Prefix Tree

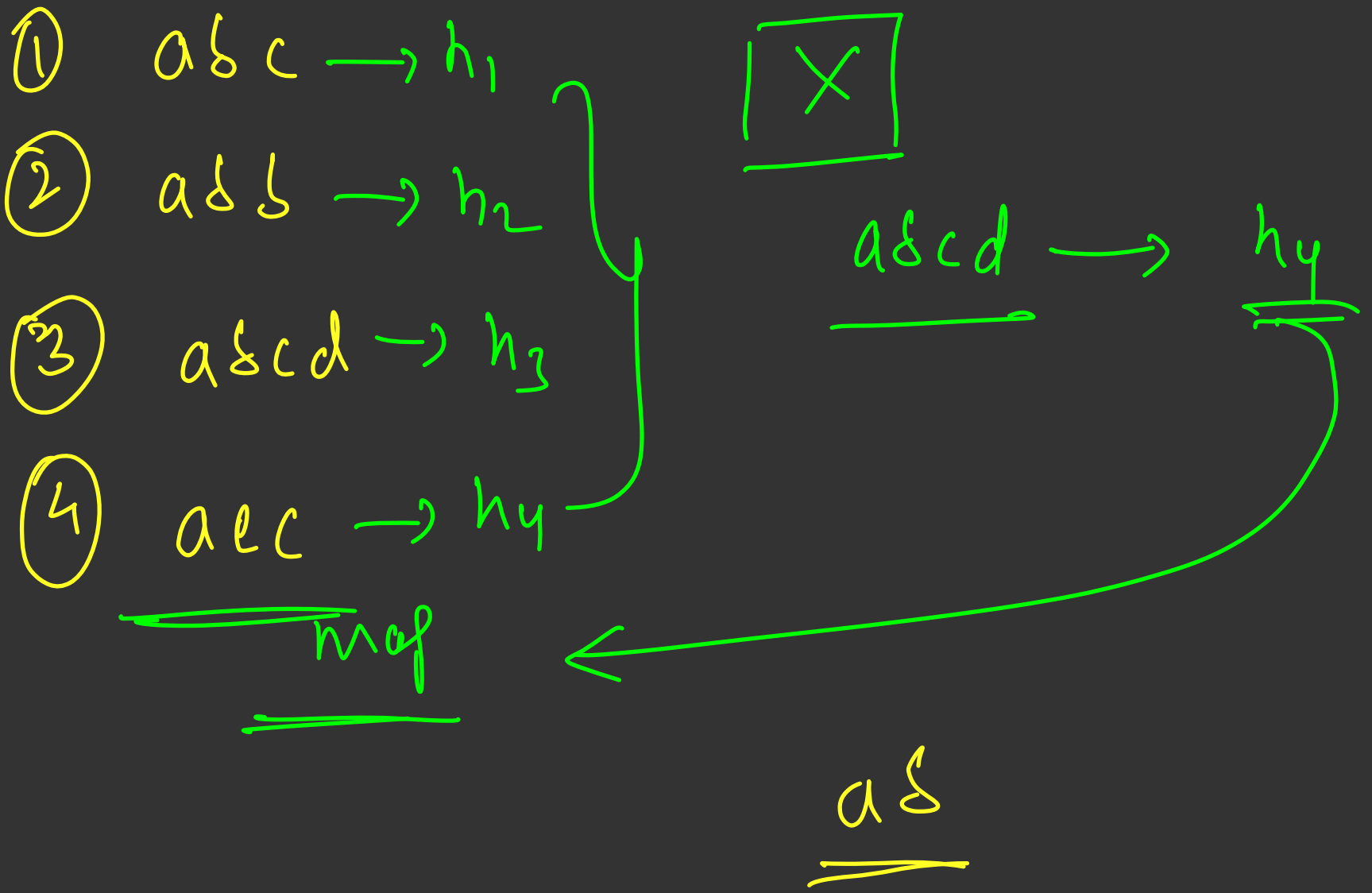
Tries

Trees ✗

- Priyansh Agarwal

Problem Statement

- Given a lot of strings a_1, a_2, a_3… a_n, do the following for a new string X
  - Find if X exists in the set of strings
  - Find number of strings with common prefix with X
- Also handle updates like add a new string a_{new} or delete an old string a_{old} from the set of strings and do the following for a new string X
  - Find if X exists in the set of strings
  - Find number of strings with common prefix with X


1 a&c → h1
2 a&b → h2
3 a&cd → h3
4 aec → h4
X
a&cd → h4
map
a&

n strings $1 \le n \le 10^4$

length of each string $\le 10^3$

Q queries $\rightarrow 1 \le Q \le 10^4$

X $\longrightarrow$ length of X $\le 10^3$

(1) check whether it is present

dictionary

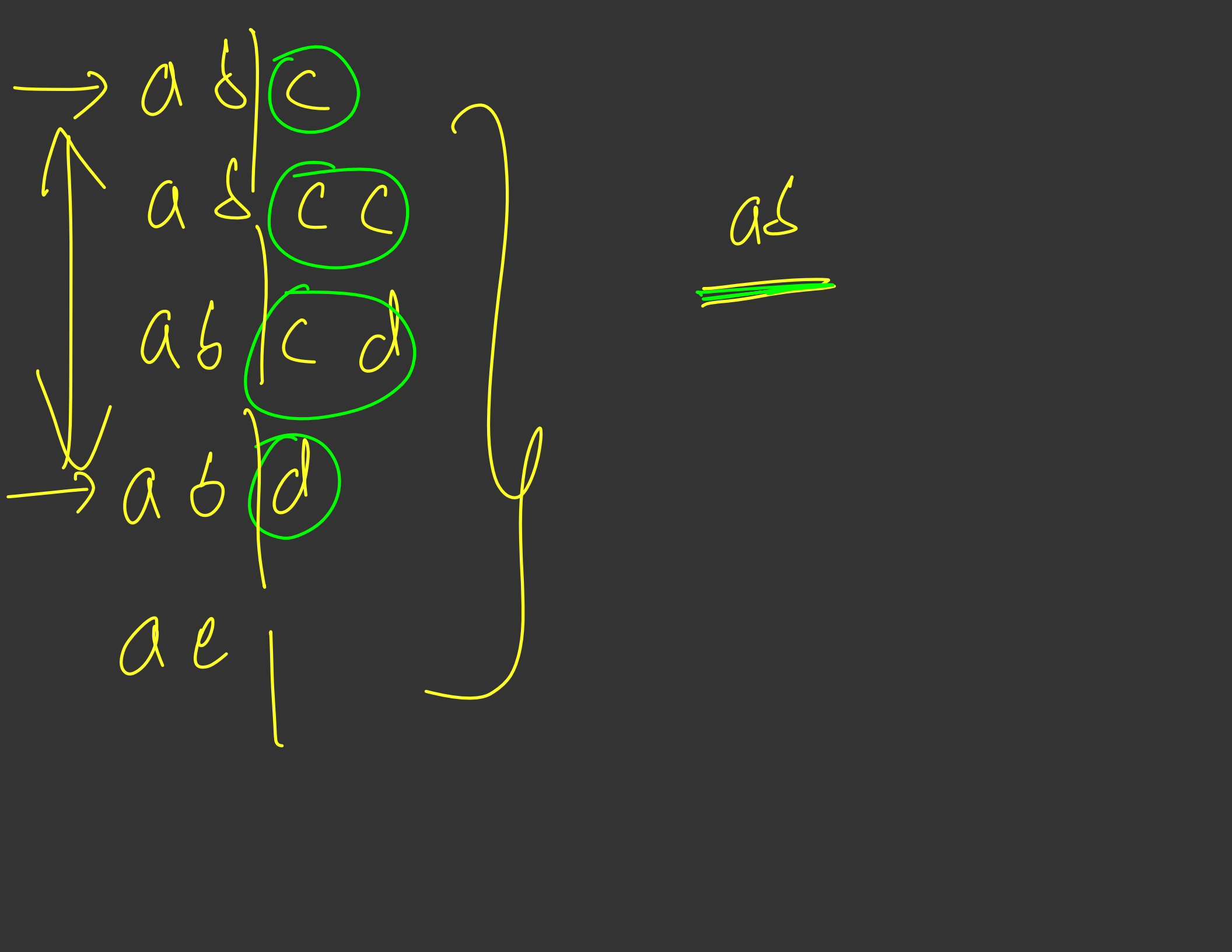
N

$\Leftrightarrow$

X → 5

O(N.Q)

a a |


as | c
as | cc
as | cd
ab | d
ae |
as

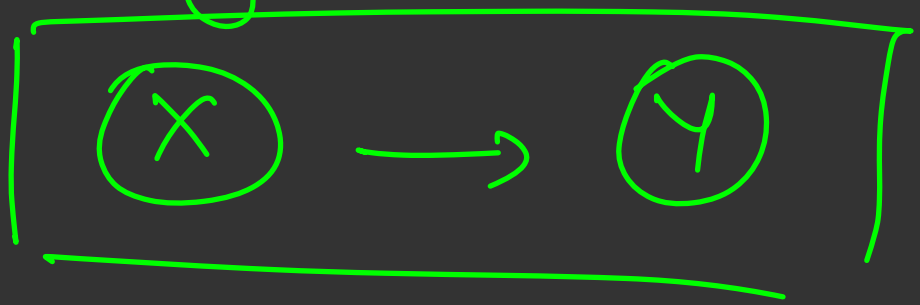
as $O(x)$

aa aaas ascc asd ae af b

$$O((\log n) \cdot \overline{x} \cdot Q)$$

① Bunch of strings (sorted)

O(logN) ↳ Binary Search



O(X)

checker → (x, y) tell

whether x > y or not

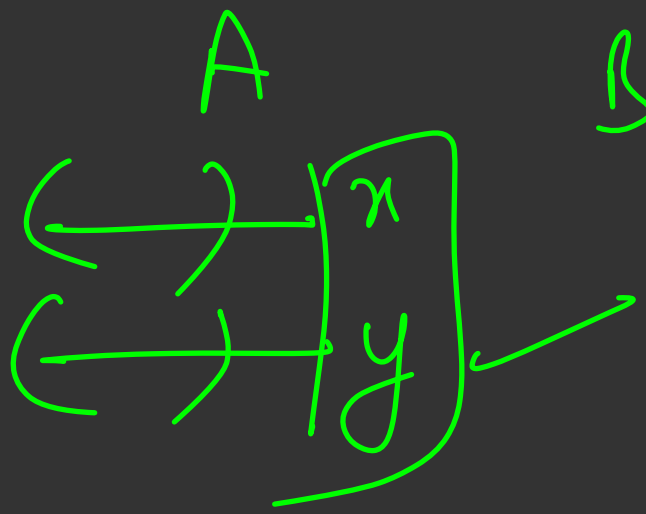
↓ ↓ ↓

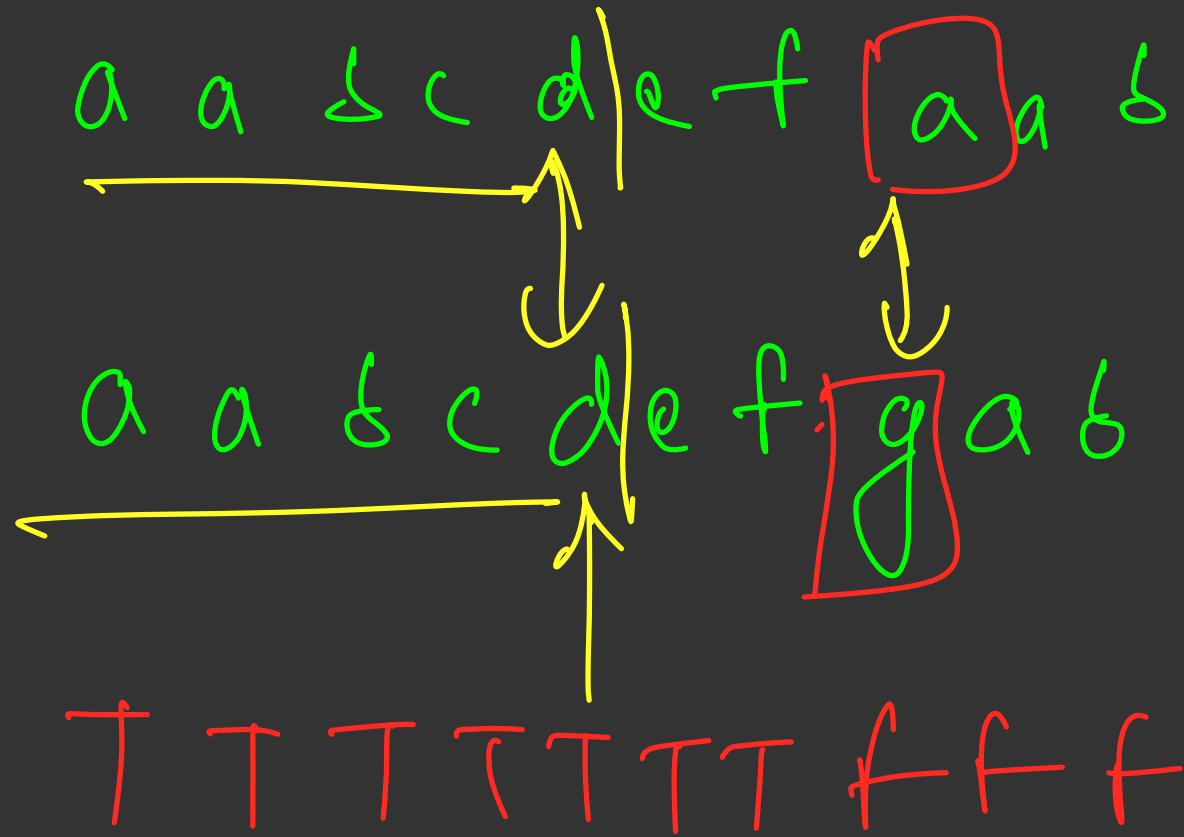
→ as

→ a a cd

Priyansh 1 2 3

Priyansi 1 2 3





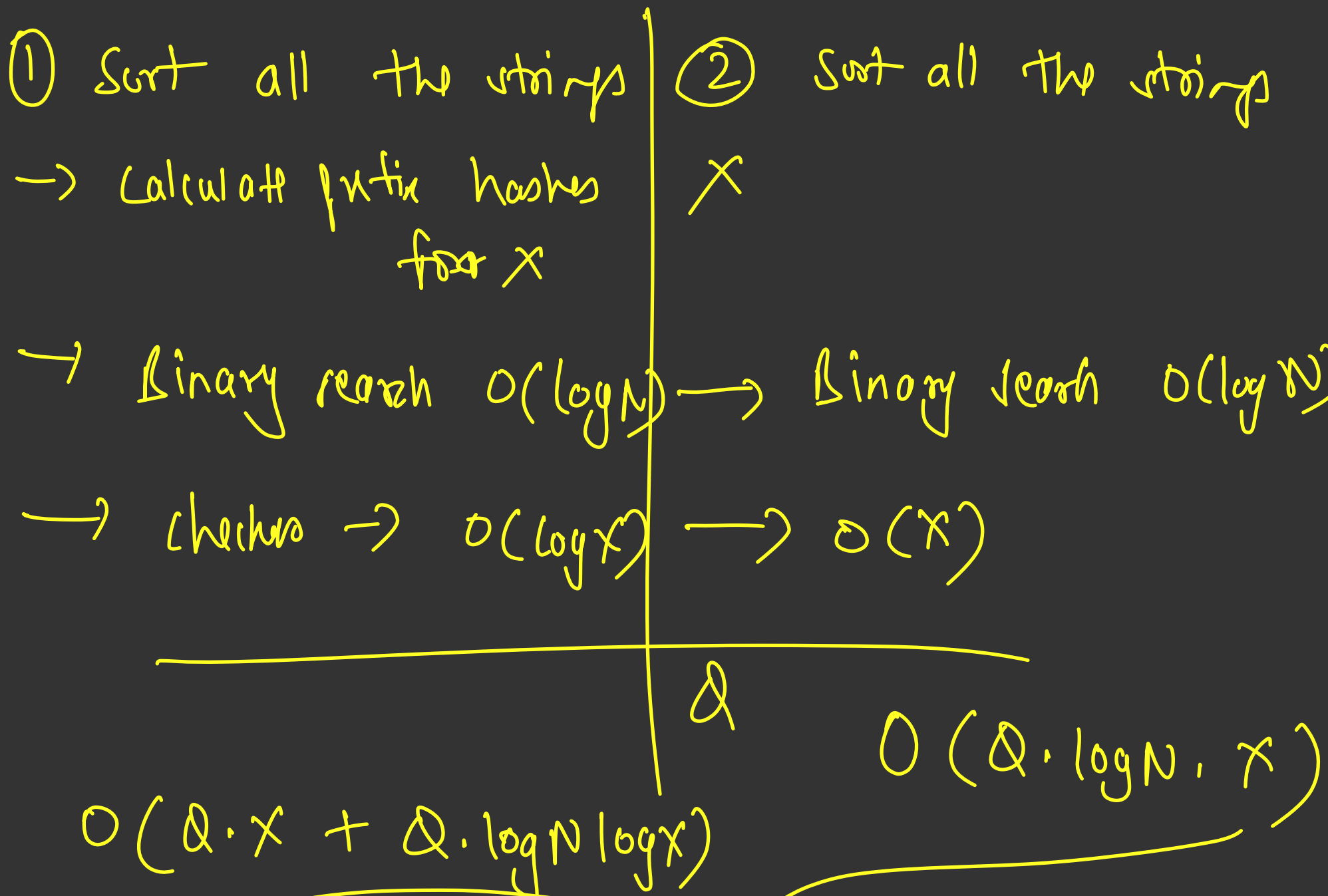
$O(x)$

$O(\log_y x)$

$$O(\log N \cdot \log x \cdot Q)$$

$$+$$

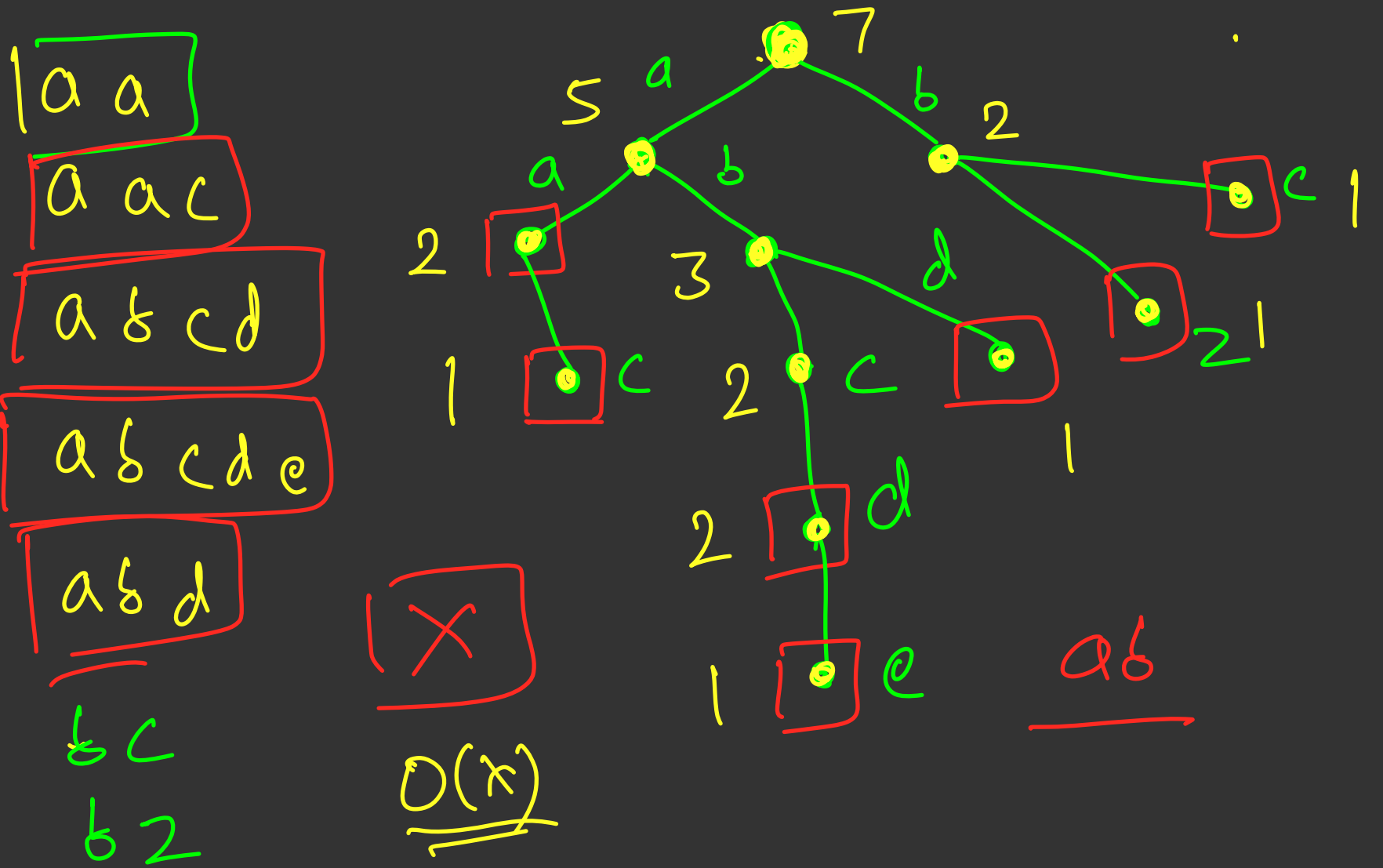
$$O(x \cdot Q)$$


① Sort all the strings
→ Calculate prefix hashes for x
→ Binary search O(log N)
→ checks → O(log x)
② Sort all the strings
x
→ Binary search O(log N)
→ O(x)
Q
O(Q · x + Q · log N log x)
O(Q · log N · x)

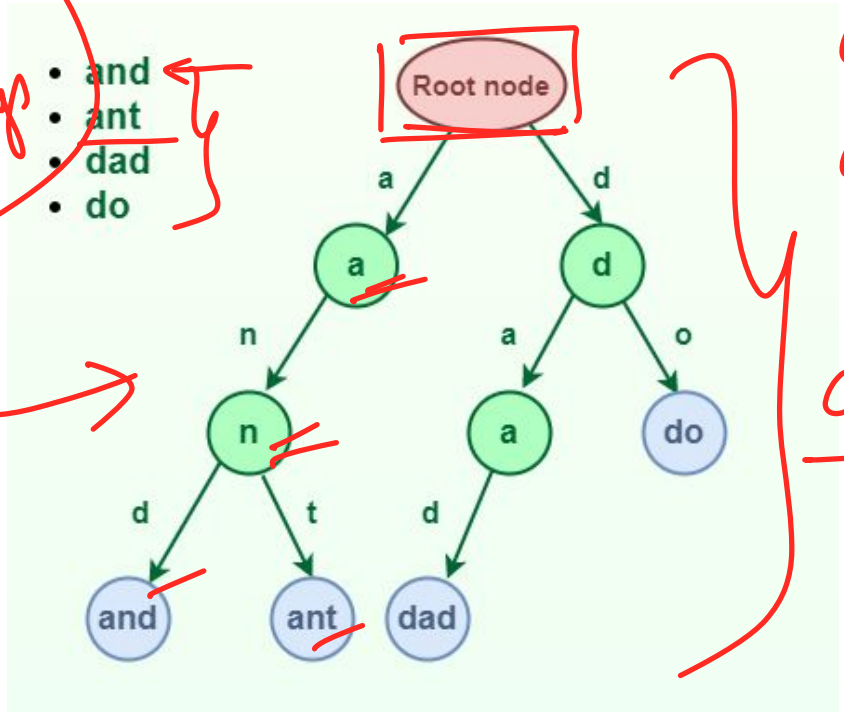
$O(a \cdot x)$ #

Solution

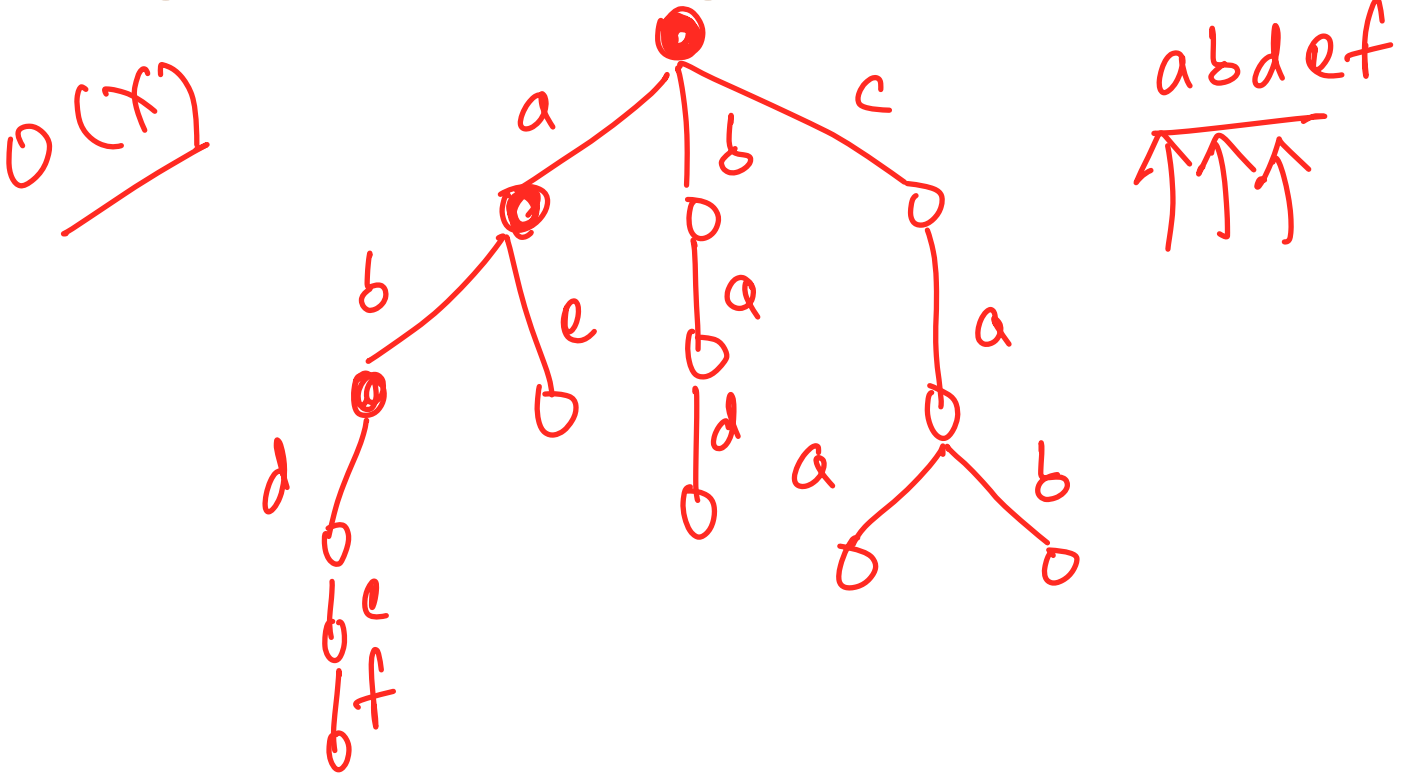
- Brute Force ✓ Hashing
- Sorting → Binary Search idea
- Let's build a Trie (no, the correct spelling is not Tree)


aa
aac
abcd
abcde
abd
bc
b2
X
O(x)
7
5
a
b
2
a
b
c
1
2
3
d
1
c
2
c
2
1
1
2
d
1
e
ab

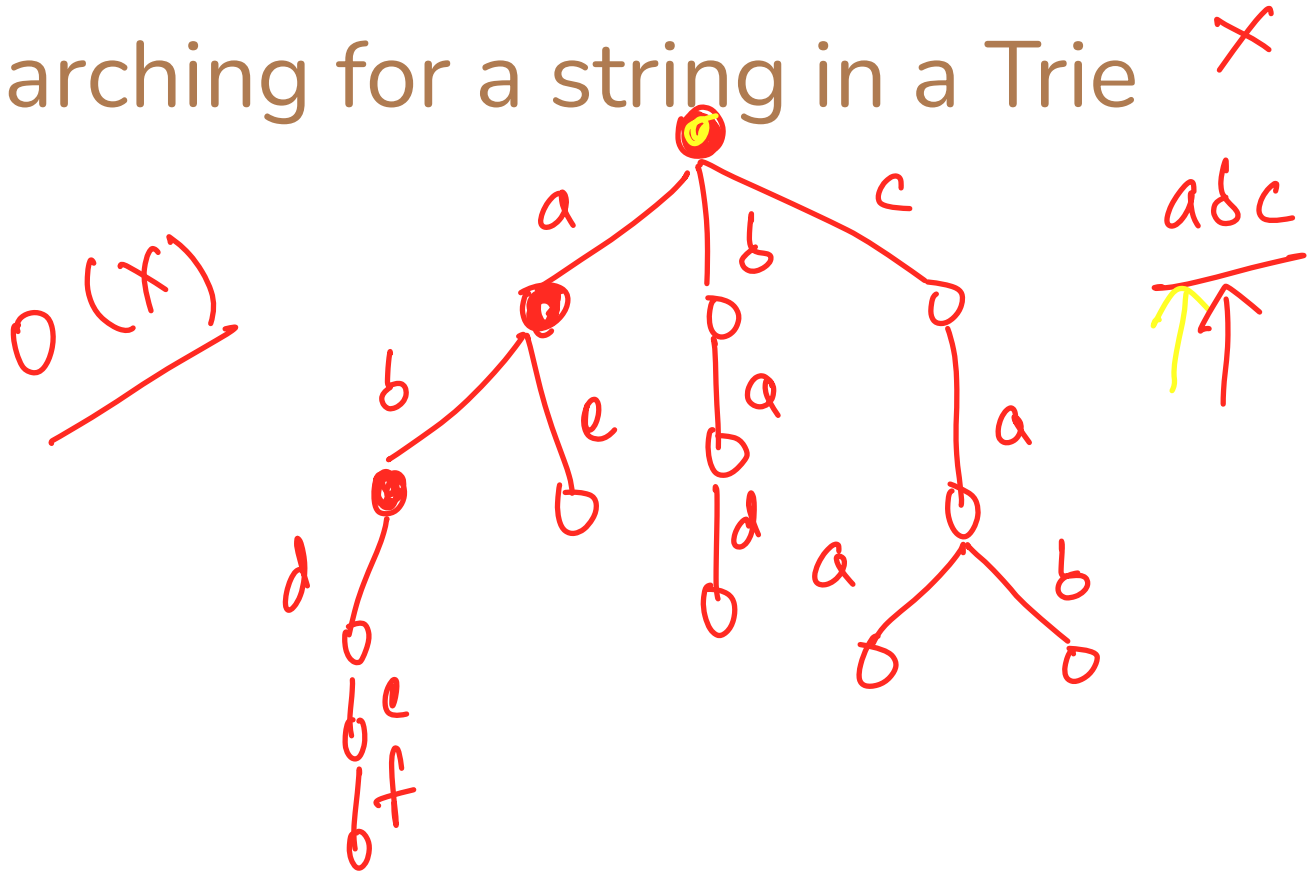
What is a Trie



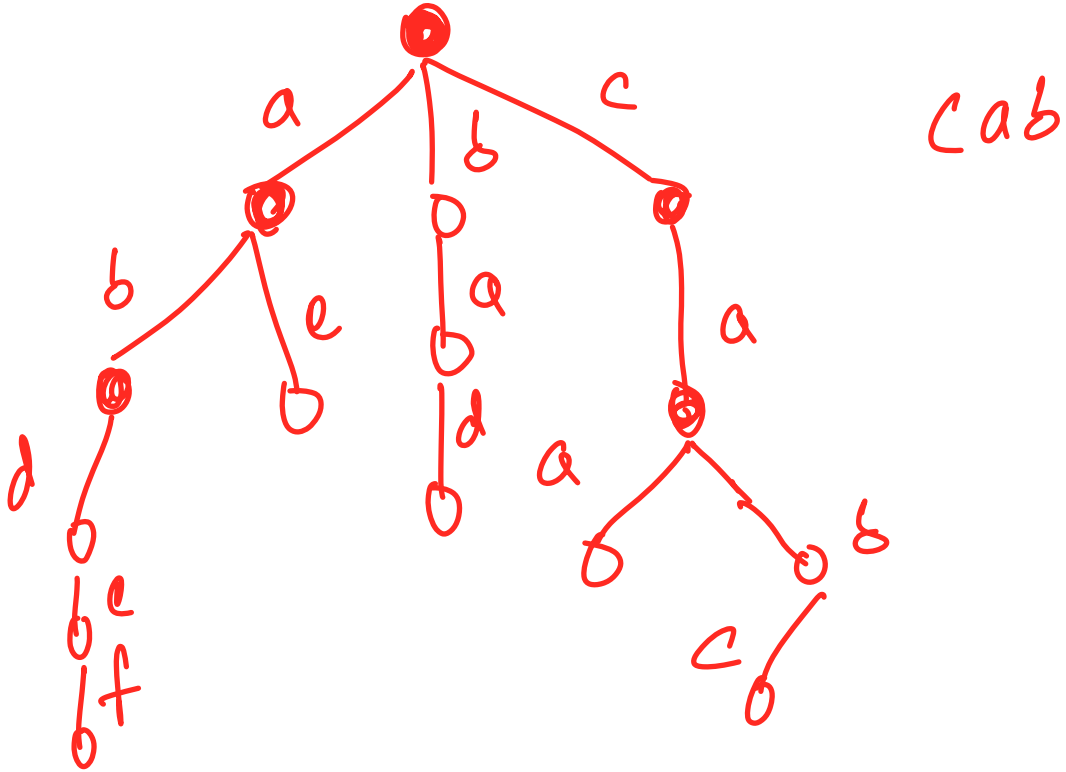
Adding a new string to Trie

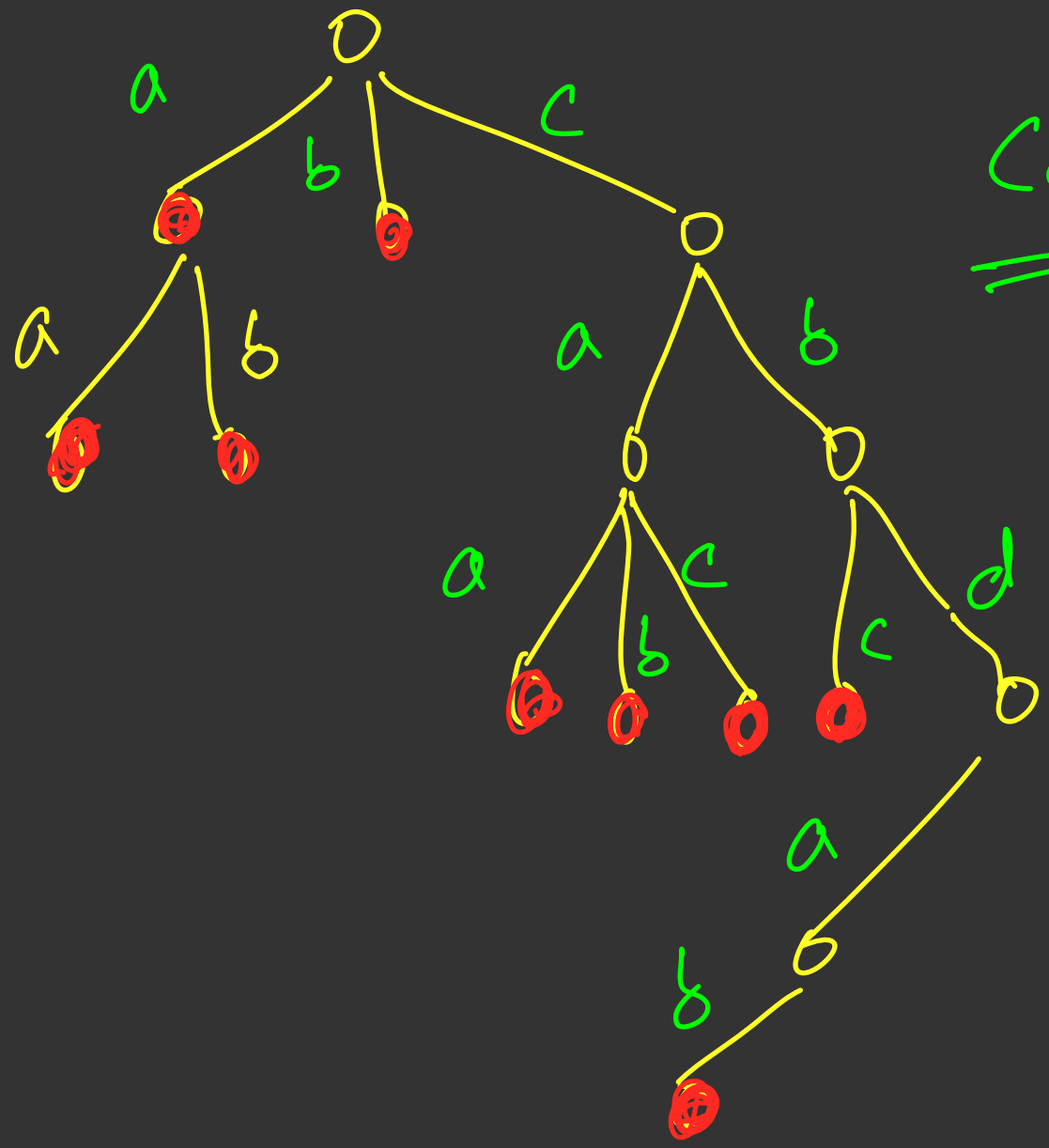


Searching for a string in a Trie



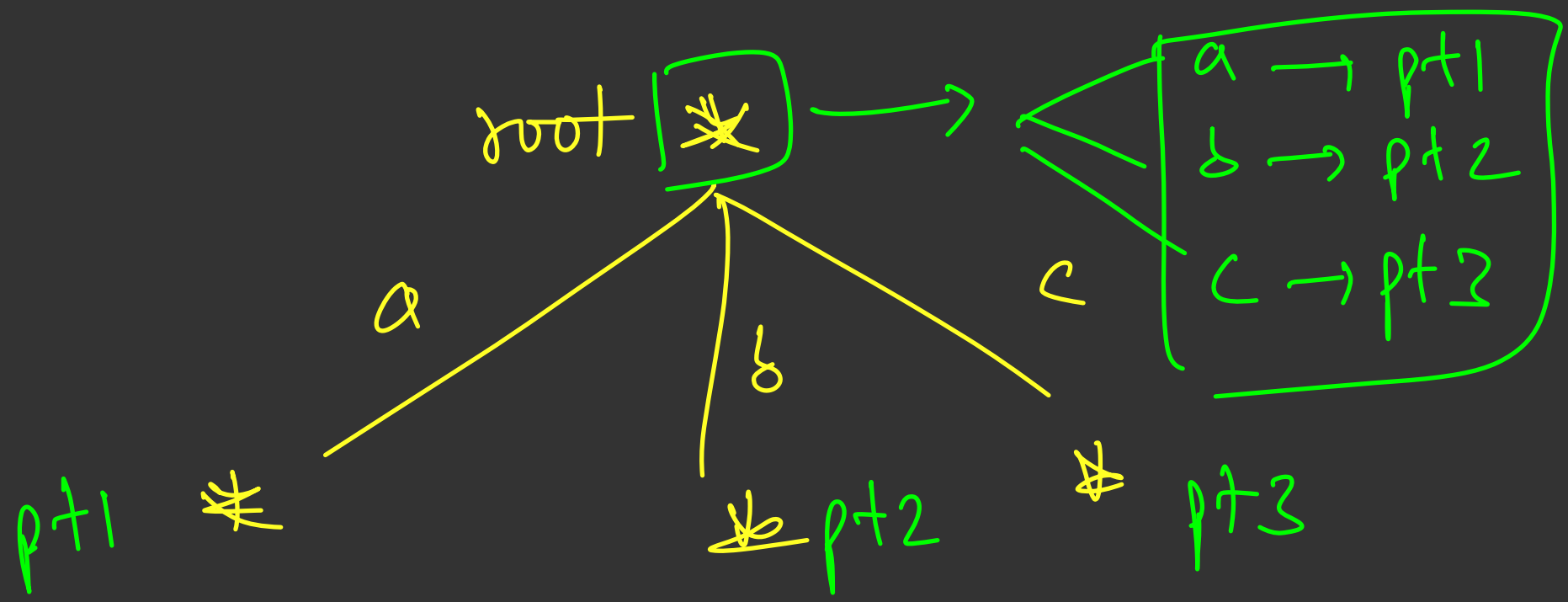
Deleting an old string from a Trie

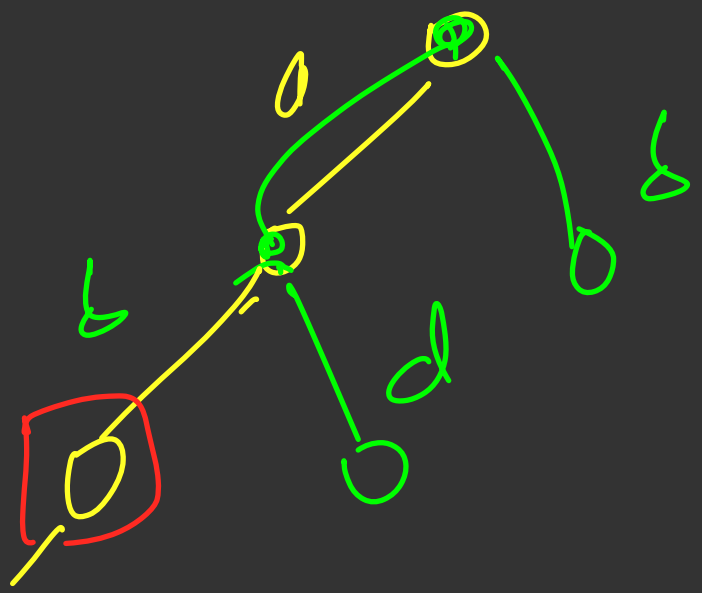




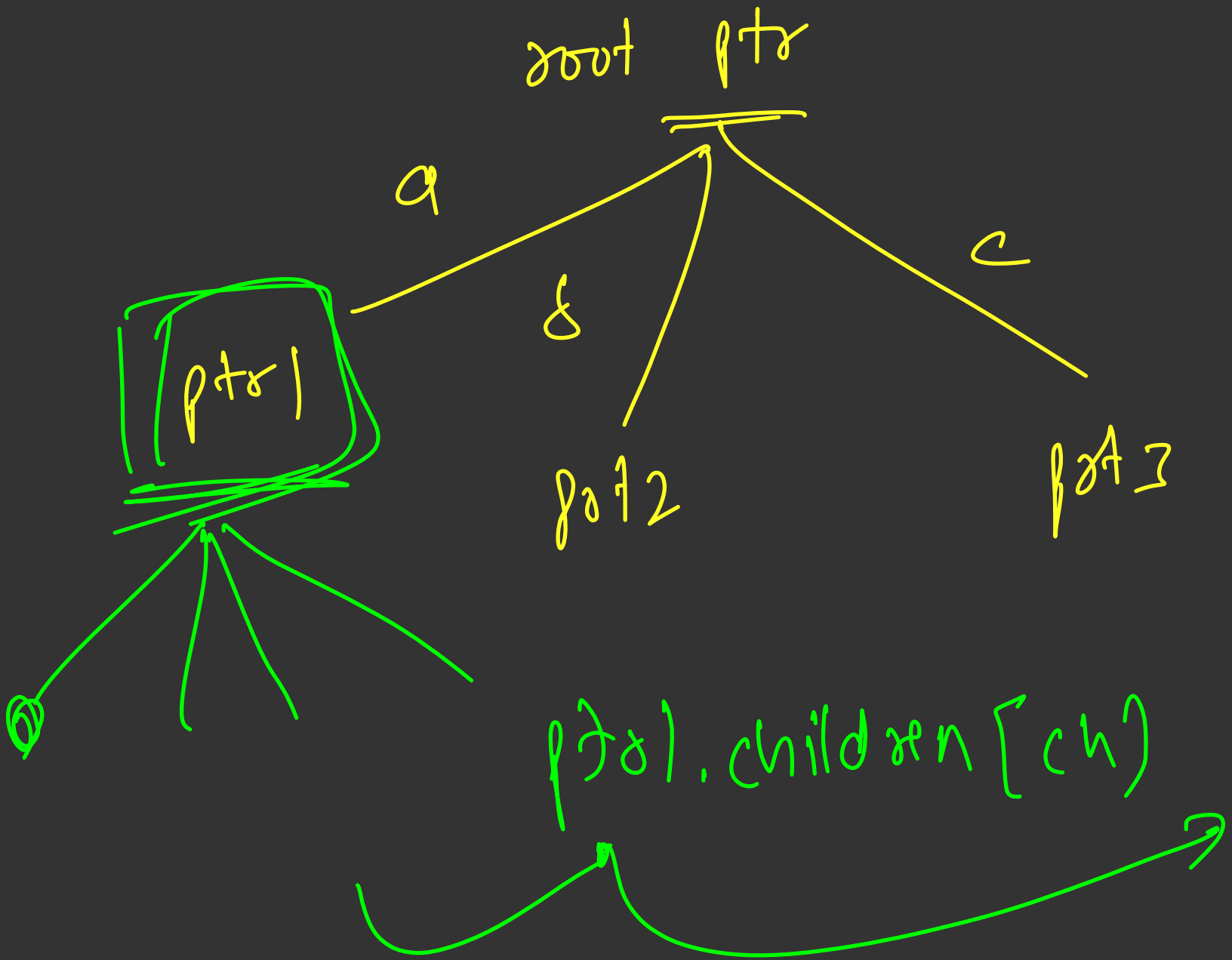
cbd ∝

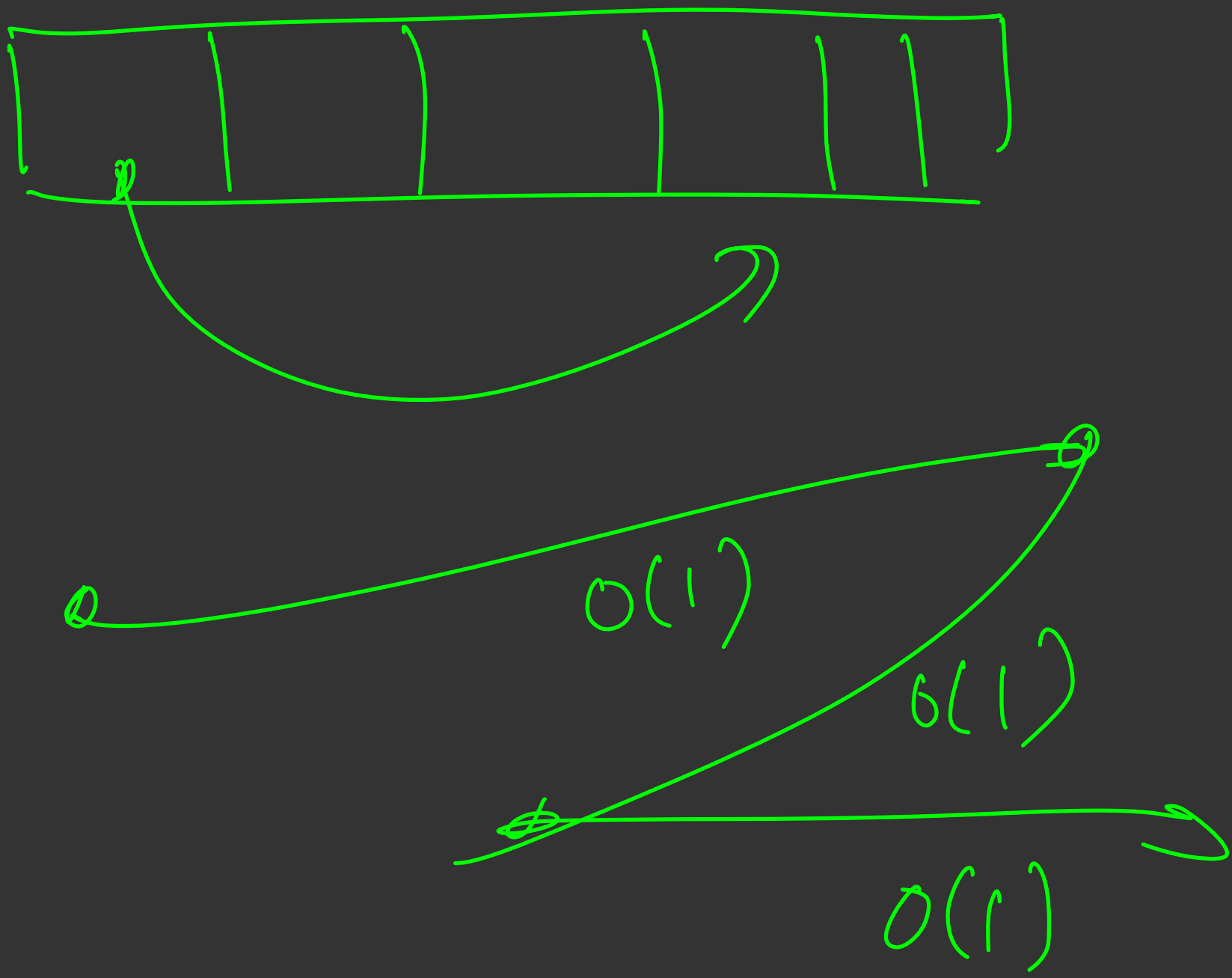
code

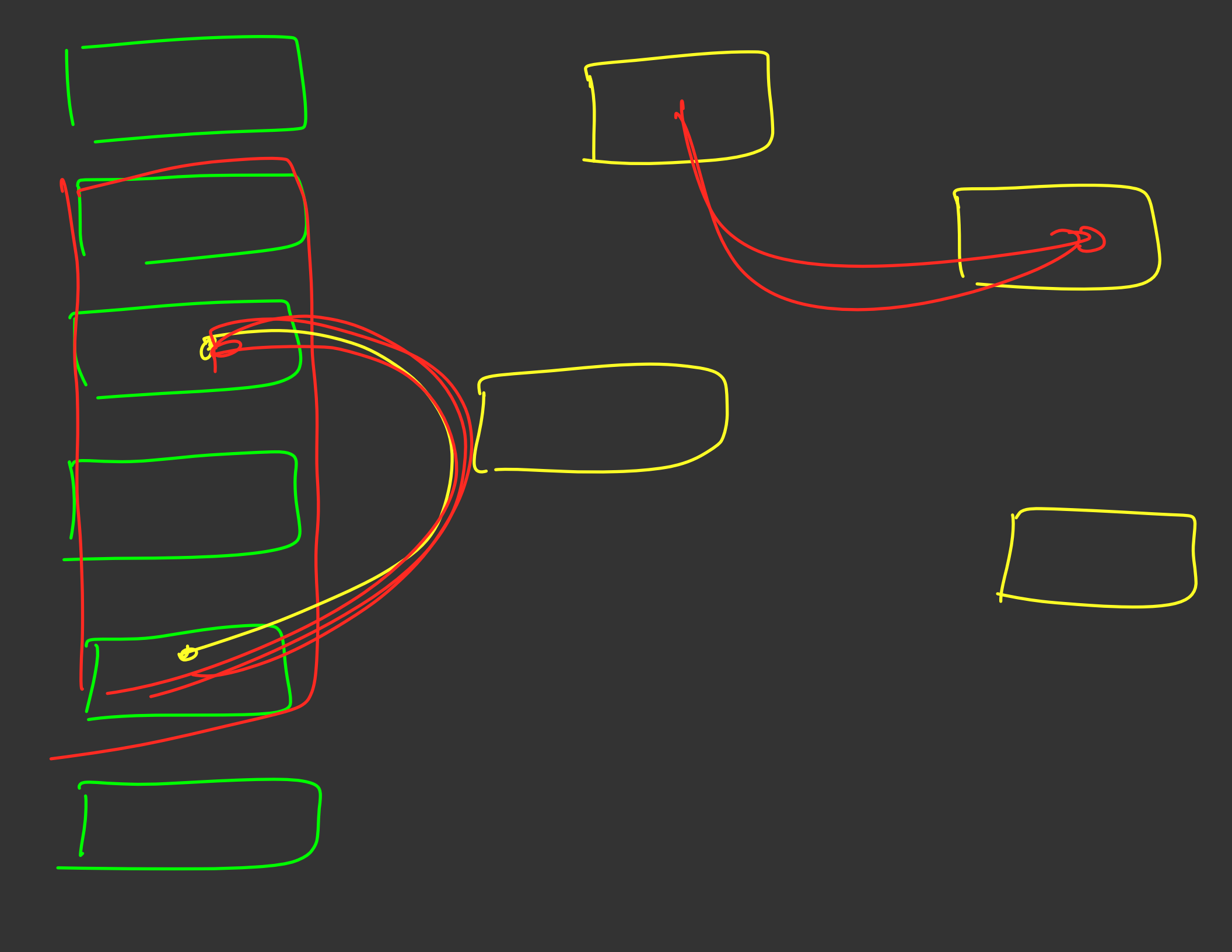

root
a
b
c
a → pt1
b → pt2
c → pt3
pt1
pt2
pt3

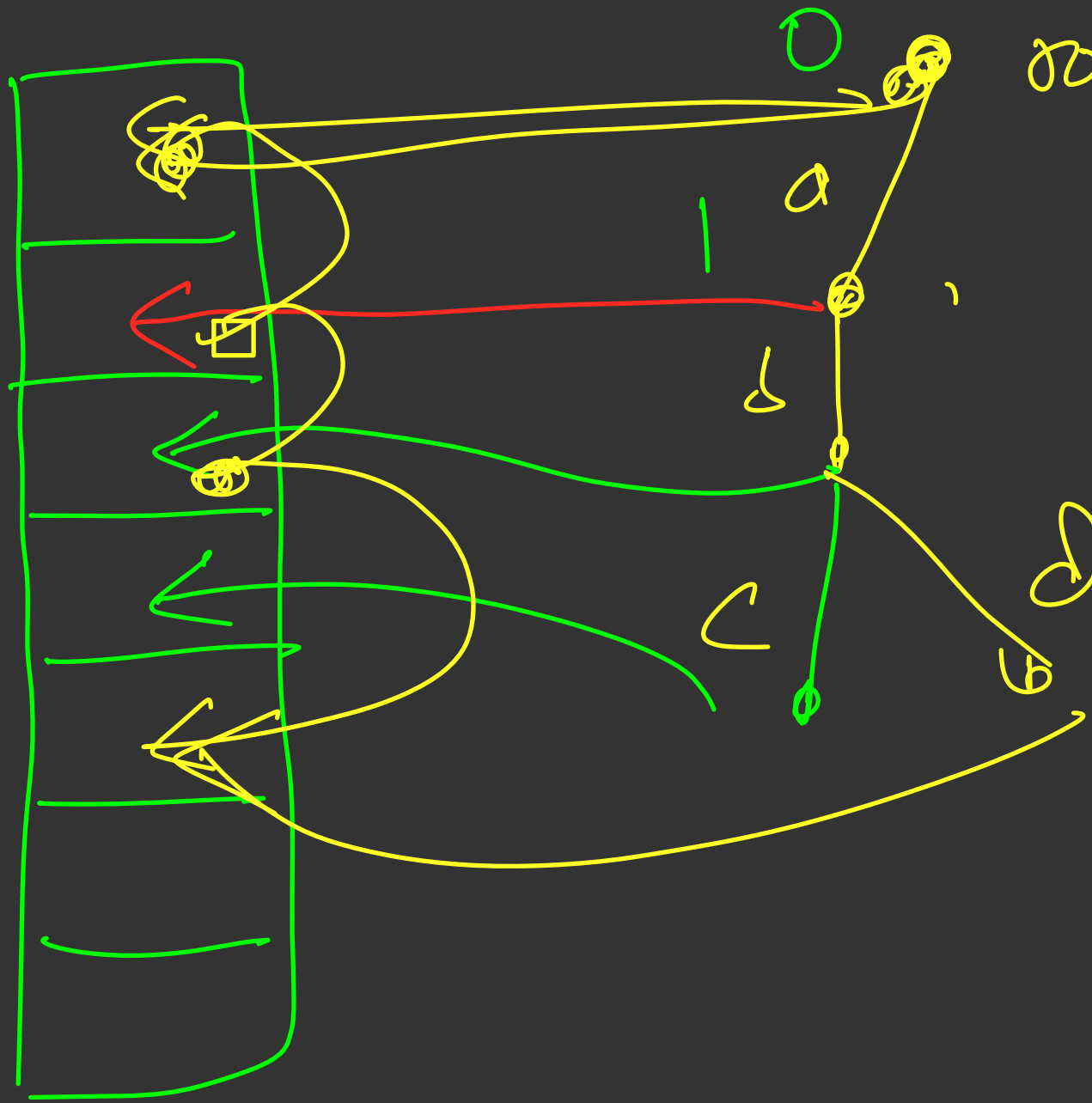


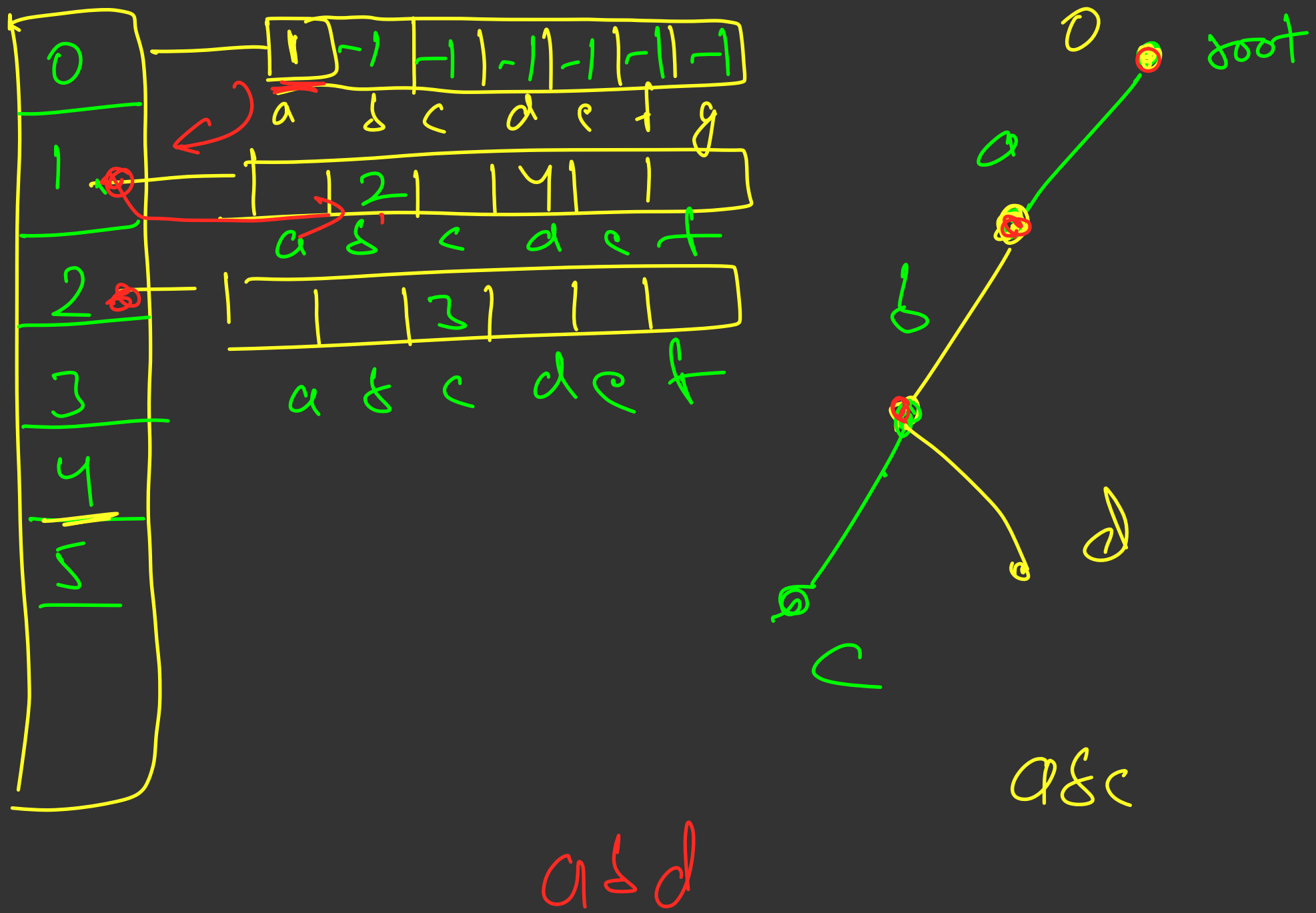
abc


root ptr
a
b
c
ptr1
ptr2
ptr3
ptr1.children[ch]

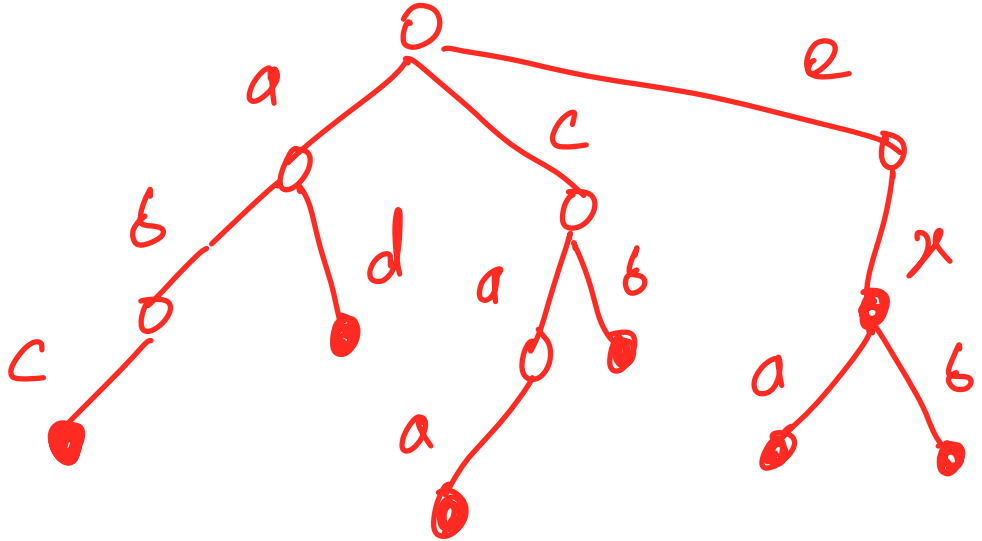

O(1)
O(1)
O(1)

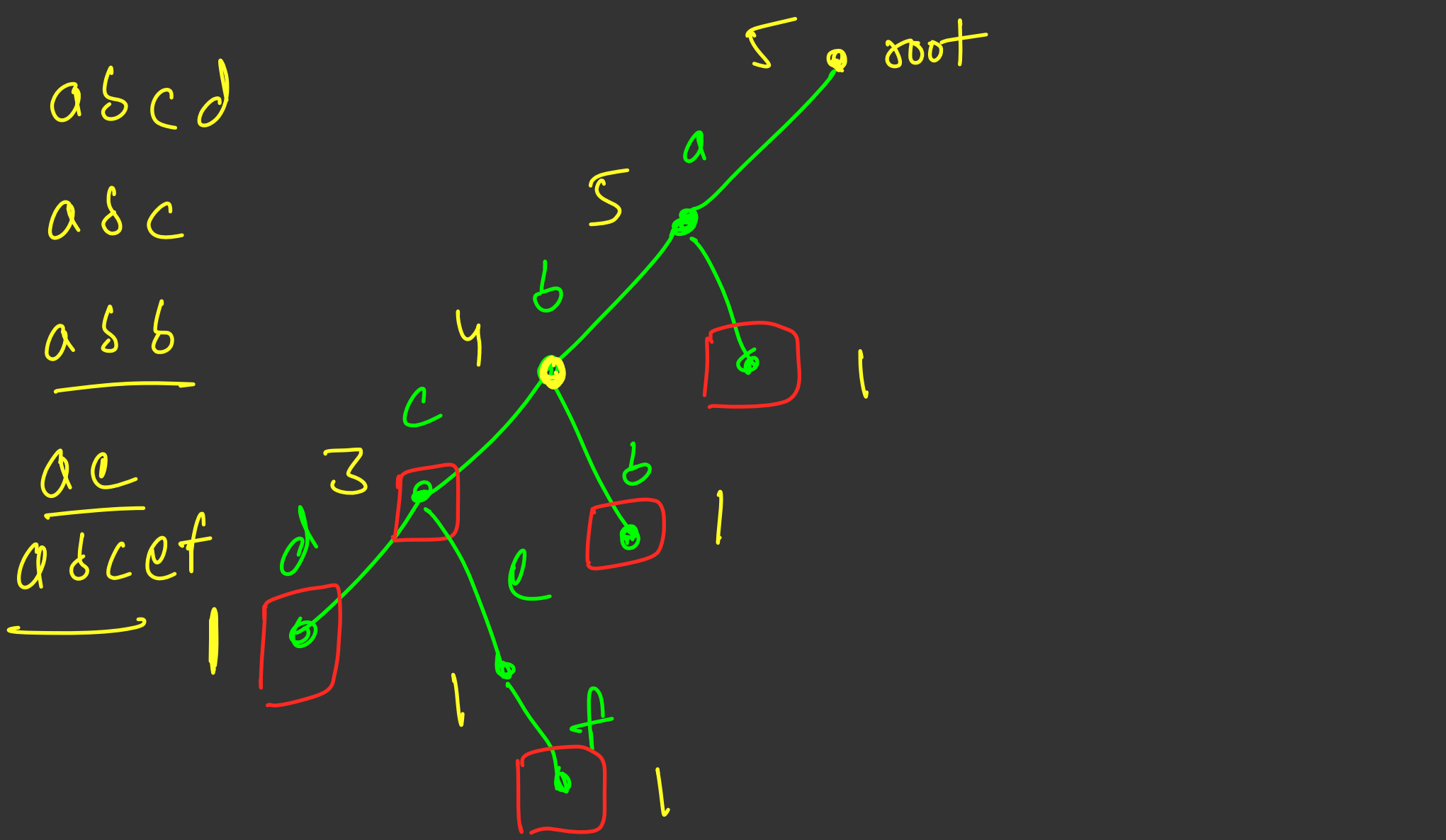



0
root
1
a
asc
asd
d
c
b


0
1
2
3
4
5
a b c d e f g
a b c d e f
a b c d e f
0
root
a
b
c
d
a&c
a&d

Precomputing subtree data




abcd
abc
abb
ac
abcef
5
root
a
5
b
4
c
3
d
1
e
1
f
1
b
1
b
1

Tries on Binary Numbers

- Given an array A of N integers ($0 <= a_i <= 1e9$), find the maximum XOR of 2 numbers.