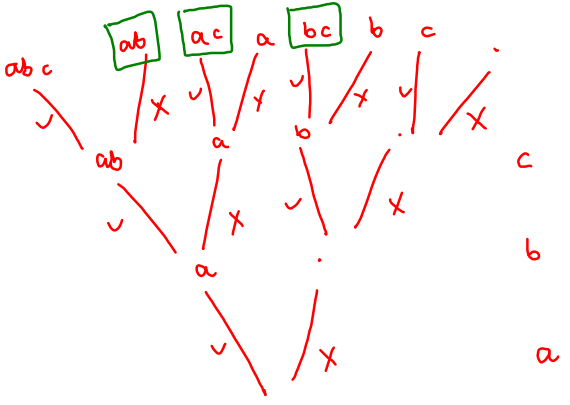
Words K selection - 1

distinct letters

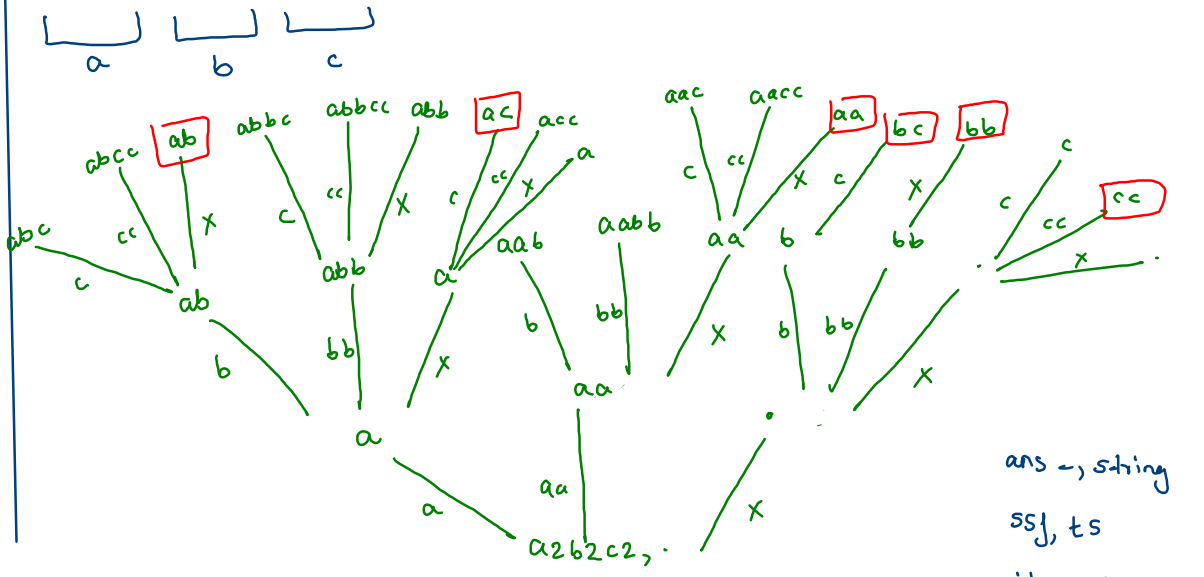
Str : abcbac K = 2

ustr : abc



Str : abcbac

K = 2



```
if(idx == ustr.length()) {
    if(ssf == ts) {
        System.out.println(asf);
    }
    return;
}

char ch = ustr.charAt(idx);
int freq = map.get(ch);

for(int i = freq; i >= 0;i--) {
    String s = "";

    for(int j = 1; j <= i;j++) {
        s += ch;
    }

    words_k_selection_3(idx+1,ustr,map,asf + s, ssf + i,ts);
}
```

str : a b c b

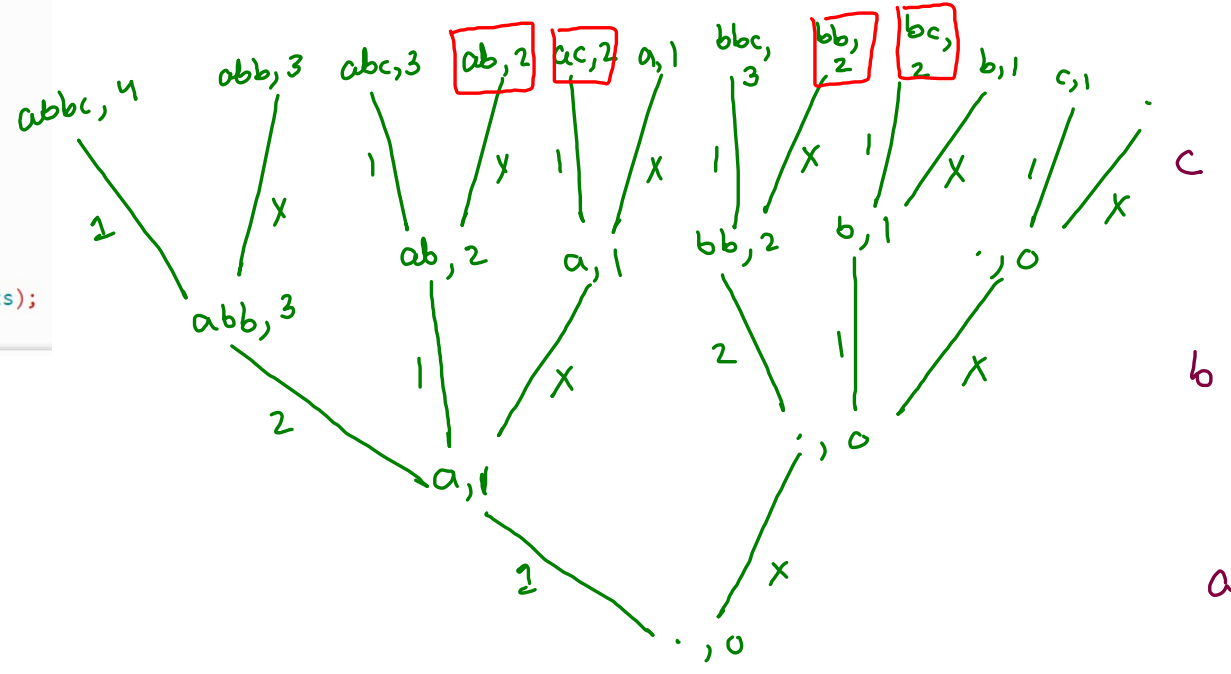
k = 2

ab

ac

bb

bc



ustr : abc

map : a -> 1

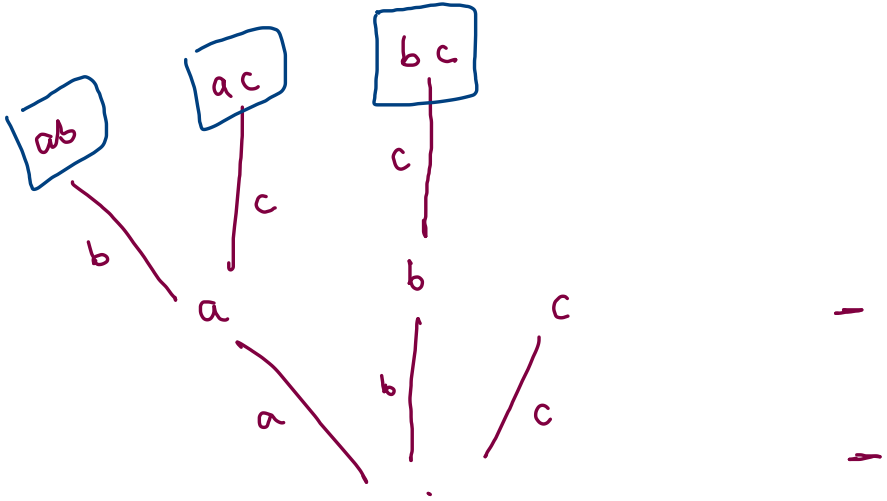
b -> 2

c -> 1

words - k - selection 2

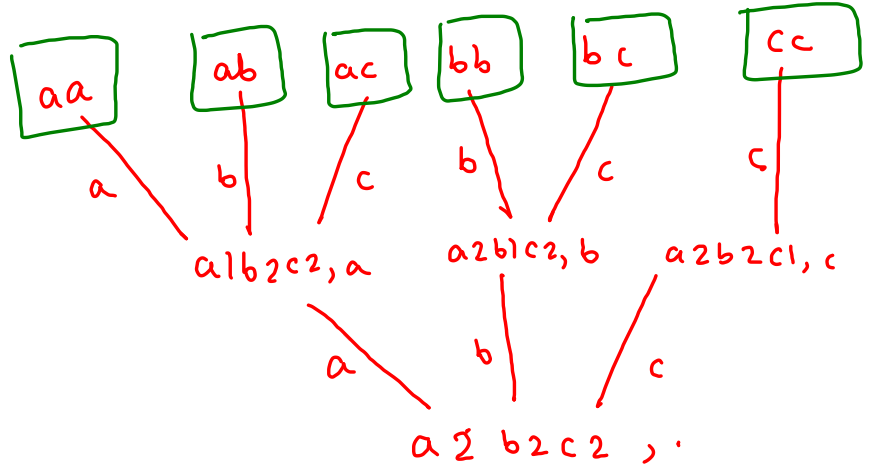
Str : a b c b a c (distinct)

k = 2



Str : a b c b a c

k = 2



```java
public static void words_k_selection_4(int cs,int ts,String ustr,HashMap<Character,Integer>map,String asf,int lci) {
    if(cs > ts) {
        System.out.println(asf);
        return;
    }

    for(int i = lci; i < ustr.length();i++) {
        char ch = ustr.charAt(i);

        if(map.get(ch) > 0) {
            map.put(ch,map.get(ch)-1);
            words_k_selection_4(cs+1,ts,ustr,map,asf + ch,i);
            map.put(ch,map.get(ch)+1);
        }
    }
}
```

str: abcac

k = 2

ustr : abc

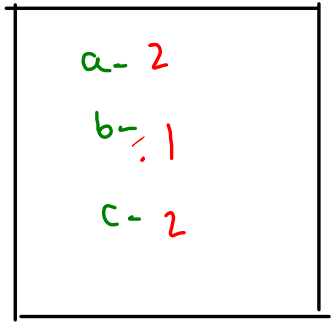
aa

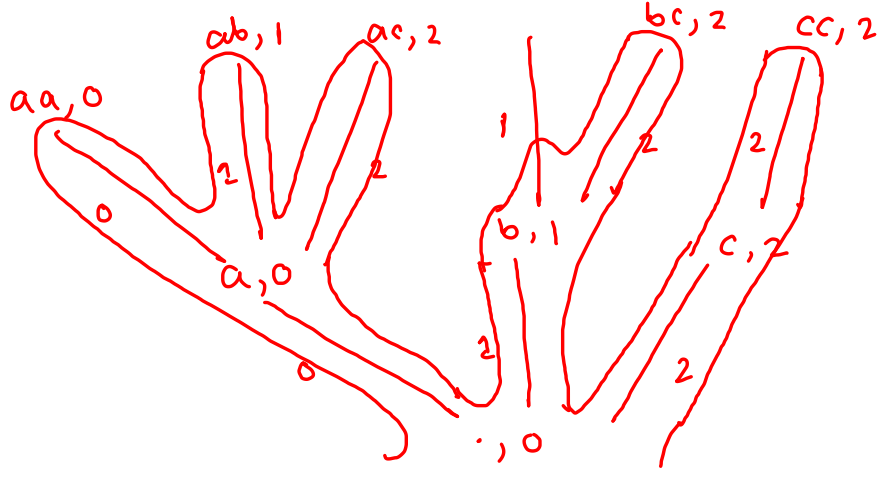
ab

ac

bc

cc

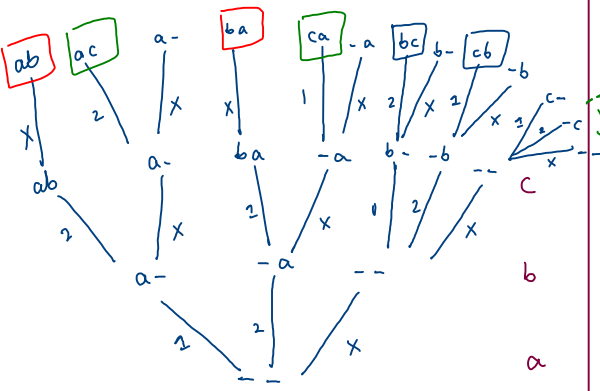




Words lk arrangement - 1

str: abcbac ustr: abc

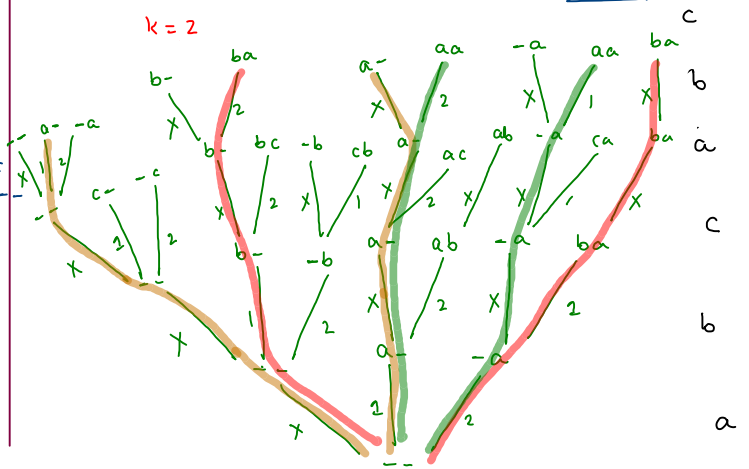
k = 2 (distinct)



Words lk arrangement - 3

str: abcabc incorrect

k = 2

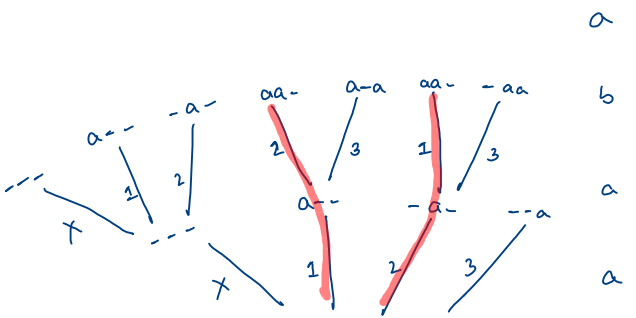


(i) don't give no choice to second 'a'.

(ii) last occurence ahead call to repeated char.

a1 -> x a2 -> ∪(1) a-

a1 -> ∪(1) a2 -> x a-



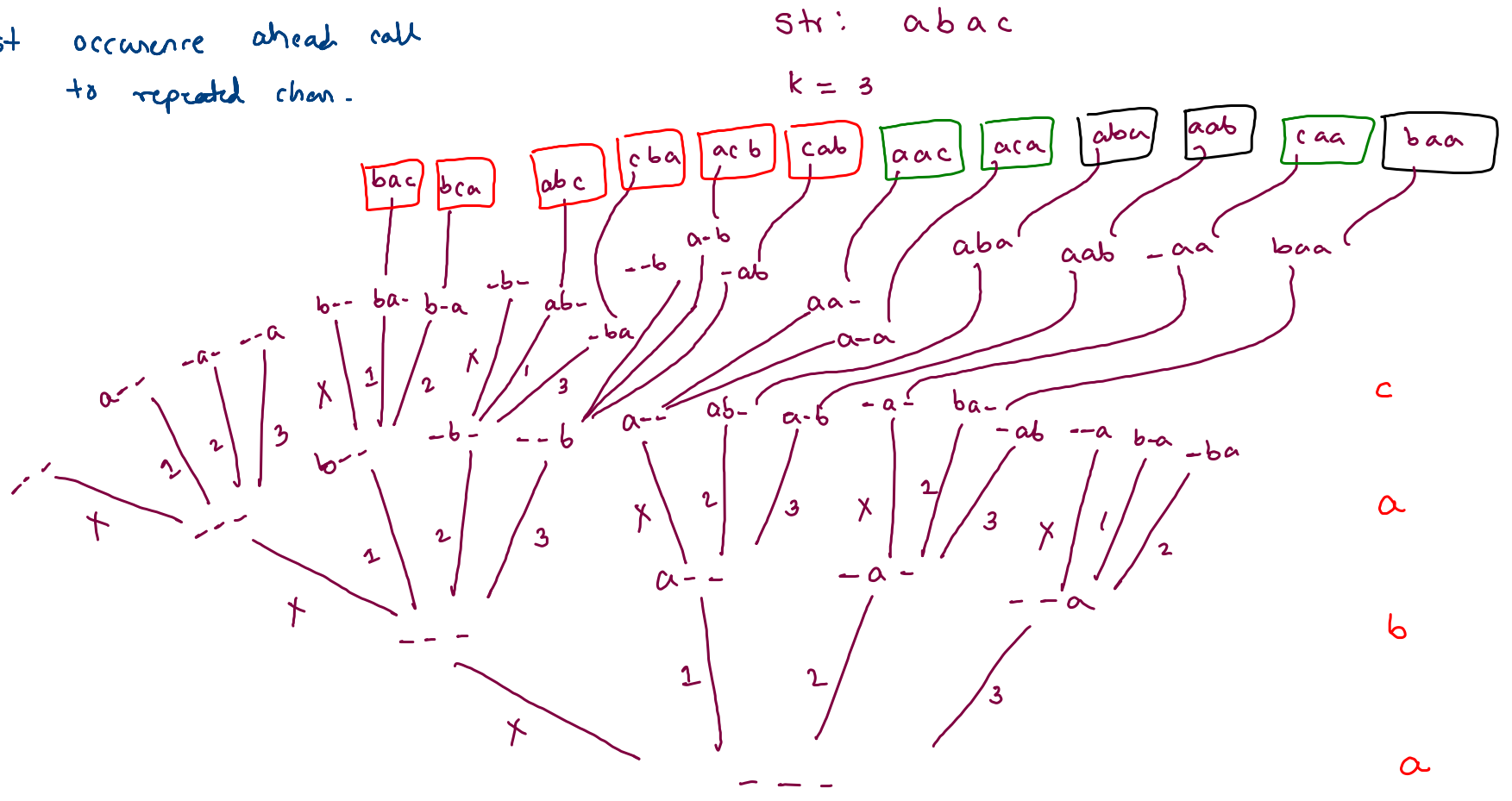
a1 a2 _

a2 a1 _

when char is present as:

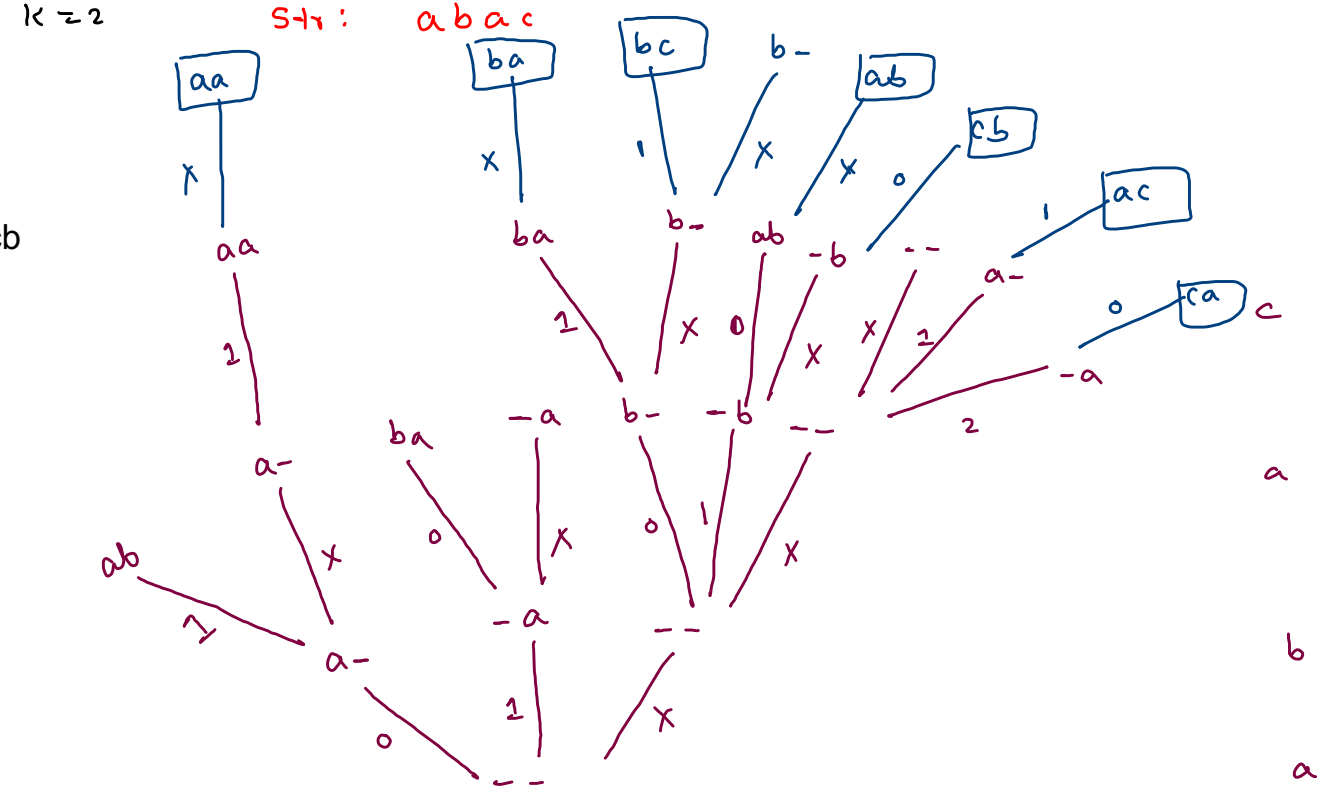
(i) don't give no choice to repeated char.

(ii) last occurence ahead call to repeated char.



```java
if(idx == str.length()) {
    if(ssf == ts) {
        for(int i=0; i < spots.length;i++) {
            System.out.print(spots[i]);
        }
        System.out.println();
    }
    return;
}


char ch = str.charAt(idx);
int loch = lastOcc.get(ch);

//yes call
for(int i = loch + 1; i < spots.length;i++) {
    if(spots[i] == '0') {
        spots[i] = ch;
        lastOcc.put(ch,i);
        words_k_arrangement_3(idx+1,str,ssf+1,ts,lastOcc,spots);
        lastOcc.put(ch,loch);
        spots[i] = '0';
    }
}

//no call
if(loch == -1) {
    words_k_arrangement_3(idx+1,str,ssf,ts,lastOcc,spots);
}
```

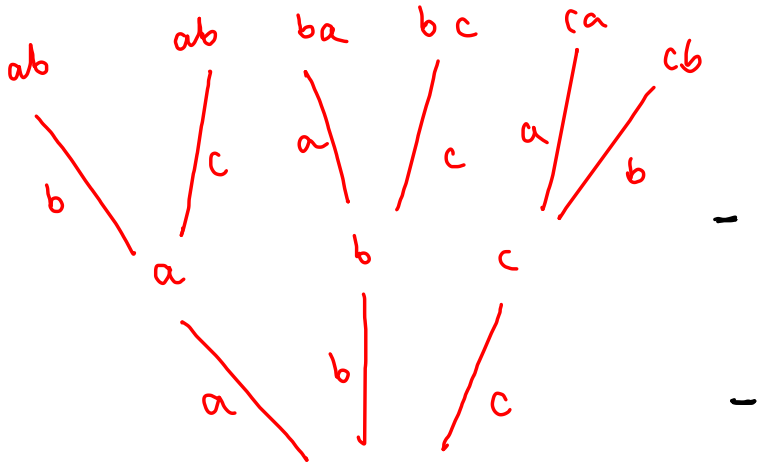
aa ba bc ab cb
ac ca



Words - k arrangement 2

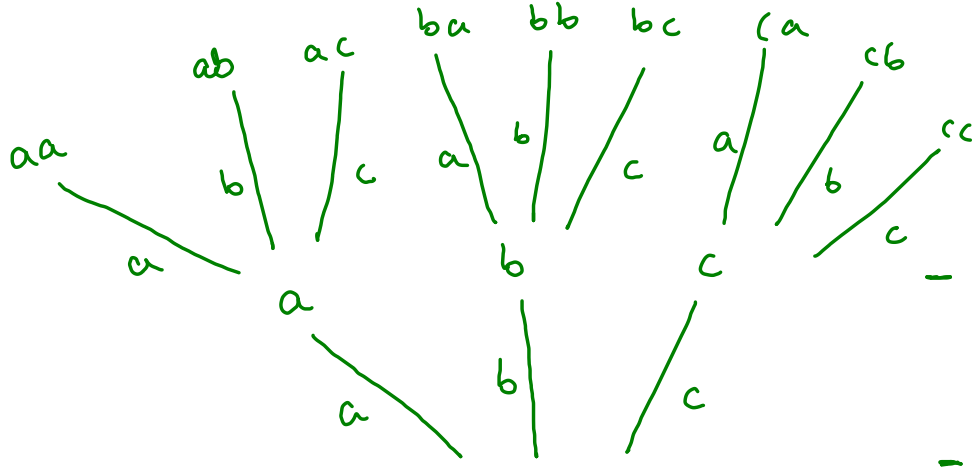
abcbac

k = 2 (distinct)

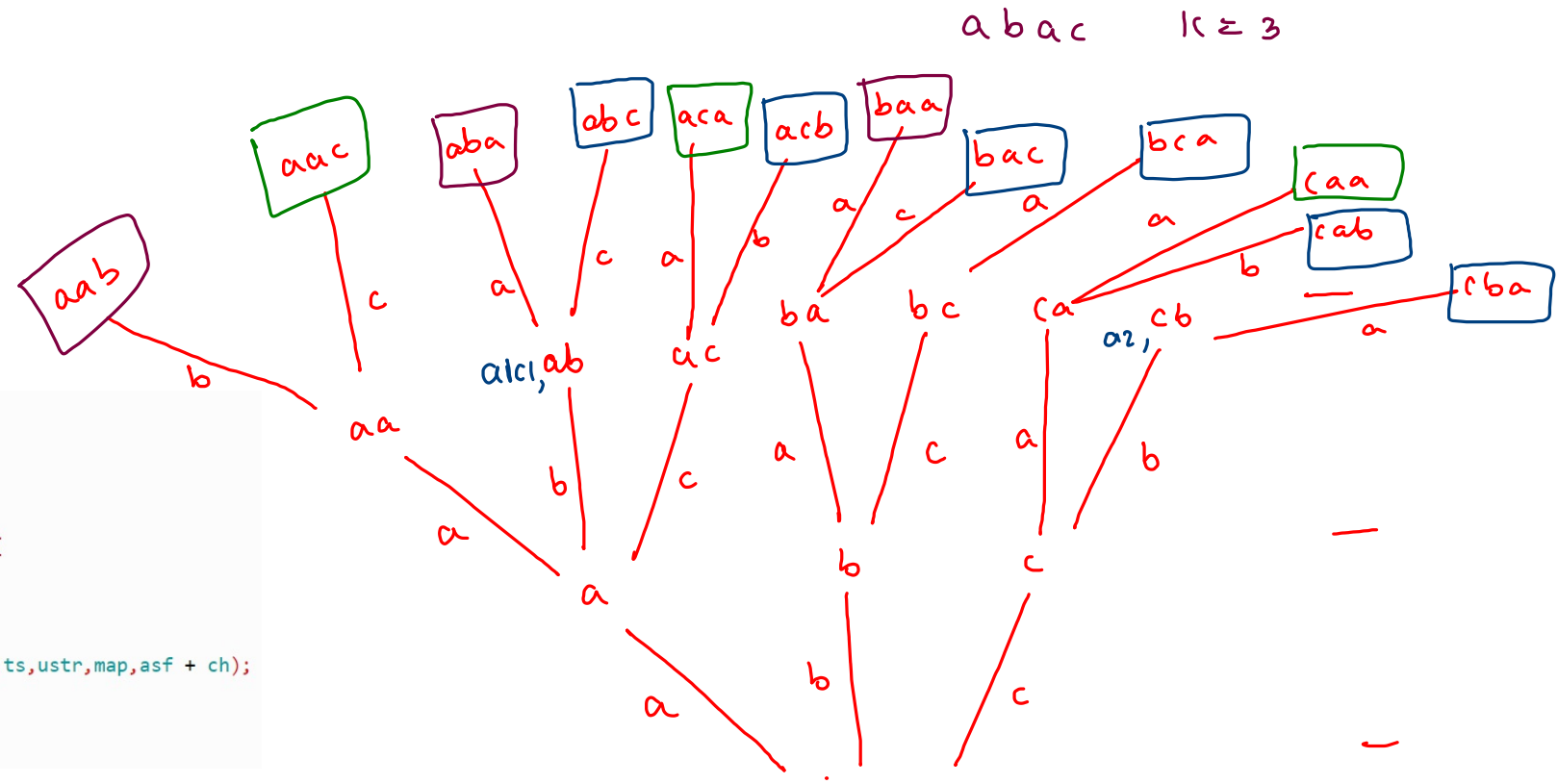


abcbac

k = 2



```
if(cs > ts) {
    System.out.println(asf);
    return;
}


for(int i=0; i < ustr.length();i++) {
    char ch = ustr.charAt(i);

    if(map.get(ch) > 0) {
        map.put(ch,map.get(ch)-1);
        words_k_arrangement_4(cs + 1,ts,ustr,map,asf + ch);
        map.put(ch,map.get(ch)+1);
    }
}
```



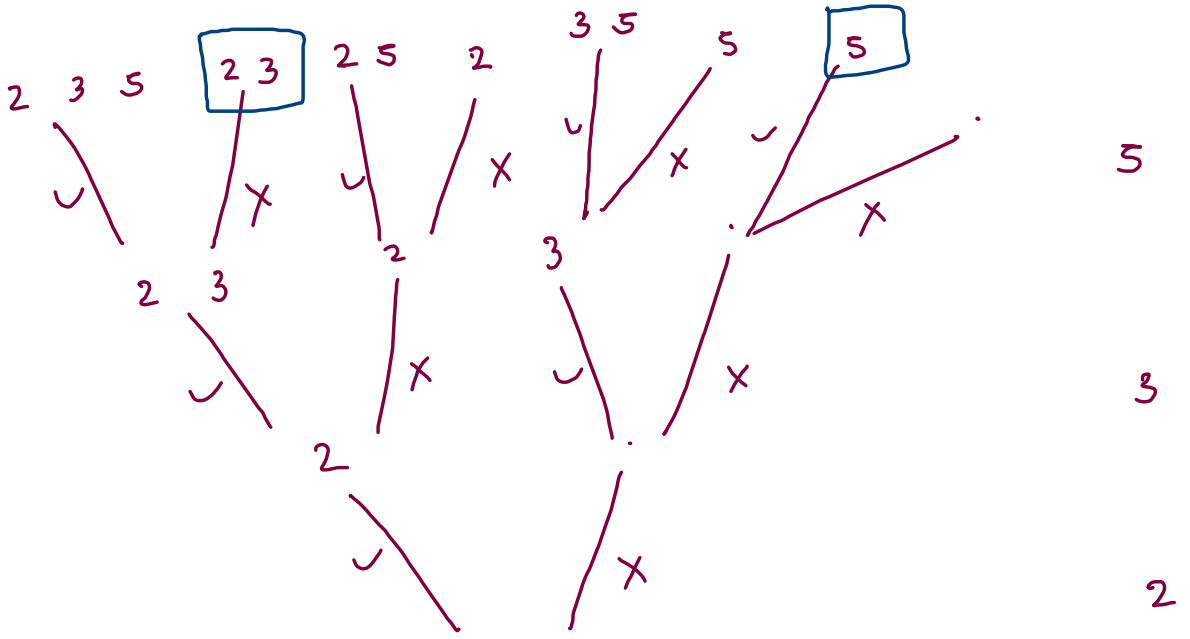
Coin change combinations: 1

(non-duplicate)

target sum subset

coins: 2, 3, 5

amt: 5

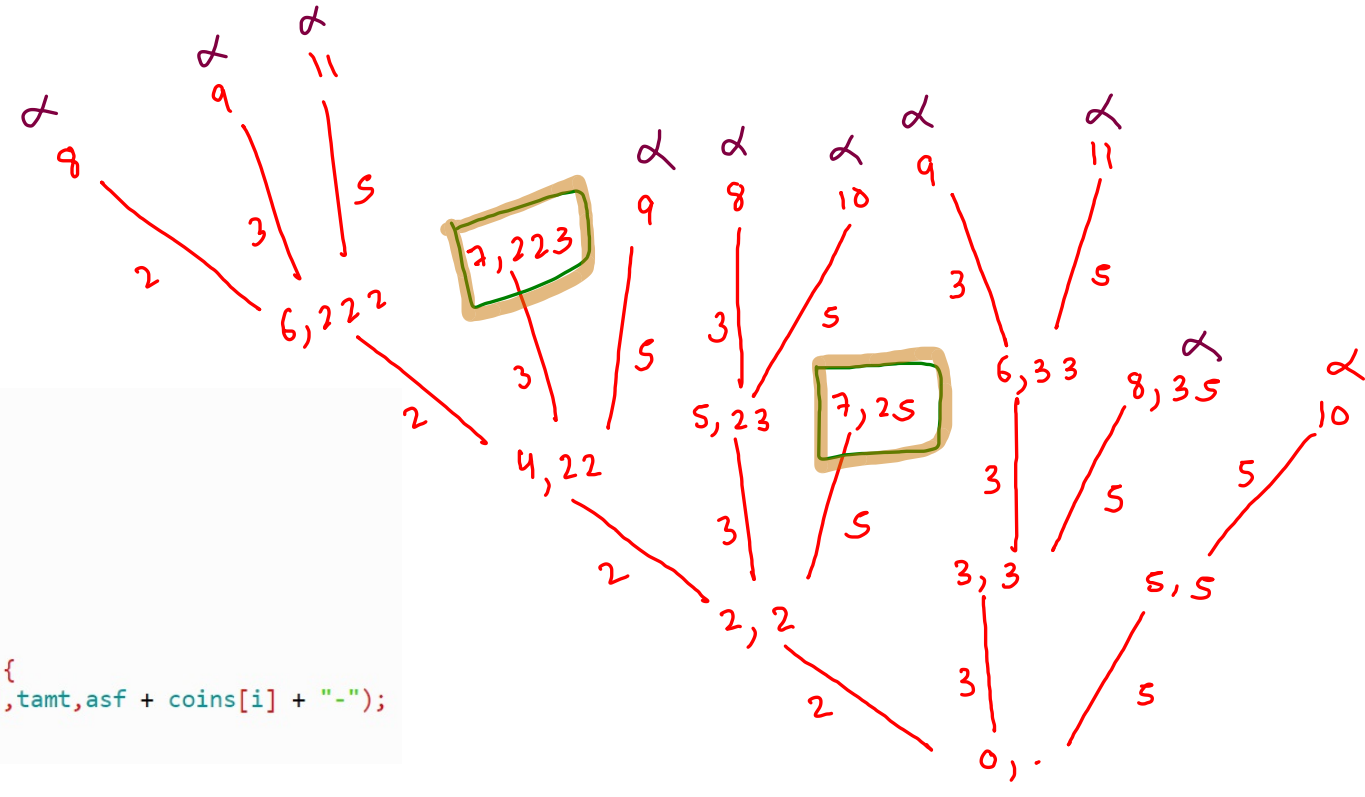


Coin change combinations: 2

(duplicate)

coins : 2 , 3 , 5

amt : 7



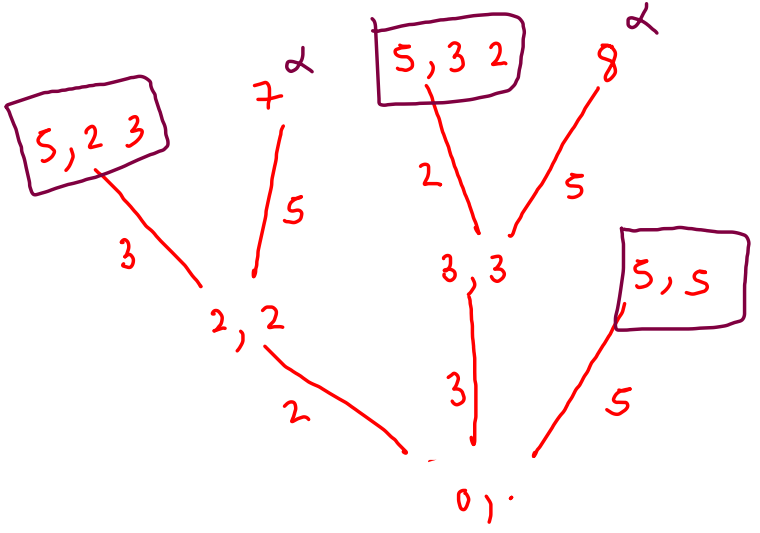

```
if(amtsf == tamt) {
    System.out.println(asf + ".");
    return;
}

if(amtsf > tamt) {
    return;
}

for(int i = lci; i < coins.length;i++) {
    coinChange(i,coins,amtsf + coins[i],tamt,asf + coins[i] + "-");
}
```

Coin Change - Permutations - 1

(non-duplicate)

Coins : 2 3 5

amt : 5



2 3

3 2

5

Coin Change - Permutations - 2

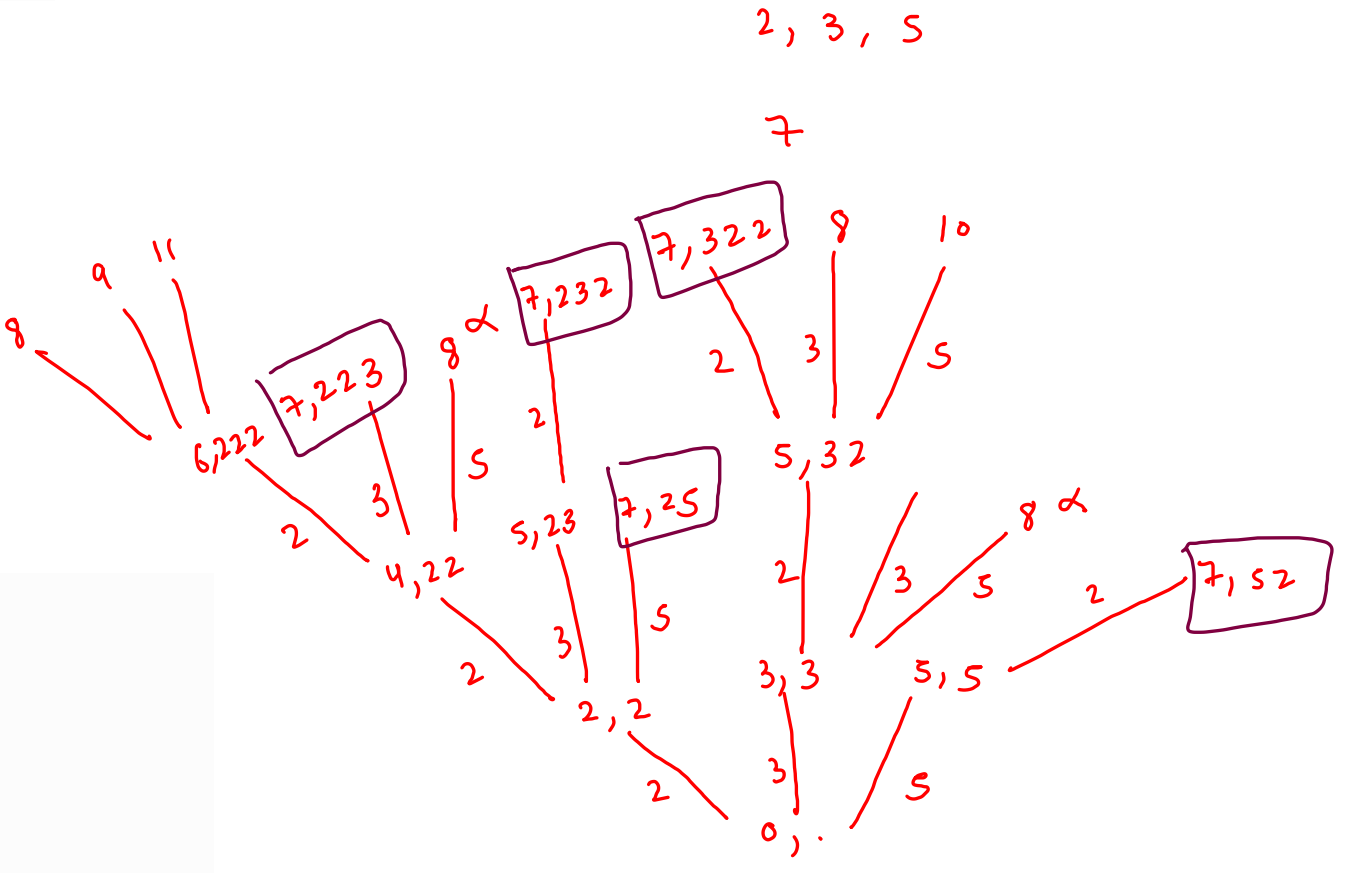
223

232

322

52

25




```java
if(tamt == amtsf) {
    System.out.println(asf + ".");
    return;
}

if(amtsf > tamt) {
    return;
}

for(int i=0; i < coins.length;i++) {
    coinChange(coins,amtsf + coins[i],tamt,asf + coins[i] + "-");
}
```