Kartik

Max height by stacking cuboid

Solving Hard Questions - Level 1 - LIVE

Special class

Vishal Pachauri • Sept 18, 2022

→ Max Height by stacking cuboid

o/p → 190

i/p → {

w l h

[50, 45, 20]

[95, 37, 53]

[45, 23, 12]

}

Step 1 → sort all dimension for each cuboid

w l h

{

[20, 45, 50]

[37, 53, 95]

[12, 23, 45]

}

$$\left\{ \begin{array}{l} \quad w \quad\; l \quad\; h \\ [20, 45, 50] \\ [37, 53, 95] \\ [12, 23, 45] \end{array} \right\}$$

Step 2 -> sort according to width

$$\left\{ \begin{array}{ll} & [12, 23, 45] \\ 1 & [20, 45, 50] \\ 2 & [37, 53, 95] \end{array} \right\}$$

C1

C1

C3

9

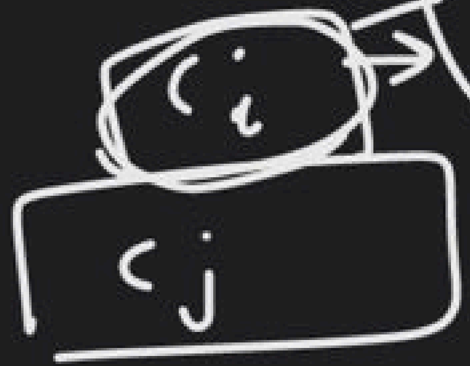
50 <= 45 -> F

[12, 23, 45]
[20, 45, 50]
[37, 53, 95]

C_i $\quad$ C_j

Question padho

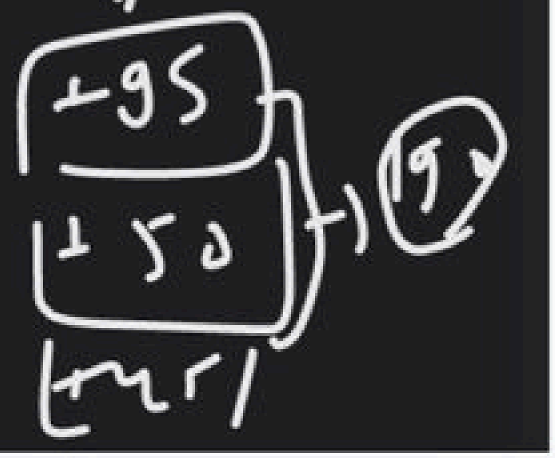
$l_i < l_j$
$h_i < h_j$
$w_i < w_j$

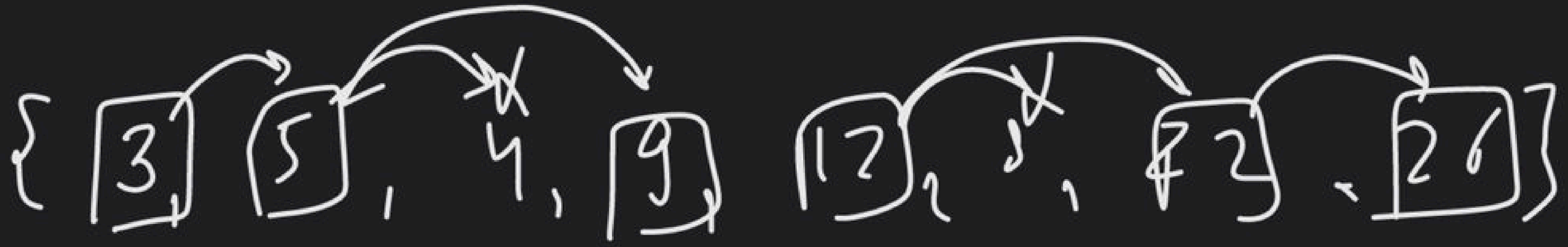


12 <= 20 T
23 <= 45 T
45 <= 50 T

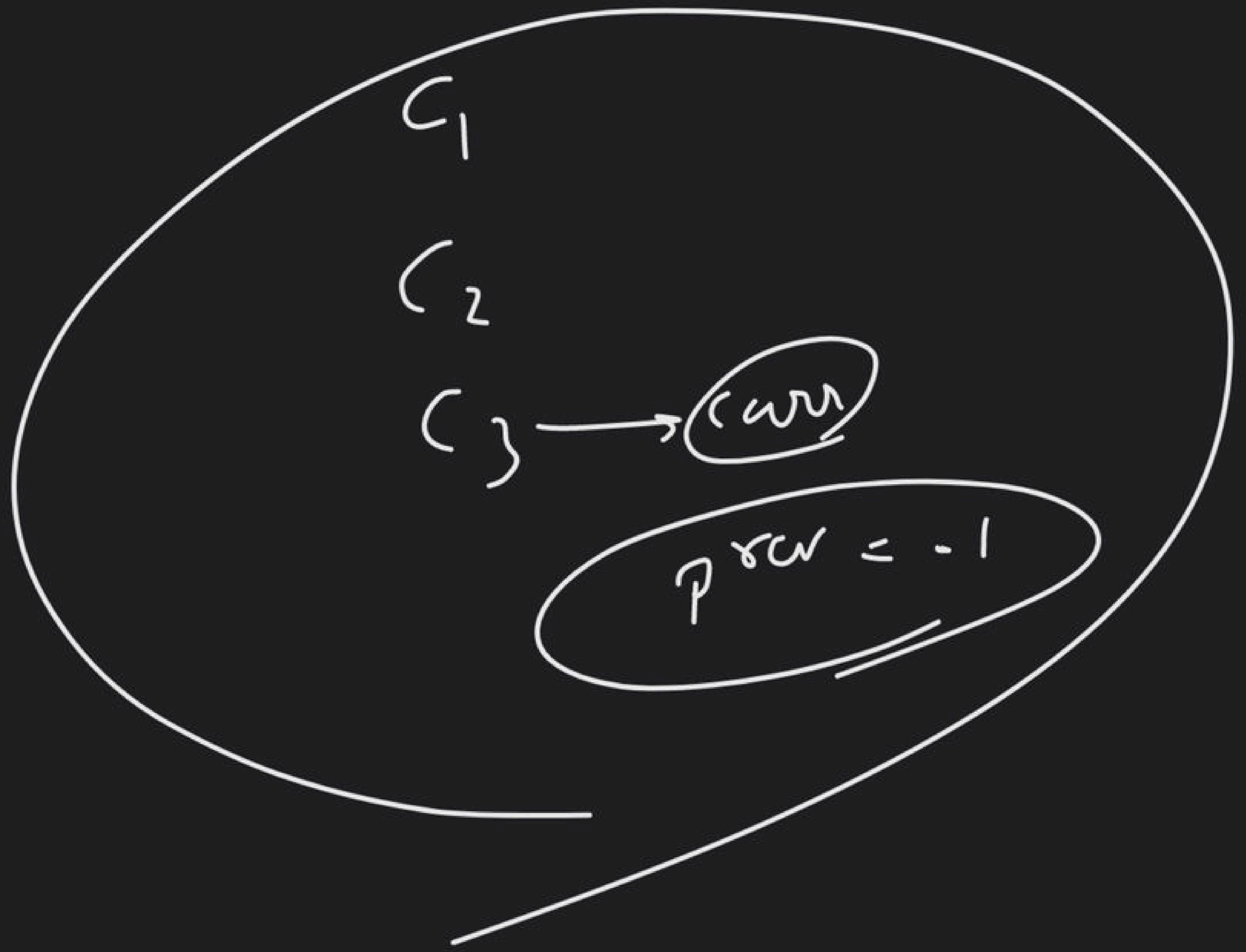
20 <= 37 T
45 <= 53 T
50 <= 95 T

ans





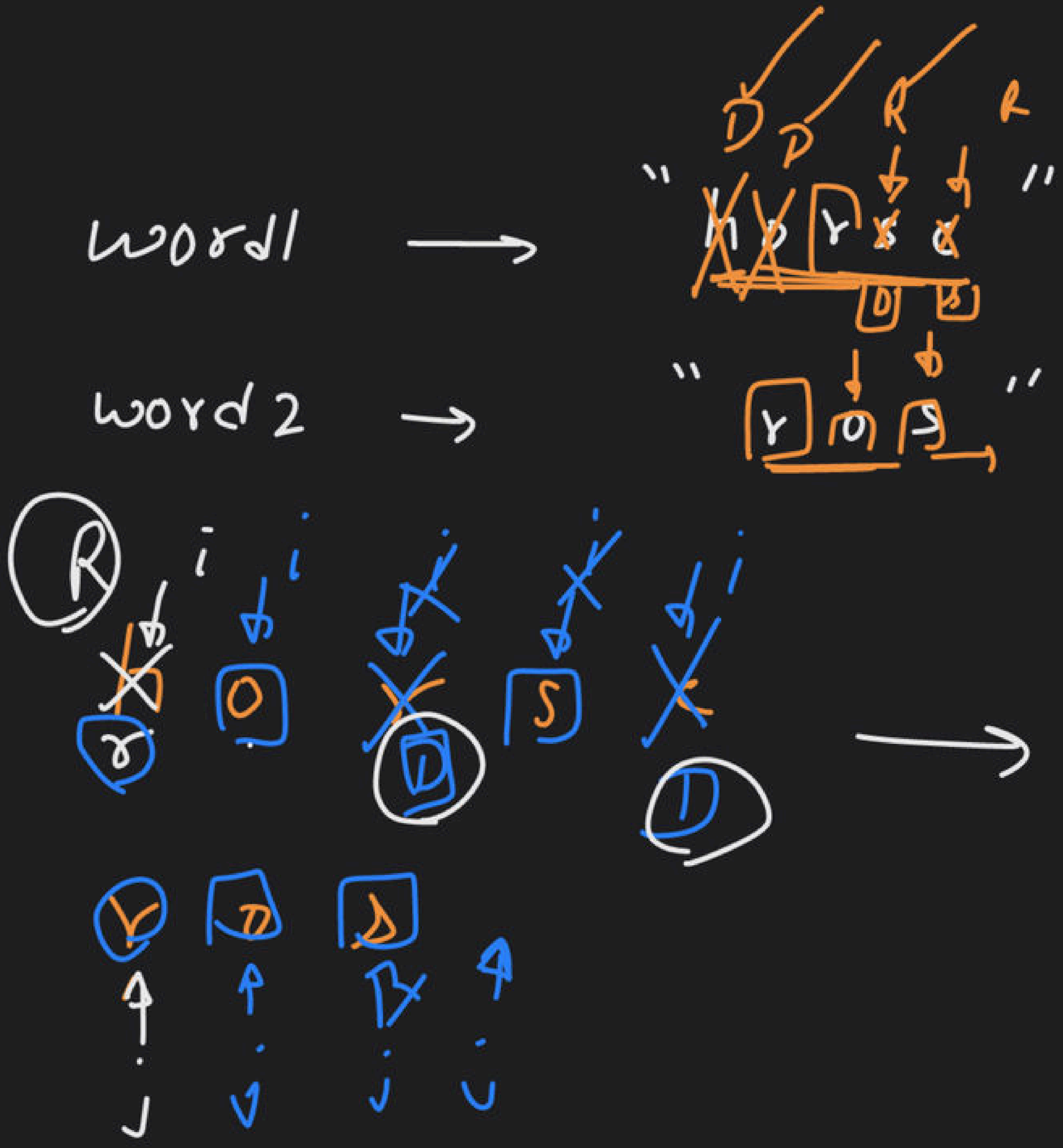
Longest Inc Subseq


C_1
C_2
C_3
curr
prev = -1

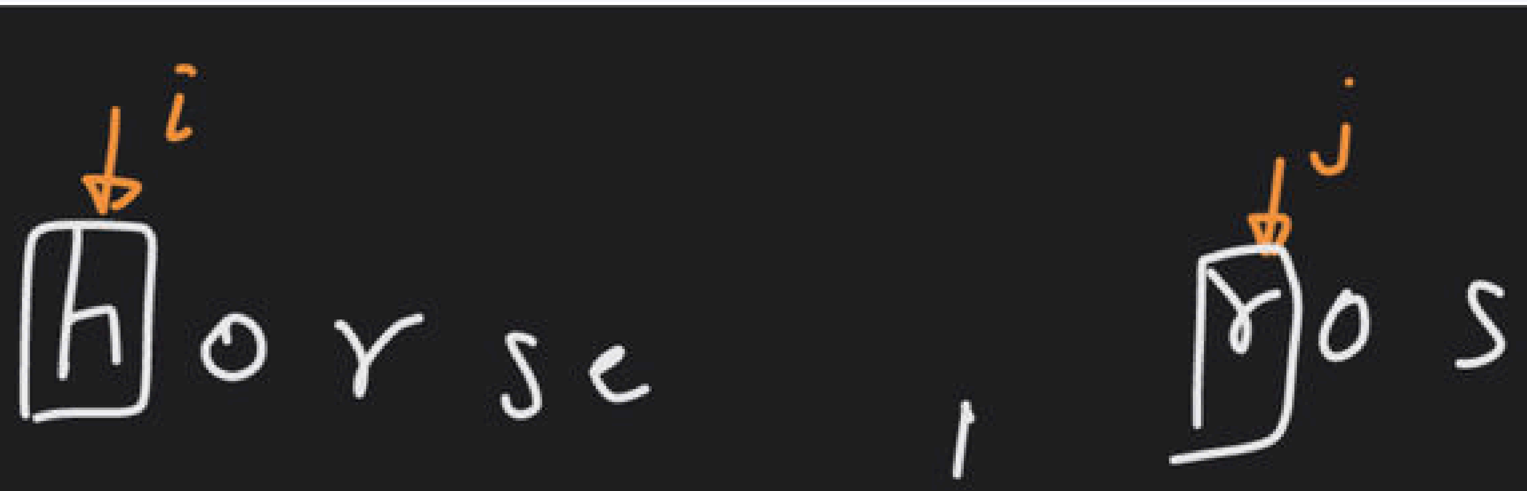
→ Edit Distance

I	D	R

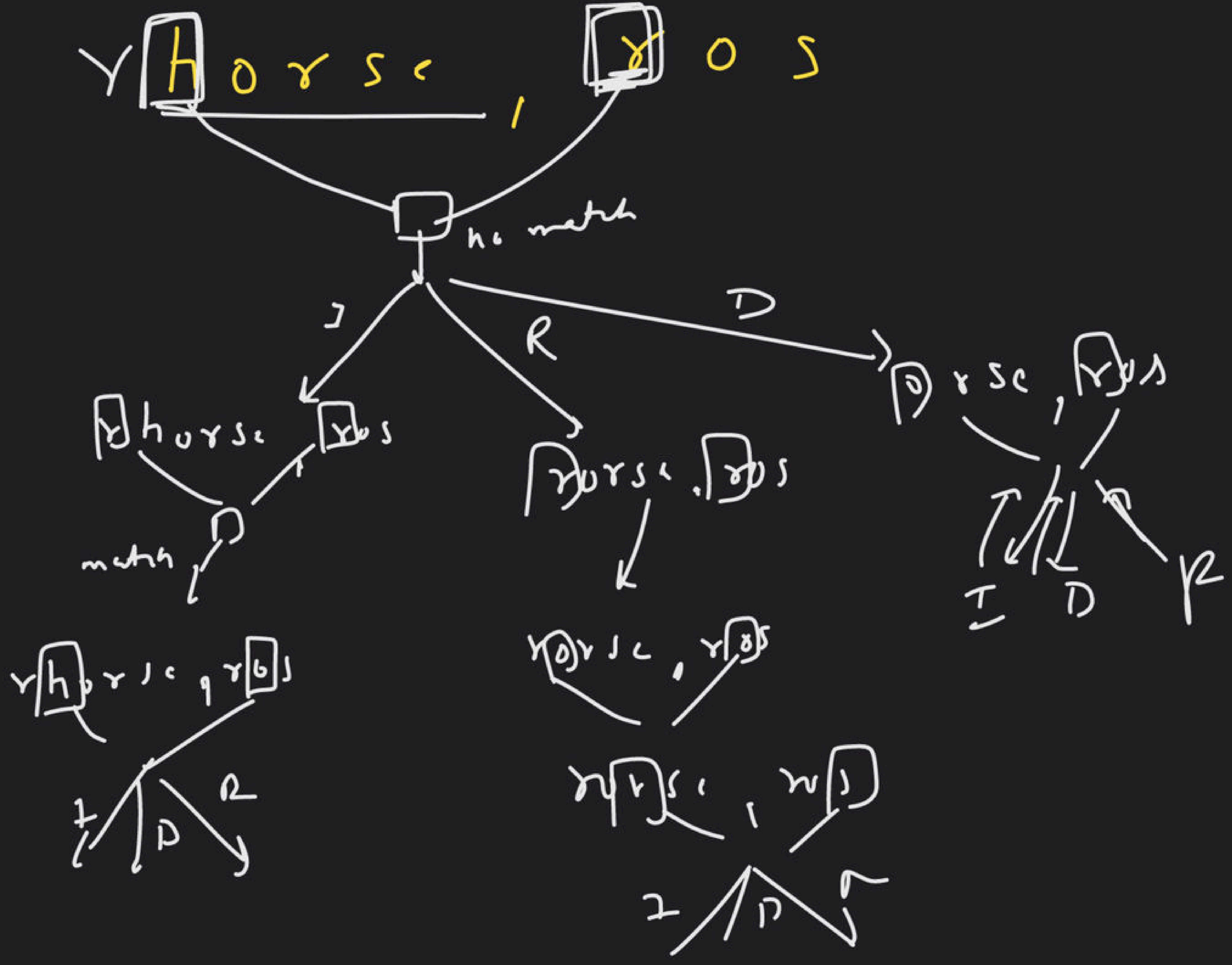
4 operation

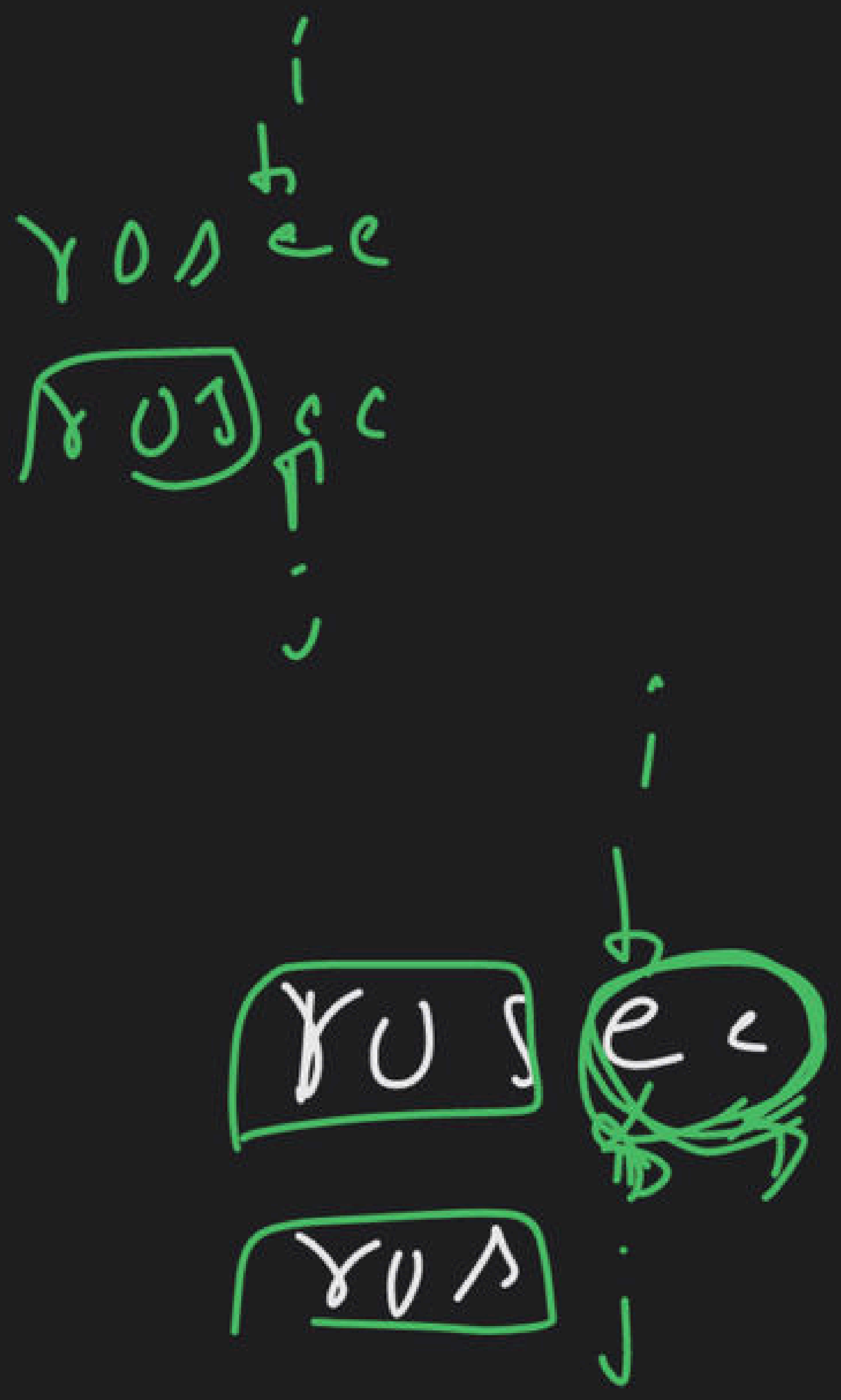


→ 3 operation



```
if (word1[i] == word2(j]) //match
{
    return sol(i+1, j+1)
}
else
{ //not match
        I     R     D
    min( _ , _ , _ ) , return
}
```


horse , ros
no match
I
R
D
rhorse rros
match
rhorse , rros
horse , ros
rorse , ros
orse , ros
I
D
R
I
D
R
I
D
R

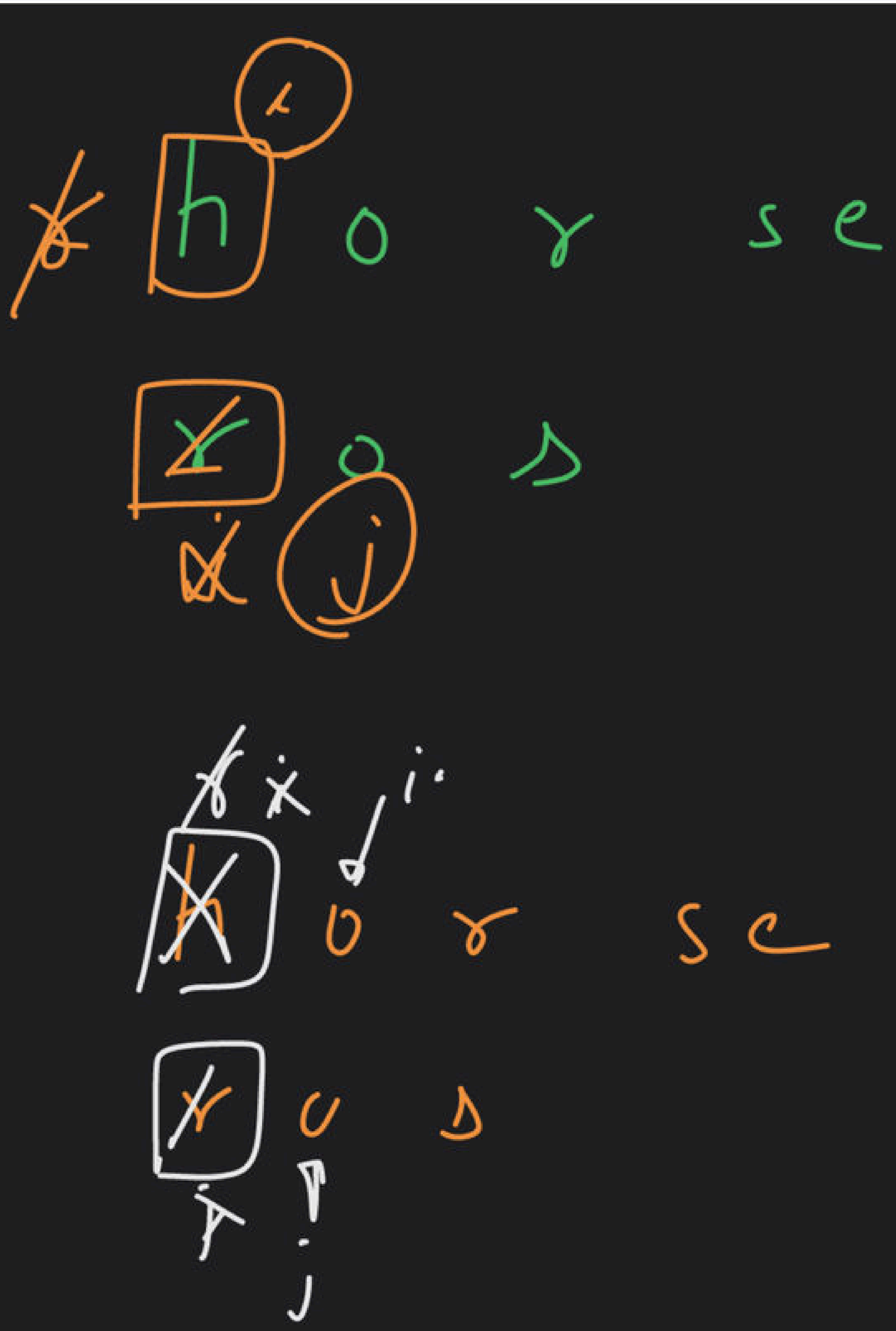


i → word1 → pencil

j → word2 → Khtm

```
if (j == word2.length())
    word1.length - i;
```

```
if (i == word1.length)
    word2.length - j
```

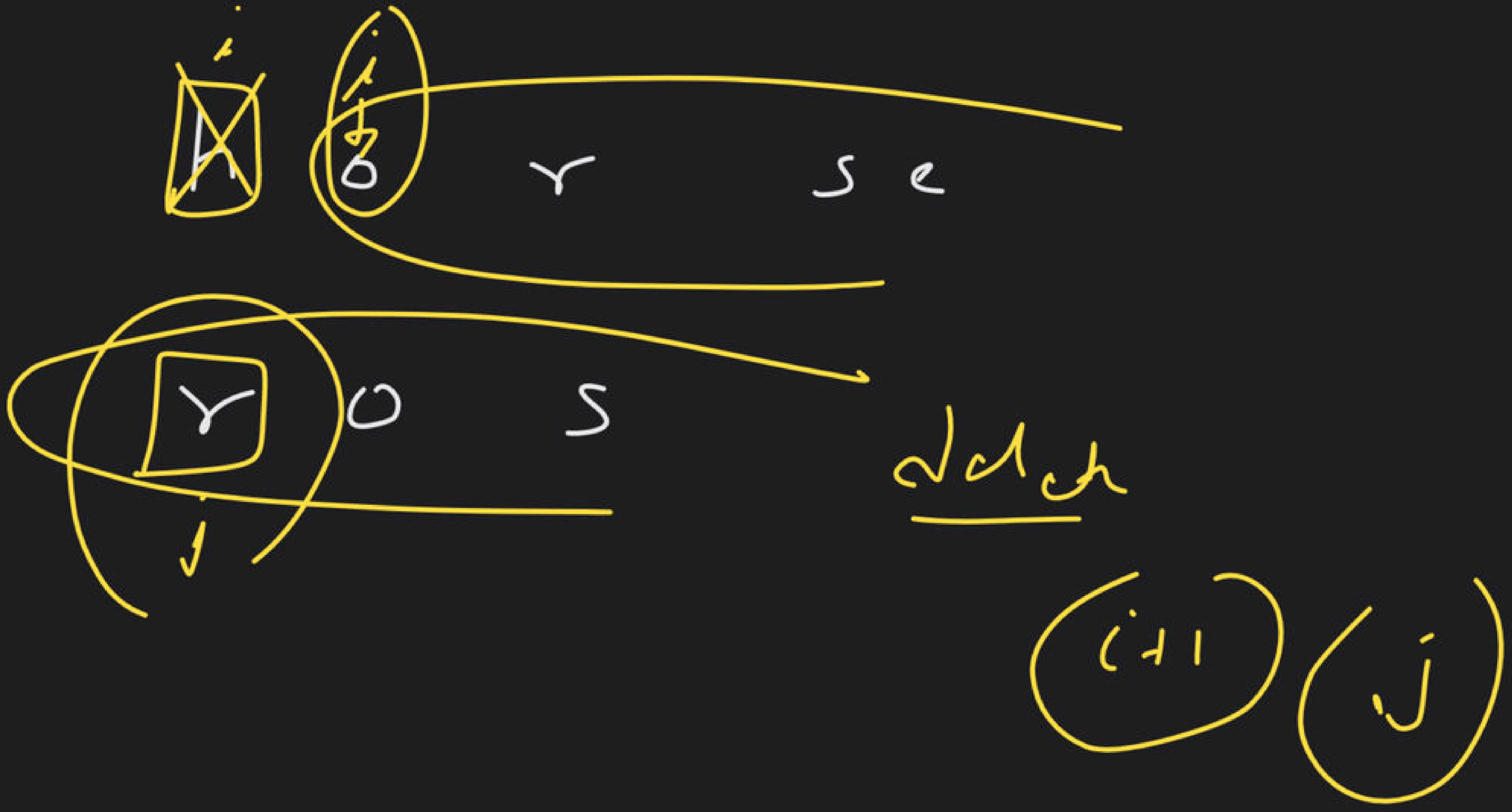


insert

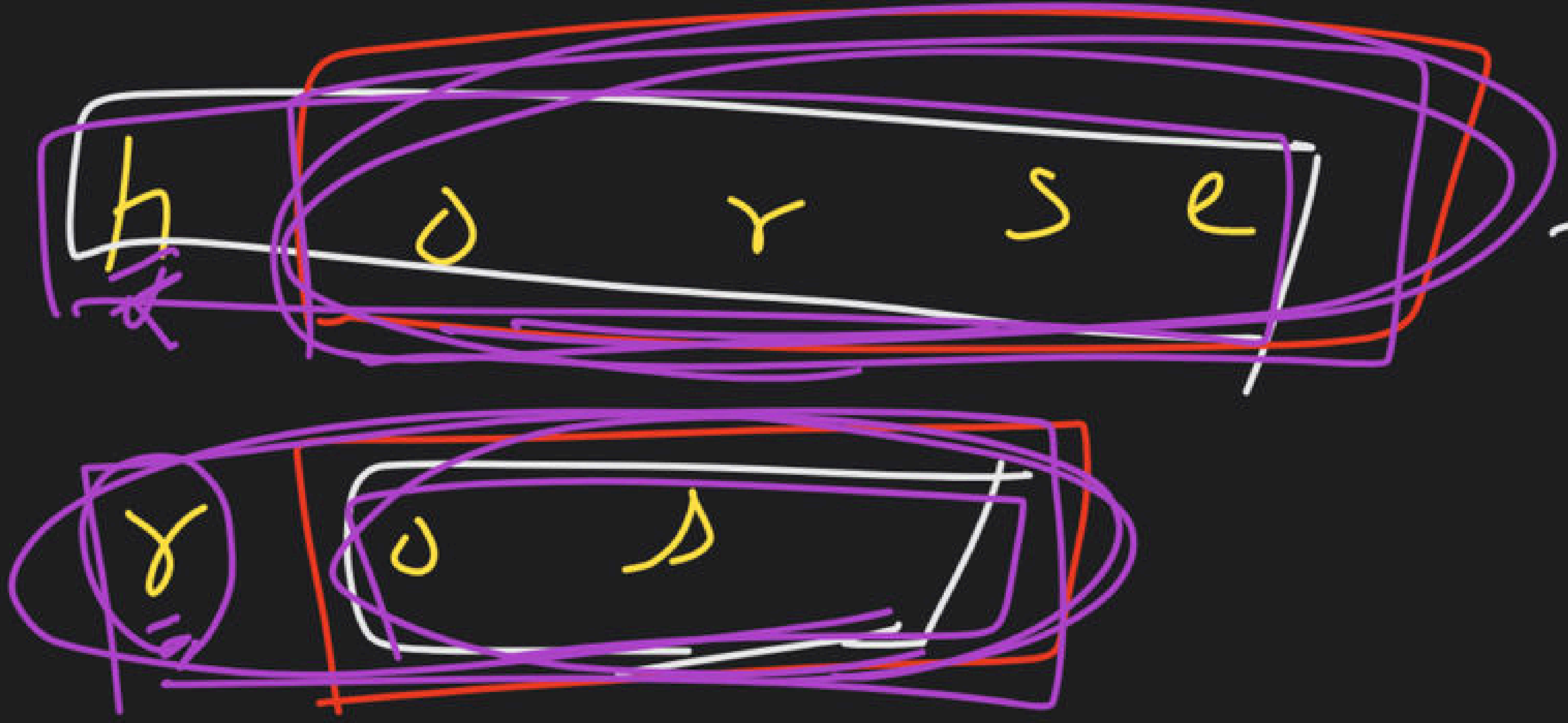
↳ i , j+1

replace

↳ i+1 j+1



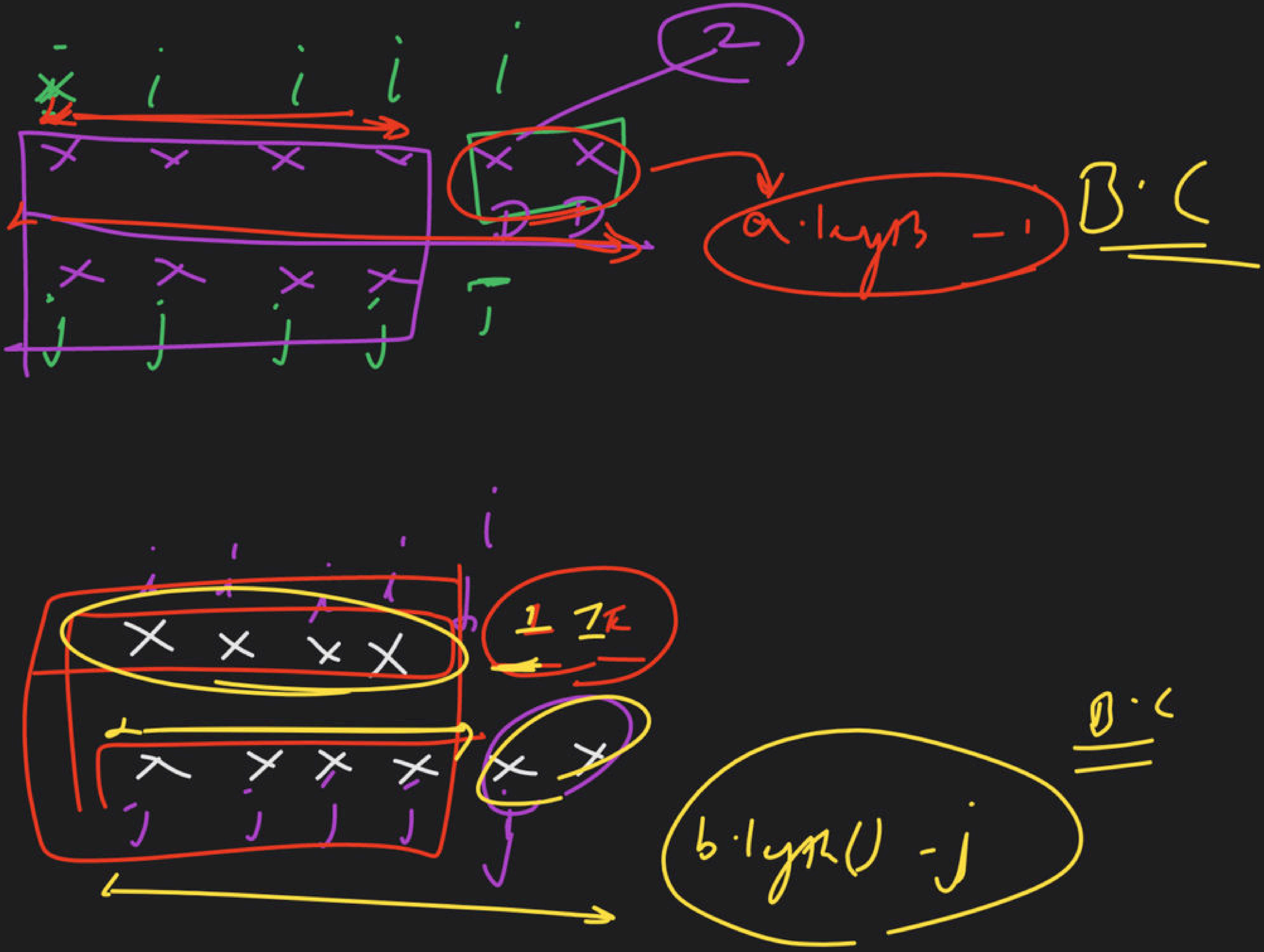
i
i
h o r s e
r o s
j
delete
i+1
j



insert → i, j+1

replace → i+1, j+1

delete → i+1, j



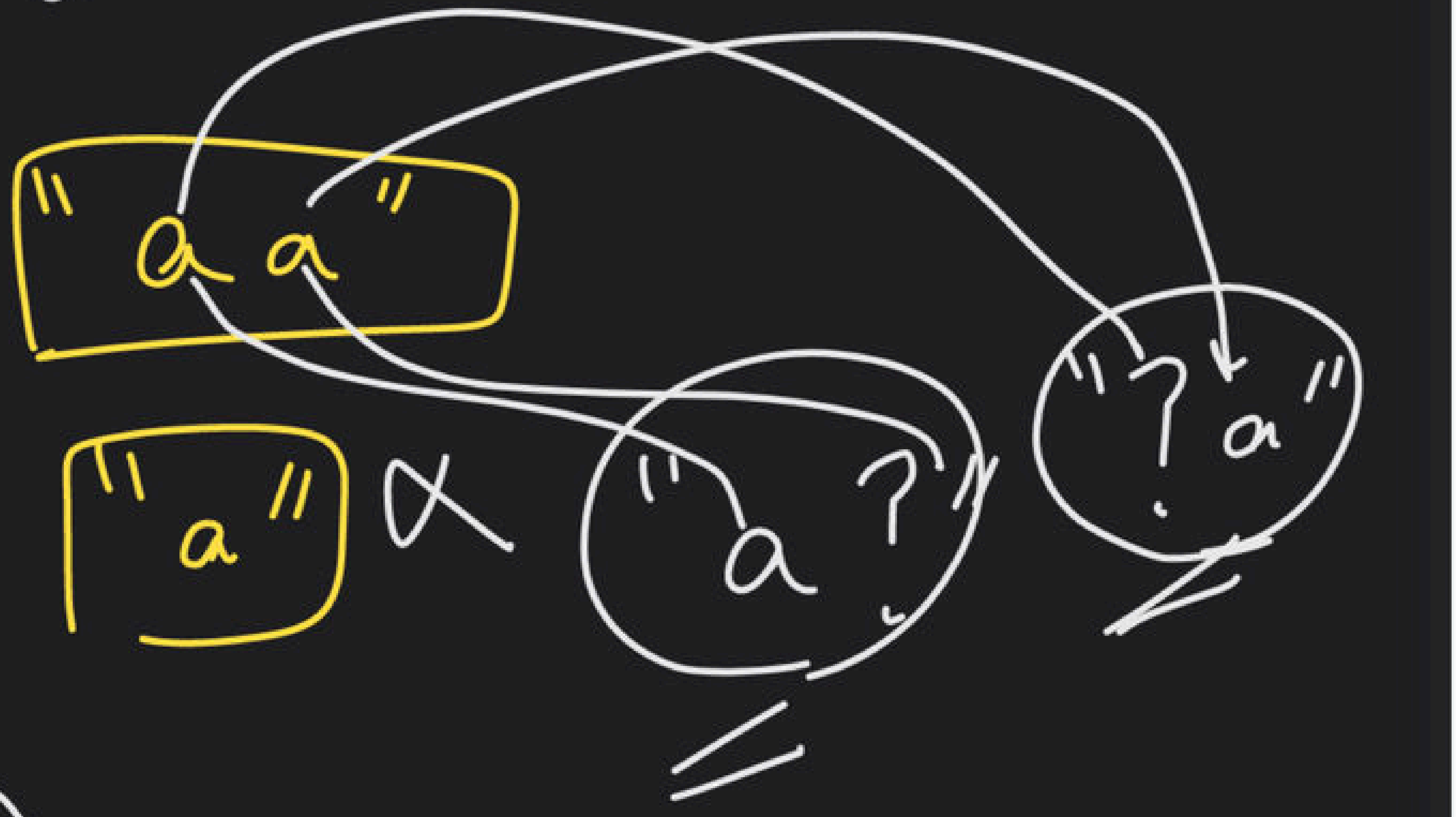
2
a.length - 1
B.C
1 1k
B.C
b.length() - j

→ WildCard Matching

i/p →

str → "aa"

pattern → "a" ✗

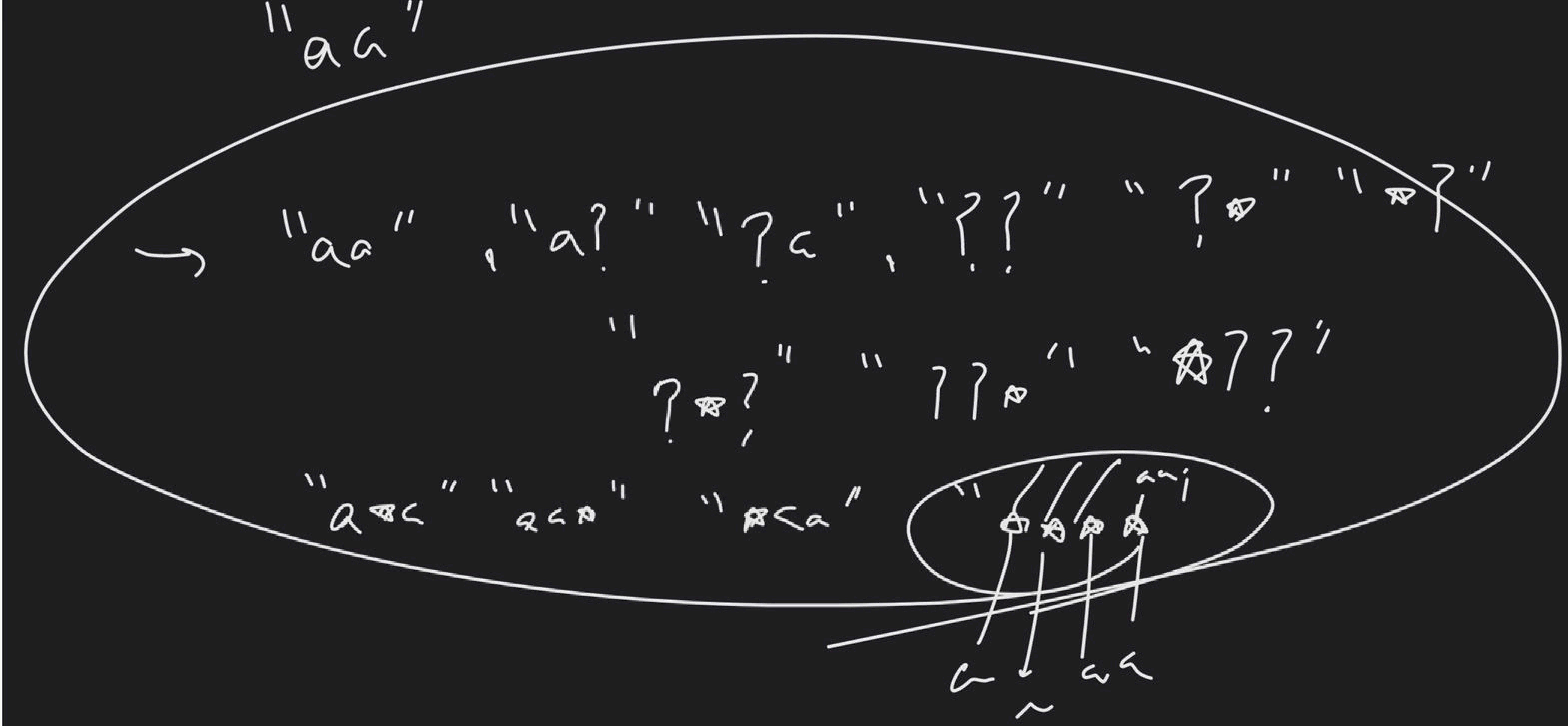


"aa" "aa" "*aa" ✗

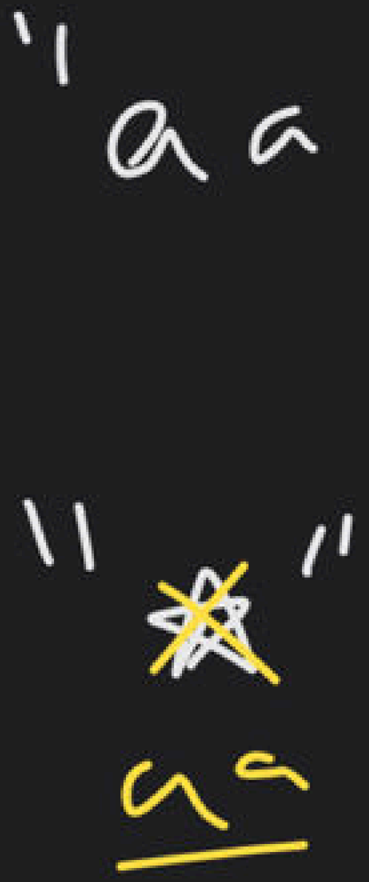
"*aa*" = "aa"

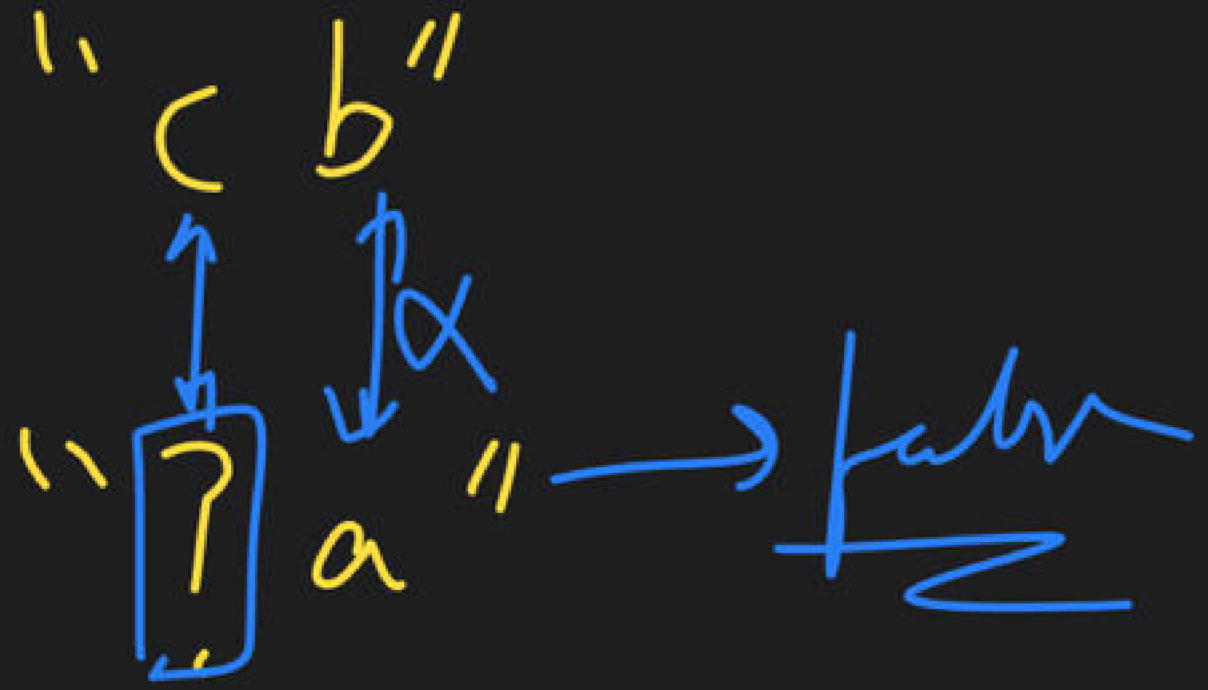
"*" "*a" = "*aa" = "aa"

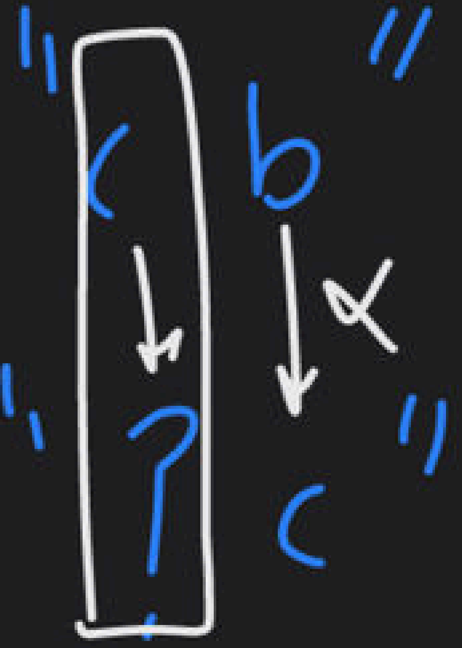
"a*" = "a*a" =



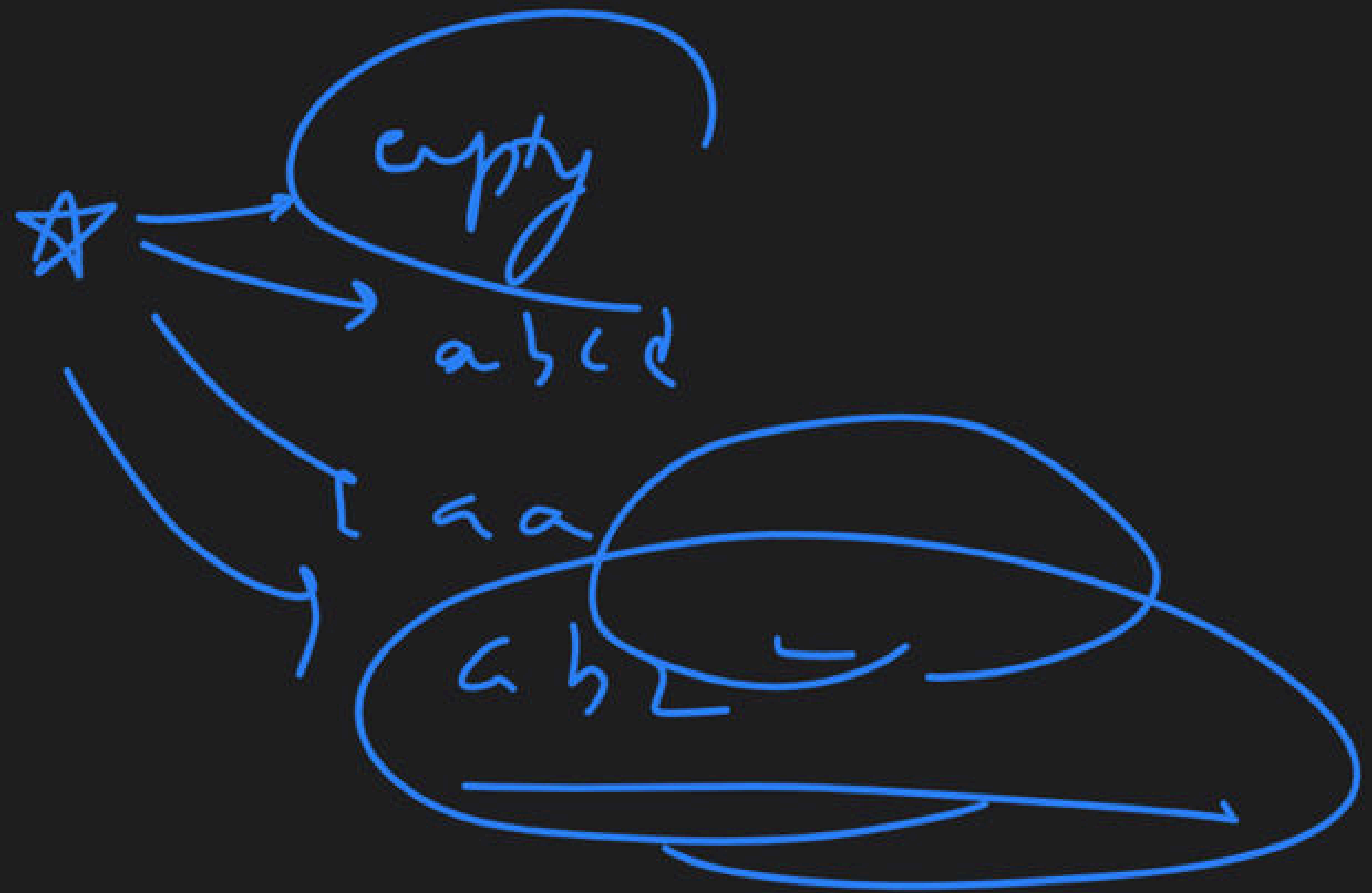
"aa"
"aa", "a?", "?a", "??", "?*", "*?"
"?*?", "??*", "*??"
"a*a", "aa*", "*aa"
"****"

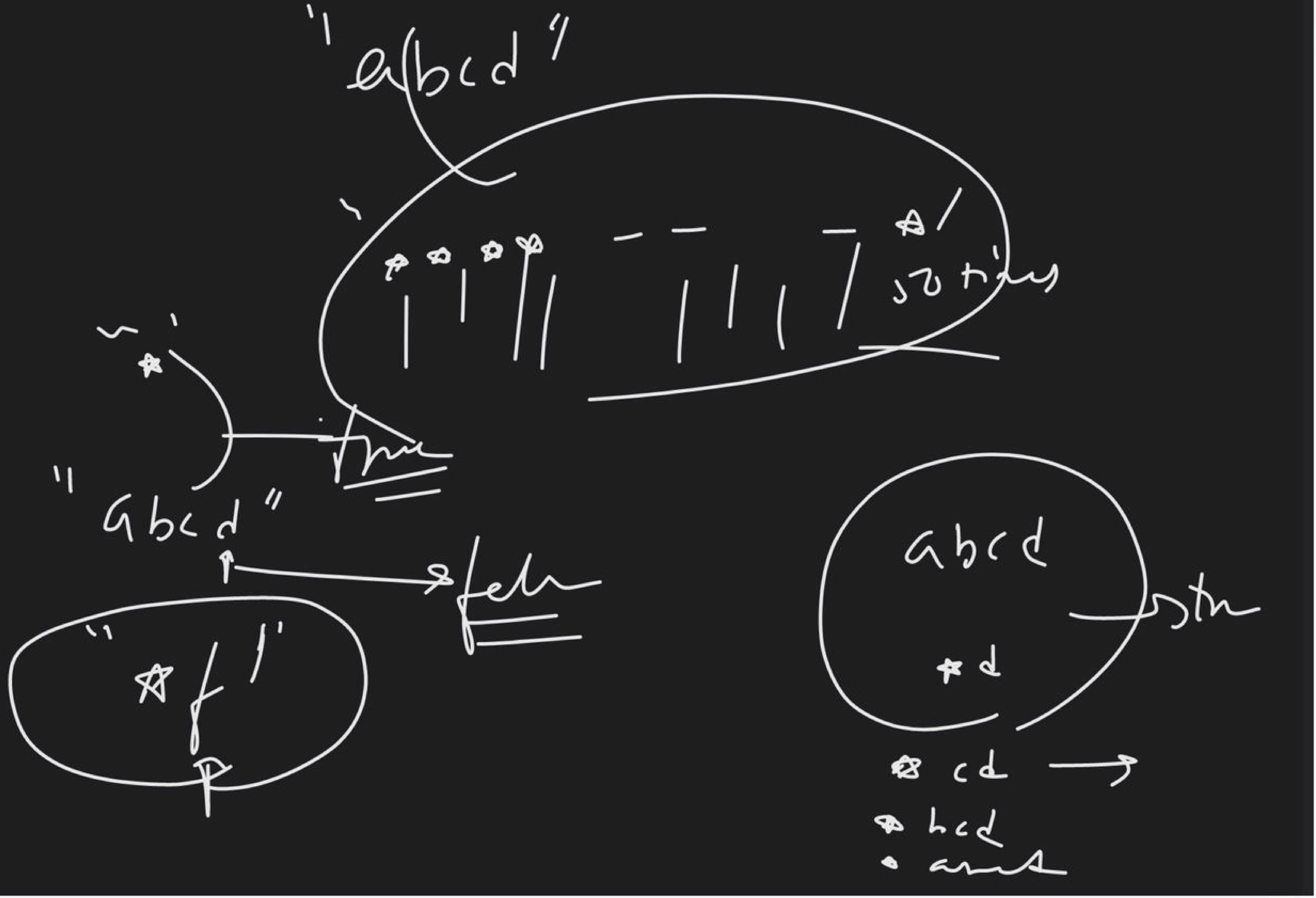

"aa"
"*"
aa

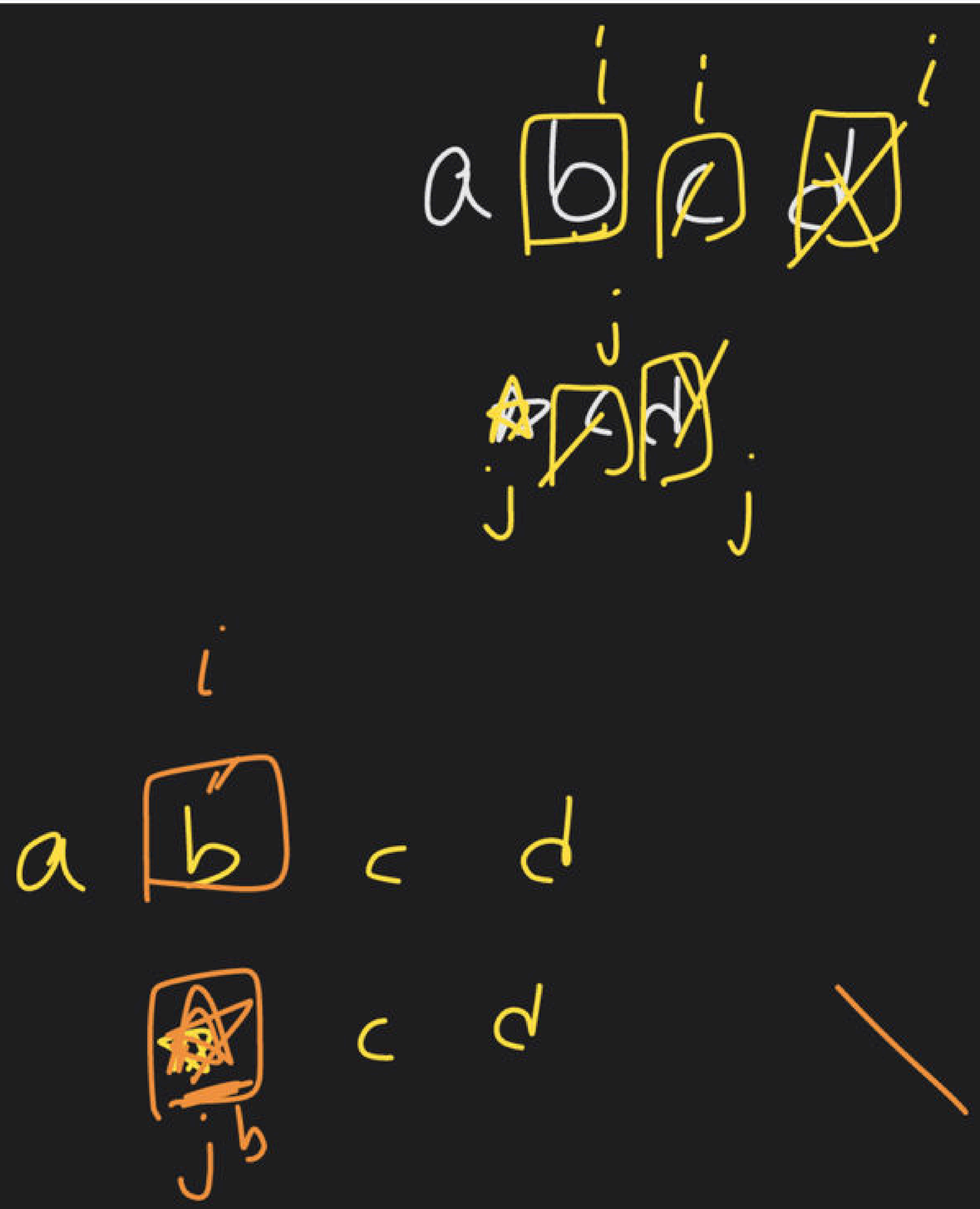

"cb"
"?a"
false


"cb"
"?c"

"abc"
"*"
ab

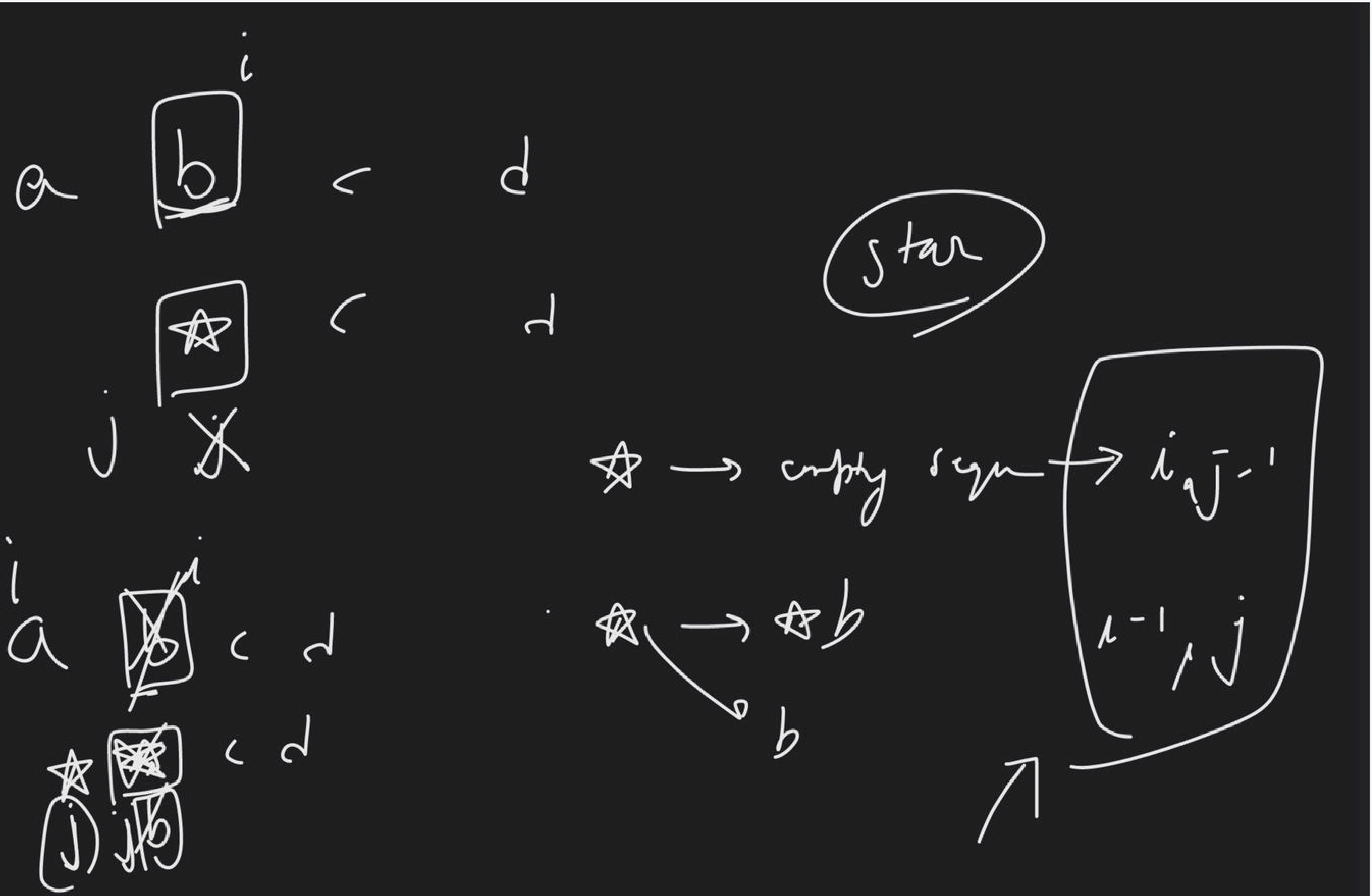

*
empty
abcd
aa
abc

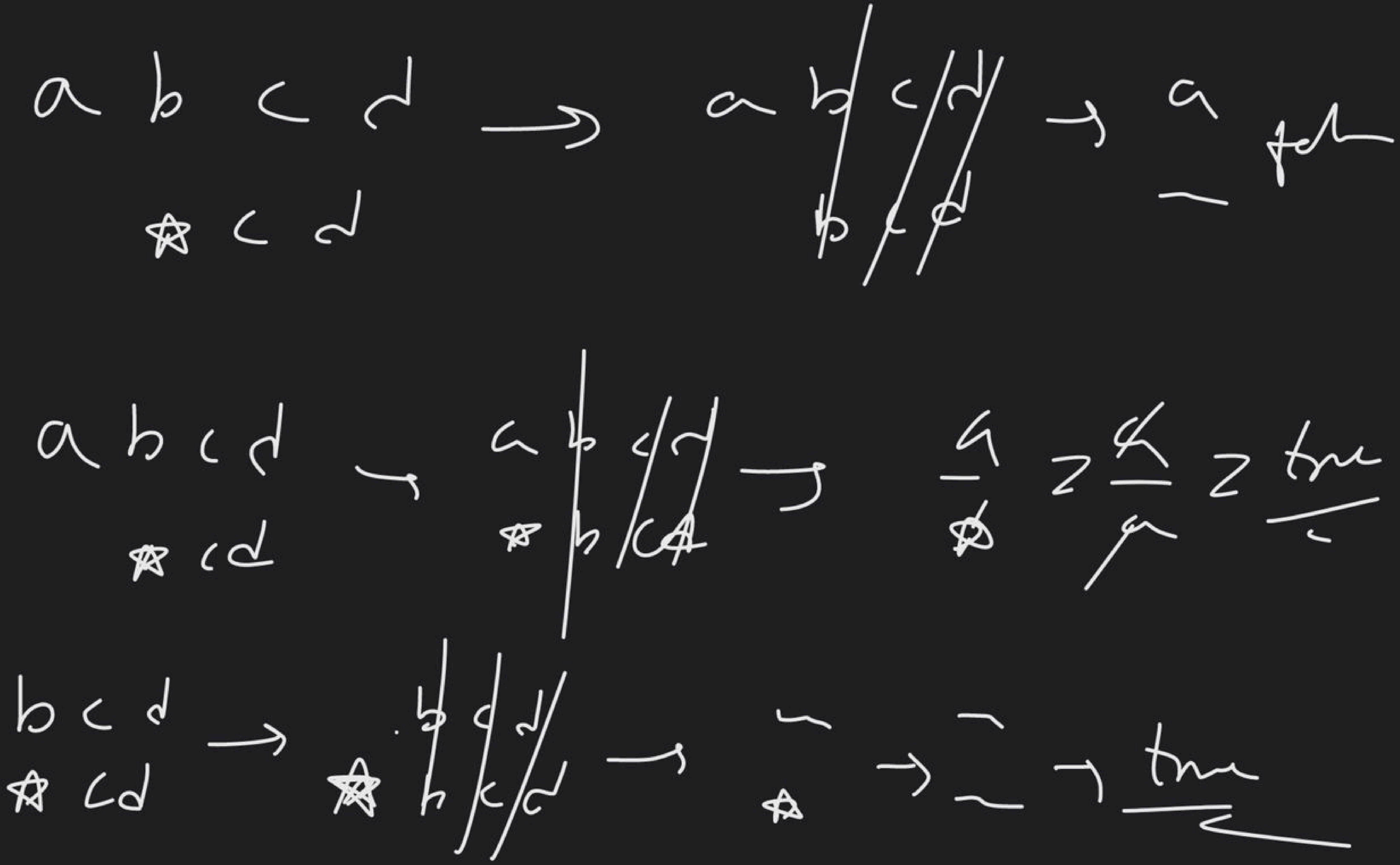

"abcd"
strings
true
"abcd"
false
"*f"
abcd
str
*d
*cd
*bcd
*abcd

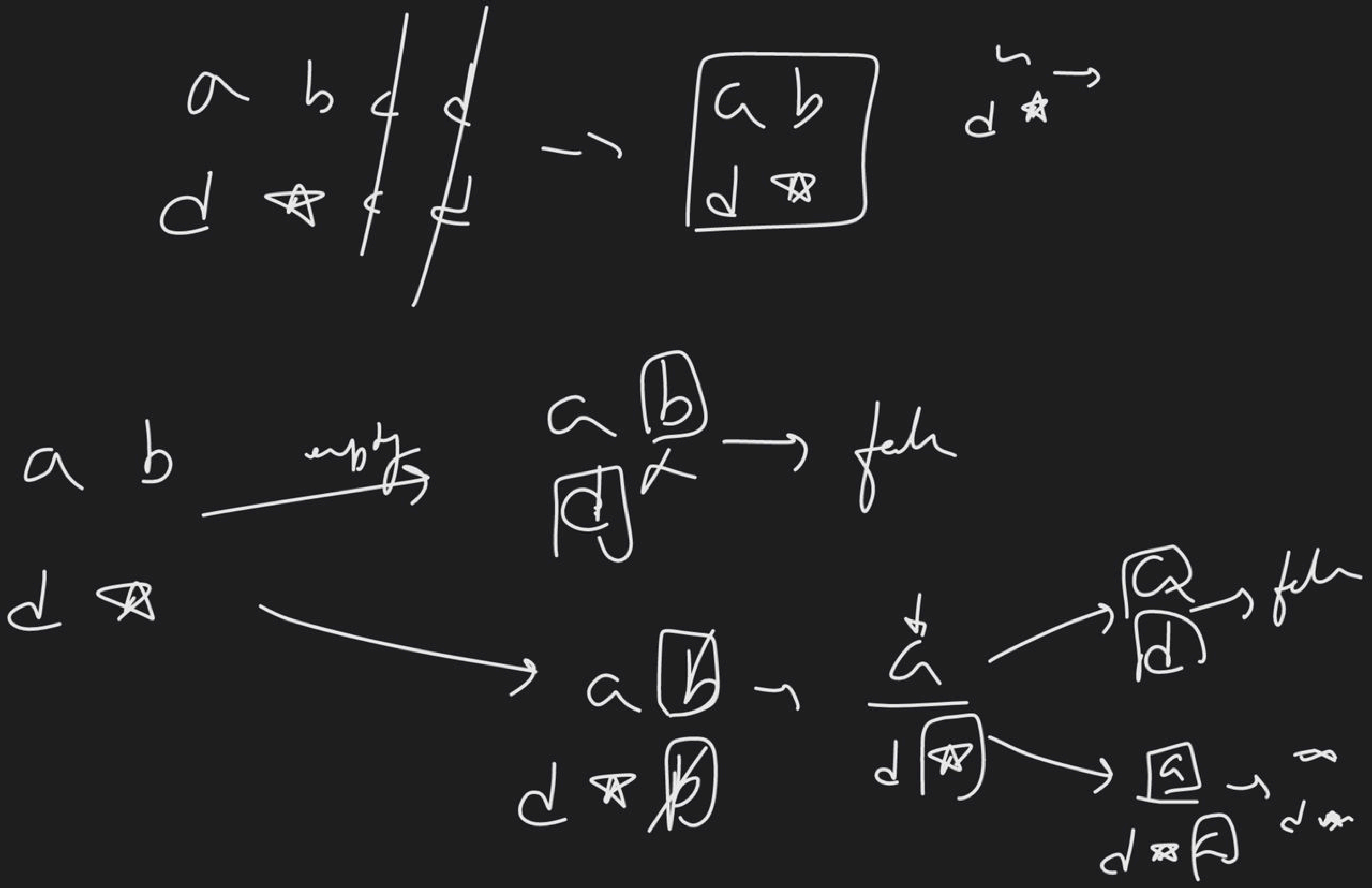


match -> i -> i-1
j -> j-1

star -> pattn me
star


i
a b c d
★ c d
j j
i i
a b c d
★ c d
j j b
star
★ → empty seqn → i, j-1
★ → ★b
b
i-1, j

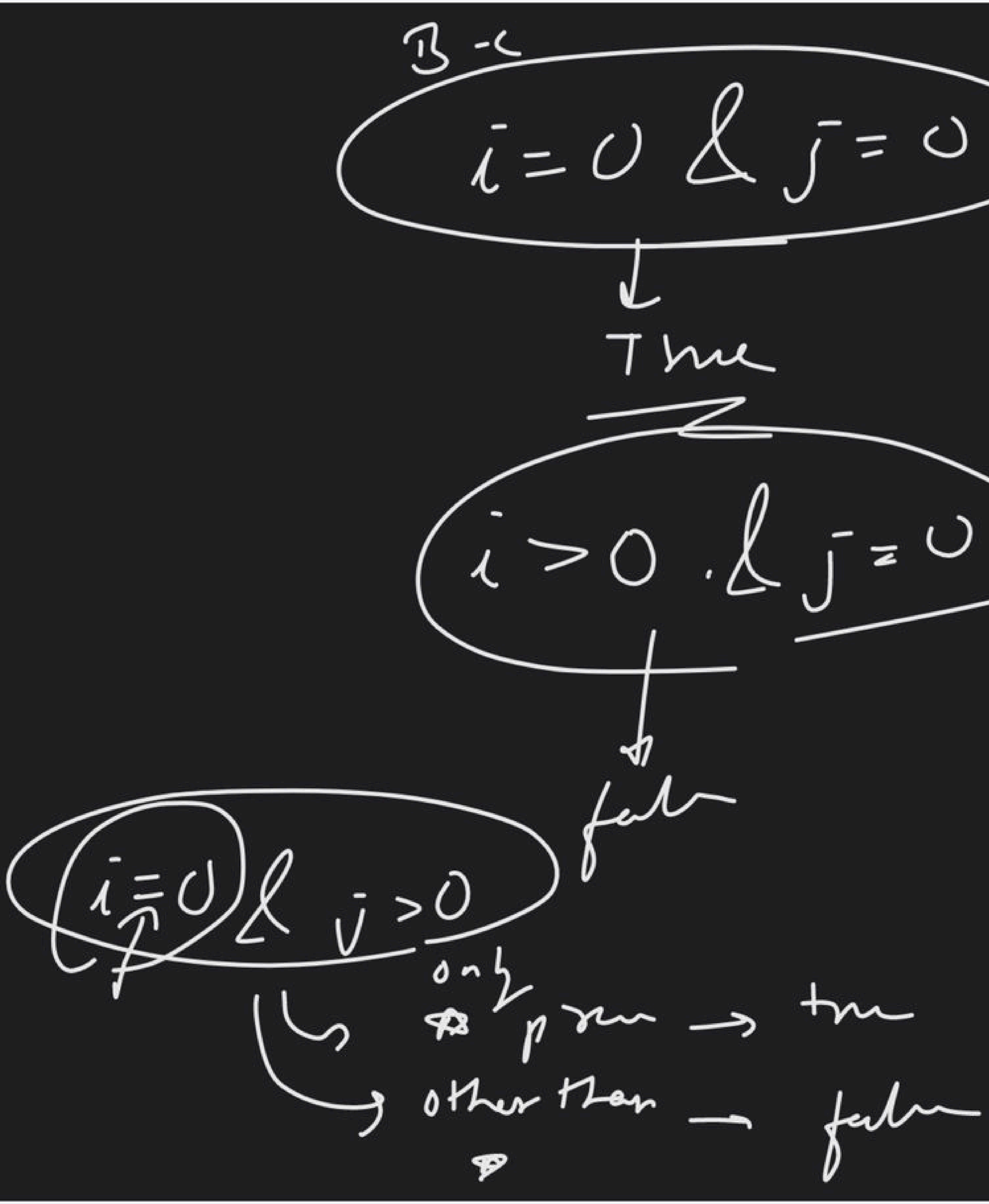

a b c d
☆ c d
a b c d
b c d
a
false
a b c d
☆ c d
a b c d
☆ b c d
a
☆
z
z
true
b c d
☆ c d
b c d
☆ b c d
☆
true


a b c d
d ☆ c c
a b
d ☆
d ☆
a b
d ☆
empty
a b
d
fail
a b
d ☆ b
a
d ☆
a
d
fail
a
d ☆ a
∞
d ☆

d
d~
B-C
false

~~a b c d~~

~~ab bc cd~~


B-C
i=0 & j=0
True
i>0 & j=0
false
i=0 & j>0
only
p ☆ present → true
other than ☆ → false

Basic Case

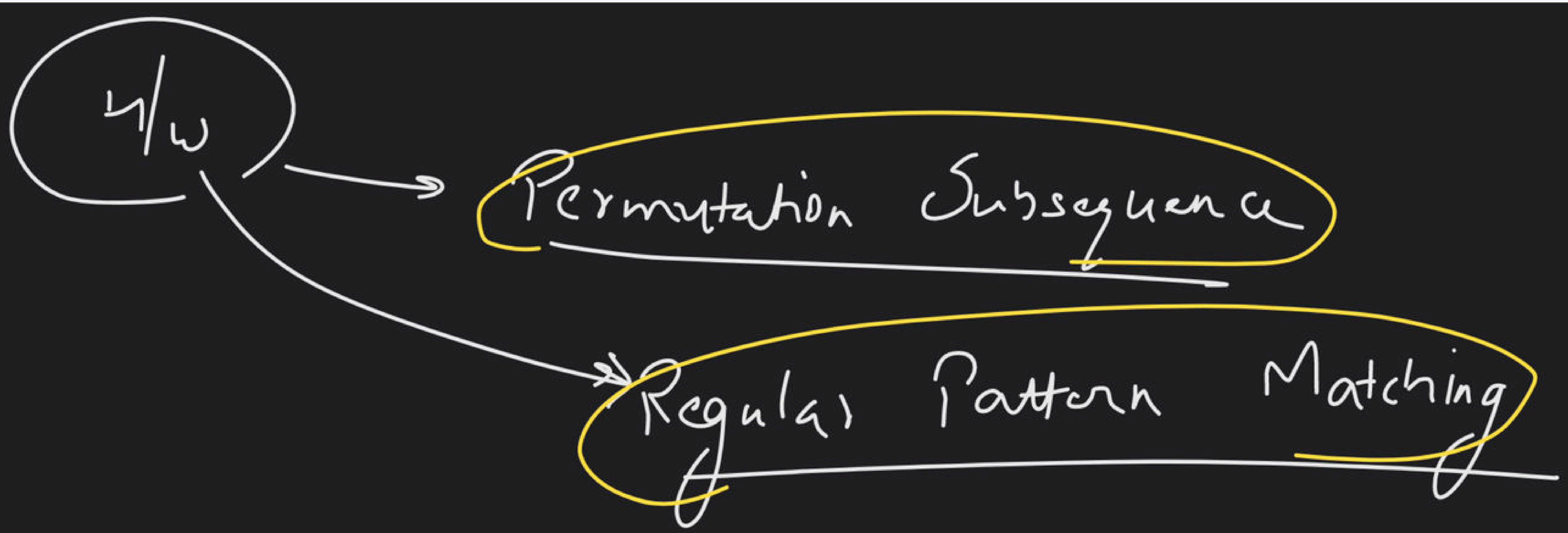
1. $i == 0$ && $j == 0$ → true

2. $i > 0$ && $j == 0$ → false

3. $i == 0$ && $j > 0$
   - * → only *
     - → true
   - other than star → false

i → a → str → l——

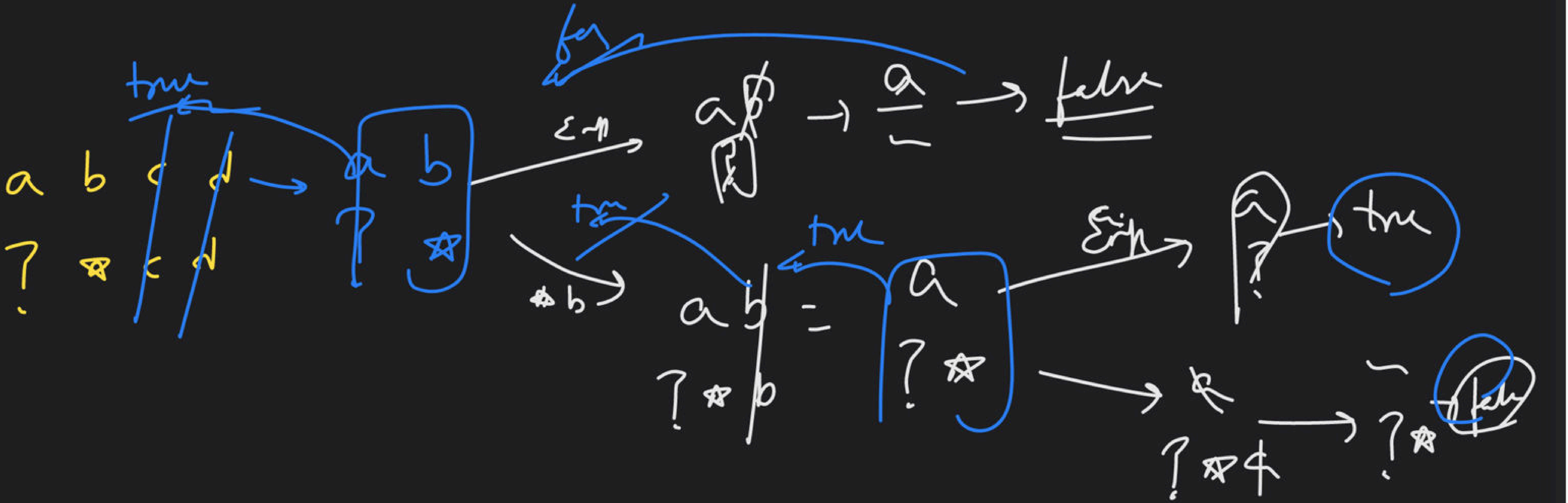
j → b → str → len

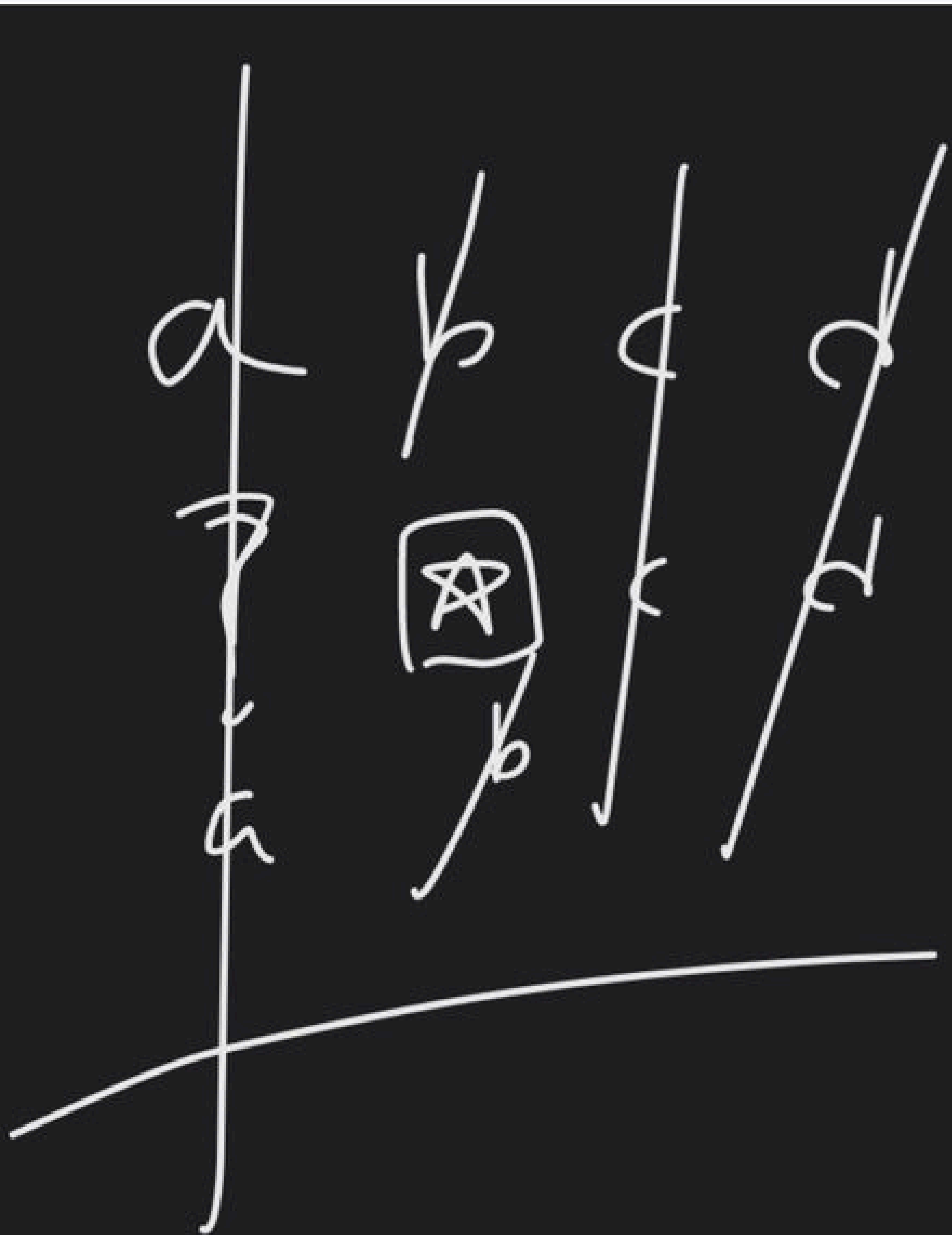


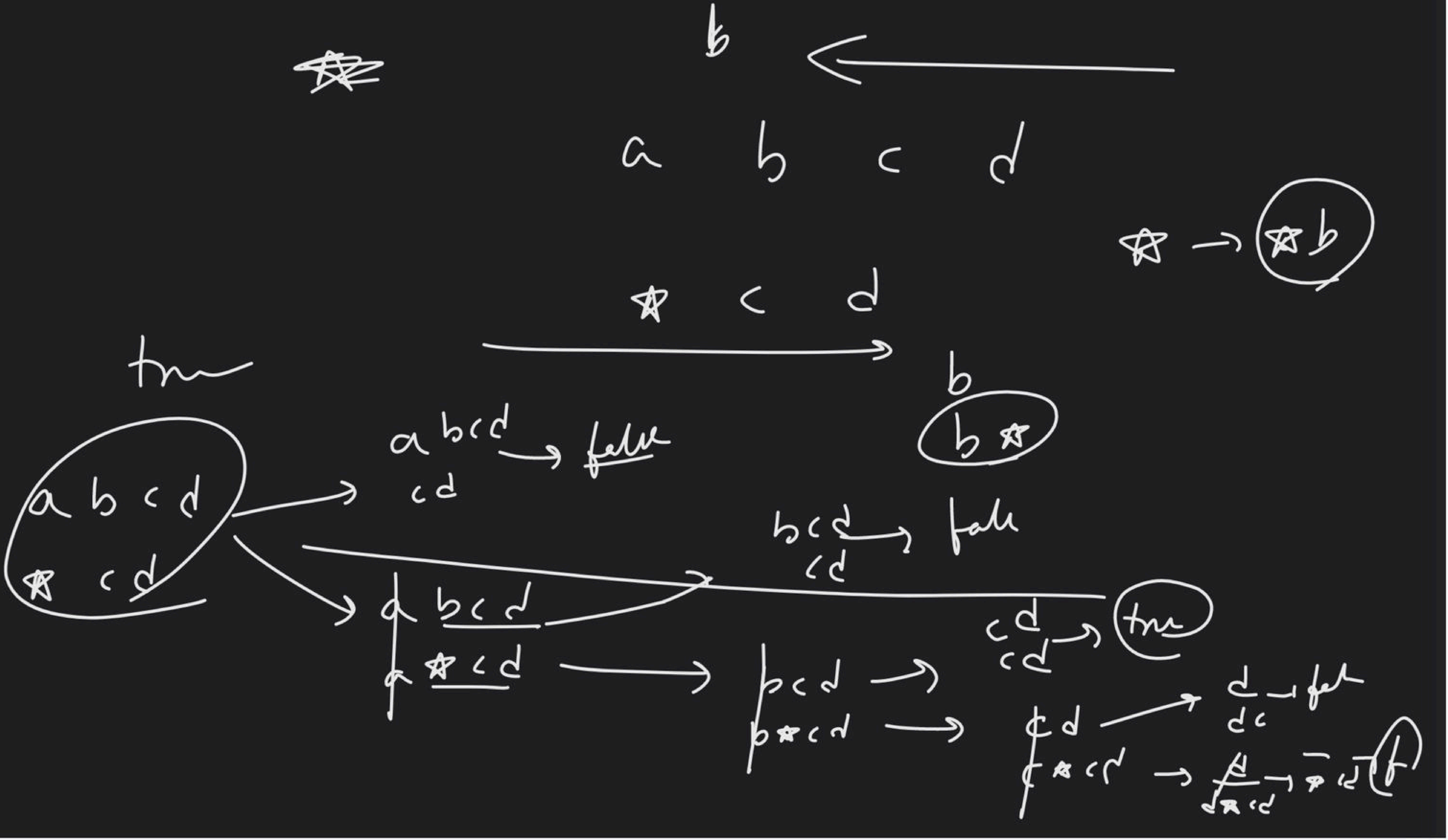
H/W

Permutation Subsequence

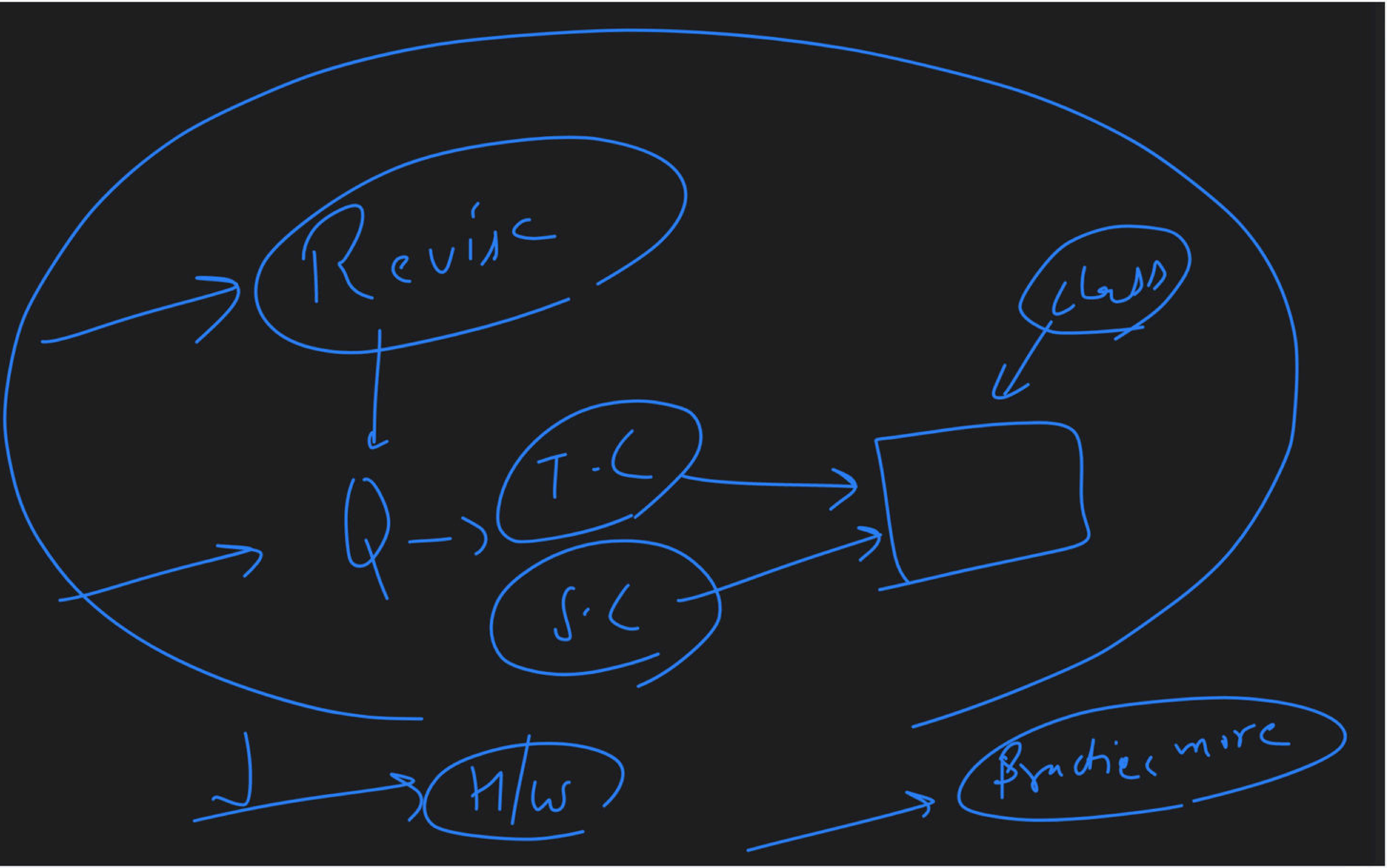
Regular Pattern Matching


a b c d
? ★ c d
true
a b
? ★
false
ε→p
a b
a
false
true
★b
true
a b
? ★ b
=
a
? ★
ε;
ε→p
a
?
true
? ★ b
? ★
false


a
b
c
d
c
d
b
a


b
★
a b c d
★ c d
★ → ★b
b
b★
true
abcd
★ cd
abcd
cd
→ false
bcd
cd
→ fall
cd
cd
→ true
abcd
a★cd
bcd
b★cd
cd
c★cd
d
dc
→ fal
d
d★cd
→ ★cd
f

2 min break


Revise
Q
T.C
S.C
class
H/W
Practice more