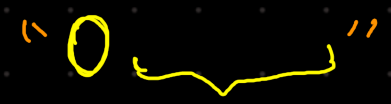
Recursion and Strings

⇓

making a problem small and solving that

Strings :- characters are stored.



remove z

"dragon ball z" ⇒
'good morning333' ⇒ 'good morning'

str

① Base case
if (len(str) == 0)
return str

② recursion work
str[1:]

③ our work

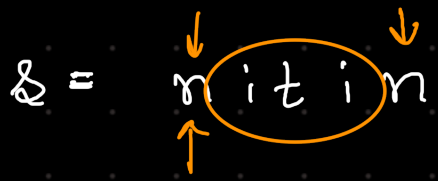
'abcz'
⇓
abc

'zandz'
'and'

Palindrome

nitin ←

→ malayalam



palindrome(s) = s[0] == s[len-1] and palindrom[s[1:len-1]]

① Base case

(len(s) <= 1):

return True

"" 'a' 'ab'

② recursive call

len(s) = 5

s[1:len(s)-1]

③ Your work s[0] s[len(s)-1]

Medium to Hard Level

→ don't run behind solution

→ in-depth understanding and intuition

array

string

abc

subarray

substring
(same order and continous)

a b c
ab bc ~~ac~~
abc

$\frac{n(n+1)}{2}$

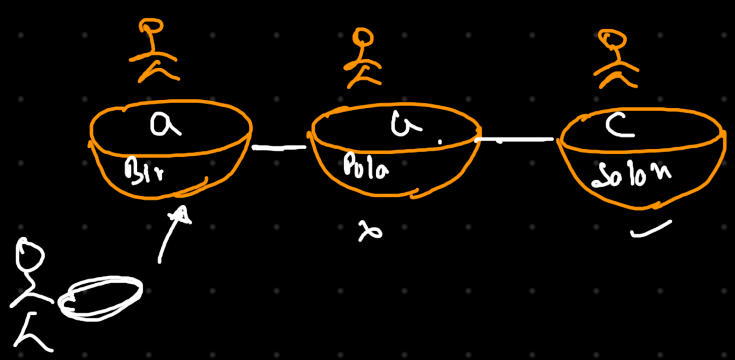
subset

subsequences
(order maintained can be non-continous)

a b c
ab bc ac
abc ϕ

2^n

Hostel



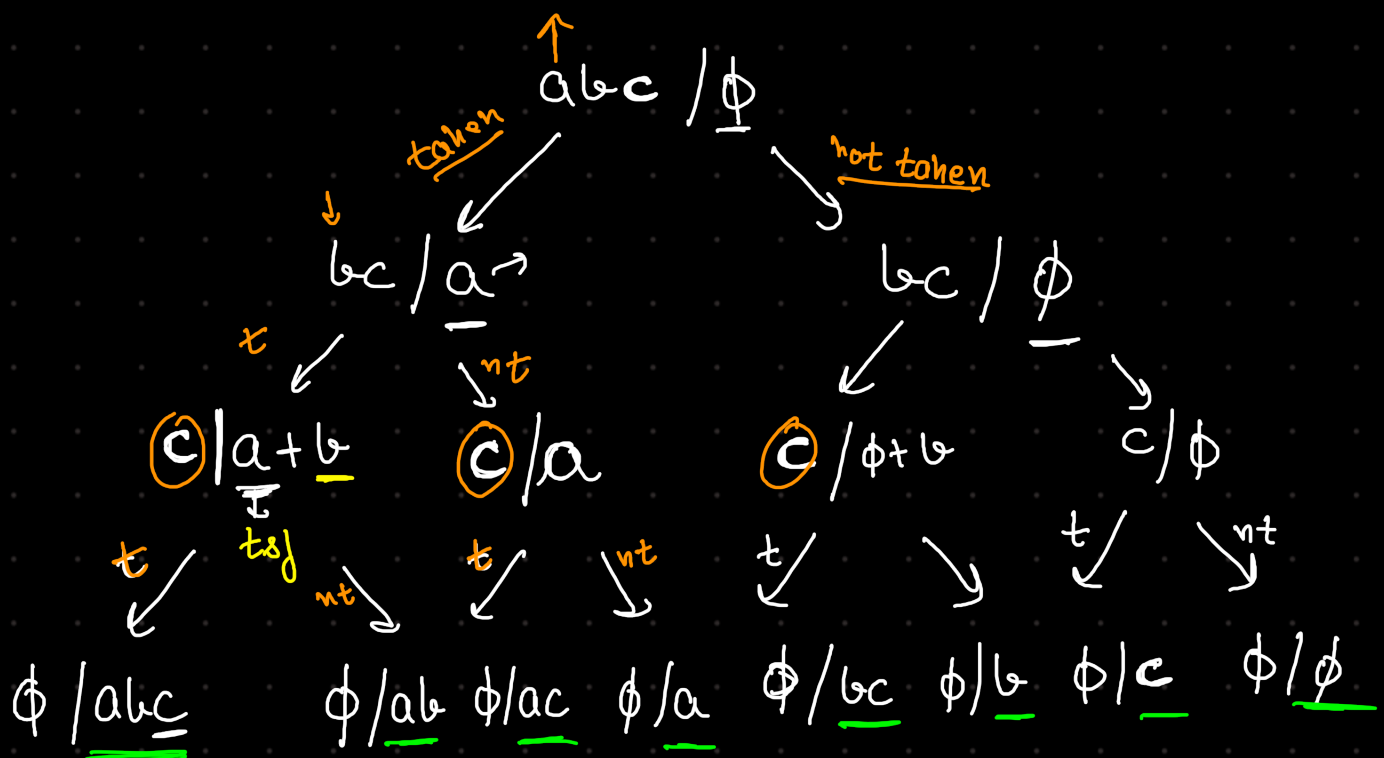
buffet

Subsequences of given string

abc → a b c
ab bc ca
abc ϕ

taking and not-taking → subsequence

Print all subsequence



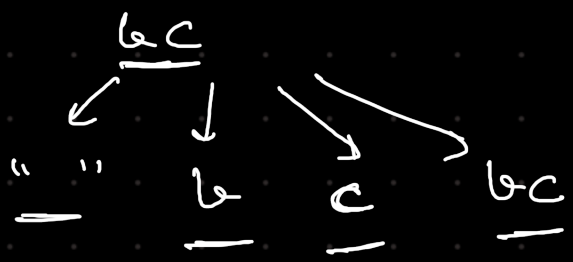
ϕ
a b c
ab bc ac
abc

Return as a list

create an ans = [ ]

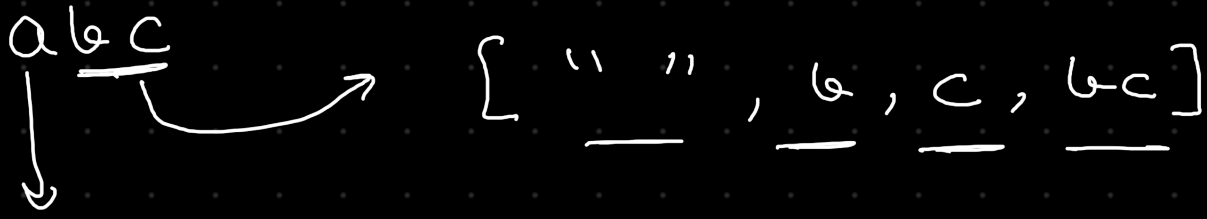
extend with other ans returned from recursion.

"" → ""



Subsequences of bc

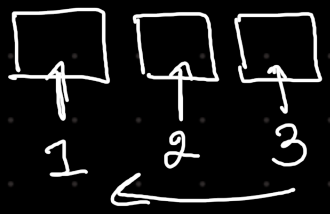
""	→ ""		a
c	→ c	abc	ac
bc	→ b		ab
	→ bc		abc



a ab ac abc

Permutations

abc → abc bac bca
cab cba acb



$n!$ $len(s) = n$

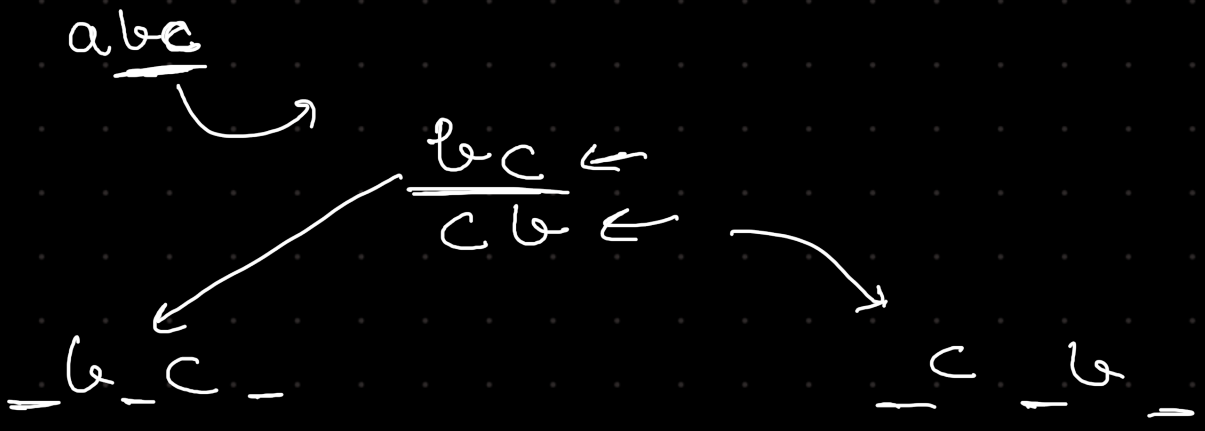
abc

abc | ϕ
1 b
bc | a

c | ba c | ab

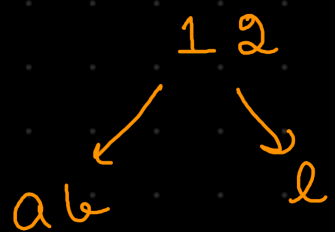
ϕ | cba ϕ | bca ϕ | bac ϕ | cab ϕ | acb ϕ | abc


1
2
2

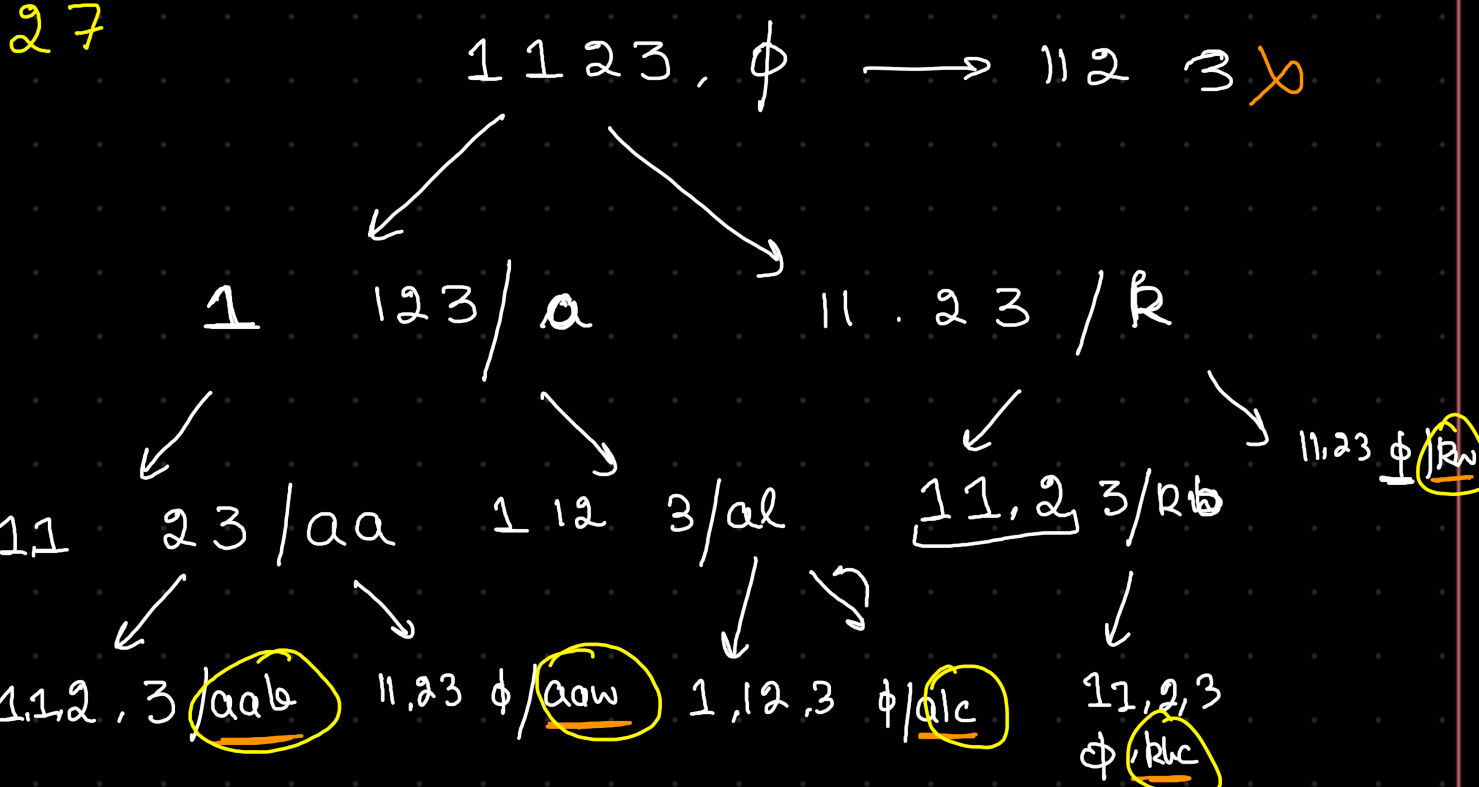

abc
bc
cb
_b_c_
_c_b_

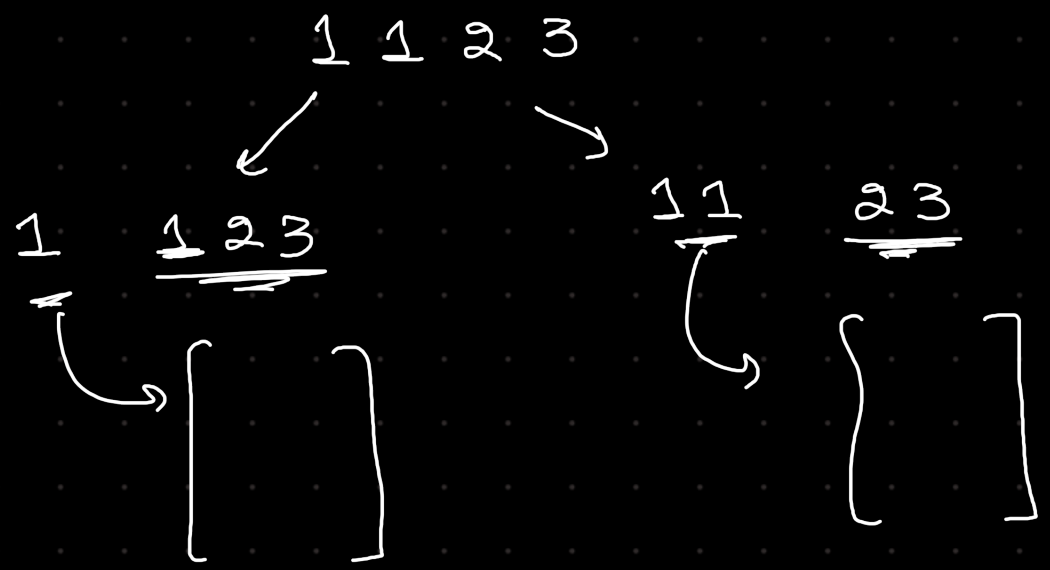
Print all Combinations

1 → 'a'
2 → 'b'
⋮
26 → 'z'



27 → bg




1 1 2 3
1
1 2 3
1 1
2 3

Letter Combination of a phone number

2 → 'a' 'b' 'c'

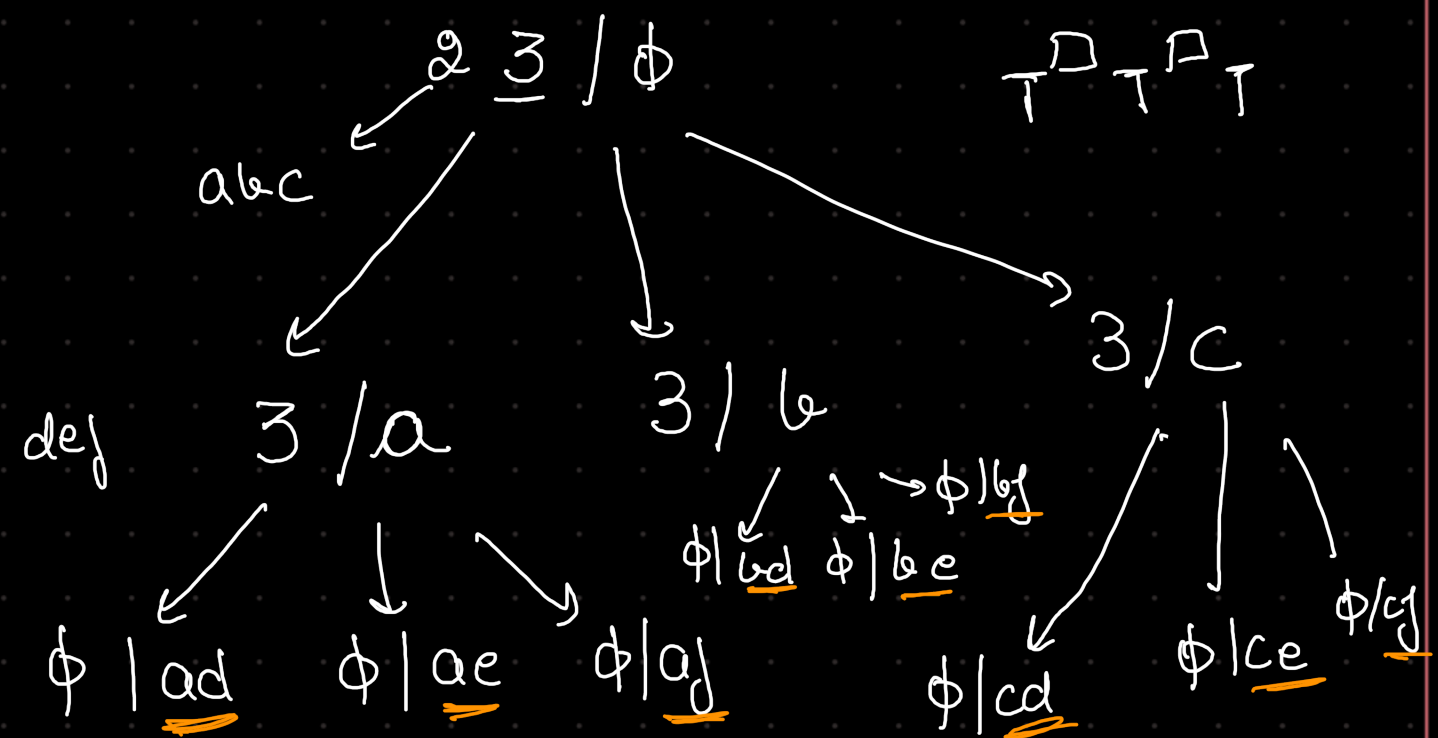
23 → 'a' 'b' 'c' → 'd' 'e' 'f'

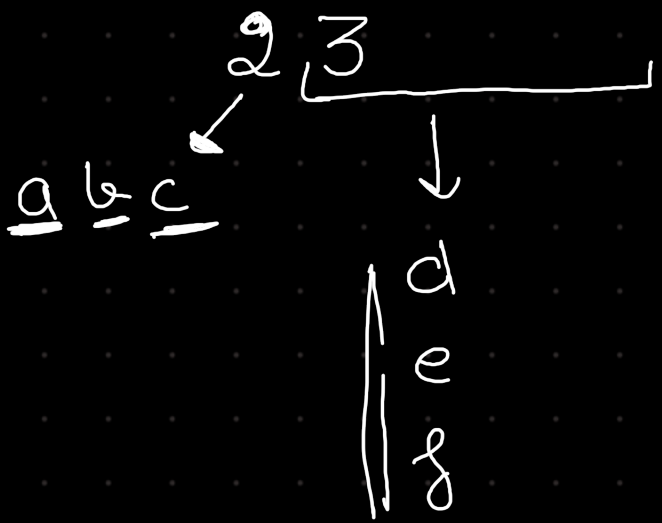
ad bd cd
ae be ce
af bf cf

223 / osf

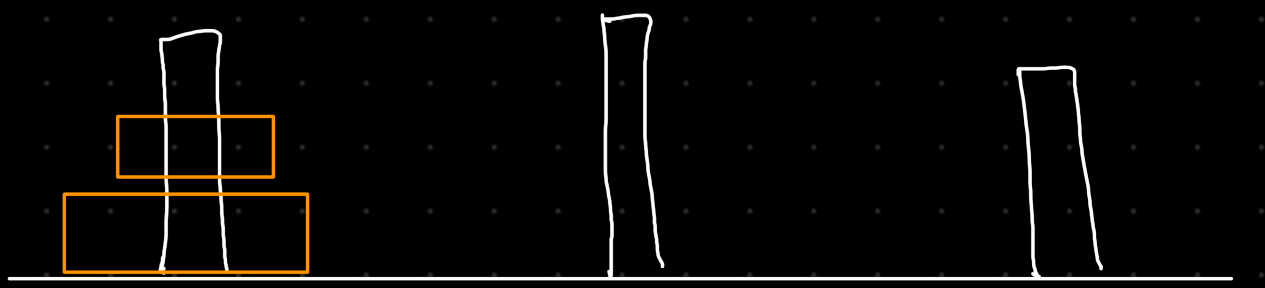
↓

a o


2 3 / ϕ
abc
def
3 / a
3 / b
3 / c
ϕ | ad
ϕ | ae
ϕ | af
ϕ | bd
ϕ | be
ϕ | bf
ϕ | cd
ϕ | ce
ϕ | cf


2 3
a b c
d
e
f

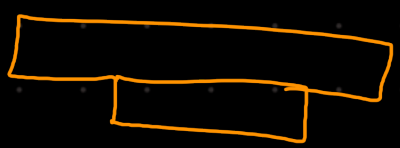
Tower of hanoi



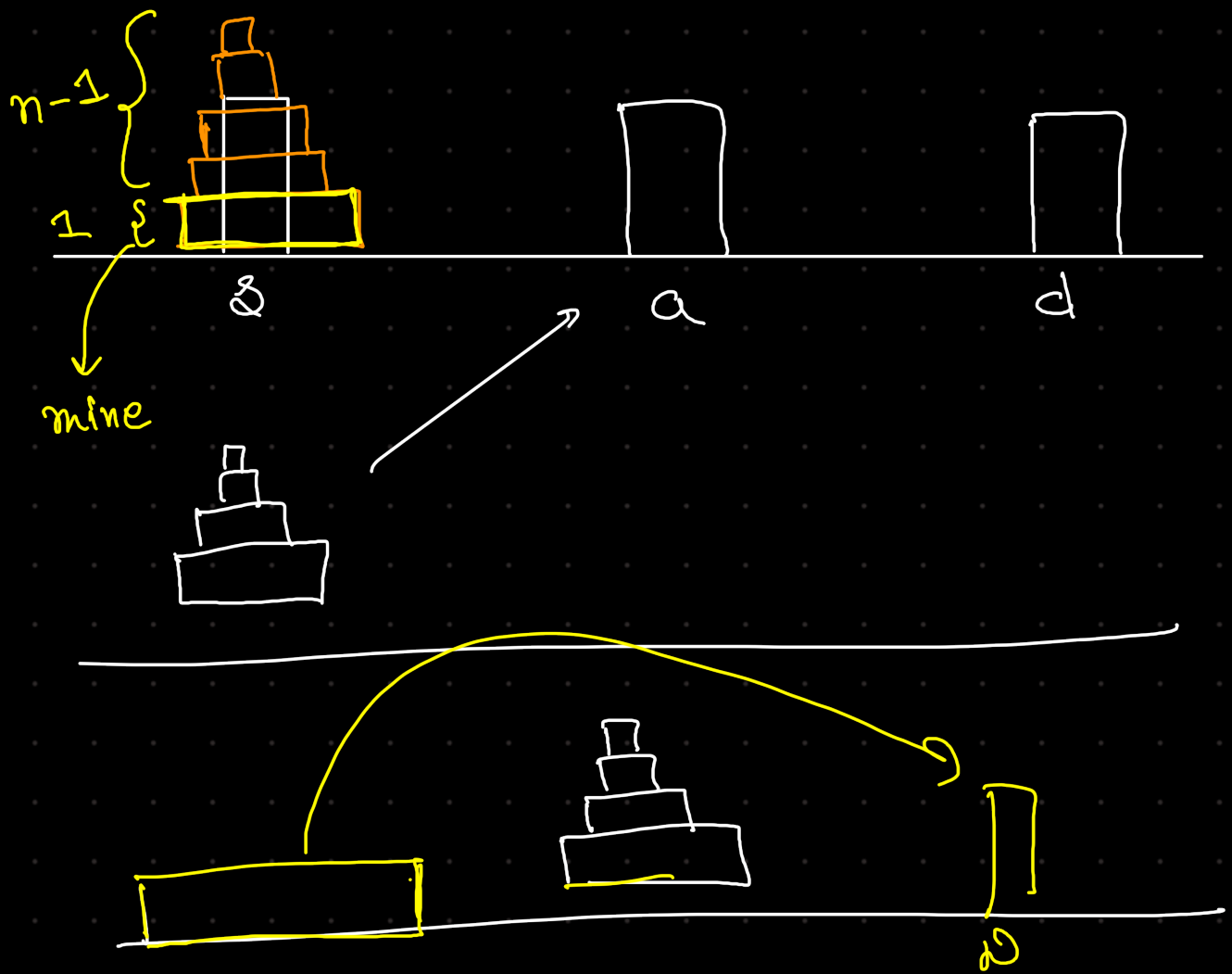
src
a

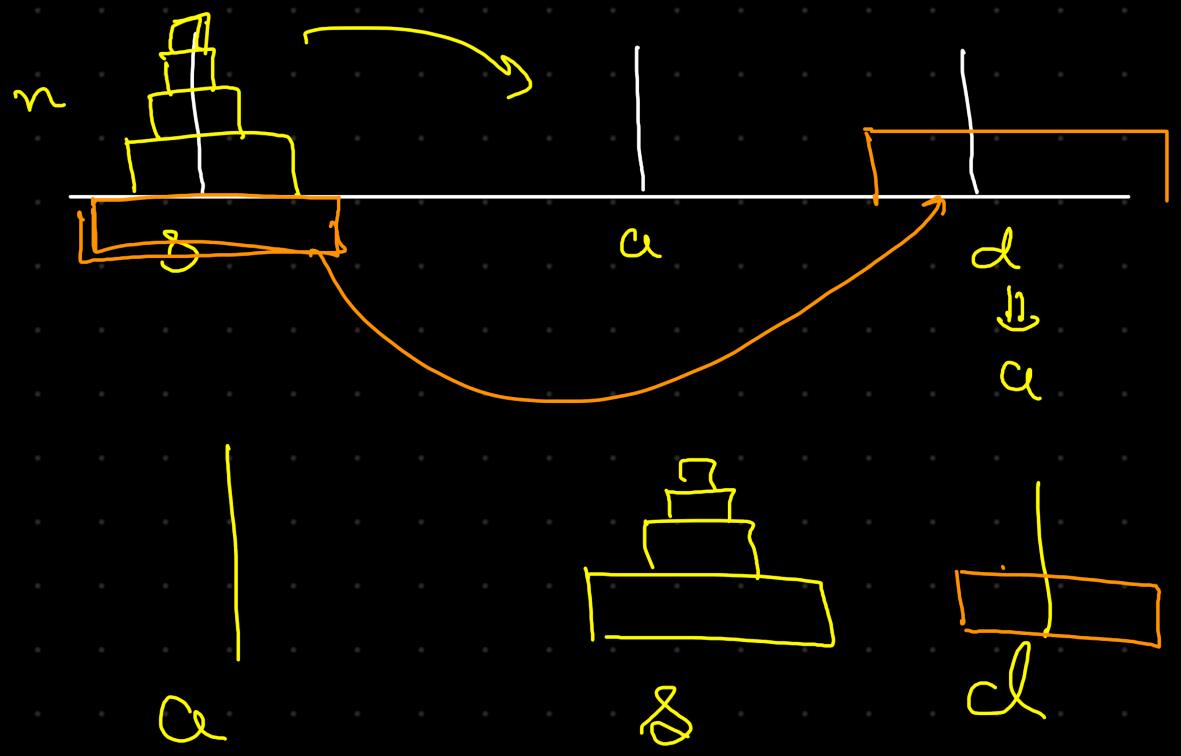
extra / aux
b

destination
c



a c


n-1
1
s
a
d
mine


n
g
a
d
11
a

a
g
d