Q⇒ Given a string of alphabets, print all possible subsequences of the given string. (ordering doesn't matter)

Ex → "abc"

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

Subsequence → It is a sequence of elements of the set obtained by deletion of some elements of the set.

Subsequence might sound similar to subset but here relative ordering of elements matters.

{1, 2, 3} → given set (collection of items)
↳ array / string

↳ {1,2} {2,3} {1,3}
{1,2,3} {1} {2} {3}
{}

{3,2}

Subsets → ordering doesn't matter

Subset is any possible combination of the original set elements

Subarray → These are contiguous cross-section of the given array. (They are different from subsequences)

→ In case strings → Substrings

↳ continuity doesn't matter

1	2	3	4

→ [1], [2], [3], [4]

[1,2] [2,3] [3,4]

[1,2,3] [2,3,4]

[1,2,3,4]

Every subarray is a subsequence but not all subsequence is a subarray.

Q⇒ Given a string of alphabets, print all possible subsequences of the given string. (ordering doesn't matter)

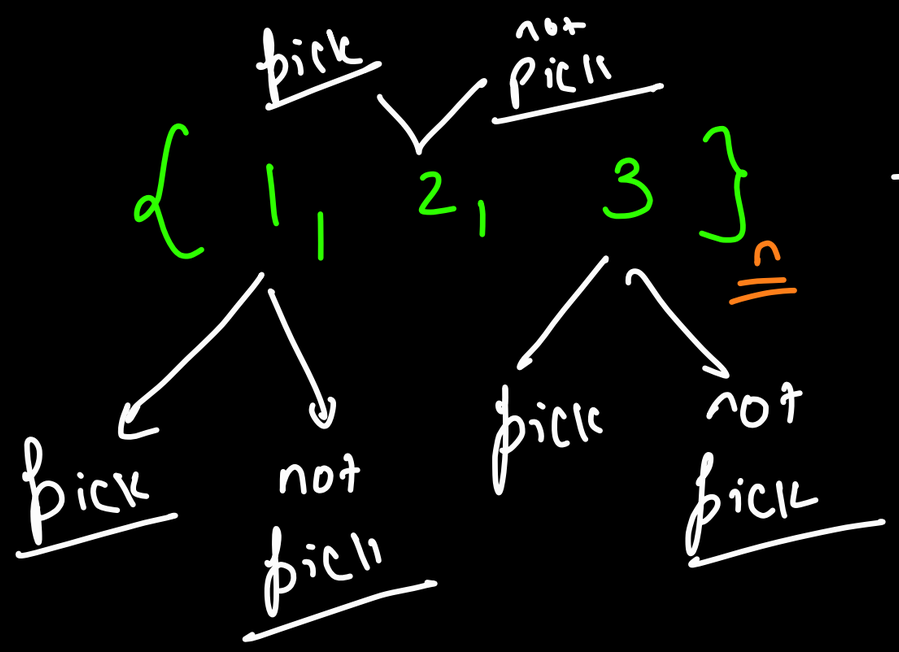
(Recursively)

Ex → "abc"

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

↳ abc → abc
ab
bc
ac
a
b
c
....

{1, 2, 3 ... n}
↳ Total no. of Subsets → 2^n



$\rightarrow 2^n$

$2 \times 2 \times 2 \ldots\ldots 2$ (n)

{1, 2} {2, 3} {1,3}

{1, 2, 3} {}

{1} {2} {3}

pick → 1

not pick → 0

{1, 2, 3}

0 0 0 → {}

0 0 1 → {3}

0 1 0 → {2}

1 0 0 → {1}

1 0 1 → {1, 3}

0 1 1 → {2, 3}

1 1 0 → {1, 2}

1 1 1 → {1, 2, 3}

a b c

all possible Subsequence of
↓
b c

all possible Subsequence of
↓
c

all possible subseq of
...

abc
ab
ac
a

bc
b
c
...

include
a

exclude

"bc"
"b"
"c"
"..

include
b

exclude
b

c
...

include c

exclude
c

...

a

$f(str, i, output) = f(str, i+1, output + str[i])$

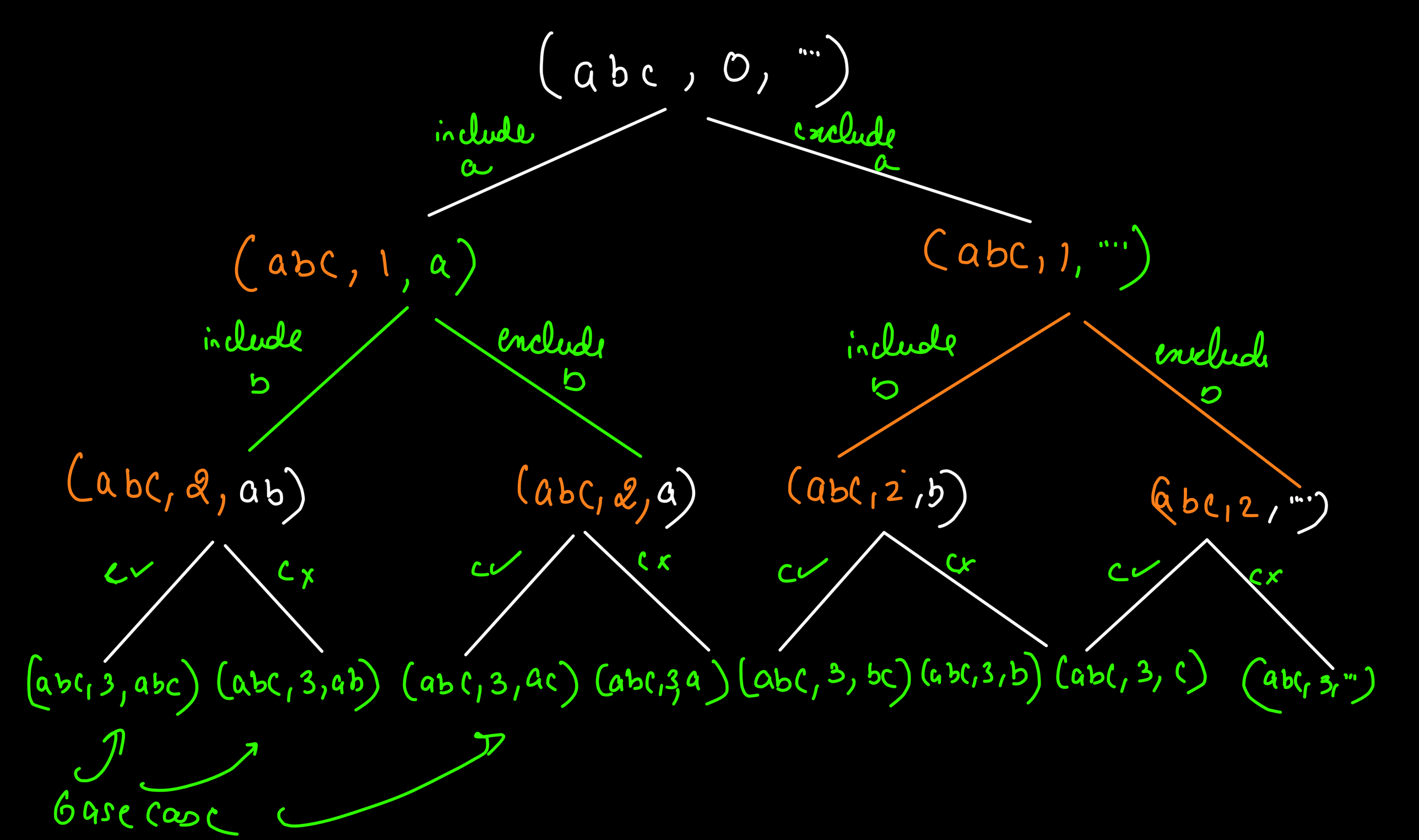
$f(str, i+1, output)$

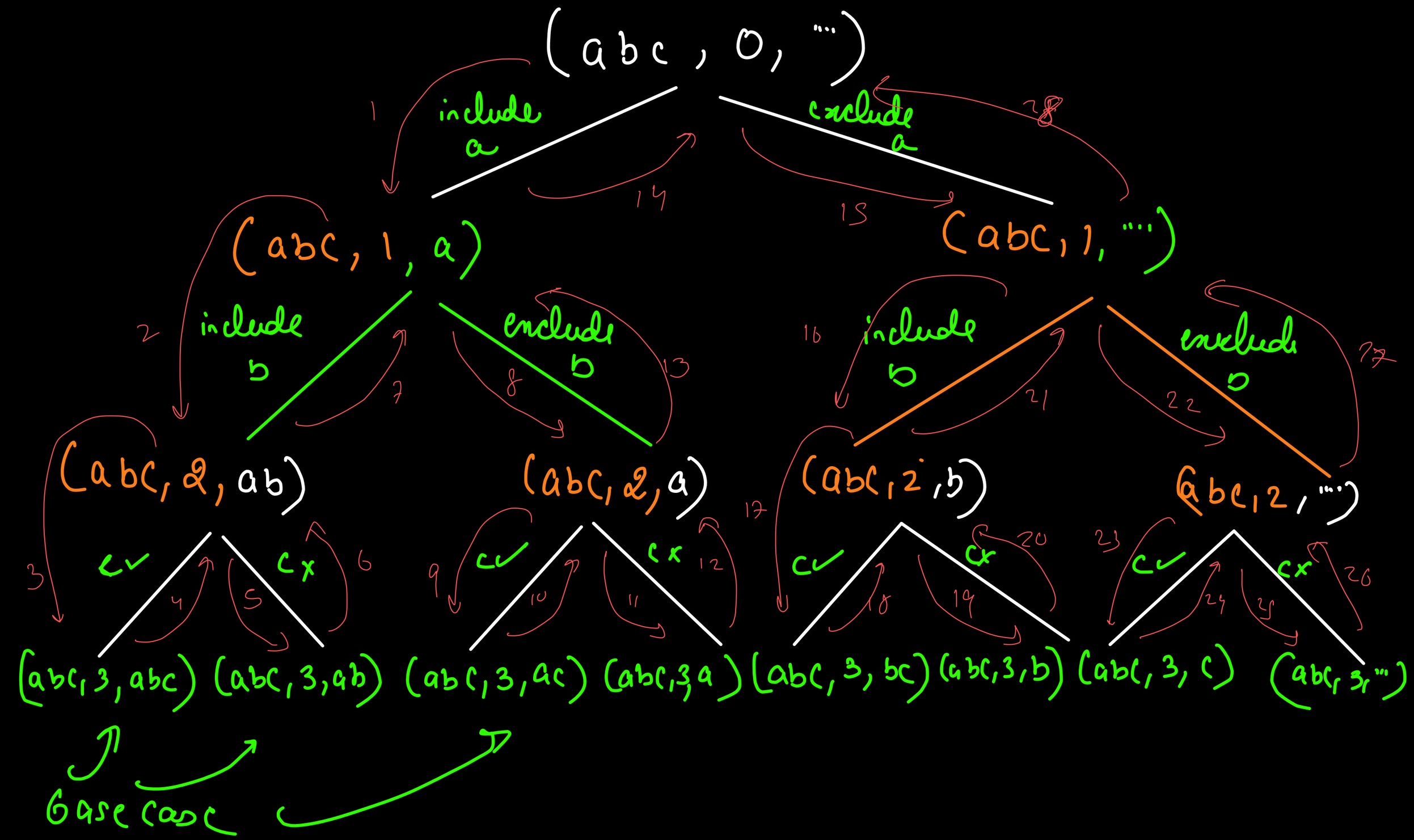
↓

computes all possible subsequences from index i to $n-1$ of the given string str in the output string 'output'

Base ↘

```
if (i == str.size) {
    print(output);
    return;
}
```





Q→ Given a string str, remove all occurence of the character "x" in it and return a new string.

Ex → "maabxxdxcxyx"

ans → "maabdcy"

f(str, i, output) =

{ f(str, i+1, output) if str[i] == 'x'

{ f(str, i+1, output + str[i]) else

↓

Computes str without "x" from index i to n-1

Base → if (i == str.size()) {
print(output)
return
}

(maab xx dx, 0, "")

↓

(moabxx dx, 1, m)

↓

(moabxxdx, 2, mo)

↓

(moabxxdx, 3, moa)

↓

(moabxxdx, 4, moab)

↓

(moabxxdx, 5, moab)

↓

(moabxxdx, 6, moab)

↓

(moabxxdx, 7, moabd)

↓

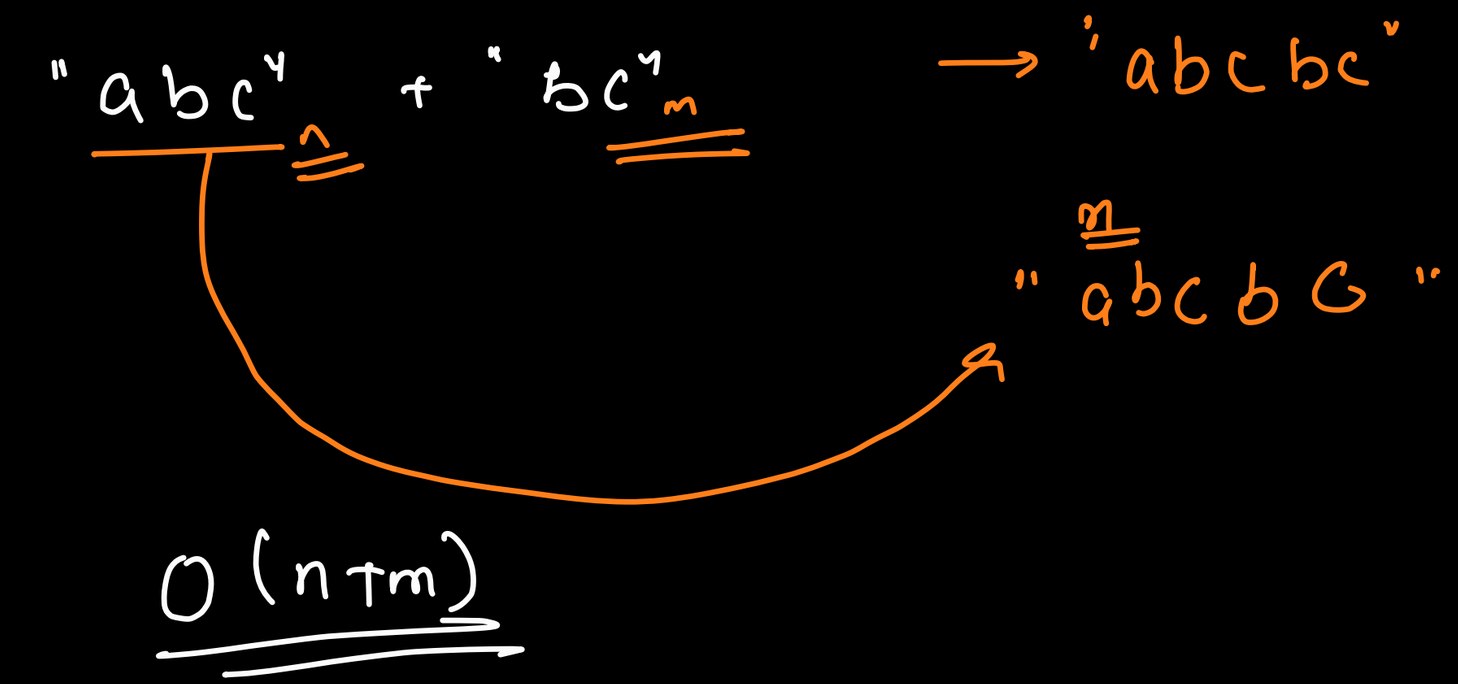
(moabxdx, 8, moabd)

n → using going to all char

O(n)

O(n²)

Space → O(n)



1 → x

2 → abc

3 → def

4 → ghi

5 → jkl

6 → mno

⋮ ⋮

⋮ ⋮

⋮ ⋮

"23"

"23"
↳ abc

mapping comb of 3

ad
ae
af

bd
be
bf

cd
ce
cf

attach a → d

attach b → e

attach c → f

2 → abc 3 → def 4 → ghi

2 3 4

Combinations

Combinations

Combinations

g 3 4

g 4

g

...

adg, adh, adi

aeg, aeh, aei

afg, afh, afi

a

bdg bdh bdi

beg beh bei

bfg bfh bfi

b

cdg cdh cdi

ceg ceh cei

cfg cfh cfi

c

dg

dh

di

eg

eh

ei

fg

fh

fi

d

e

f

g

h

i

g

4

i

...

mapping

$f(str, i, output) = f(str, i+1, output + char)$

$\forall char \in$ mapping of digit at i^{th} index

2

Compute all possibe keypad comb. of the str from index i to $n-1$

mapp → 2 → abc