All Palindromic Partitions

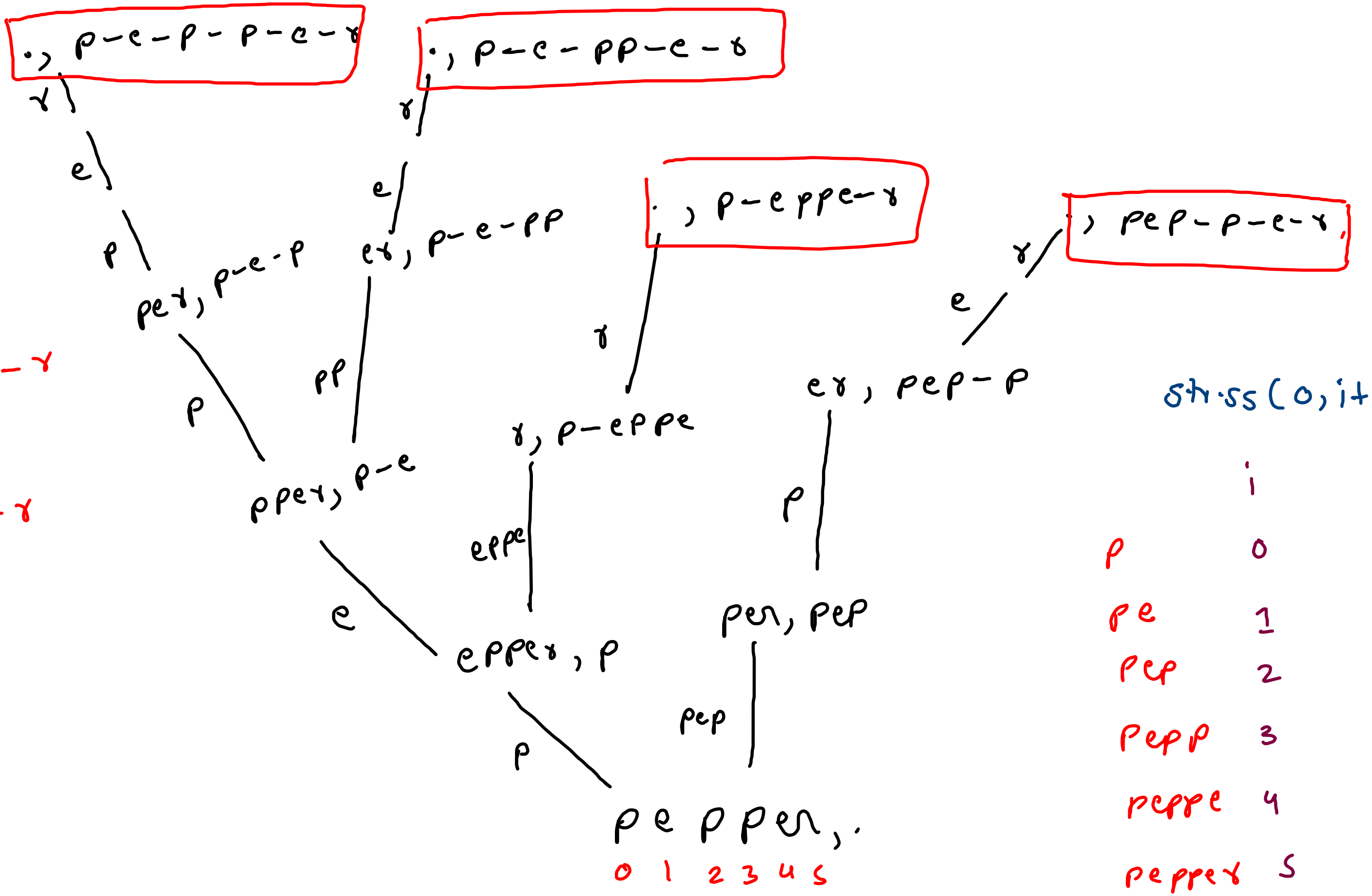
str: pepper

(i) p - e - p - p - e - r

(ii) p - e - pp - e - r

(iii) pep - p - e - r

(iv) p - eppe - r



str.ss(0, i+1)

	i
p	0
pe	1
pep	2
pepp	3
peppe	4
pepper	5

All Palindromic Permutations

palindromic

(P1) (P1') abccba

(P1) ch (P1') abcba

P1' -> rev (P1)

str : aabb

distinct perm

aabb

abab

abba

baba

bbaa

baab

Palindromic perm ⟶

abba

baab

str: abaabccda

odd j -> 'd'

$$\frac{a4\ b2\ c2\ d1}{2} = a2\ b1\ c1$$

aabcdcbaa

aacbdbcaa

bacadacab

bcaadaacb

baacdcaab

caabdbaac

cbaadaabc

cabadabac

abacdcaba

abcadacba

acbadabca

acabdbaca

str: a b a c c b

odd j = '.'

$$\frac{a2\ b2\ c2}{2} = a1\ b1\ c1$$

a b c c b a

a c b b c a

b a c c a b

b c a a c b

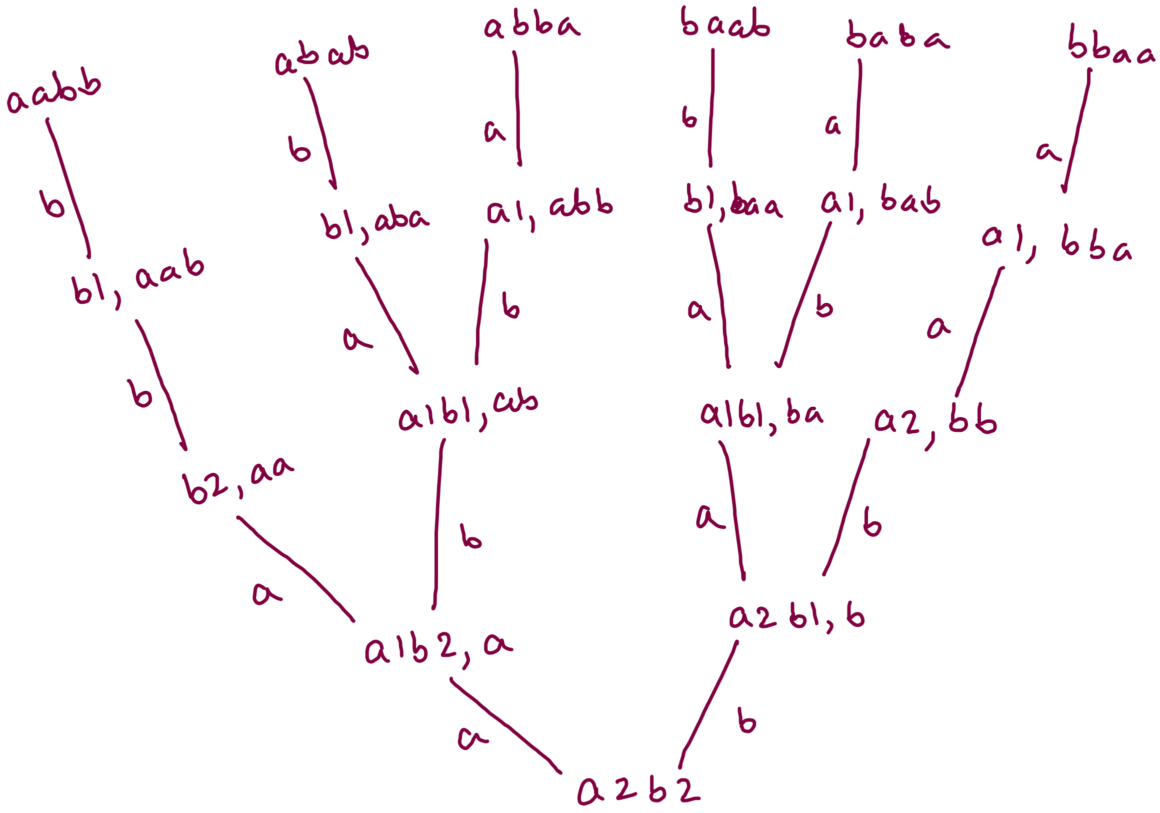
c a b b a c

c b a a b c

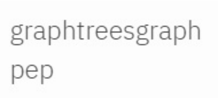
Str: abababab

$\frac{a4b4}{2} \rightarrow a2b2$

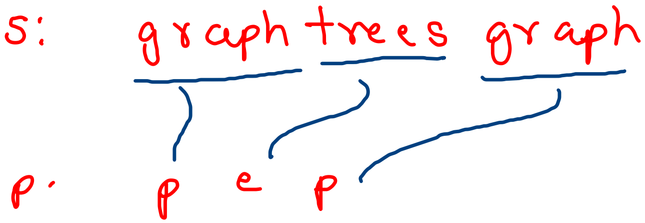
aabb



Pattern Matching

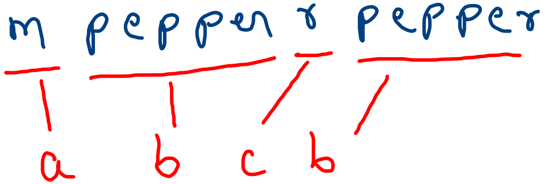


graphtreesgraph
pep



p -> graph

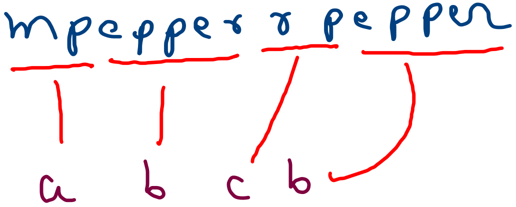
e -> trees



a - m

b - pepper

c - r



a - mp

b - epper

c - rp

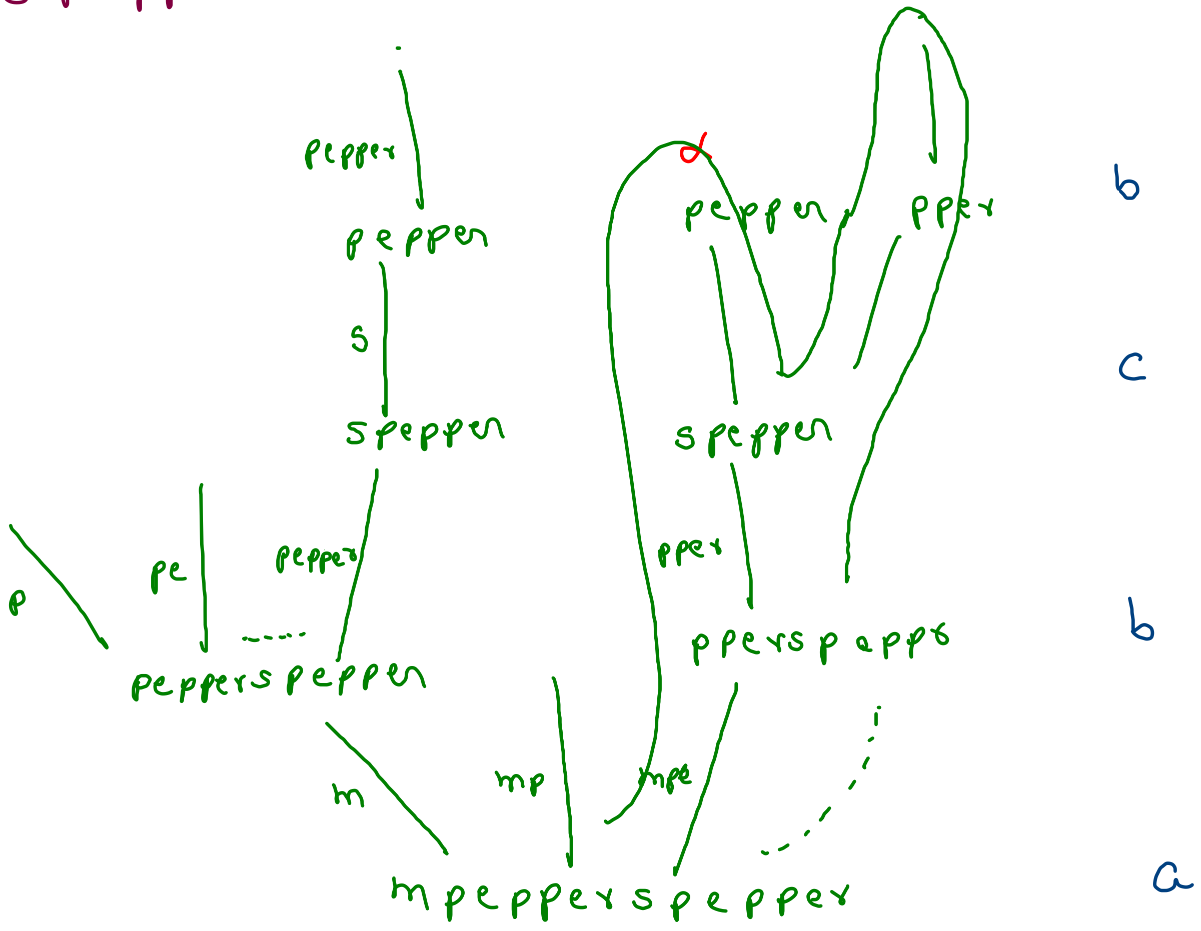
Str : mpeppers pepper

pat: abcb

a -> m

b -> pepper

c -> s



a -> mpe

b -> pper

c -> spe

Word Break - I

str : i love mango icecream

i – love – mango – icecream

i – love – mango – ice – cream

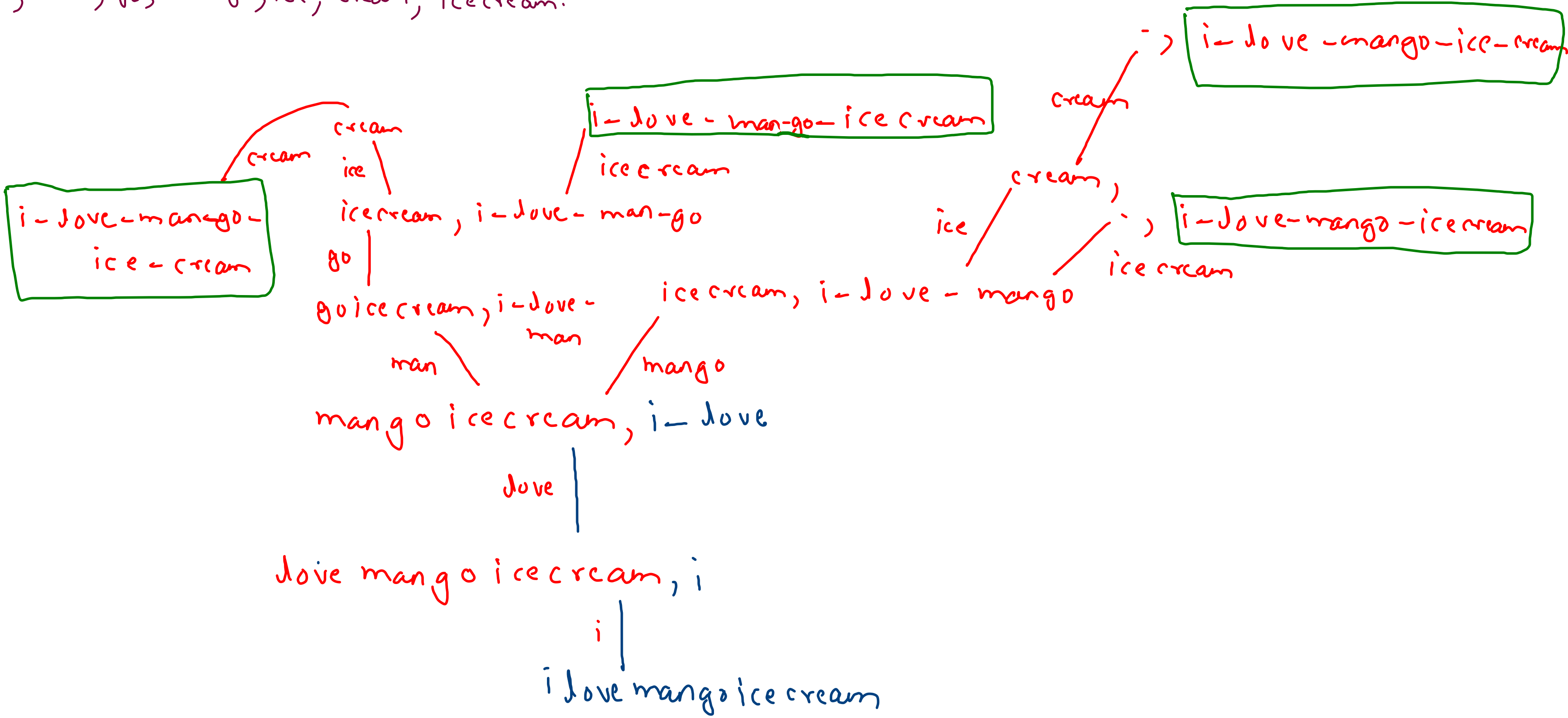
i – love – man – go – icecream

i – love – man – go – ice – cream

dict : i, love, man,
go, mango,
ice, cream,
icecream.

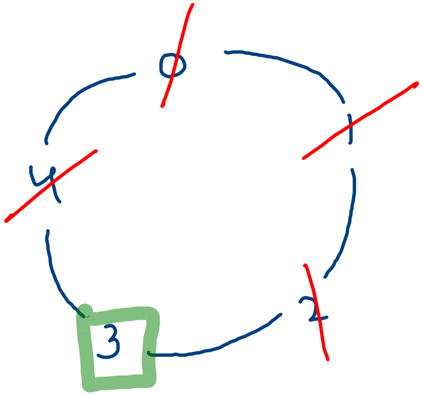
str. i love mango icecream

dict : i, love, man, go, mango, ice, cream, icecream.

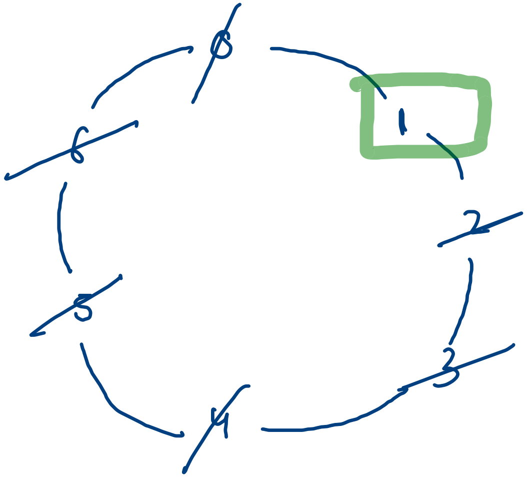


Josephus Problem

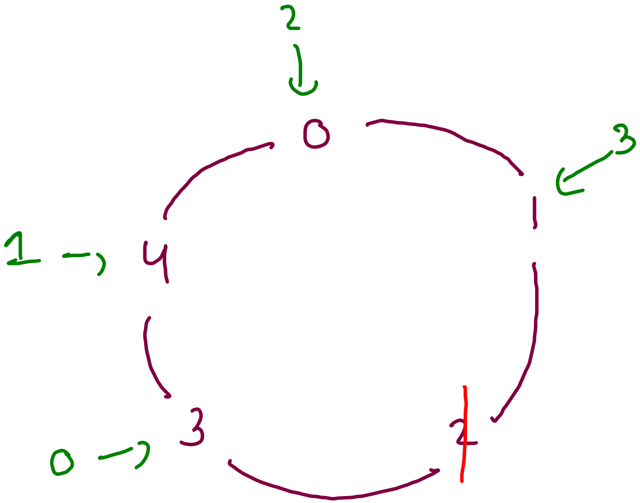
$n = 5$, $k = 3$

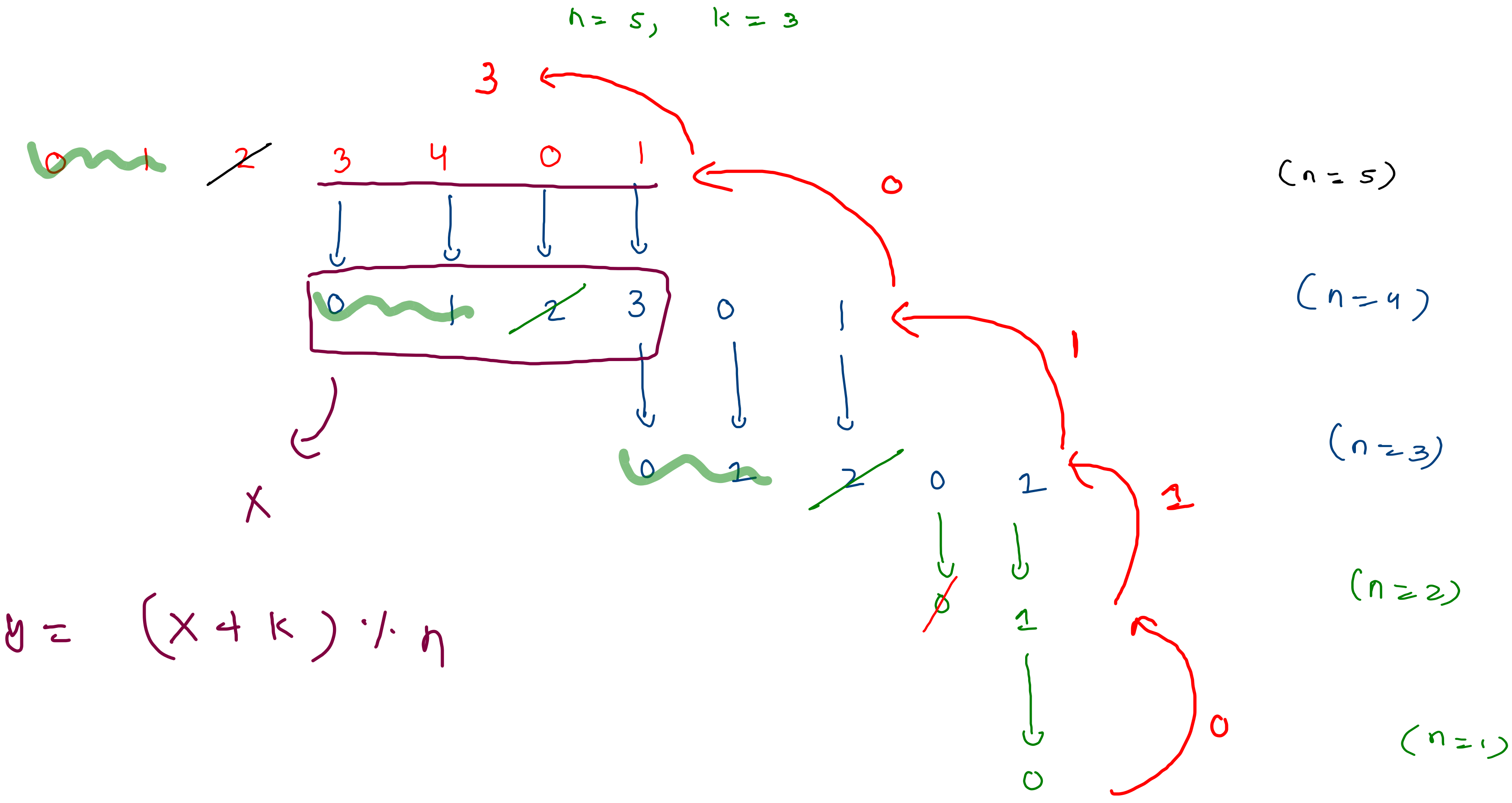


$n = 7$, $k = 4$

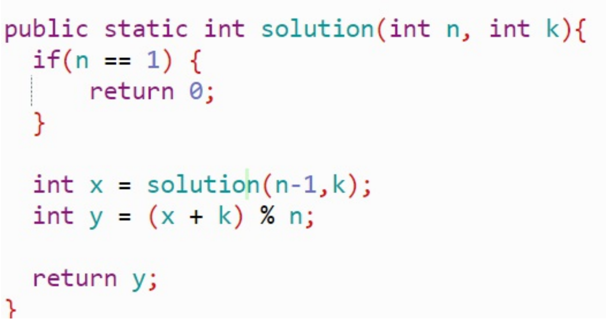


$n = 5 \qquad k = 3$





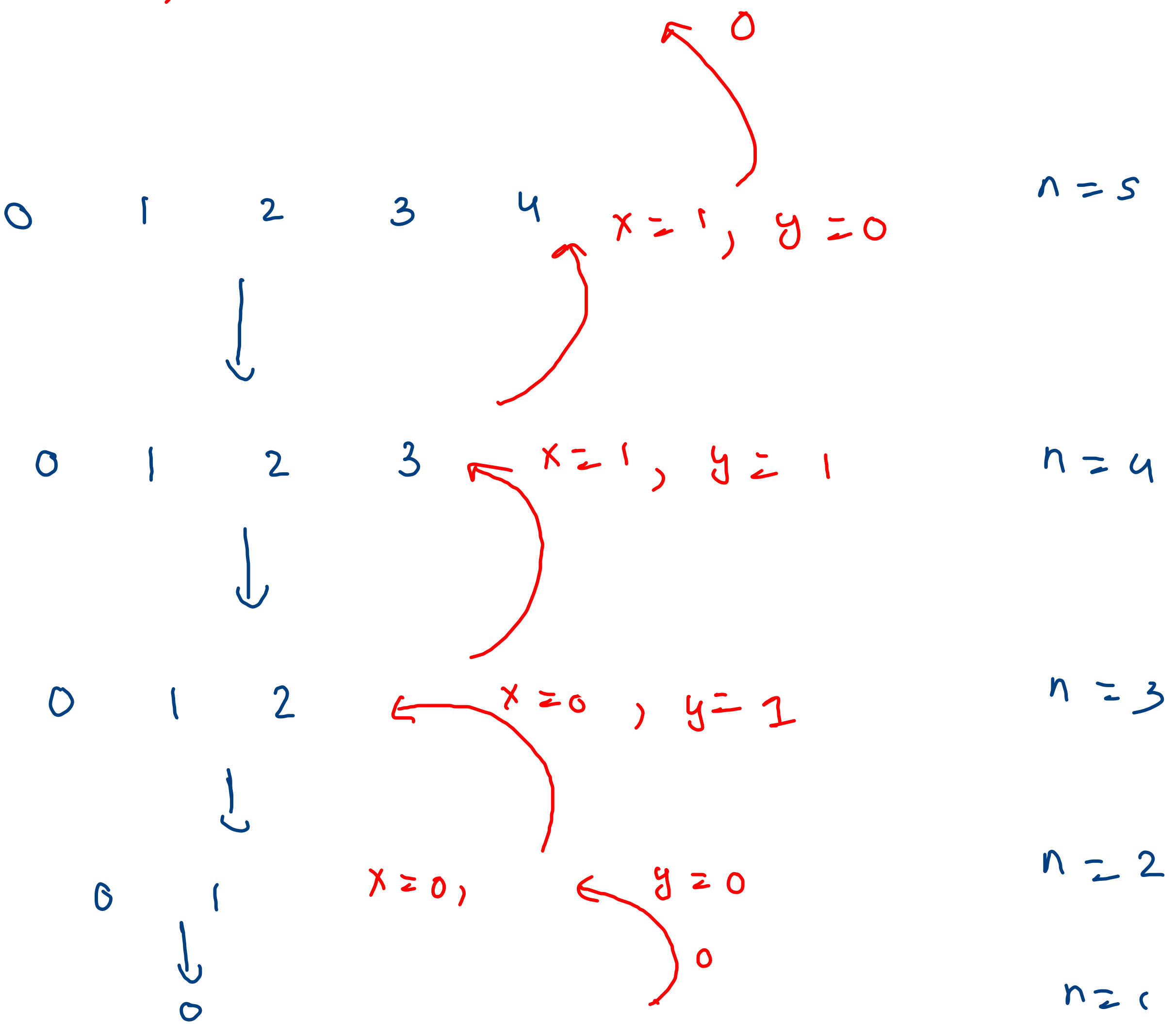
$$y = (x + k) \% n$$

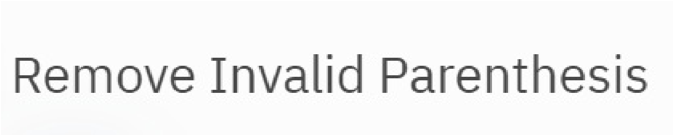


```
public static int solution(int n, int k){
  if(n == 1) {
      return 0;
  }

  int x = solution(n-1,k);
  int y = (x + k) % n;

  return y;
}
```

n = 5, k = 4

k = 4





Remove Invalid Parenthesis

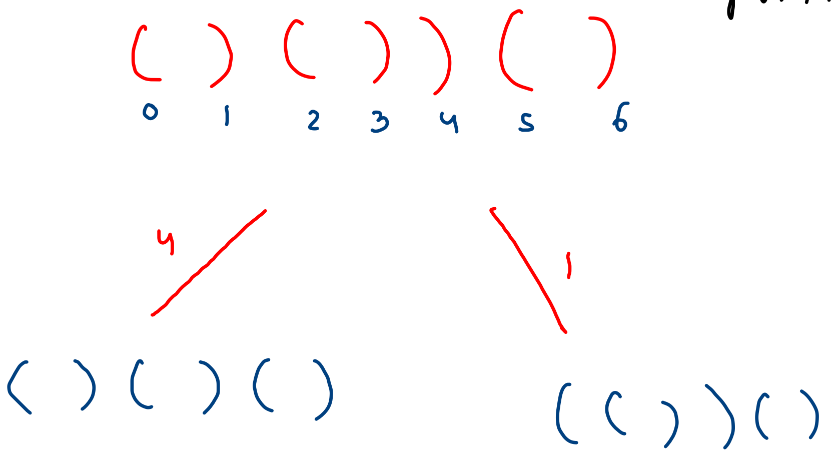
()())()

min no. of brackets remove

⇓

print all the balanced ans (distinct)

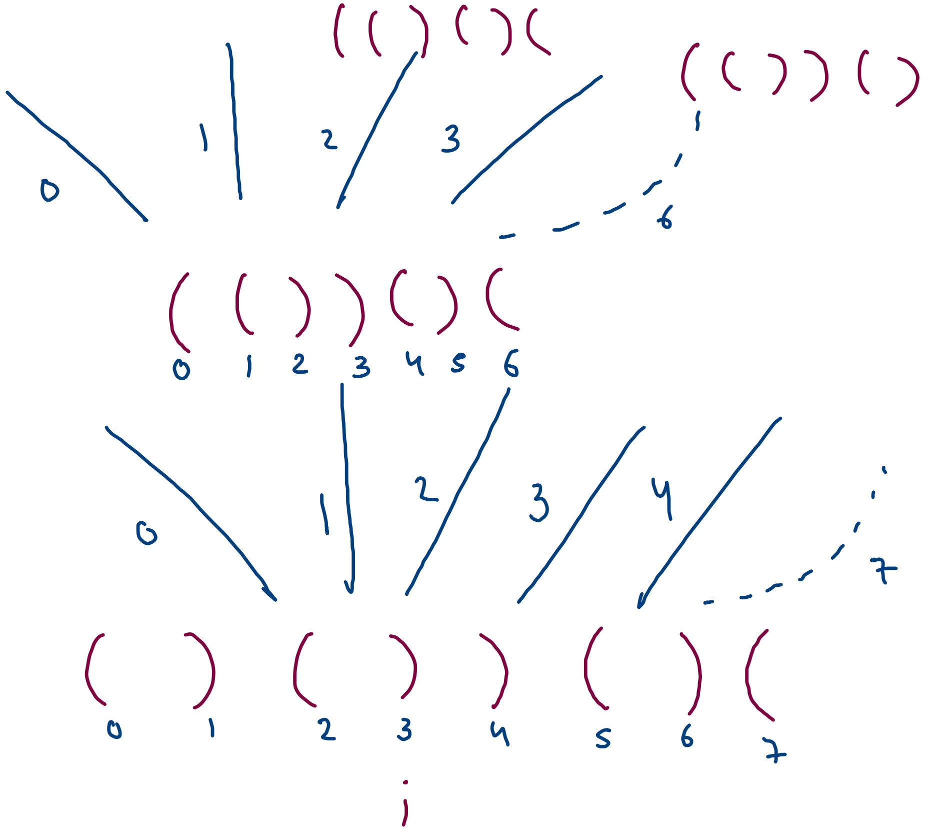
minrem = 1



str : ( ) ( )) ( ) (

0 1 2 3 4 5 6 7

minremovals = 2



(())()

2nd removal

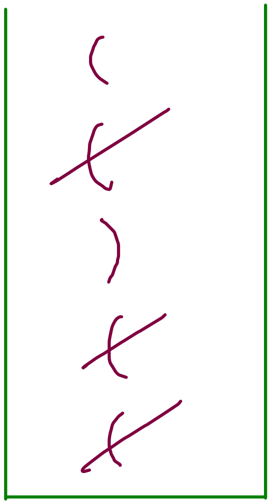
1 removal

res = str.ss(0,i) + str.ss(i+1);

str : ~~( )~~ ~~( )~~) ~~( )~~ (

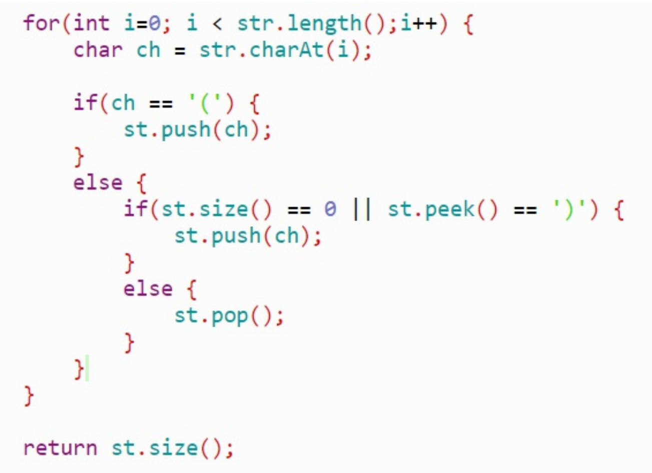
0 1 2 3 4 5 6 7

↑



st . size ( ) → min removals

```
ij (ch = = '(' ) {
        st.push (ch);
}
else {
      ij (st.size( ) = = 0 || st.peek( )==')'){
            st.push( );
      }
      else {
            st. pop( );
      }
}
```



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

    if(ch == '(') {
        st.push(ch);
    }
    else {
        if(st.size() == 0 || st.peek() == ')') {
            st.push(ch);
        }
        else {
            st.pop();
        }
    }
}

return st.size();
```