Recursion Problems - 2

In This Lecture

1. Palindrome String
2. Count occurrences of String in another String
3. Print All Subsets of a given String

Palindrome String

S = a b a , a a , c , racecar

l = 0
r = 0

Problem ⟶ subproblem

a b c b a

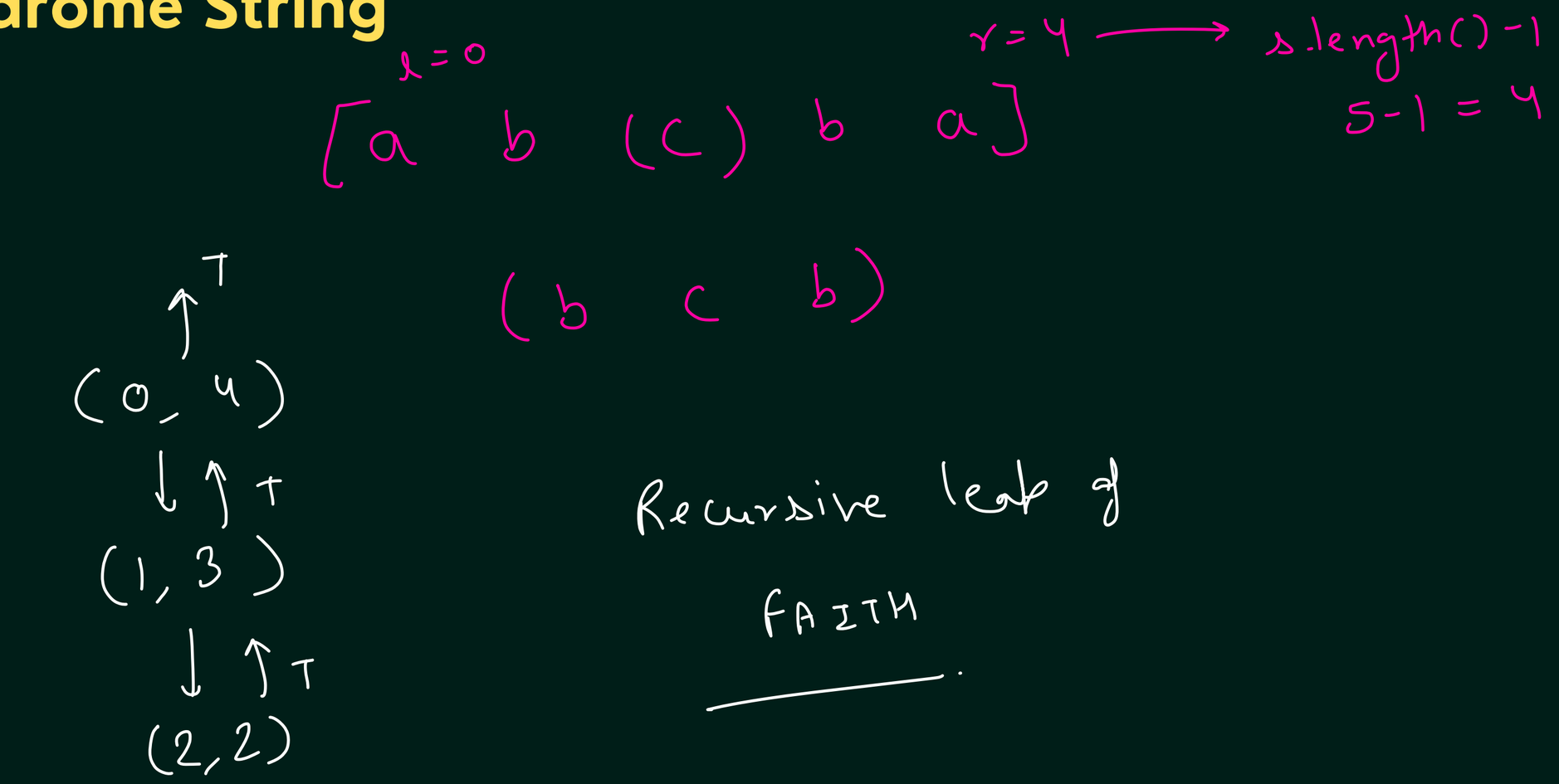
r a c e c a r

l, r

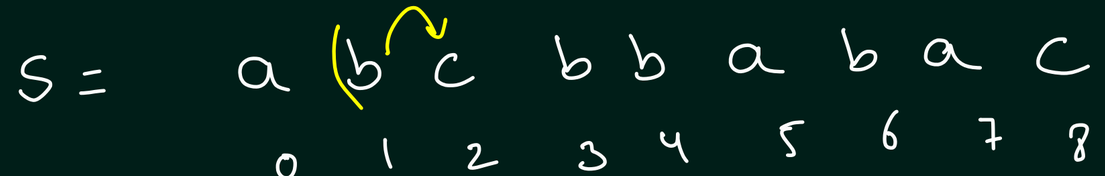
l == r

```
P(s, l, r)
    if (l >= r) return true;
    if (extream ends match)
        return P(s, l+1, r-1);
```

Palindrome String



Count substrings in a String



6 + 3 = 9

(s, 9)

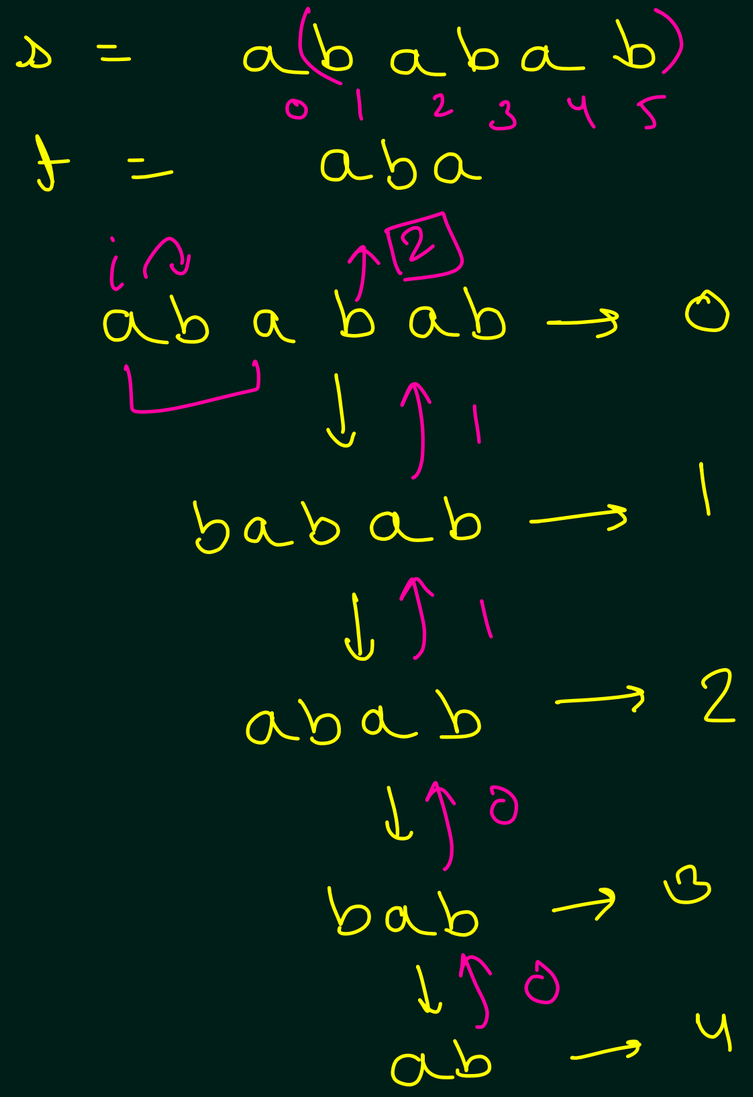
t = a b a

Base case

[i + t.length > s.length]

i > s.length - t.length

Count substrings in a String




```
2 usages
static int countOccurrencesHelper(String t, String s, int i) {
    if(i > s.length() - t.length()) {
        return 0;
    }
    int subProblemKaAnswer = countOccurrencesHelper(t, s, i: i+1);

    boolean doStartingCharsMatch = s.substring(i, i+t.length()).equals(t);

    if(doStartingCharsMatch) return subProblemKaAnswer + 1;
    else return subProblemKaAnswer;
}
```

Print All Subsets of a given String

S = abc → '', a, b, c, ab, ac, bc, abc

Count → length = n → Total → 2^n

(bc) → '', b, c, bc

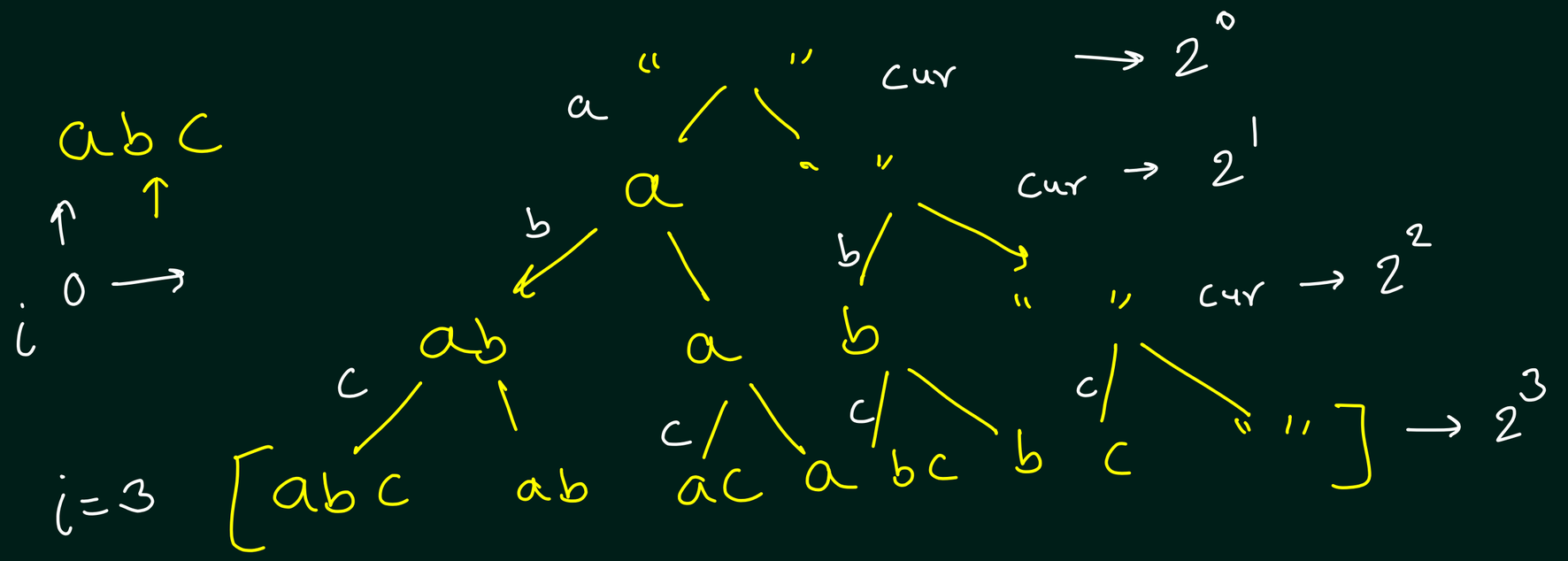
(a, b, c)

↓↑ → [a * ('', b, c, bc), '', b, c, bc]

(bc)

[a, ab, ac, abc, '', b, c, bc]

Print All Subsets of a given String



$T(n) = 2T(n-1) + C$

$2^0 + 2^1 + 2^2 + \dots 2^n$

$= 2^{n+1}$

$\hookrightarrow \boxed{O(2^n)}$