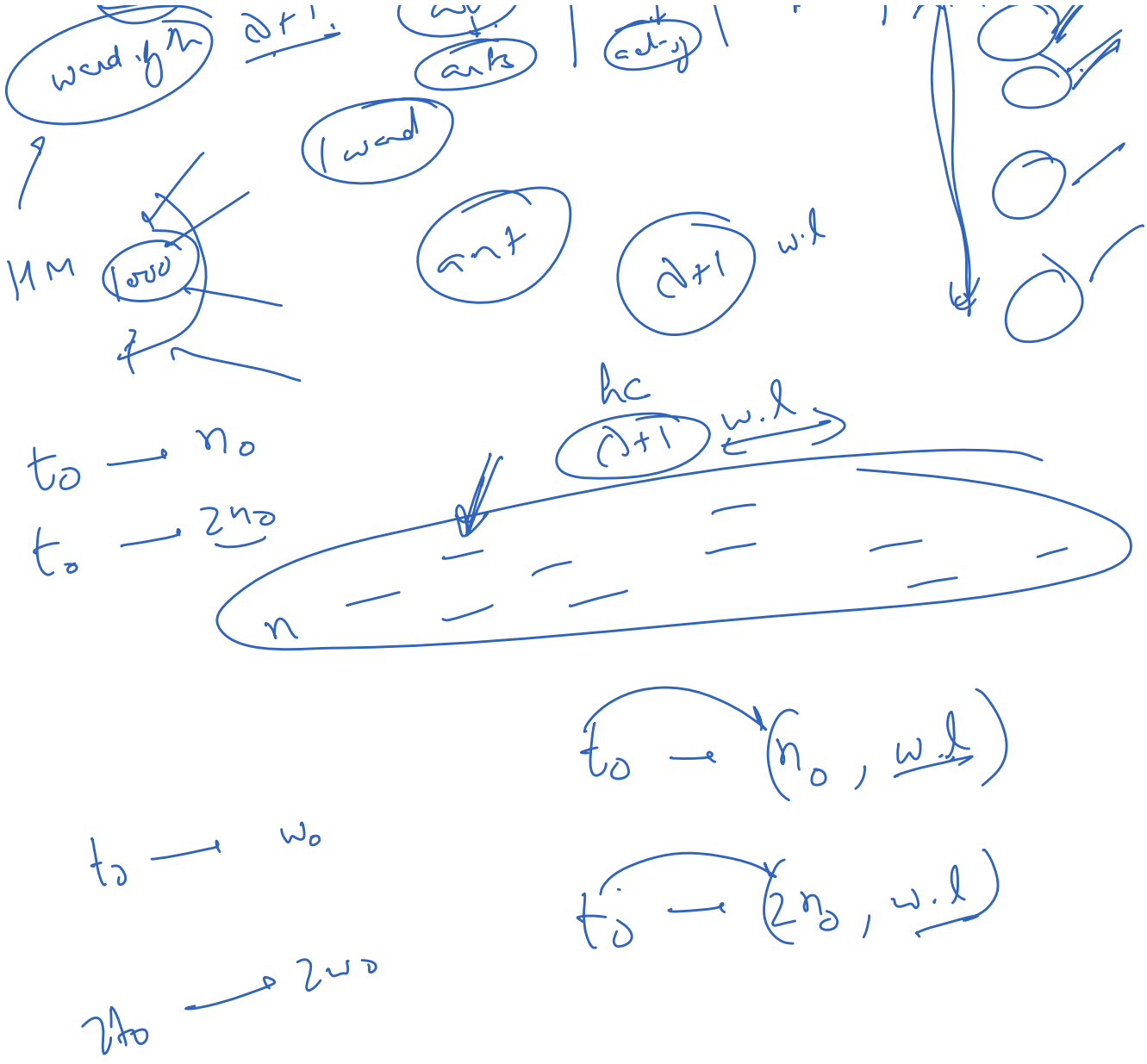
sea

see

seen

```
Node root;
public Trie() {
    root = new Node();
}

public void insert(String word) {
    Node node = root;
    for(char ch: word.toCharArray()){
        if(node.children[ch - 'a'] == null){
            node.children[ch - 'a'] = new Node();
        }
        node = node.children[ch - 'a'];
    }

    node.eow = true;
}

public boolean search(String word) {

}
```





211. Design Add and Search Words Data Structure

Medium 5446 286 Add to List Share

11:15 - 11:25

Design a data structure that supports adding new words and finding if a string matches any previously added string.

Implement the `WordDictionary` class:

- `WordDictionary()` Initializes the object.

211. Design Add and Search Words Data Structure

11:15 - 11:25

Medium 5446 286 Add to List Share

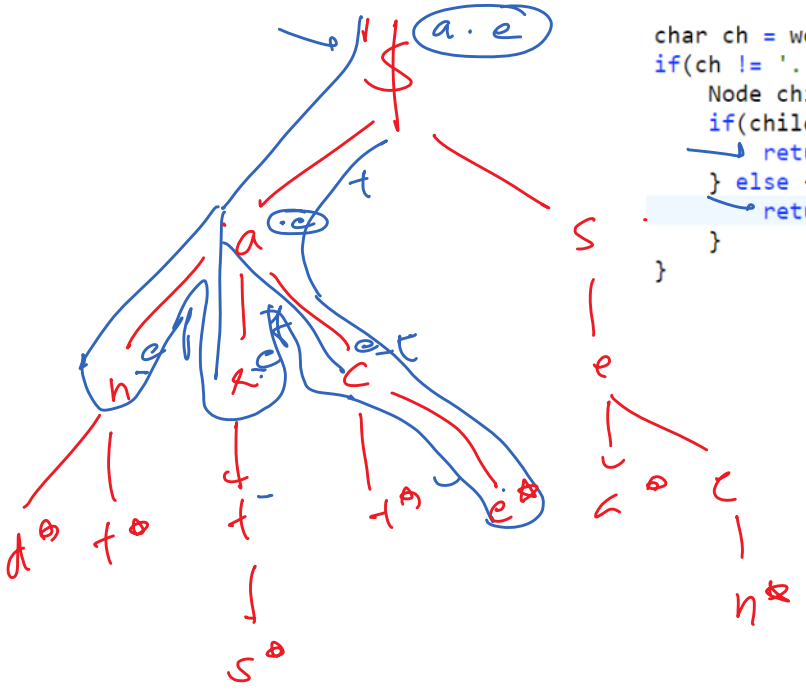
Design a data structure that supports adding new words and finding if a string matches any previously added string.

Implement the `WordDictionary` class:

- `WordDictionary()` Initializes the object.
- `void addWord(word)` Adds `word` to the data structure, it can be matched later.
- `bool search(word)` Returns `true` if there is any string in the data structure that matches `word` or `false` otherwise. `word` may contain dots `'.'` where dots can be matched with any letter.

insert
same

search
change




```
char ch = word.charAt(idx);
if(ch != '.'){
    Node child = node.children[ch - 'a'];
    if(child == null){
        return false;
    } else {
        return helper(word, idx + 1, child);
    }
}
```

and
ant
ants
act
sea
seen

11:47 - 12:00

Input

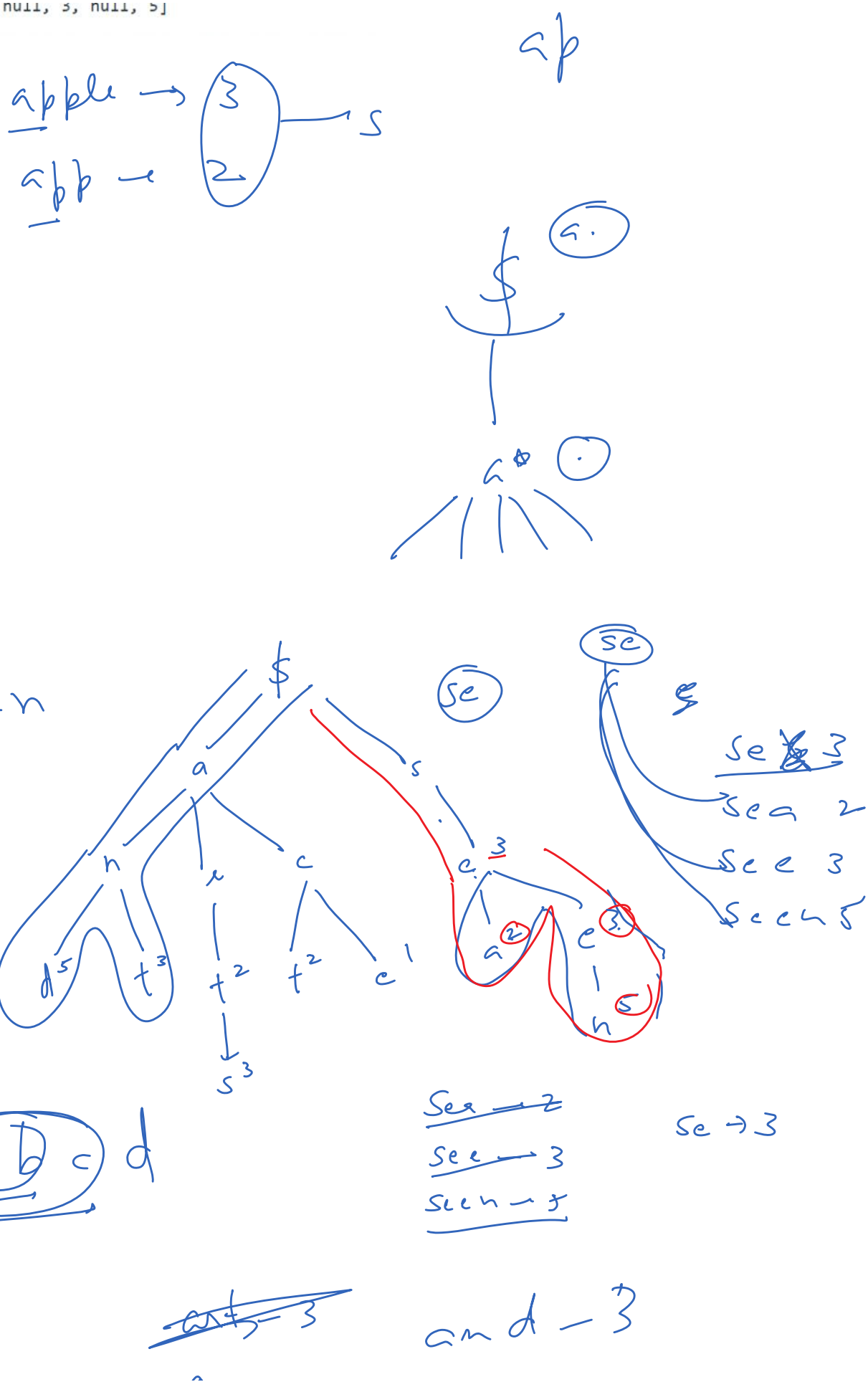
```
["MapSum", "insert", "sum", "insert", "sum"]
[[], ["apple", 3], ["ap"], ["app", 2], ["ap"]]
```

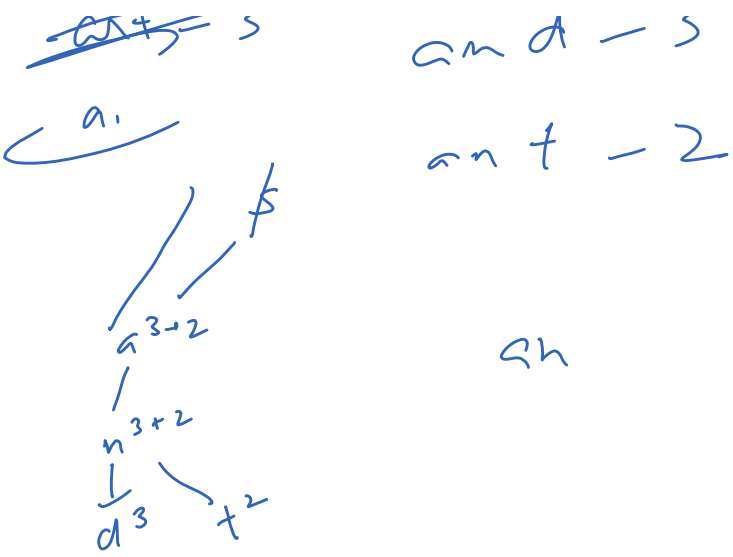
Output

```
[null, null, 3, null, 5]
```

ap

[null, null, 5, null, 5]





Work Page 7