You are given a string `word`, and an integer `numFriends`.

Alice is organizing a game for her `numFriends` friends. There are multiple rounds in the game, where in each round:

- `word` is split into `numFriends` **non-empty** strings, such that no previous round has had the **exact** same split.
- All the split words are put into a box.

Find the lexicographically largest string from the box after all the rounds are finished.

Example 1:

> **Input:** `word = "dbca", numFriends = 2`
>
> **Output:** `"dbc"`
>
> **Explanation:**
>
> All possible splits are:
>
> - `"d"` and `"bca"`.
> - `"db"` and `"ca"`.
> - `"dbc"` and `"a"`.

Example 2:

> **Input:** `word = "gggg", numFriends = 4`
>
> **Output:** `"g"`
>
> **Explanation:**
>
> The only possible split is: `"g"`, `"g"`, `"g"`, and `"g"`.

"lexicographically Largest" → which appear later in the dict

Ex : "abc" < "abd"

"a" = "a" → move forward
"b" = "b" → move forward
"c" < "d" → d appears later in dict than c
∴ "abc" < "abd"

Ex : "abcd" < "e"

"a" < "e" → e comes after a in dict.

∴ "abcd" < "e"

"e" comes later in dict than "abcd".

⇒ To find lexicographically largest start comparing strings character wise.

Ex : "ebcde" > "e"

"e" = "e"

"b" > "" → Empty string no more letters/ chars left.

∴ "ebcde" appears after "e" in dict.

Thought Process

word = "dbca", numFriends = 2.

⇒ look for largest char (that comes later in alphabet)

Ex: "d" in dbca.

How to distribute among friends?

Ex:

⇒ we know we have to start with "largest char" i.e. "d"

Friend 1	Friend 2.
d	bca
db	ca
dbc	a
~~dbca~~	" " (NOT ALLOWED)

largest length string.

"we gave the last friend "one char" & gave all

"the other chars" to the first friend"

Ex : 3 friends.

⇒ we start with "d"

"dbca"

friend 1	friend 2	friend 3.
db.	c	a

NOTE: give one-one char to the remaining friends whatever chars are left add it with "d" & give it to first friend.

This way we will get our lexicographically largest string.

⇒ if we have numFriends.

⇒ Fr 1. (numFriends - 1) give 1 char each.

whatever remaining give to fr 1 to add with largest char.

length of lexicographically largest string =

word.length() − (numFr − 1 chars);

Ex: "dbca"
⇓
4.

giving 1 char each to remaining friends.

Max length substring we would get.

Ex: "eabcde", numFriends = 3.

⇓
Total length = 6

F1 (6 − 2) ((word.len − (NumFr − 1)))
4

F2 1 char
F3 1 char
2 (NumFr − 1)

⇓ largest length substring we can get.

"eabc" "d" "e"

another Ex :

" a b c e d "
0 1 2 3 4
↑

Fr = 3 .

F1	F2	F3.
3 ch	1 ch	1 ch.
ed → lexico-graphically largest	ab	c

↓

after "e" in og string there was only "d" left

although we can make 3 ch largest substring with "e" but take whatever is available it will still be lexicographically largest .

e ⟶ idx = 3 .

longest possible = n - (numFr - 1);
(starting with largest char)

substring (i , 3)
possible ↓
index of largest char

its not possible in this string as "ed" is only available.

∴ add a check, [jitna mil jaye utna hi lelo]

longest possible = n - (numFr - 1);

take possible length = min (longest possible , n - i);
(canTake length)

char avail.. to the right side of it

Substring (i , take possible length);

Ex: "abced"

takepossible length = 2

Substring (i , 2) → length

"ed"

starting index.

Ex : 5 - 3 = 2

"abced"

i → idx of ith char.

(in this ex, i is idx of largest char)

Ex: "eabced" (0 1 2 3 4 5) Friends = 3.

largest char at two indexes → 0 & 4.

longest Possible = n - (Fr - 1)

= 6 - (3 - 1)

= 4.

if we start from $i = 0$

we can make longest possible substring.

"eabc"

if we start from $i = 4$

we get, "ed"

NOTE : "eabc" < "ed"

∴ lexicographically largest → "ed"

what to do for this kind of edge case ??

1st way

⇒ find largest char "e"

⇒ check in which indices it is present in {0, 4}

⇒ go to each i & check if we are able to make "largest possible" length substring or not.

"eabc" "ed"

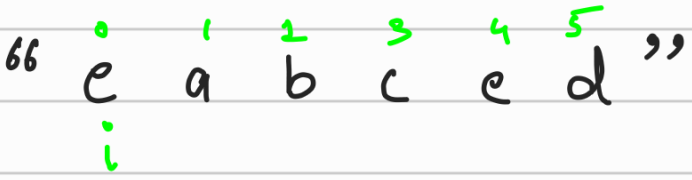
⇒ then compare l/w which is bigger. (lexicographically)

"eabc" < "ed"

"ed"

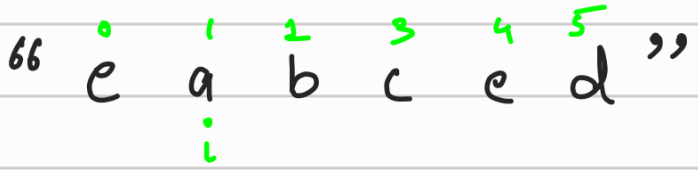
2nd way

⇒ go to every index & try to make "longest possible" length substring.



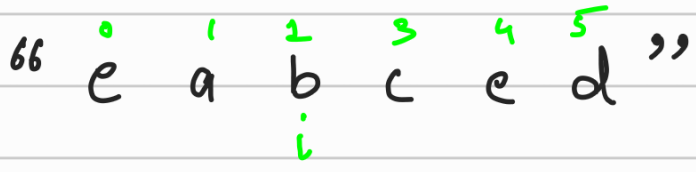
" e a b c " ⇒ best answer.

⇓



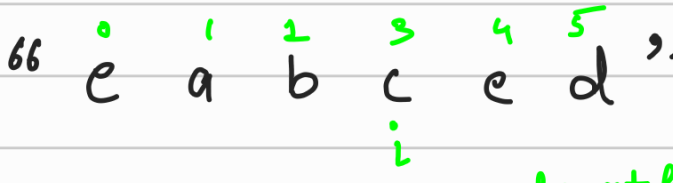
" a b c e " ⇒ still the above is best answer.

⇓



" b c e d " ⇒ still the .....

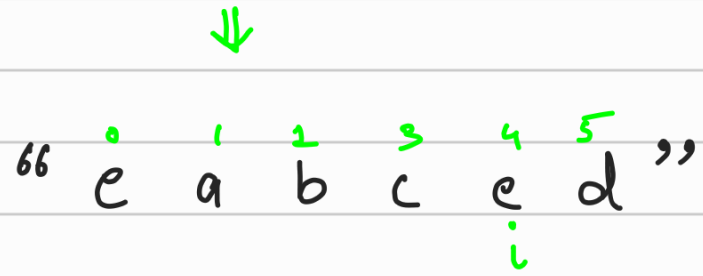
⇓



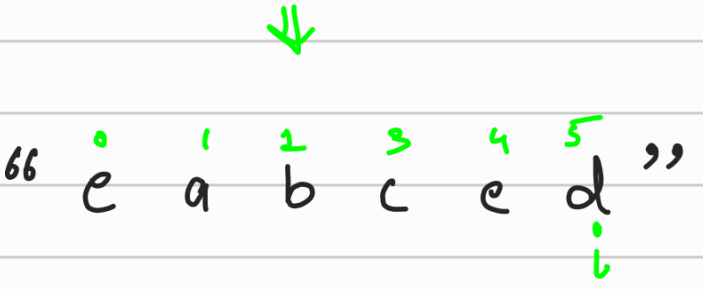
↗ largest possible.

can't take lim = 4, doesn't matter take whatver length is possible.

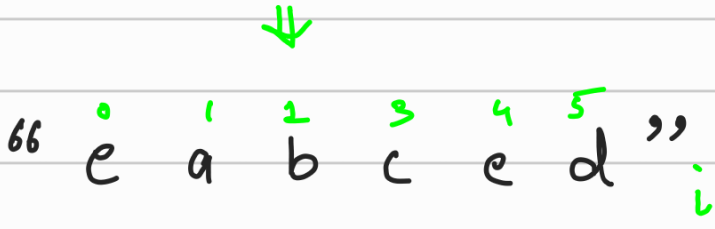
" c e d " ⟶ still .....



"ed" → better than previous best answer
∴ update it.



"d" → still .....
.....



i oob

Answer found.

longestPossible = n - (numFriends - 1);

for (i = 0 ; i < n ; i++) { → O(n)

canTakeLength = min (longestPossible , n - i);

result = max (result , word.substr (i , canTakeLength)

↓

O(n)

}

return result ;

T.C = O(n²)

S.C = O(1)

optimisation

↓

O(n) .

```
class Solution {
public:
    string answerString(string word, int numFriends) {
        int n = word.length();
        if( numFriends == 1)
            return word;

        string result;

        int longestPossible = n - (numFriends - 1);

        for(int i = 0; i < n; i++) {
            int canTakeLength = min(longestPossible, n-i);

            result = max(result, word.substr(i, canTakeLength));
        }

        return result;
    }
};
```

} → edge case if no of friend = 1 give the whole word to it .