Words - K Length Words- 3

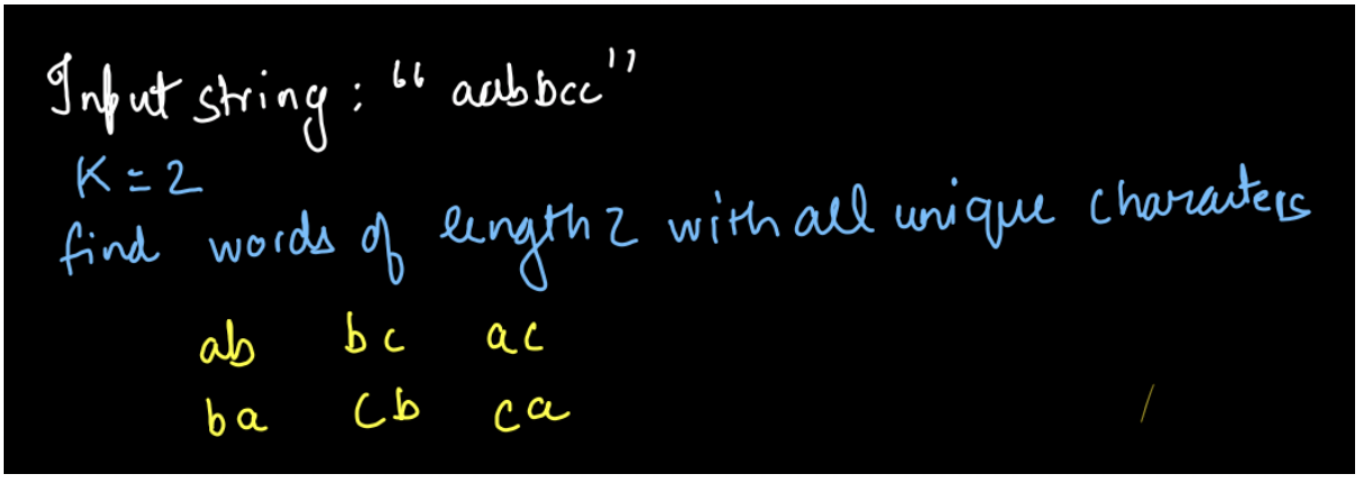
1. You are given a word (may have one character repeat more than once).
2. You are given an integer k.
2. You are required to generate and print all ways you can select k characters out of the word.
Note -> Use the code snippet and follow the algorithm discussed in question video. The judge can't force you but the intention is to teach a concept. Play in spirit of the question.

aabbbccdde
3
['aab', 'aac', 'aad', 'aae', 'aba', 'aca', 'ada', 'aea', 'baa', 'caa', 'daa', 'eaa', 'abb', 'abc', 'abd', 'abe', 'acb', 'adb', 'aeb', 'acc', 'acd', 'ace', 'adc', 'aec', 'add', 'ade', 'aed', 'bab', 'bac', 'bad', 'bae', 'cab', 'dab', 'eab', 'cac', 'cad', 'cae', 'dac', 'eac', 'dad', 'dae', 'ead', 'bba', 'bca', 'bda', 'bea', 'cba', 'dba', 'eba', 'cca', 'cda', 'cea', 'dca', 'eca', 'dda', 'dea', 'eda', 'bbb', 'bbc', 'bbd', 'bbe', 'bcb', 'bdb', 'beb', 'cbb', 'dbb', 'ebb', 'bcc', 'bcd', 'bce', 'bdc', 'bec', 'bdd', 'bde', 'bed', 'cbc', 'cbd', 'cbe', 'dbc', 'ebc', 'dbd', 'dbe', 'ebd', 'ccb', 'cdb', 'ceb', 'dcb', 'ecb', 'ddb', 'deb', 'edb', 'ccd', 'cce', 'cdc', 'cec', 'dcc', 'ecc', 'cdd', 'cde', 'ced', 'dcd', 'dce', 'ecd', 'ddc', 'dec', 'edc', 'dde', 'ded', 'edd']

```
def wordsKLengthIII(s,k):
    # s = ''.join(list(set(s)))
    n = len(s)
    ans = []
    spots = [None]*k
    fmap = {}
    for el in s:
        fmap[el]=-1
    helper(s,k,0,fmap,ans,spots,0)
    return ans

def helper(s,k,count,fmap,ans,spots,idx):
    if idx==len(s):
        if count==k:
            ans.append(''.join(spots))
        return
    ch = s[idx]
    lIdx = fmap[ch]
    for i in range(lIdx+1,len(spots)):
        if spots[i]==None:
            spots[i]=ch
            fmap[ch] = i
            helper(s,k,count+1,fmap,ans,spots,idx+1)
```

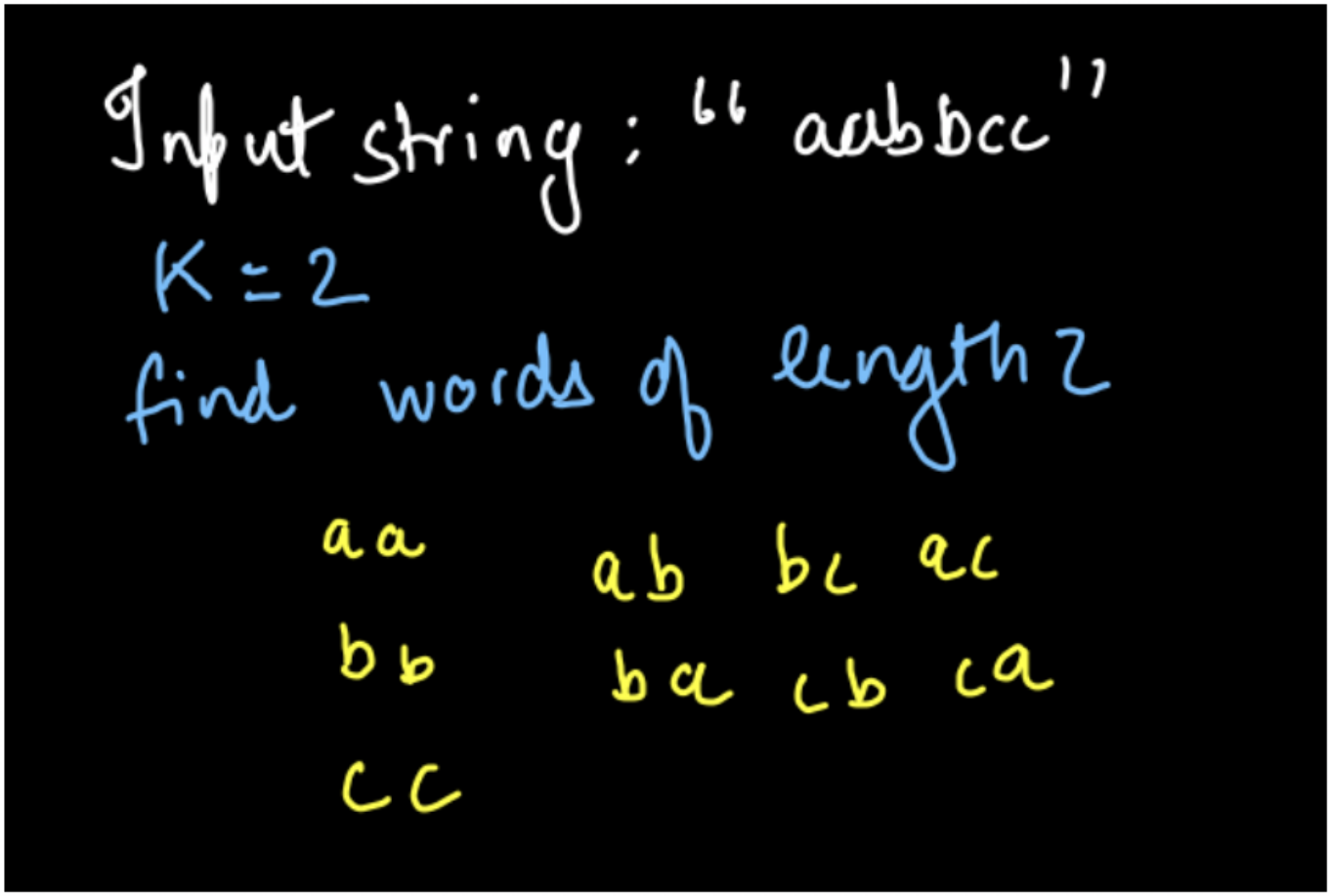
```
            spots[i]=None
            fmap[ch] = lIdx
    if lIdx==-1:
        helper(s, k, count + 0, fmap, ans, spots, idx + 1)


print(wordsKLengthIII('aabbbccdde',3))
```

Welcome back, dear reader. So, how is it going? We hope you are doing well and learning the concepts and enjoying your coding journey. So, in this article, we will discuss the problem WORDS- K LENGTH WORDS-3 . So, what does this problem say? Before we jump into this problem, do you remember the problem K-LENGTH WORDS-1 that we have solved previously. If you have not solved this problem, we recommend you try to solve this problem on your own first and refer to the solution video for K-LENGTH WORDS-1 problem because this is an extended version of the same problem. So, let us recall a little bit about that problem first and then we will tell you about this problem. We will be given an input string which can have repeated characters also. In the k-length words-1 problem, we had to find the words of length k, which do not have any repeating character. For instance:



We were given the input string "aabbcc". The 2 letter words where both the characters are unique are shown in the image above. Now, there is a slight variation in this problem.

We still have to print k letter words but the words can have repeating characters. For instance:



So, we hope you have got the question and the difference between these two problems. You may refer to the K-LENGTH WORDS-3 video to understand the question if you have any doubts about the same. We recommend you try to solve this problem on your own first and then go to the solution.

Approach

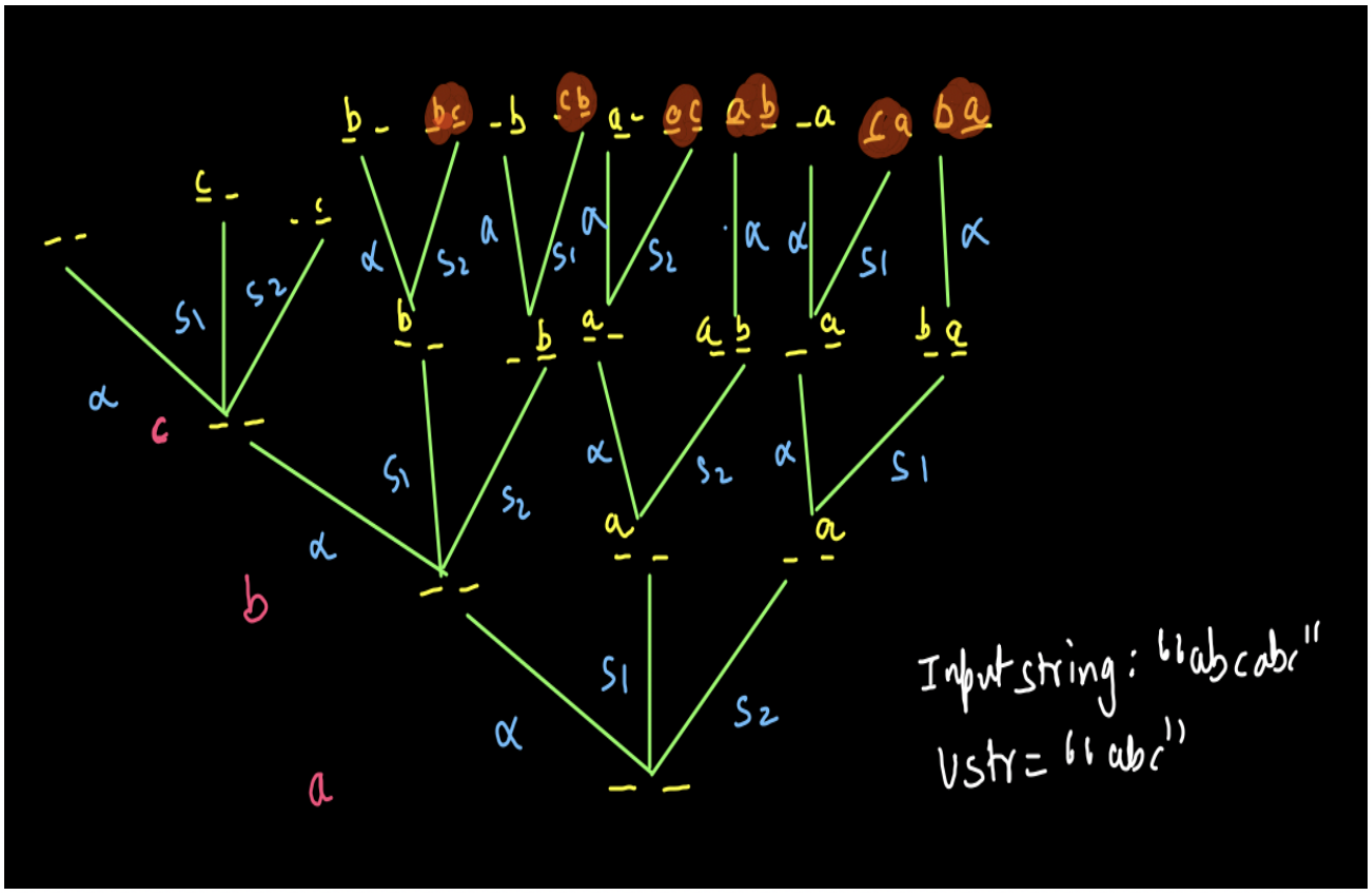
Time Complexity: O(n!) where n is the length of the input string **Space Complexity:** O(h) where h is the height of the recursion stack. O(n!) where n is the length of the input string

Explanation

Let us have a quick recap of the WORDS-K LENGTH WORDS-1 problem and see how we used to solve it.

So, let us say that the input string is "abcabc" and we have to find the words of length 2. So, in this question, since we had to find the words with every character unique, we selected the unique string "ustr" first from the complete input string. So, we have "abc" as the string now. After this, we used to keep the characters at the levels and the spots used to be our options (Remember we compared this with the box and items problem?).

So, at every level, we had a character and it had three options: either to go at spot1, or at spot2 or not to come in the word. The entire tree for the same is shown below:

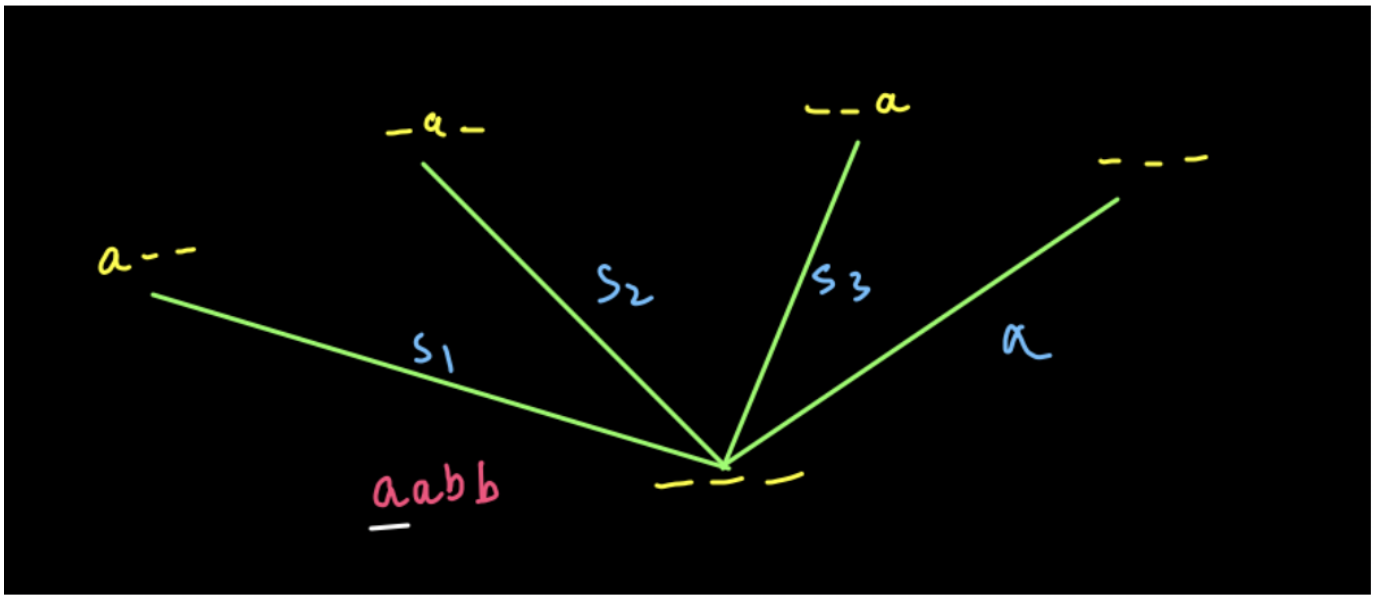


So, at every level, we used to have options for the characters to fit at a spot or not to get selected in the word. Here, we are having a little bit of a different situation.

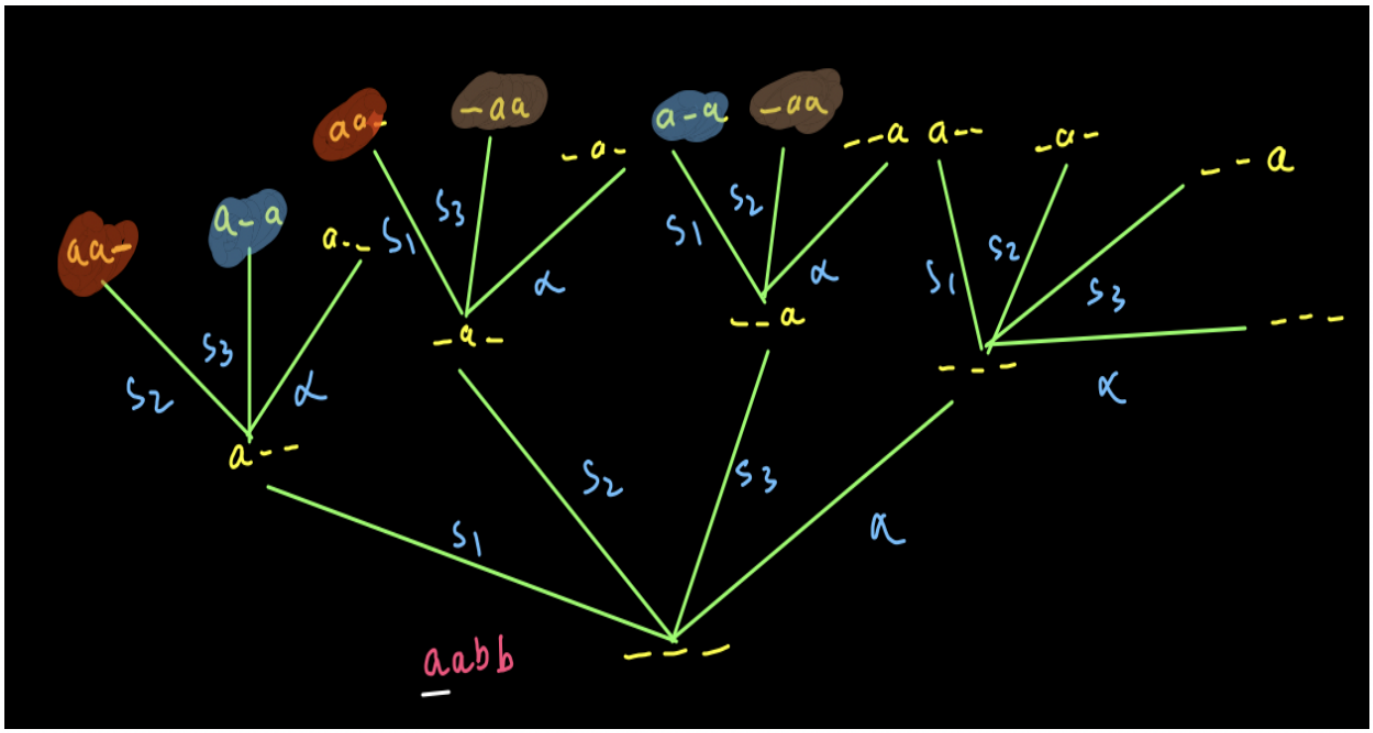
First of all, we do not have to make a "ustr" i.e. a string of unique characters. This is because our words can have repeating characters. So, at least we know where to start now,right? So, let us begin the procedure for our question.

Constructing Recursion Tree

So, let us start by taking another example. Let us say, we have an input string "aabb" and we have to make words of length 3 i.e. k=3. So, we will put the first character of the input string at the 0th level of the tree. (This is indicated by white underline under the first character of the string). This character will have 4 choices. The first three choices will be the spot at which it wants to go and the 4th choice will be not to go at any spot.

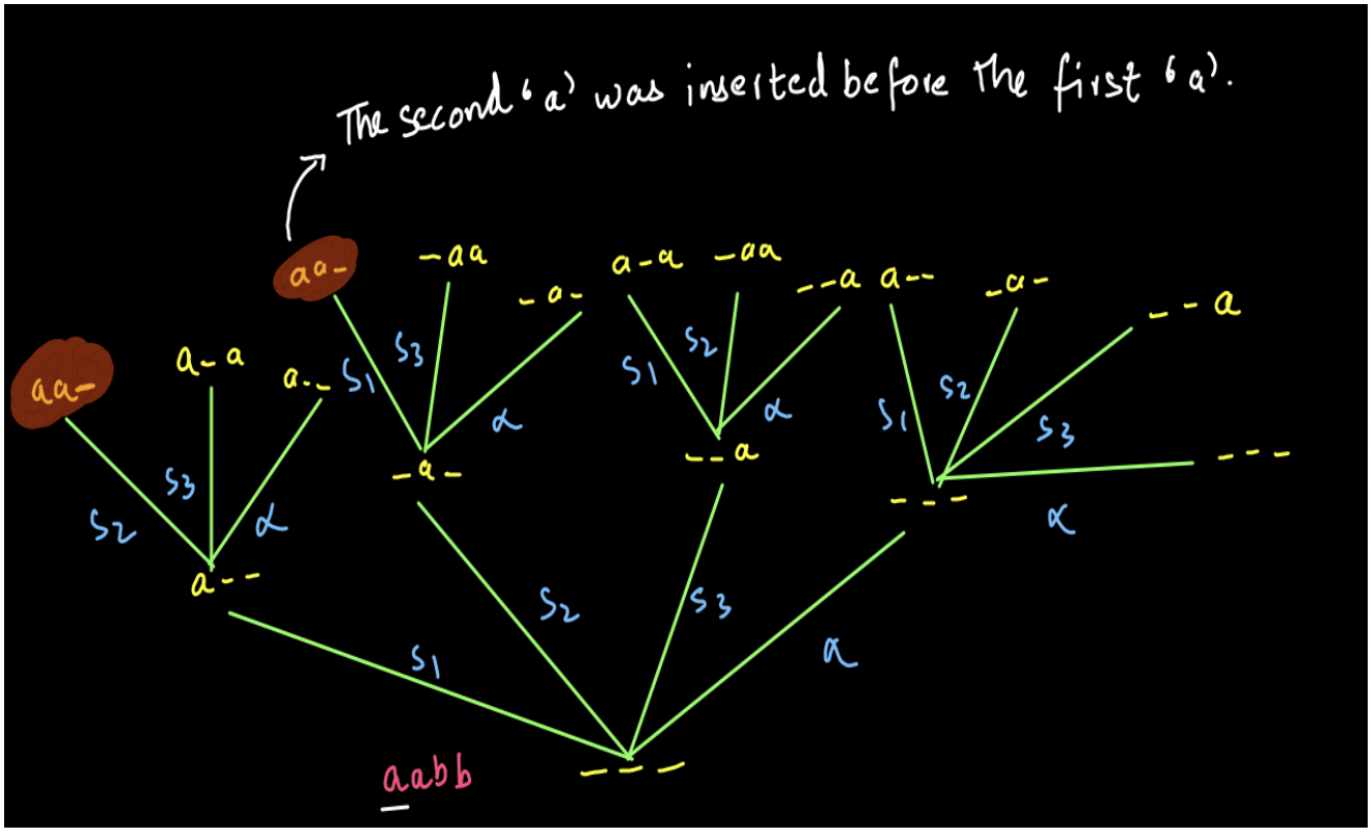


So, at the 0th level, we have the first character 'a'. There are 4 choices since the word is of length 3 which are shown in the diagram above and their consequences are also shown.



Now if you look at the image shown above, we can see that we have drawn the tree up to the next level where we have shown the placement of the second character 'a'. There is however, a problem with this tree. If you look at the elements highlighted in the orange color, they both are the same i.e. the elements are repeating. Also, the elements highlighted in the blue color are also the same and the elements highlighted in the brown color are also the same.

Can you think why this perception is happening? The reason is very simple. Have a look at the diagram below:



The repetition occurred because we inserted the second 'a' after inserting the first 'a' but the position of insertion was before the first 'a'. For instance, we inserted the second "a' at the first spot before the first 'a' which was already present at the second spot.

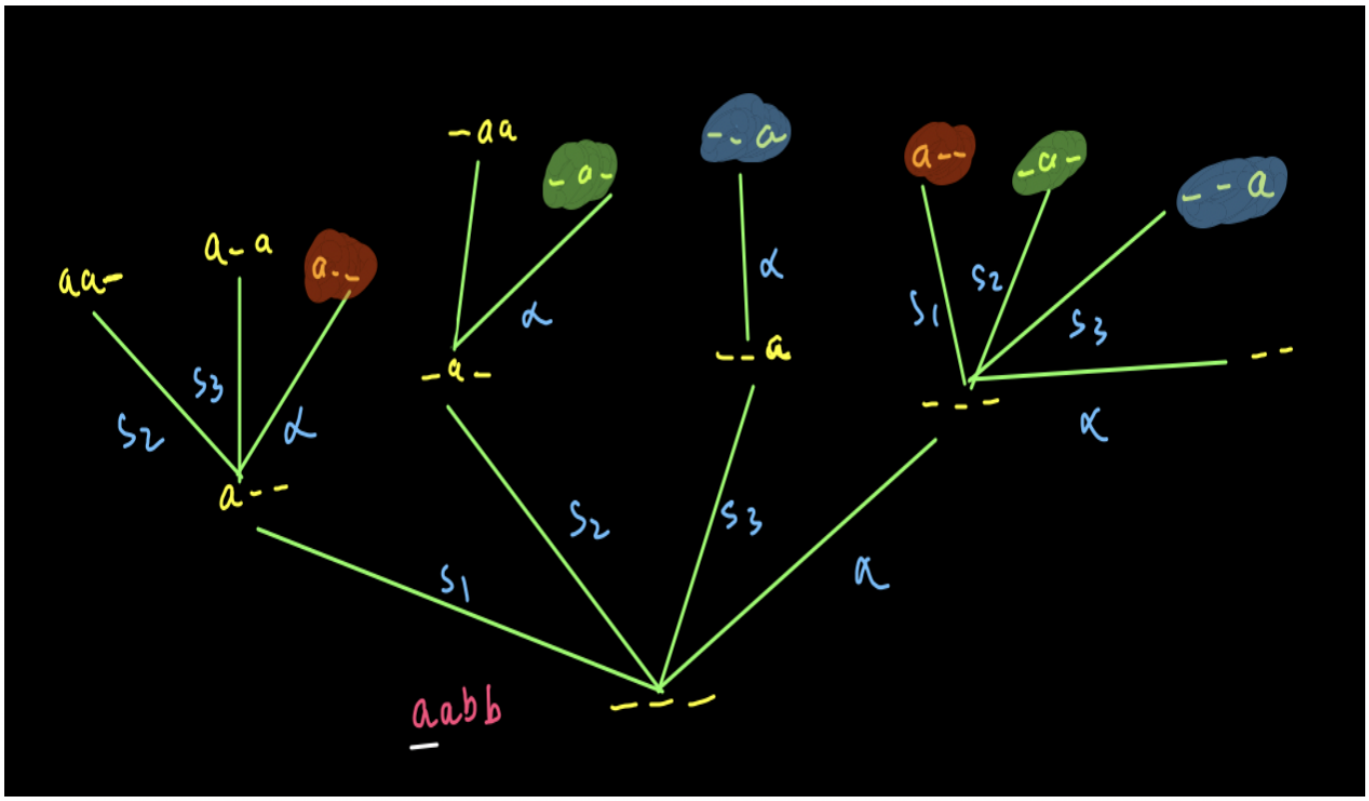
Dear reader, we want you to look at the tree carefully and meditate why this is the reason for repetition. This is because, after inserting the first 'a' at spot 1, we already have an option for inserting the second 'a' at the second spot. This will produce the result "aa_". Now, if you have meditated on this fact and observed the recursion tree carefully, you might have understood already that this call where we put the second 'a' to spot 2 after the insertion of the first 'a' at spot 1 is made before the call where we insert the second 'a' before the first 'a' at spot1.

Therefore, the second call, where we are inserting the repeating character before a position where the character has already arrived, will cause repetition.

So, we can reduce the repetition by removing all the calls where we insert a repeating character before it's previous occurrence.

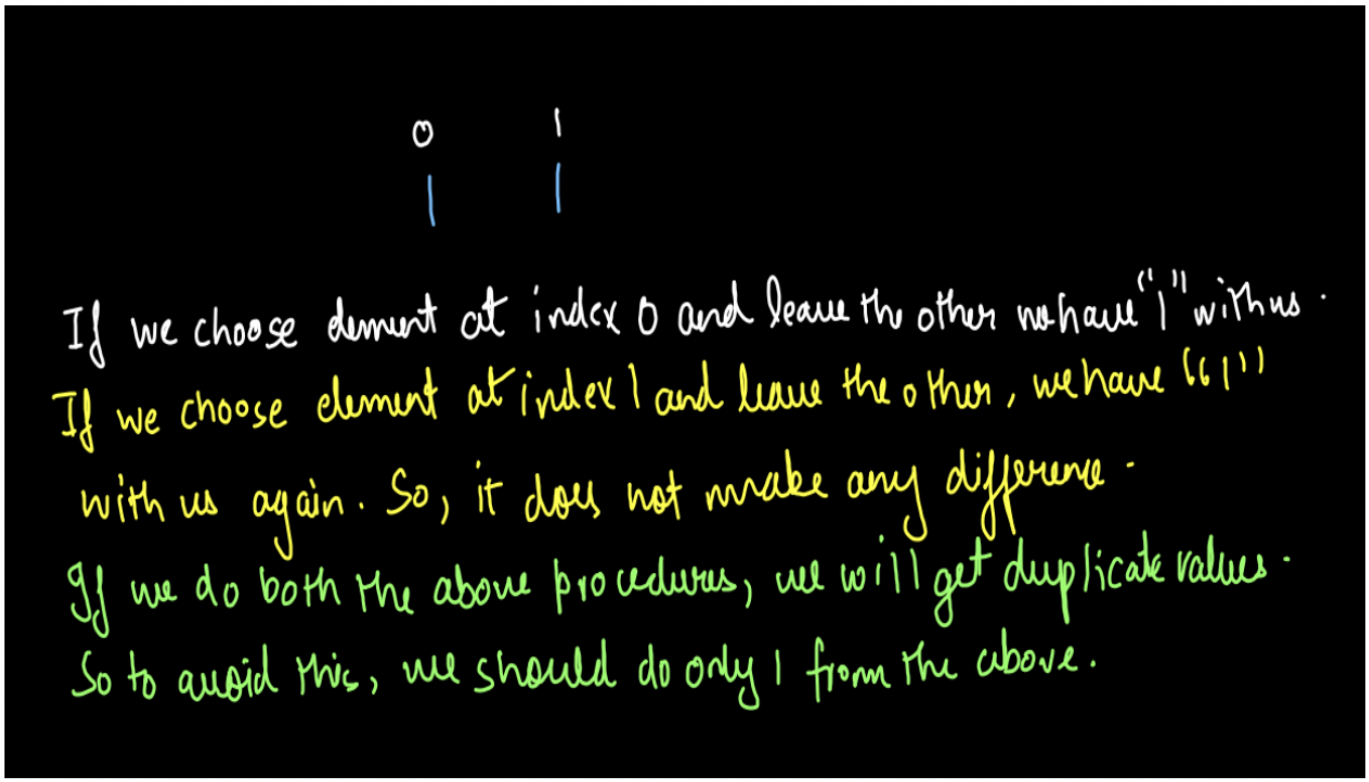
You may refer to the solution video to understand the above explained reason for duplicacy of the elements.

he same is done below:



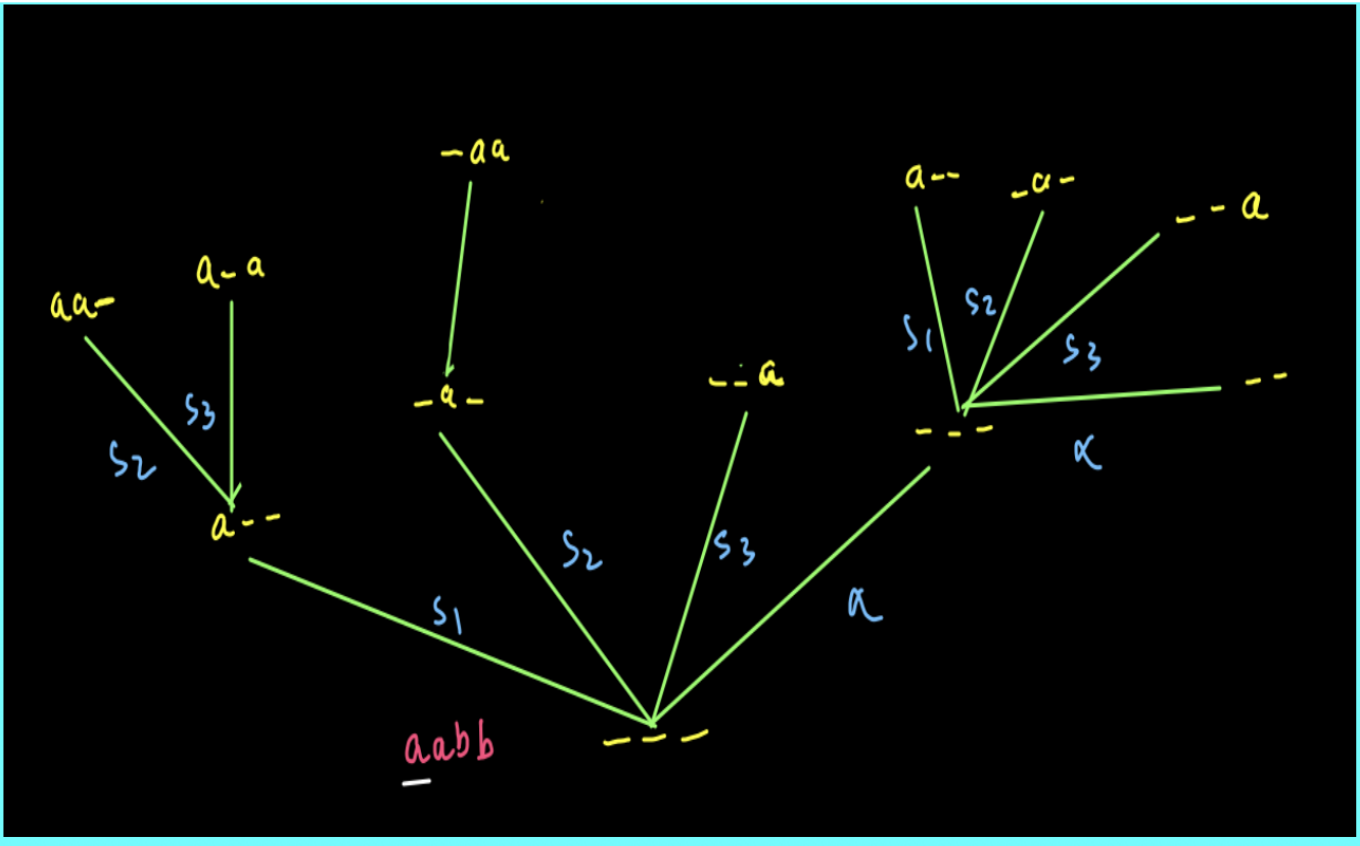
Now if you look at the image shown above, even after removing some of the duplicates, we still have some. They are highlighted with the same color i.e. the node in orange color is the duplicate of another orange color highlighted node and so on.

Can you think of any reason behind this duplicacy shown in img-7? Let us make this simple for you. Let us say we give you two elements with values 1 and 1 respectively. Now, choose any one from them but reject the other. Now, choose the other one from them and reject the other. Do you see any difference?



So, you can see from the image above that if we have two identical elements, it does not make any sense to include any one of them and exclude the other twice. So, if we have selected the first 'a' in our example and we do not have the second 'a' for now, then it is going to cause duplicacy problems if we select the second 'a' and do not select the first 'a' as we have in img-7.

So, if an element has already occupied a spot in the word and it occurs again, we will eliminate the choice of the repeating word for "not occupying a spot". This will be clear to you when you look at the diagram given below:



So, we have removed all the "no" calls from the places where 'a' already had a spot. Now there is no repetition at every level in the tree.

You may refer to the solution video to understand the above explained reason for the duplicacy of elements in the tree.

So, we have to follow these two rules to avoid the repetition at any level in the entire tree:

The repeating character cannot be inserted before the already present character.

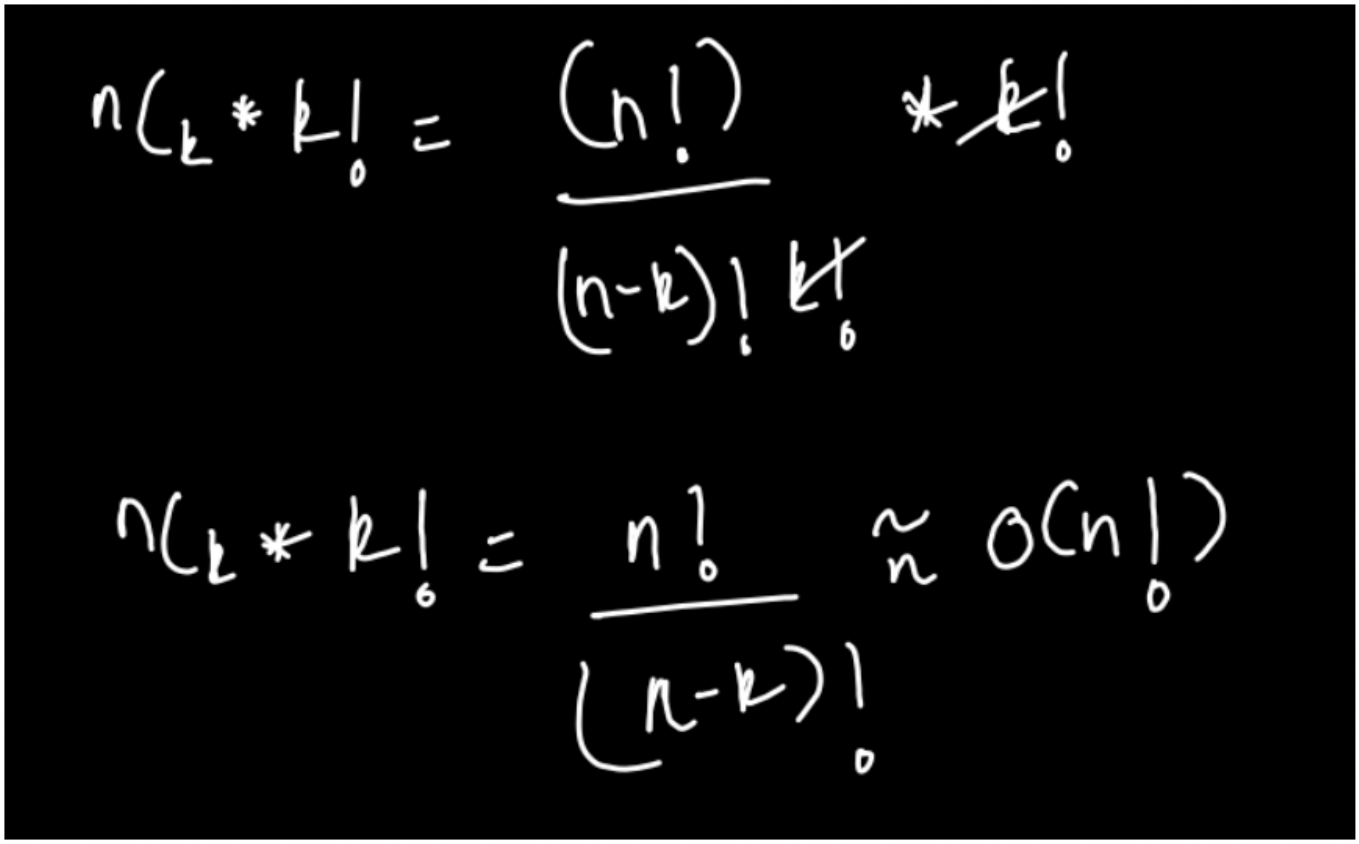
If a character is already selected and it occurs again, it does not have a choice of not getting selected.

So, dear reader, we hope that you have got these two rules and the reason behind them as well. We recommend you trace the rest of the tree and to understand the process of tracing the rest of the tree or to verify your procedure, you may refer to the solution video.

Time and Space Complexity Analysis

Time Complexity:

If we take the above example, we were to select 3 letters from 4 as we were making 3 letter words. Then, we had 3! Arrangements of those letters also. So, first we select the k letters from n i.e. nCk and then we are arranging them in k! ways. So, the time complexity should be nCk x k!



Space Complexity:

The space complexity without considering recursion will be O(1) and if we consider recursion it will be O(h) where h is the height of the tree. Also, what is the height of the tree? Since at every level, we have kept a character from the input string, we will have the height of the tree as O(n). So, the max height of the recursion stack will also be O(n) and so will be the space complexity.

So, dear reader, we hope that you understood the time and space complexity analysis also. If you have any doubts regarding the procedure or the code, refer to the complete solution video to clear all your doubts. With this, we have completed this problem.