Word - K length word - 2

String → abc abc , unique string - abc

k = 2.

ab ba
ac ca
bc cb

3 box 2 distinct items (1, 2) slots

- - -

12 -
1 - 2
- 12

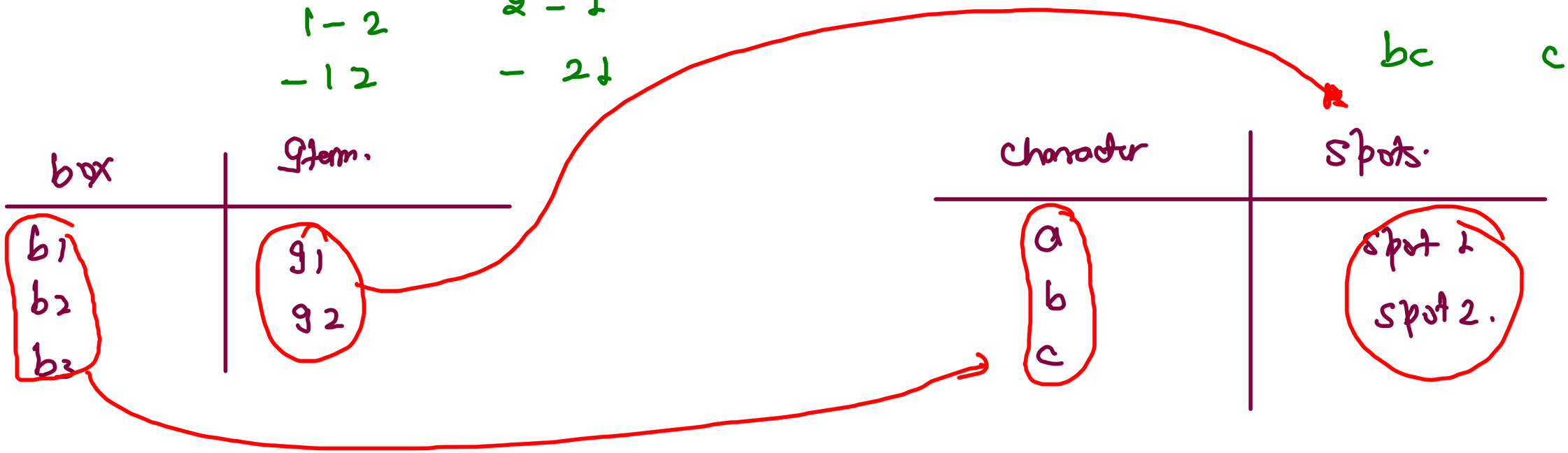
21 -
2 - 1
- 21

3 character

a b c

pick 2 distinct character

ab ba
ac ca
bc cb.



Item: level

box=3

Item: i_1 . i_2

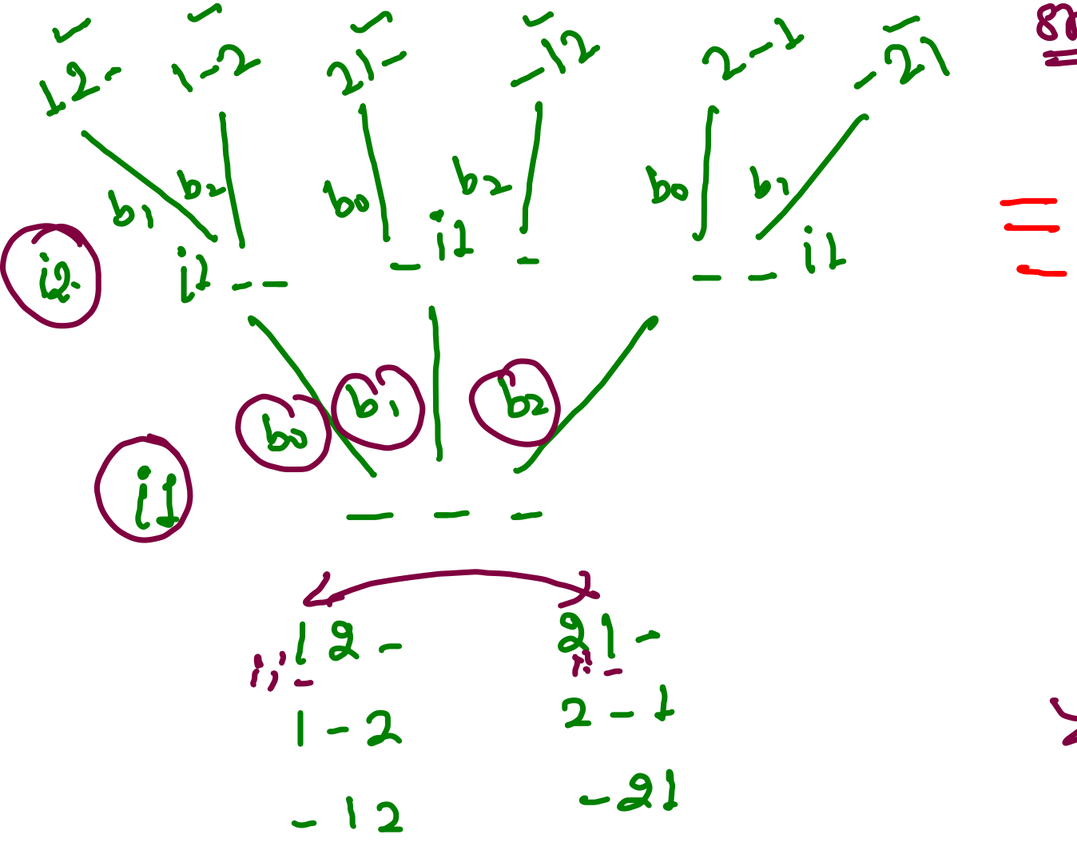
① Unique string.

② HashSet of already present element to make call

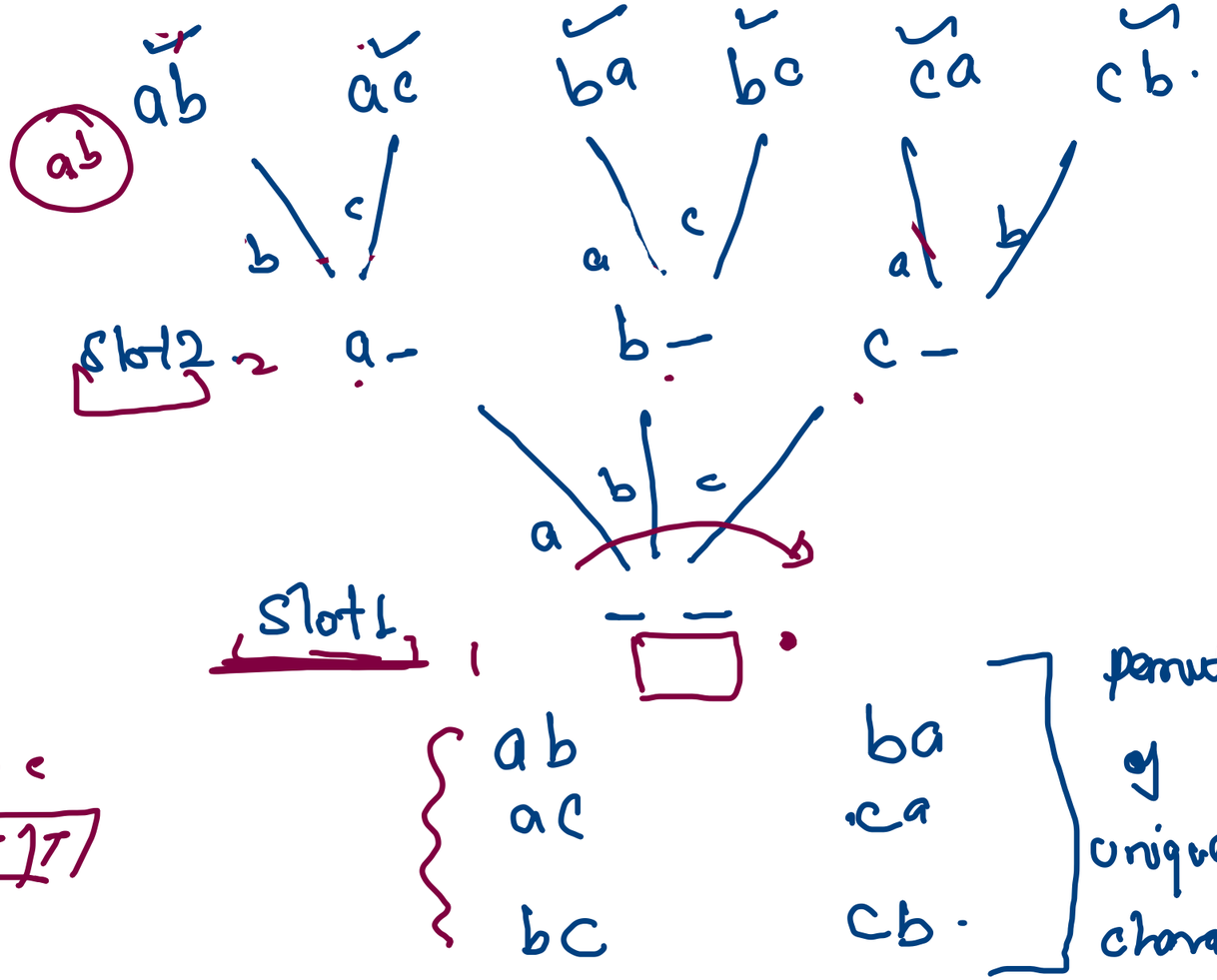
③ current slot

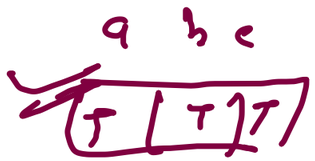
④ Total Slot

⑤ answer so for string.



=





Words- K. Selection. 3

String → a a b b a c

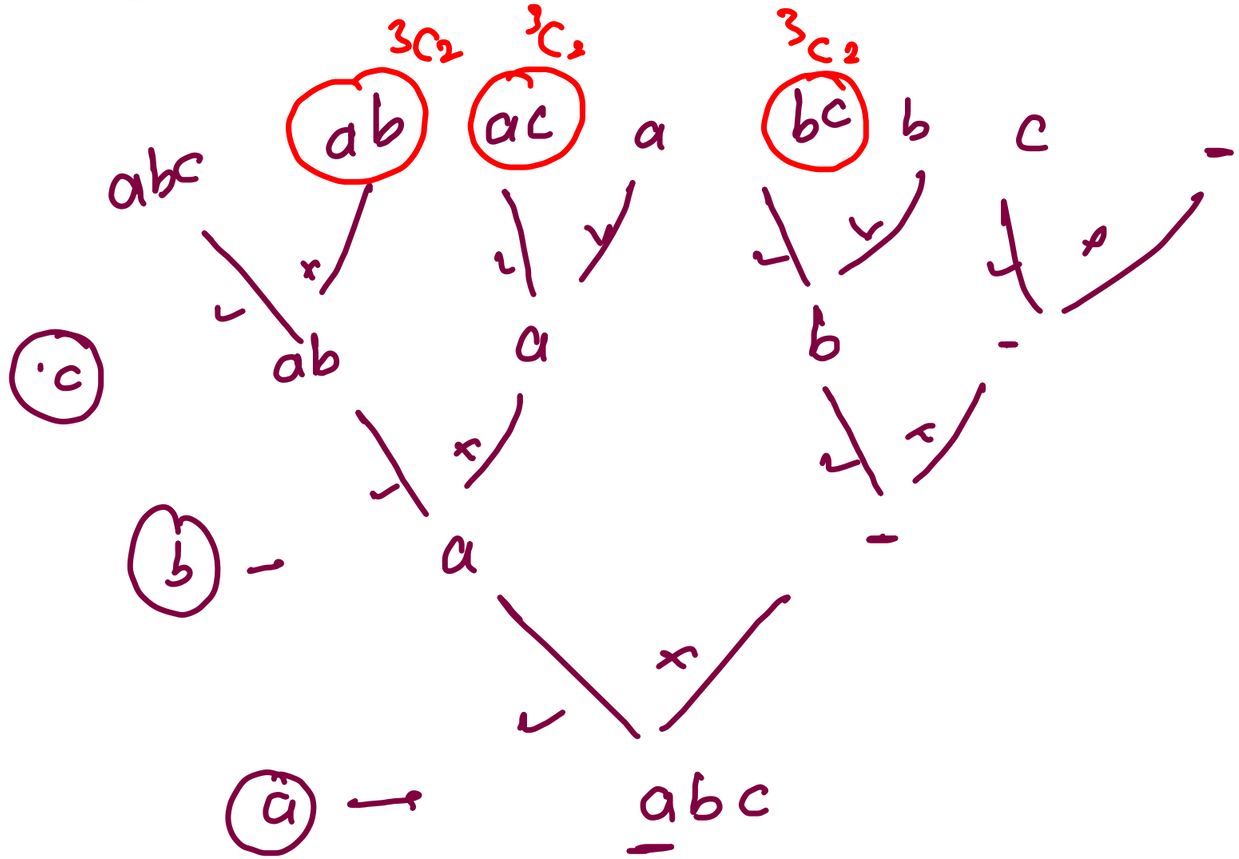
k = 3

Ustr → a b c

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

$n=3,\ r=2$

$^3C_2$ $^3C_2$ $^3C_2$



aaa
aab
aac
abb
bbc
-

$2^n = {}^nC_0 + {}^nC_1 + \cdots {}^nC_n$

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

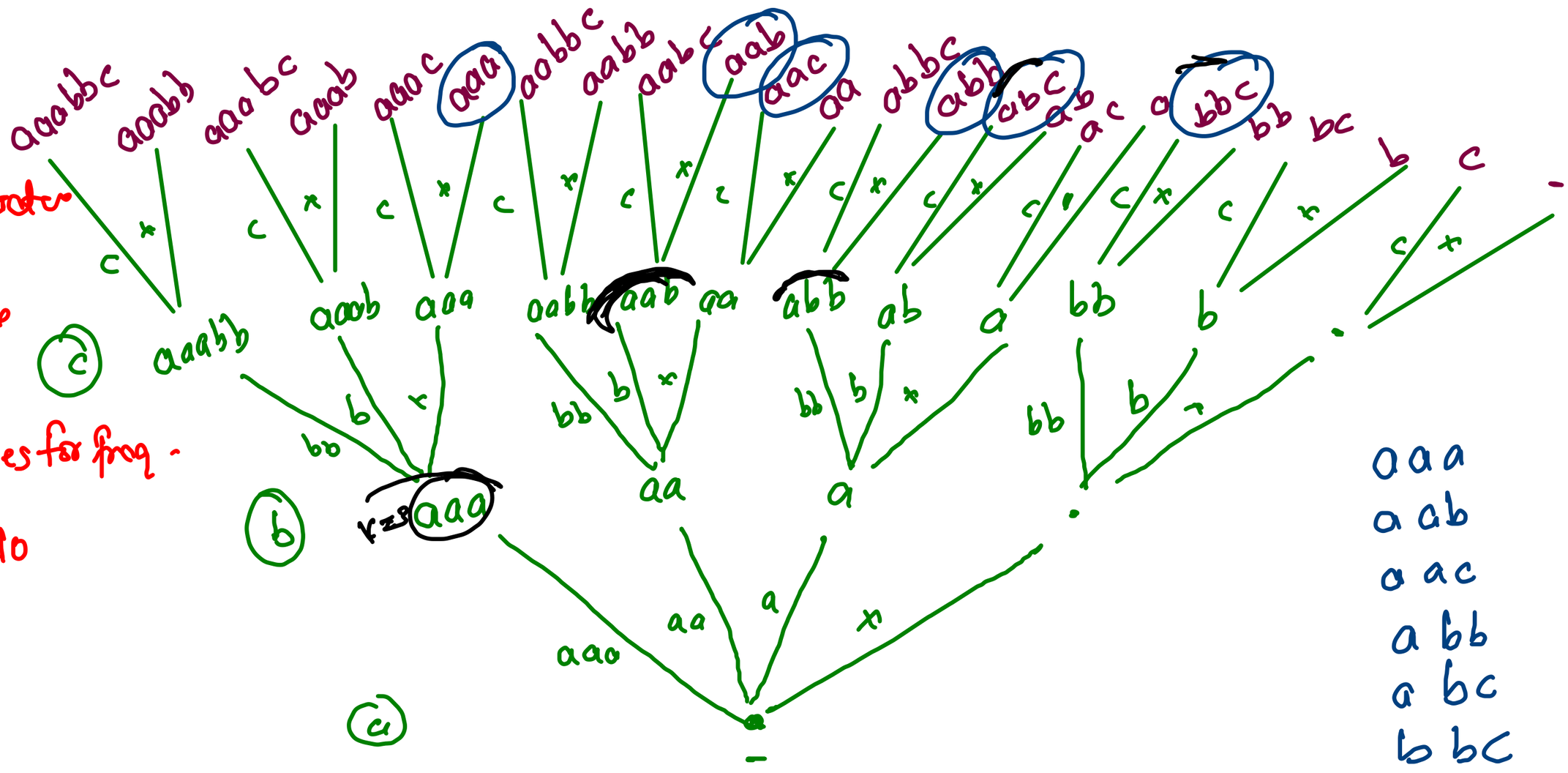
String → aabbac k = 3

a → 3
b → 2
c → 1

Level → Character of unique string

Option → Yes for freq.

No



a → 3
b → 2
c → 1

K = 2.

→ ① current character
② asf] string.
③ freq. map.
④ unique str'g.

last level

④ → string

aaa
aab
abb
aac
abc
bbc

aa
ab
a
bb
b
.

aaa
aab
abb
aa
ab
a
bb
b
.

aaa
aa
a
.

aaa
aa
a
.

③ → Pfa-3

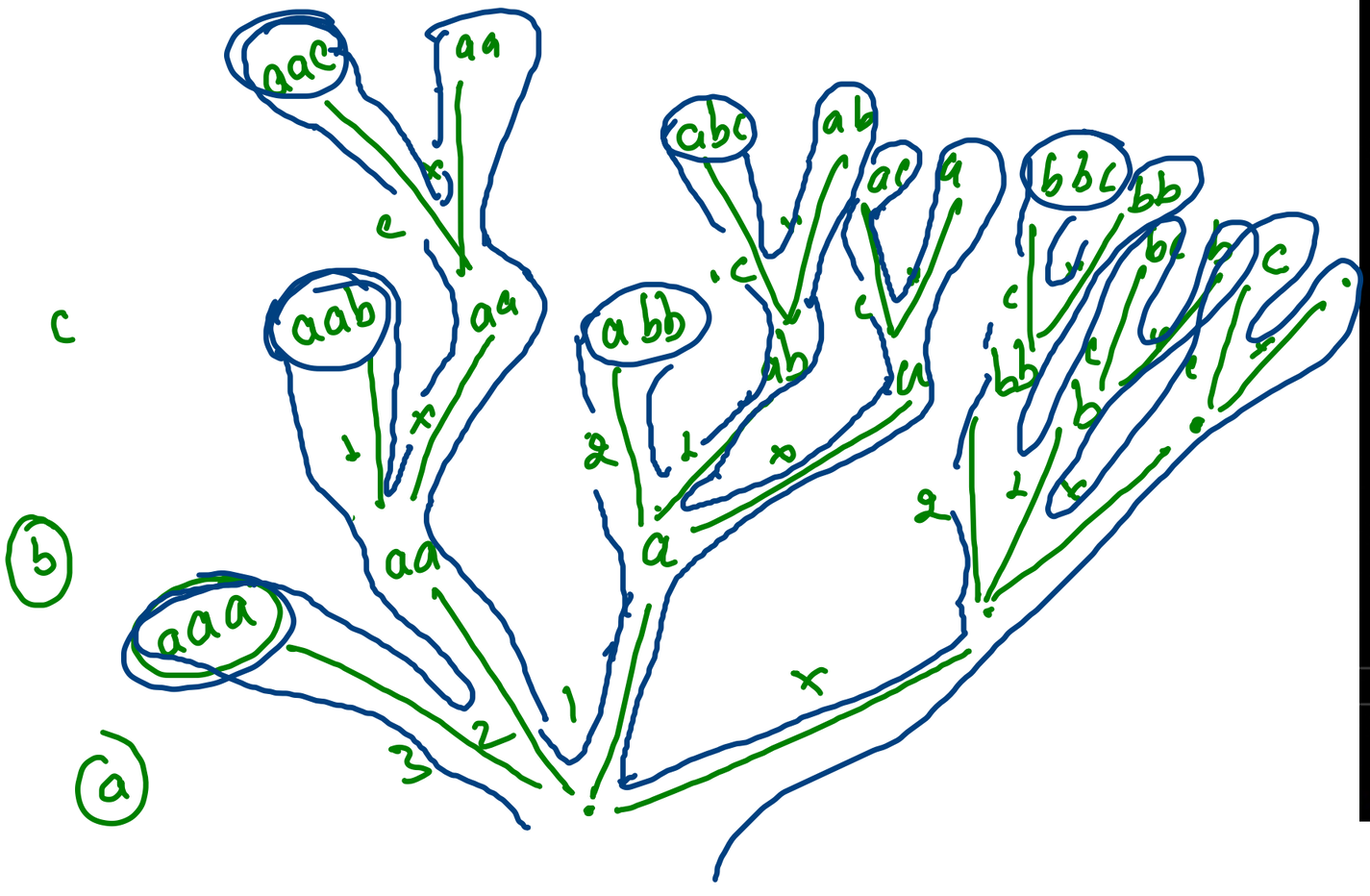
previous frequency of 'a'

① ② ③ ④

a3
b2
c1

k=3

ustr = abc




```java
// Words - K Selection - 3, cc-> current character
public static void combination(String ustr, int cc, HashMap<C
    int ssf = asf.length();

    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

Klord - K - Selection - 4

aabbac ⟶

aaa
aab
aac
abb
abc
bbc

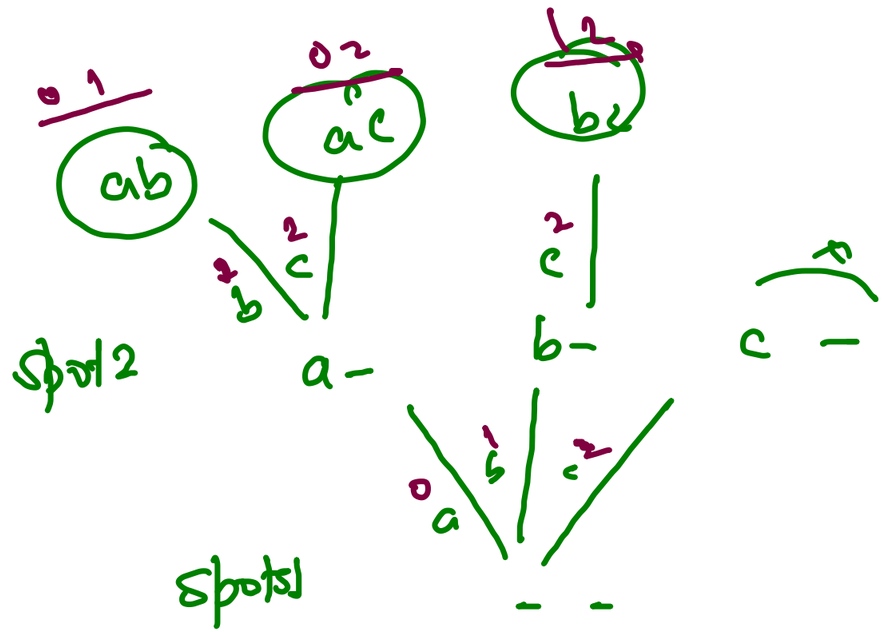
Level → Spots

option → character

String → aabbac

uniq. str = abc

, k = 2.



ab
ac
bc

Note :- Selection order are basically sorted in order

String → aabbac

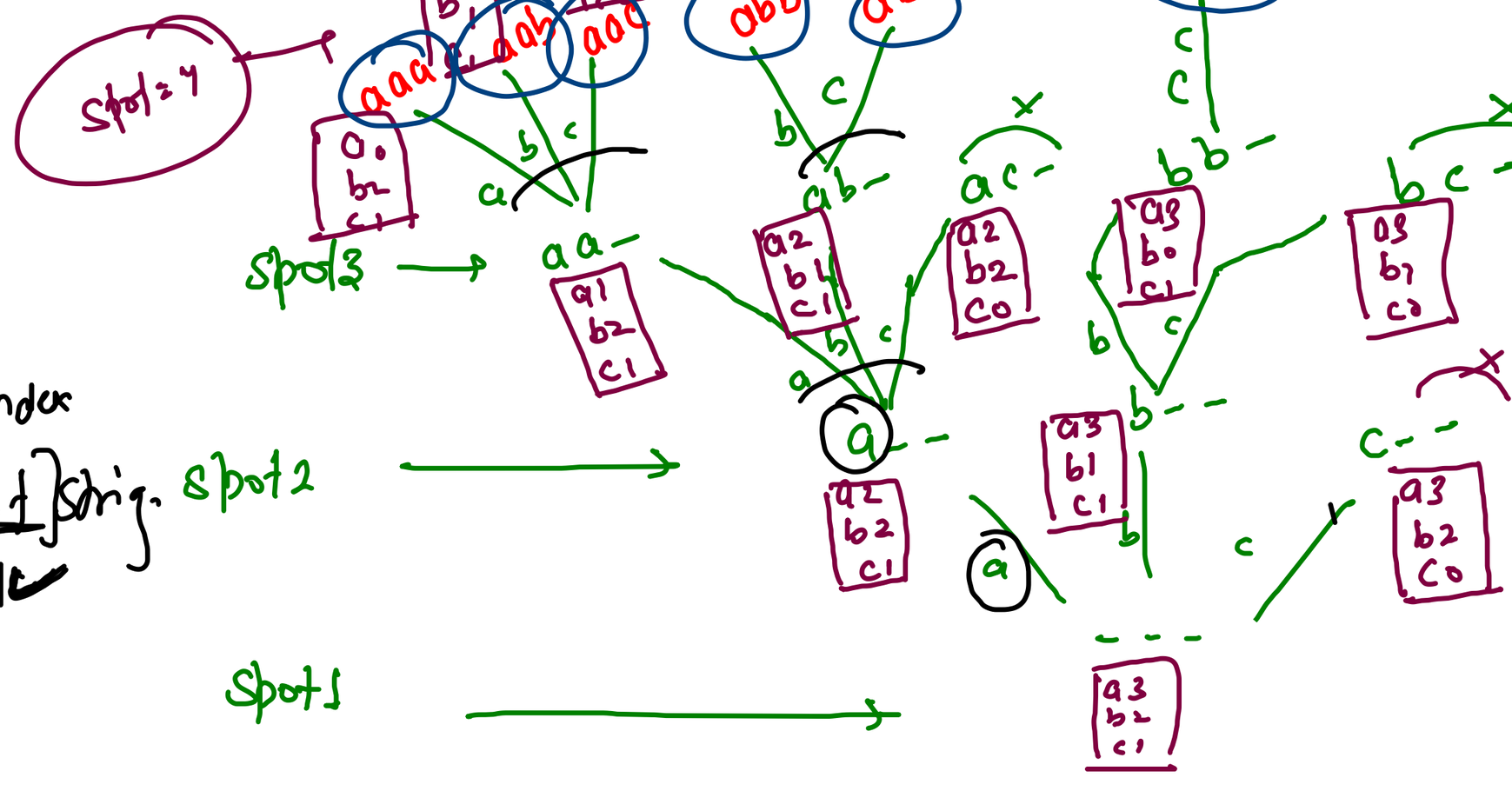
ustring → abc

spots → level

character → option.

K=3

fmap → a → 3
b → 2
c → 1



aaa
aab
aac
abb
abc
bbc

(1) unique string.
(2) fmap.
(3) [0:1] index
(4) spots → as [string.
(5) cs
(6) ts

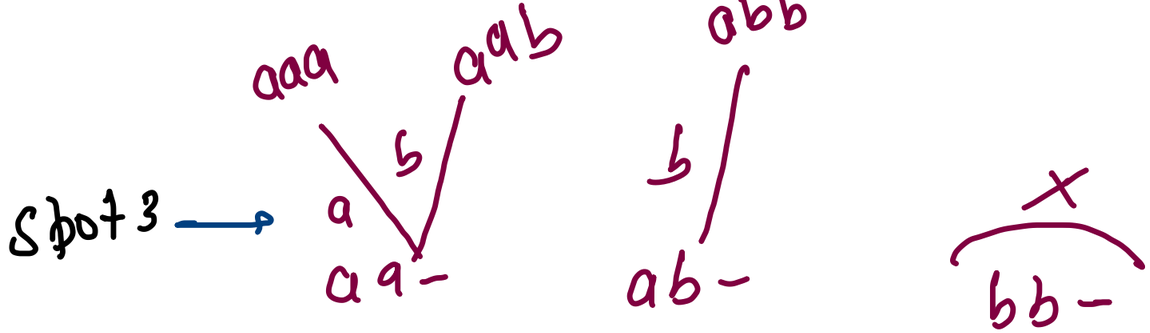
String → a a a a b b

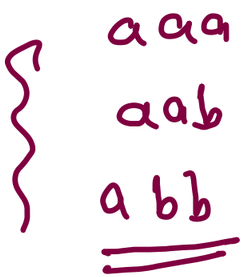
a 4

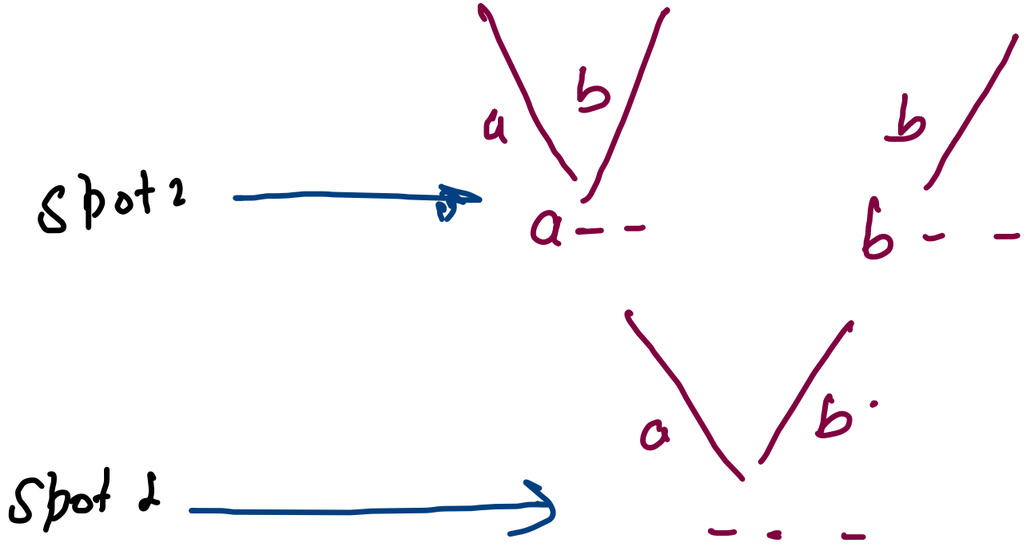
b 2

unique String - a b

k = 3








```java
// 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);
        }
    }
}
```

Summarisation :→

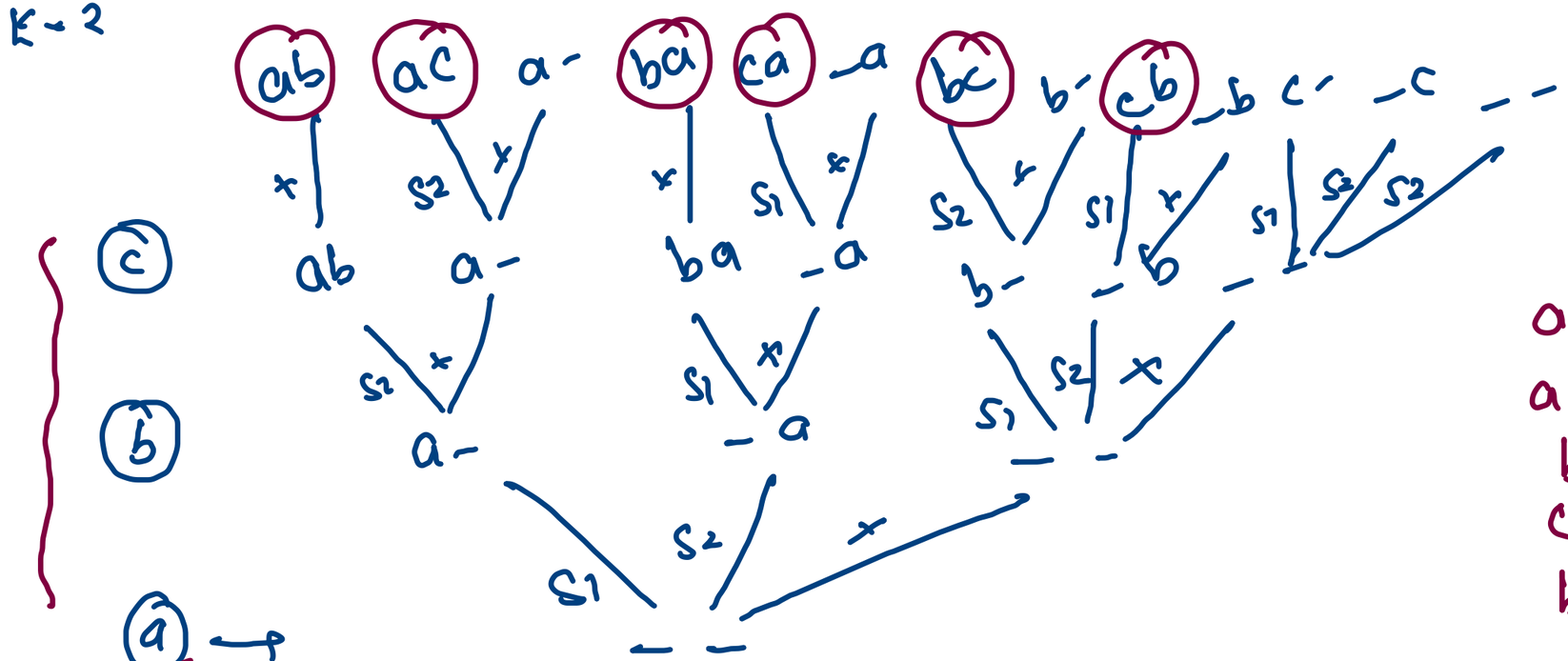
Words- k lengths word -3

Level - character. words. k length word - 1

String → aabbac

unique string → abc] $^3P_2 = \frac{3!}{1!} = 3 \times 2 \times 1 =$ (c)

K - 2



ab
ac
ba
ca
bc
cb

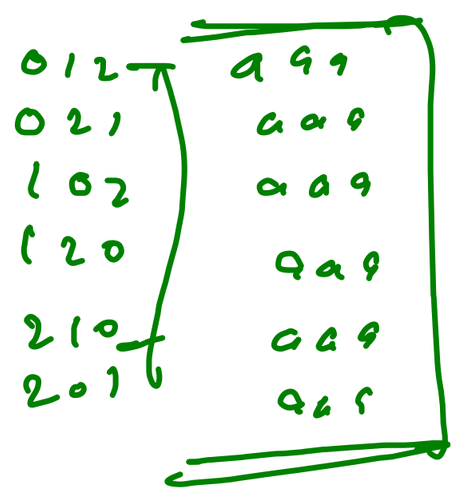
$^3P_2$

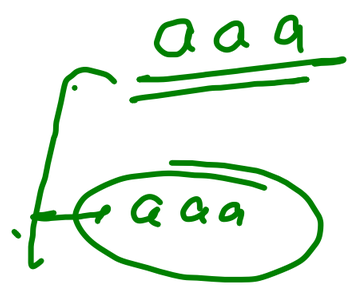
String → aabbac k-3

$q_0 q_1 q_2$

0

b:

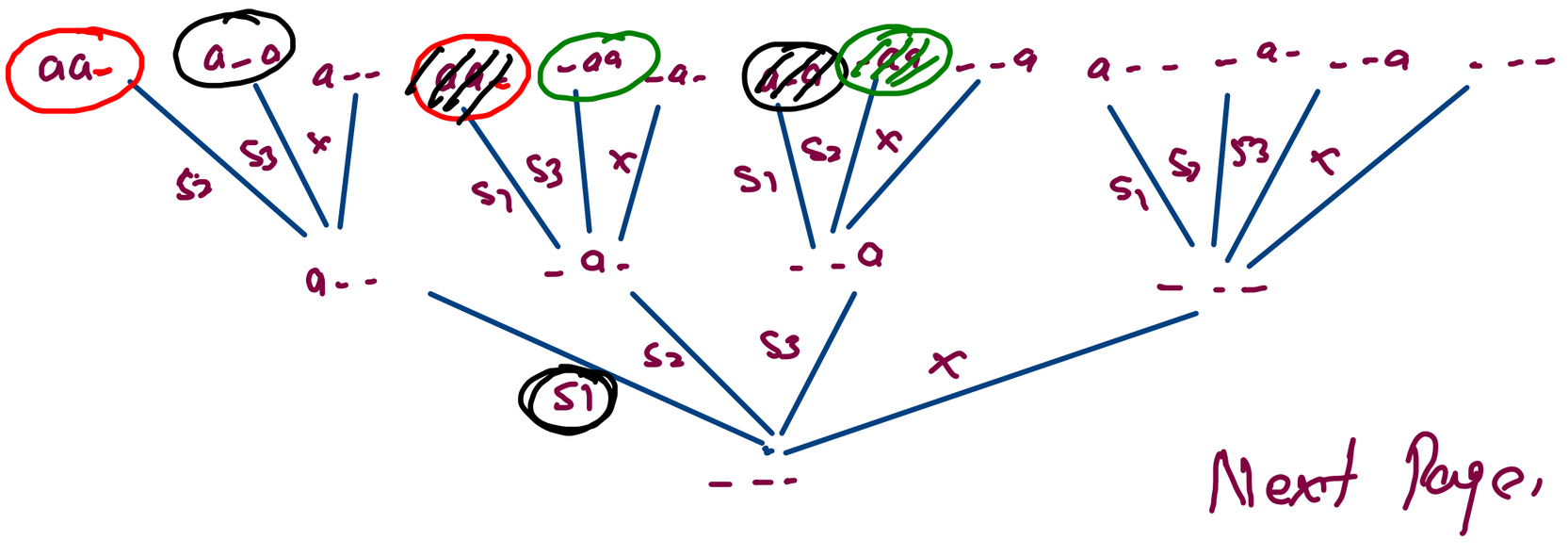




Repition [aab / aab]

To avoid permutation of same character, we Run iteration, ahead from previous selection.

b:



Next Page. →

Level Character → After last location preserve, we can't stop repitition.