

Word - K length word - 2

String \rightarrow abc abc , unique string - abc

$$\underline{\underline{K=2}}$$

ab
a c
b c
ba
c a
c b

3 box

2 distinct g-tems ($\overset{s_1 s_2}{\downarrow, \uparrow}$)

- - -

! 2 -
1 - 2
- 1 2
2 1 -
2 - 1
- 2 1

box

g-item.

b₁
b₂
b₃

g₁
g₂

3 character

a b c

pick 2 distinct
character

ab ba

ac ca

bc cb

character

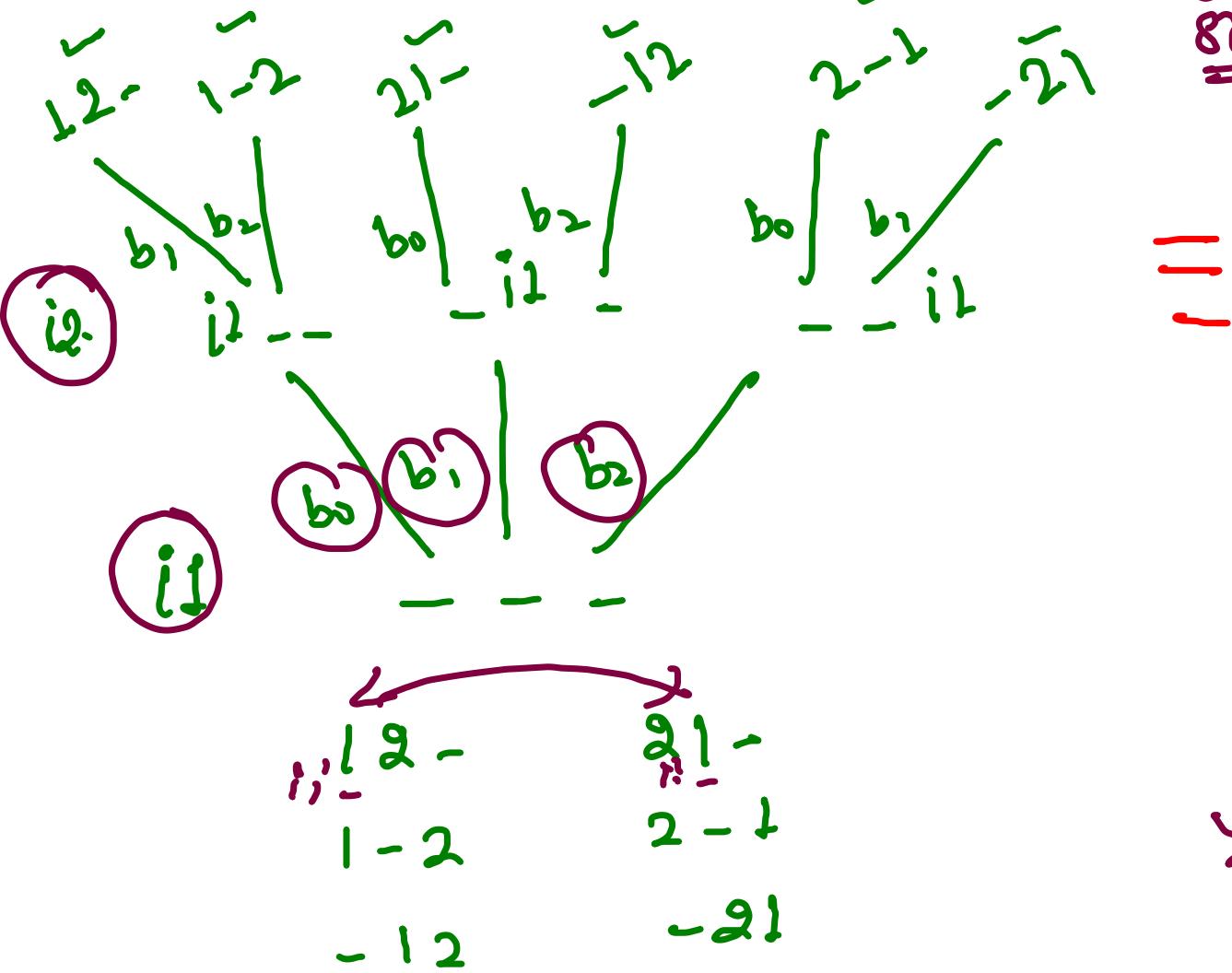
a
b
c

spots.

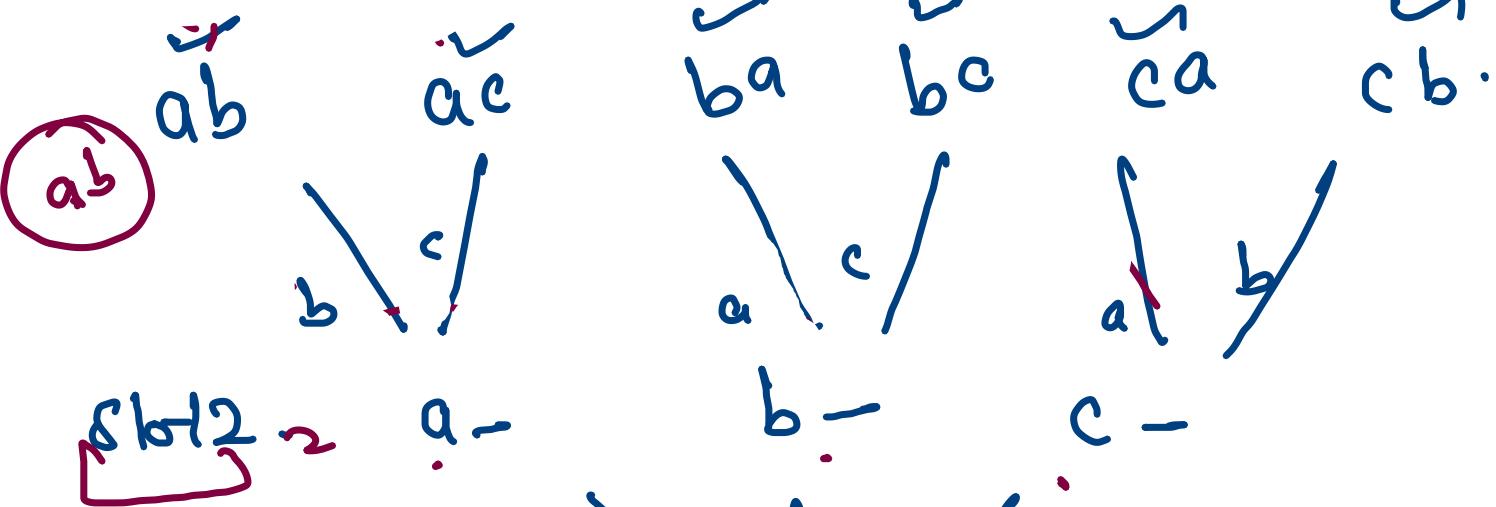
spot 1
spot 2.

Item: level

$\{ \text{box} = 3$
 $\text{Item: } i_1, i_2$



- ① Unique string.
- ② Hash set of already present elements to make call
- ③ Current slot
- ④ Total slot
- ⑤ Answer so far for string.



Slot1

ab
ac
bc

ba
ca
cb

permutation
of
unique
charact.

a b c
f t t t

Words - K. Selection - 3

String \rightarrow aabbc

$$k=3$$

UStr \rightarrow abc

$${}^n C_r = \frac{n!}{(n-r)!r!}$$

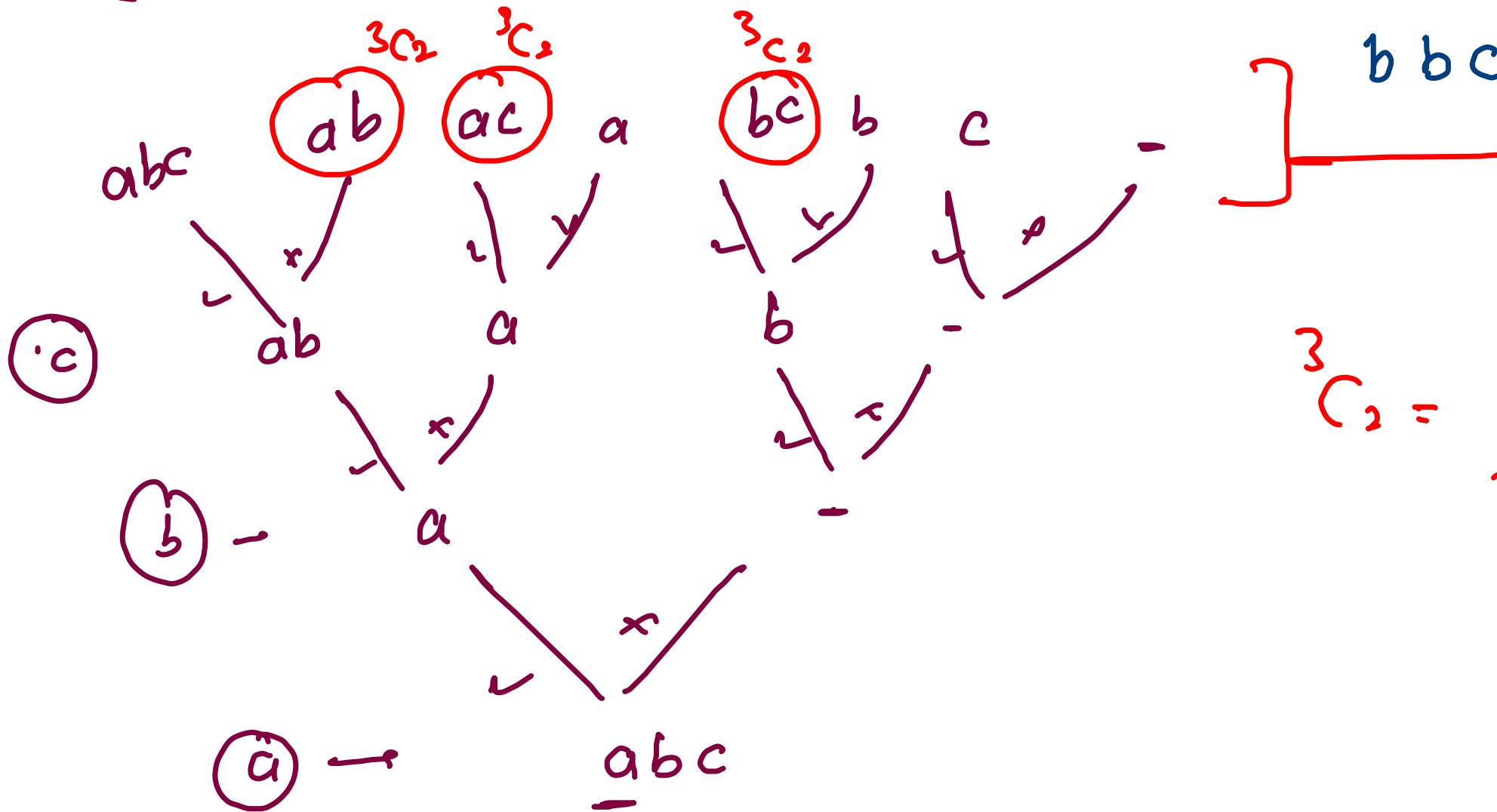
$$\underline{\underline{n=3, r=2}}$$

aaa
aab

aac
abb

bbc

$$2^n: {}^n C_0 + {}^n C_1 + \dots + {}^n C_n$$



$${}^3 C_2 = \frac{3!}{2!1!} = 3$$

String → aabbac $k = 3$

$$a \rightarrow 3$$

$b \rightarrow 2$

C → L

Level → Character

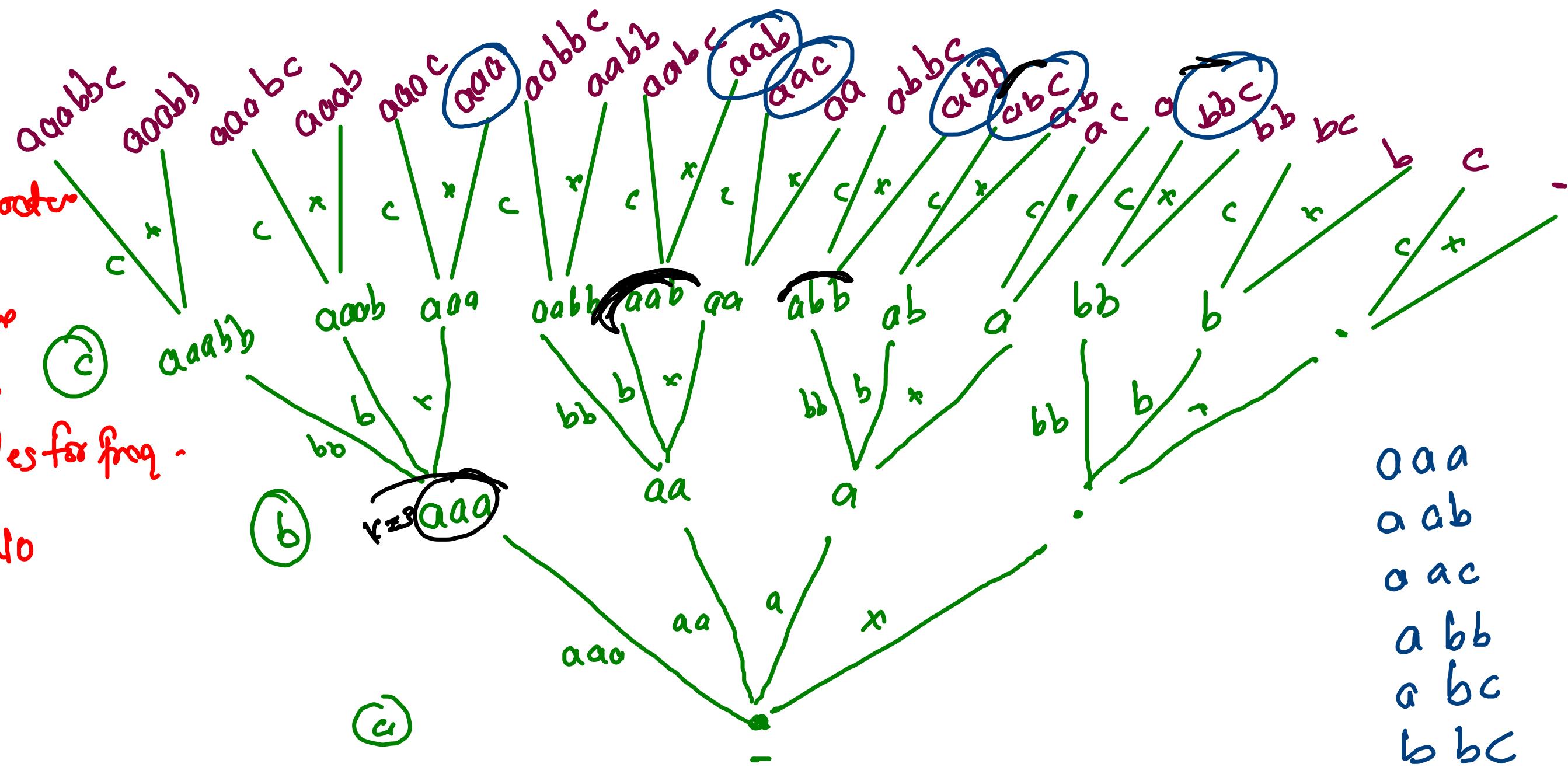
۱۰

二四

三

Option → Yes for freq.

No



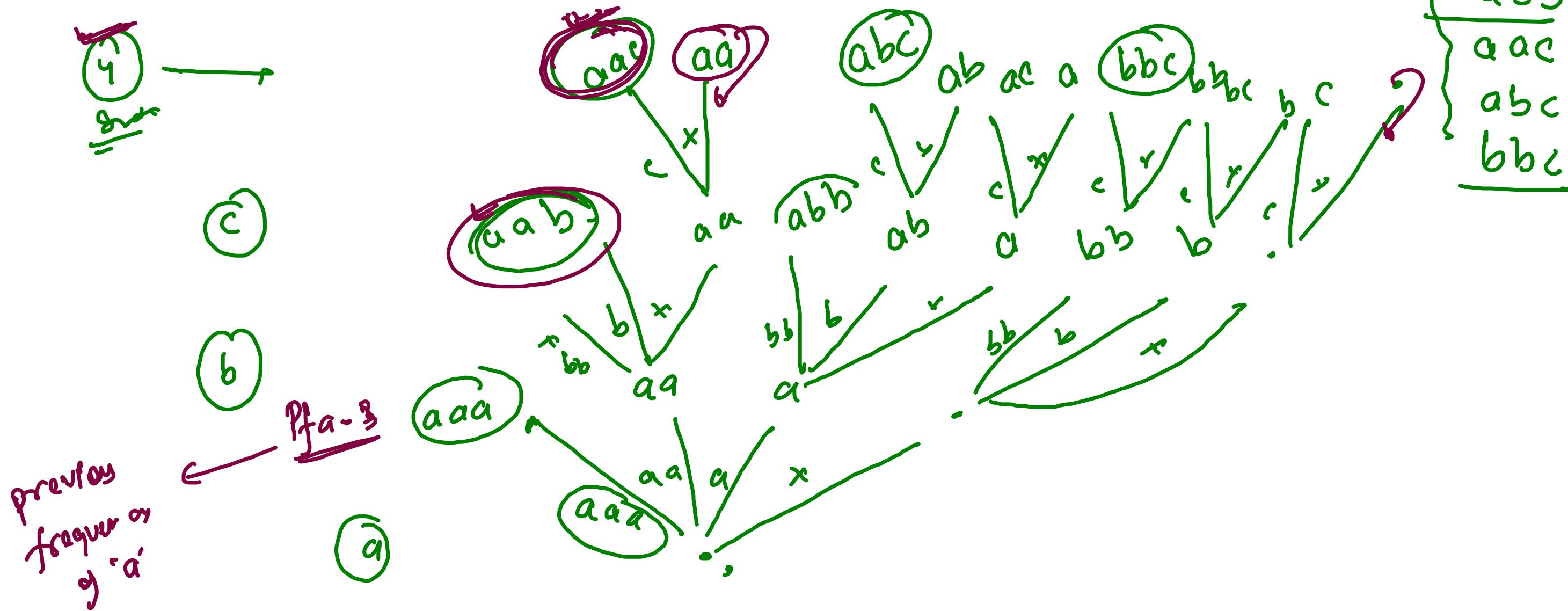
a → 3

b → 2

c → 1

K = 3

- (1) current character
 (2) asf] string.
 (3) freq, map.
 (4) unique str'g.
- last level

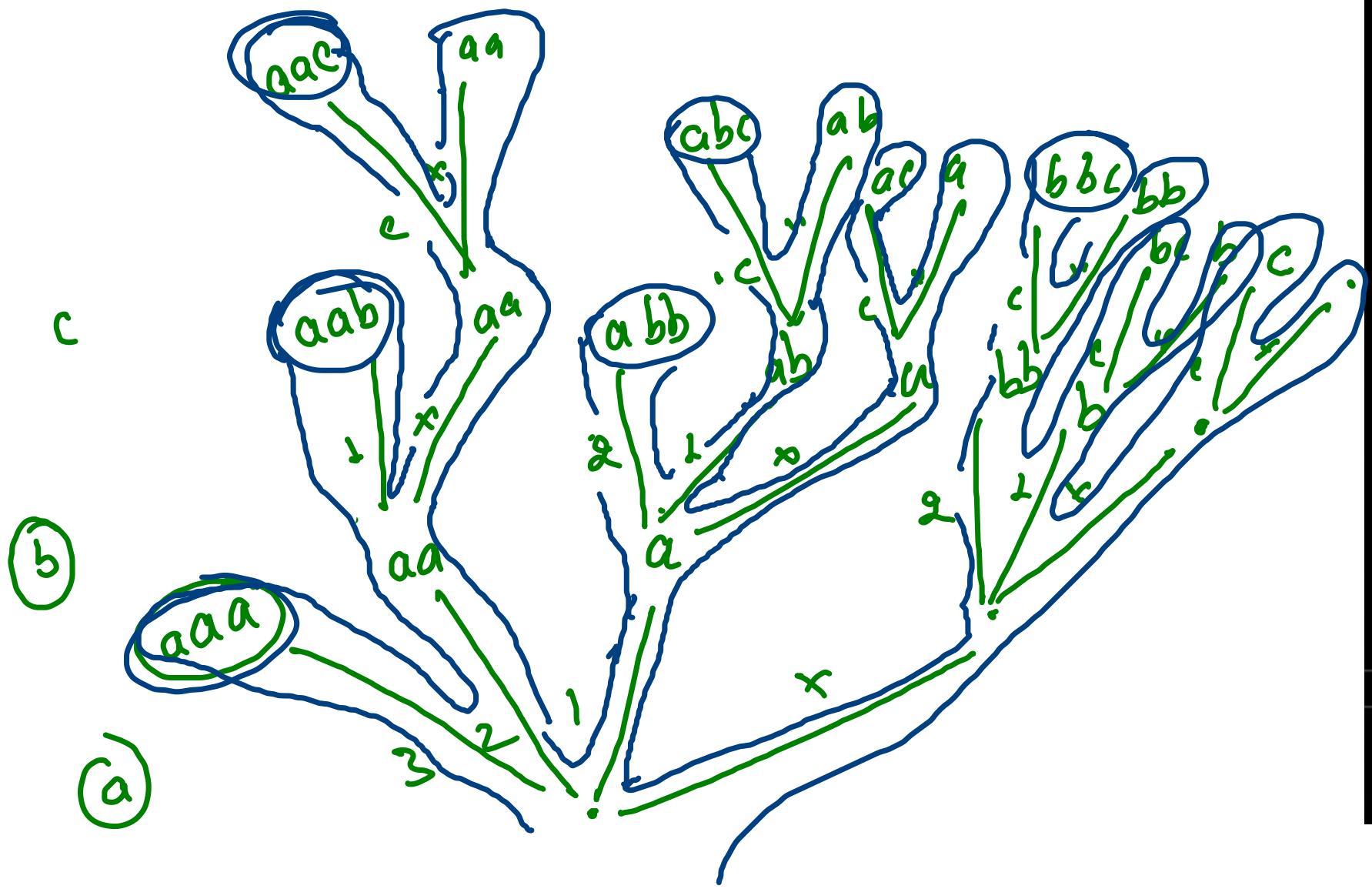


a3

b2

c1

word = abc



```
// Words -- K Selection -- 3, cc-> current character
public static void combination(String ustr, int cc, HashMap<Character, Integer> fmap, String asf, int ssf, int k) {
    if(ssf == k) {
        System.out.println(asf);
        return;
    }
    if(cc == ustr.length()) return;
    char ch = ustr.charAt(cc);
    int freq = fmap.get(ch);

    // yes call
    for(int i = freq; i > 0; i--) {
        if(i + ssf <= k) {
            String str = "";
            for(int j = 0; j < i; j++) {
                str += ch;
            }
            combination(ustr, cc + 1, fmap, asf + str, k);
        }
    }
    // no call
    combination(ustr, cc + 1, fmap, asf, k);
}
```

aaa
aab
aac
abb
abc
bbc

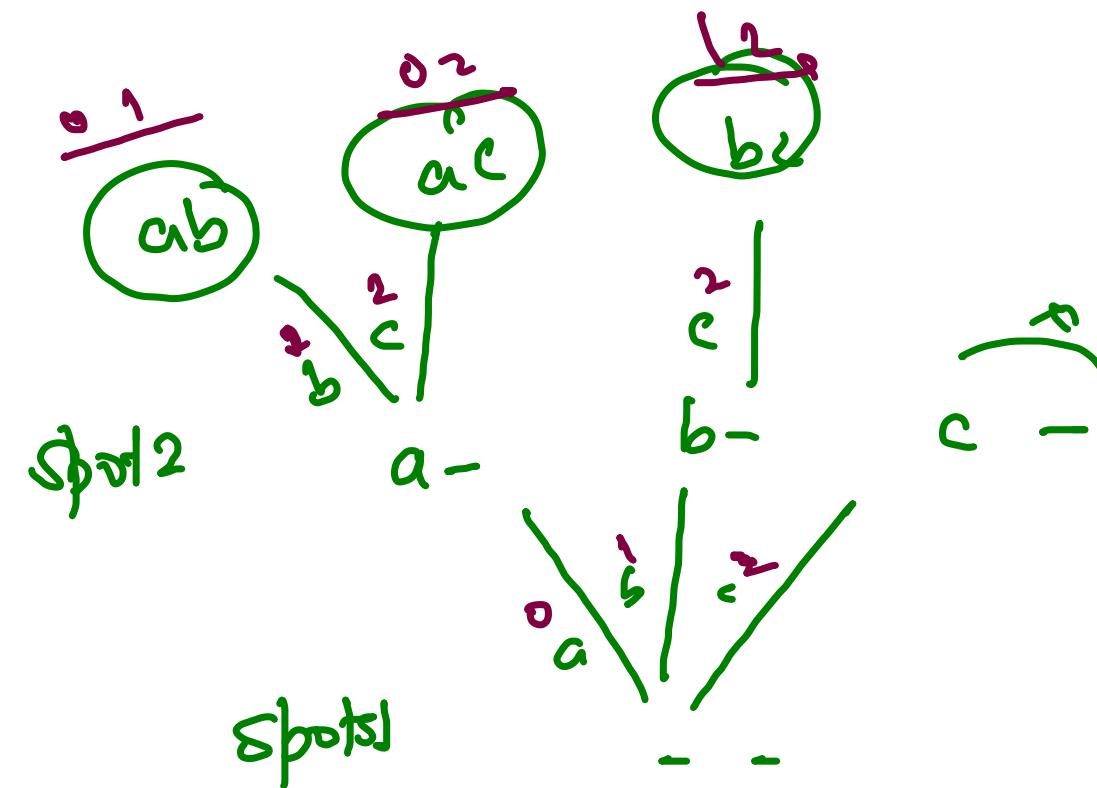
Klond-K-Selection-4

level → spots

option → character]

String → aabbac

unq. str = abc , $1 \leq 2$.



a a g
a a b
o a c
a b b
a b c
b b c

ab
ac
bc

NOTE :- Selection order
are basically
sorted in order

String \rightarrow aabbac $\underline{k=3}$

using \rightarrow abc

spots \rightarrow level

character \rightarrow option.

① unique string.

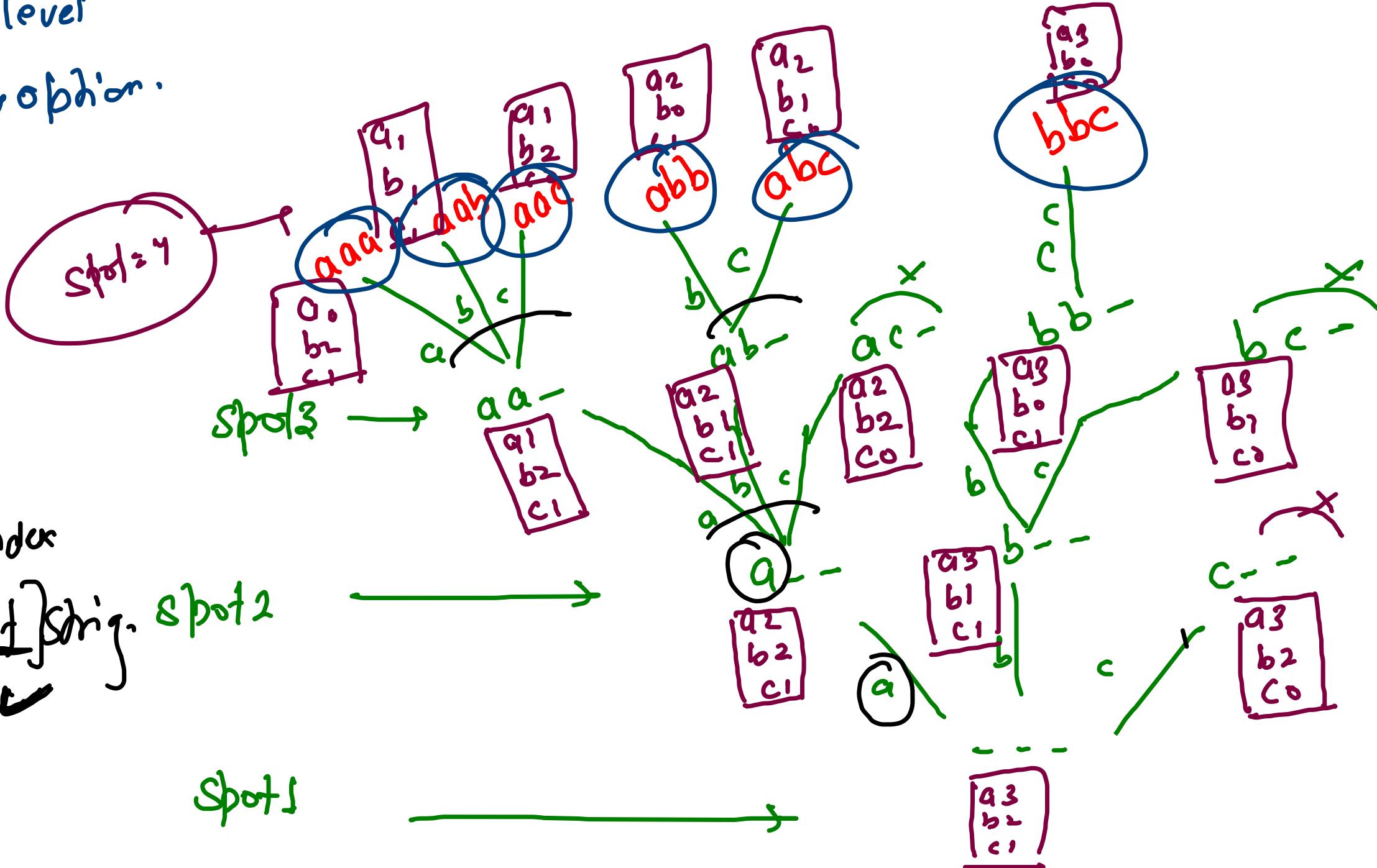
② tmap.

③ lost index

④ ~~Spot3 \rightarrow ast~~ string, spot2

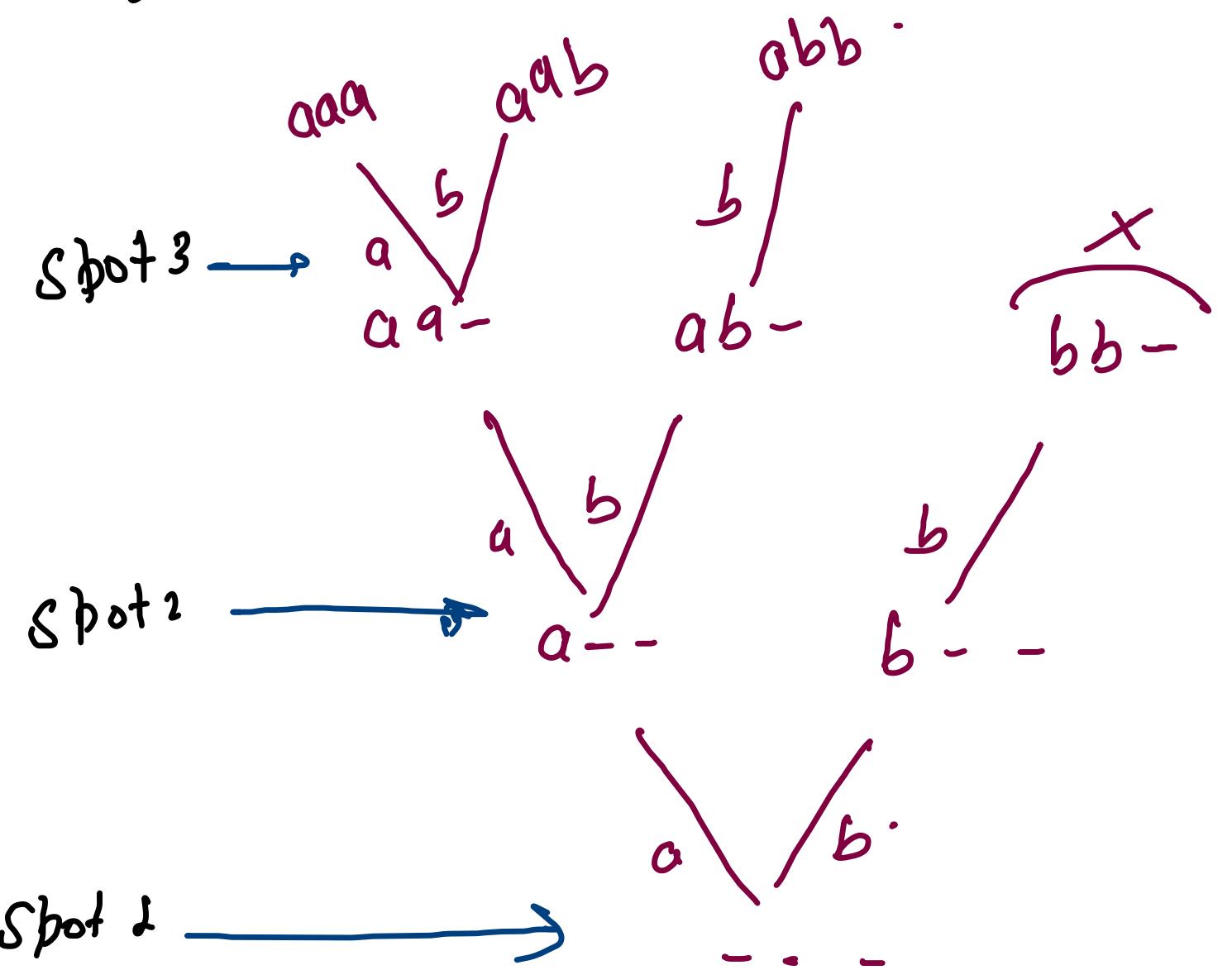
⑤ ts

tmap \rightarrow a $\rightarrow 3$
b $\rightarrow 2$
c $\rightarrow 1$



aaa
aab
aac
abb
abc
bbc

String → a a a a b b
a⁴ unique string - ab
b² k = 3



a a a
a a b
a b b

```
// Words -- K Selection -- 4, li->last index, cs-> current spot,
public static void combination(String ustr, HashMap<Character,
    if(cs == ts) {
        System.out.println(asf);
        return;
    }

    for(int i = li, i < ustr.length(); i++) {
        char ch = ustr.charAt(i);
        int freq = fmap.get(ch);
        if(freq > 0) {
            fmap.put(ch, freq - 1);
            combination(ustr, fmap, i, asf + ch, cs + 1, ts);
            fmap.put(ch, freq);
        }
    }
}
```

Summarising →

| | | | | | | | |
|--|--------|----|--------|---|----------|-------|----|
| </> Nqueens Combinations - 2d As 1d - Queen Chooses | Medium | 10 | ✓ Auth | 0 | ✓ Public | ✓ Sol | 10 |
| </> Nqueens Permutations - 2d As 1d - Queen Chooses | Easy | 10 | ✓ Auth | 0 | ✓ Public | ✓ Sol | 11 |
| </> Nknight Combinations - 2d As 1d - Knight Chooses | Medium | 10 | ✓ Auth | 0 | ✓ Public | ✓ Sol | 12 |
| </> Permutations - Words - 1 | Easy | 10 | ✓ Auth | 0 | ✓ Public | ✓ Sol | 13 |
| </> Permutations - Words - 2 | Easy | 10 | ✓ Auth | 0 | ✓ Public | ✓ Sol | 14 |
| </> Words - K Selection - 1 | Easy | 10 | ✓ Auth | 0 | ✓ Public | ✓ Sol | 15 |
| </> Words - K Selection - 2 | Easy | 10 | ✓ Auth | 0 | ✓ Public | ✓ Sol | 16 |
| </> Words - K Length Words - 1 | Easy | 10 | ✓ Auth | 0 | ✓ Public | ✓ Sol | 17 |
| </> Words - K Length Words - 2 | Easy | 10 | ✓ Auth | 0 | ✓ Public | ✓ Sol | 18 |
| </> Words - K Selection - 3 | Easy | 10 | ✓ Auth | 0 | ✓ Public | ✓ Sol | 19 |
| </> Words - K Selection - 4 | Easy | 10 | ✓ Auth | 0 | ✓ Public | ✓ Sol | 20 |
| </> Words - K Length Words - 3 | Easy | 10 | ✓ Auth | 0 | ✓ Public | ✓ Sol | 21 |
| </> Words - K Length Words - 4 | Easy | 10 | ✓ Auth | 0 | ✓ Public | ✓ Sol | 22 |
| </> Coin Change - Combinations - 1 | Medium | 10 | ✓ Auth | 0 | ✓ Public | ✓ Sol | 23 |
| </> Coin Change - Combinations - 2 | Medium | 10 | ✓ Auth | 0 | ✓ Public | ✓ Sol | 24 |
| </> Coin Change - Permutations - 1 | Medium | 10 | ✓ Auth | 0 | ✓ Public | ✓ Sol | 25 |
| </> Coin Change - Permutations - 2 | Medium | 10 | ✓ Auth | 0 | ✓ Public | ✓ Sol | 26 |

*Combination or
selection and option*

unique selection

in every unique character

selection

on unique words

in every word unique character

permutation

unique word

*character maybe
repeated in a word*

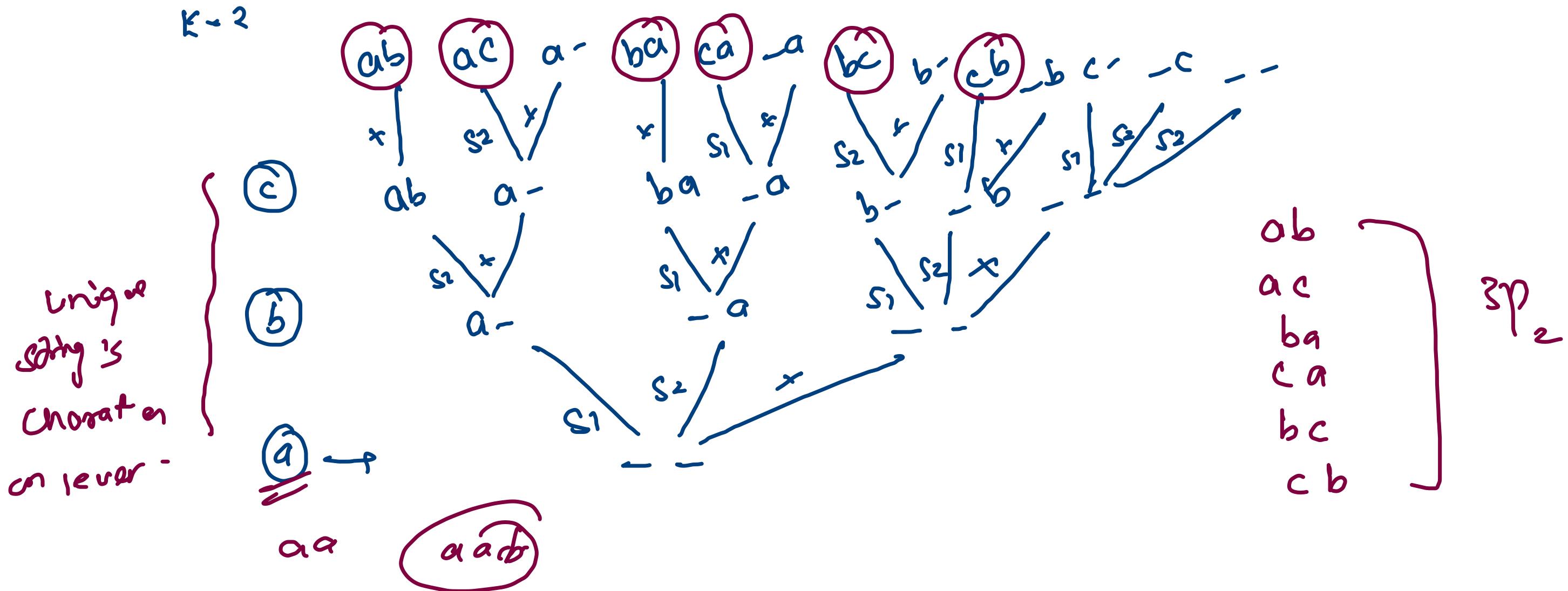
*selection
but character
no may repeat*

Words - k lengths word - 3

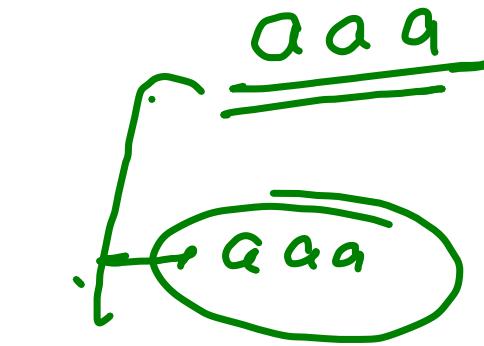
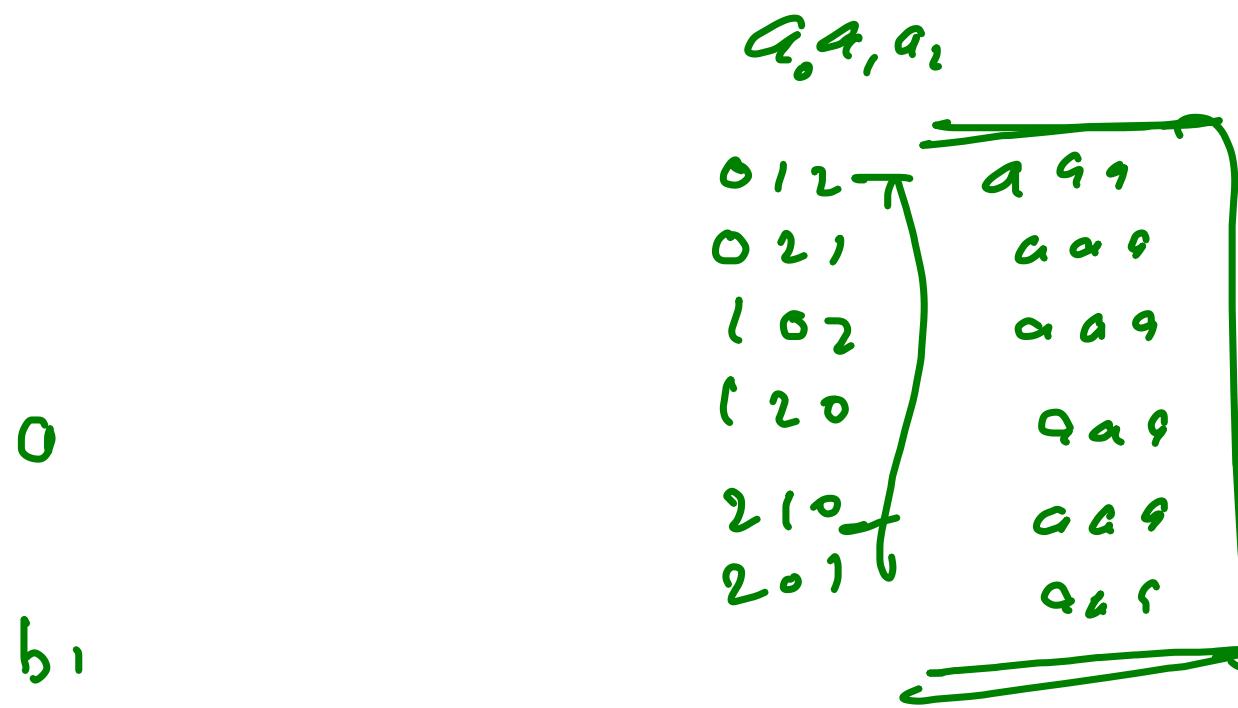
Level - character : word.k lengths word - 1

String \rightarrow aabbac
unique string \rightarrow abc]

$${}^3P_2 = \frac{3!}{1!} = 3 \times 2 \times 1 = 6$$



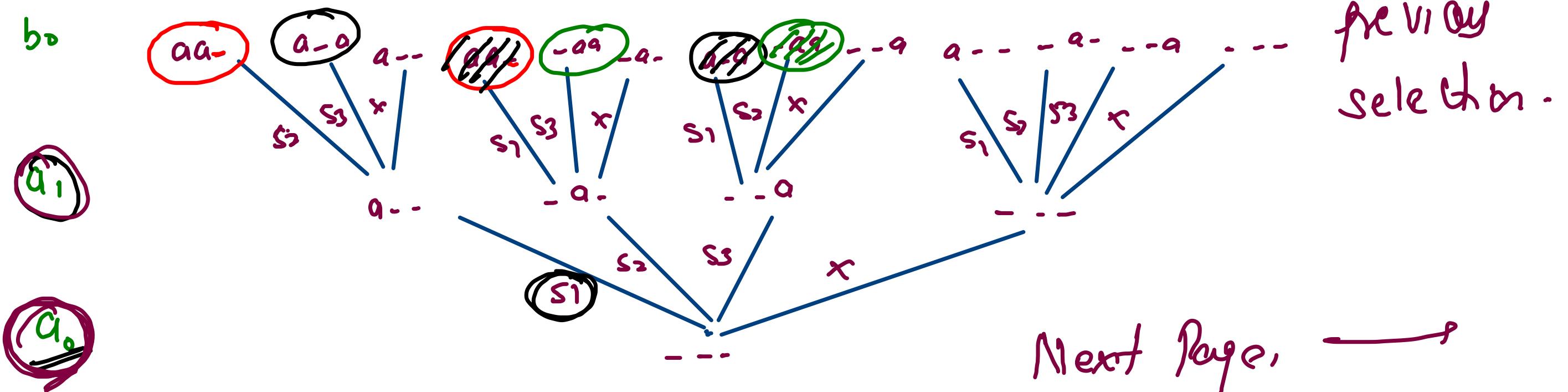
String \rightarrow aabbac $k=3$



Repetition

$\begin{cases} aab \\ aab \end{cases}$

To avoid permutation of some character, we run generation, ahead from previous selection.



Level Character →

After 'last' location preserve, we can't stop

rebirth

