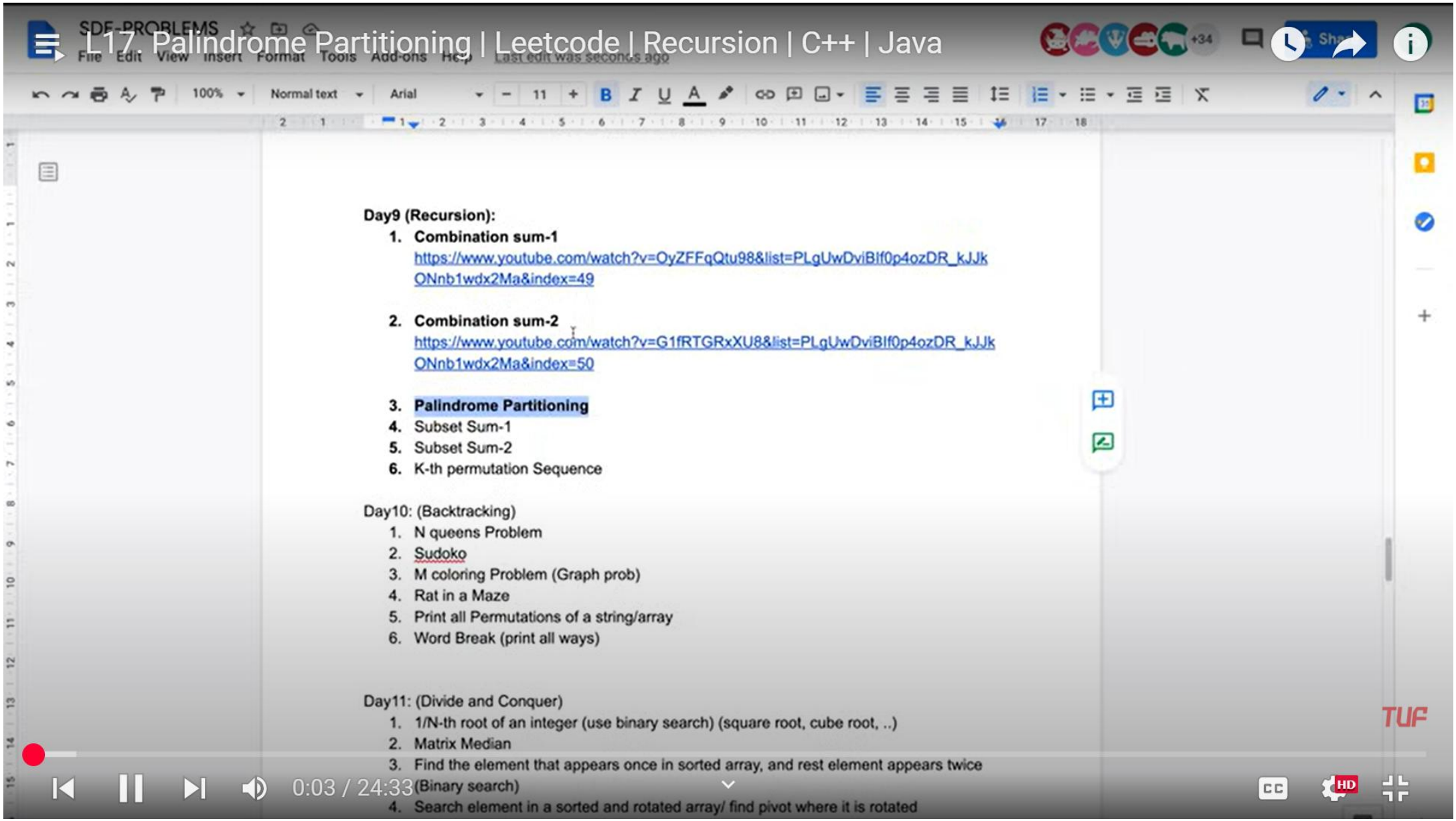

SDE-PROBLEMS
L17. Palindrome Partitioning | Leetcode | Recursion | C++ | Java
File Edit View Insert Format Tools Add-ons Help
Last edit was seconds ago
Share
100% Normal text Arial 11
Day9 (Recursion):
1. Combination sum-1
https://www.youtube.com/watch?v=OyZFFqQtu98&list=PLgUwDviBIf0p4ozDR_kJJkONnb1wdx2Ma&index=49
2. Combination sum-2
https://www.youtube.com/watch?v=G1fRTGRxXU8&list=PLgUwDviBIf0p4ozDR_kJJkONnb1wdx2Ma&index=50
3. Palindrome Partitioning
4. Subset Sum-1
5. Subset Sum-2
6. K-th permutation Sequence
Day10: (Backtracking)
1. N queens Problem
2. Sudoko
3. M coloring Problem (Graph prob)
4. Rat in a Maze
5. Print all Permutations of a string/array
6. Word Break (print all ways)
Day11: (Divide and Conquer)
1. 1/N-th root of an integer (use binary search) (square root, cube root, ..)
2. Matrix Median
3. Find the element that appears once in sorted array, and rest element appears twice (Binary search)
4. Search element in a sorted and rotated array/ find pivot where it is rotated
TUF
0:03 / 24:33
HD

131. Palindrome Partitioning

Medium 3029 97 Add to List Share

Given a string `s`, partition `s` such that every substring of the partition is a **palindrome**. Return all possible palindrome partitioning of `s`.

A **palindrome** string is a string that reads the same backward as forward.

Example 1:

```
Input: s = "aab"
Output: [["a","a","b"],["aa","b"]]
```

Example 2:

```
Input: s = "a"
Output: [["a"]]
```

Constraints:

- `1 <= s.length <= 16`
- `s` contains only lowercase English letters.

Accepted 295,647 Submissions 569,013

131. Palindrome Partitioning

Medium 3029 97 Add to List Share

Given a string `s`, partition `s` such that every substring of the partition is a **palindrome**. Return all possible palindrome partitioning of `s`.

A **palindrome** string is a string that reads the same backward as forward.

Example 1:

```
Input: s = "aab"
Output: [["a","a","b"],["aa","b"]]
```

Example 2:

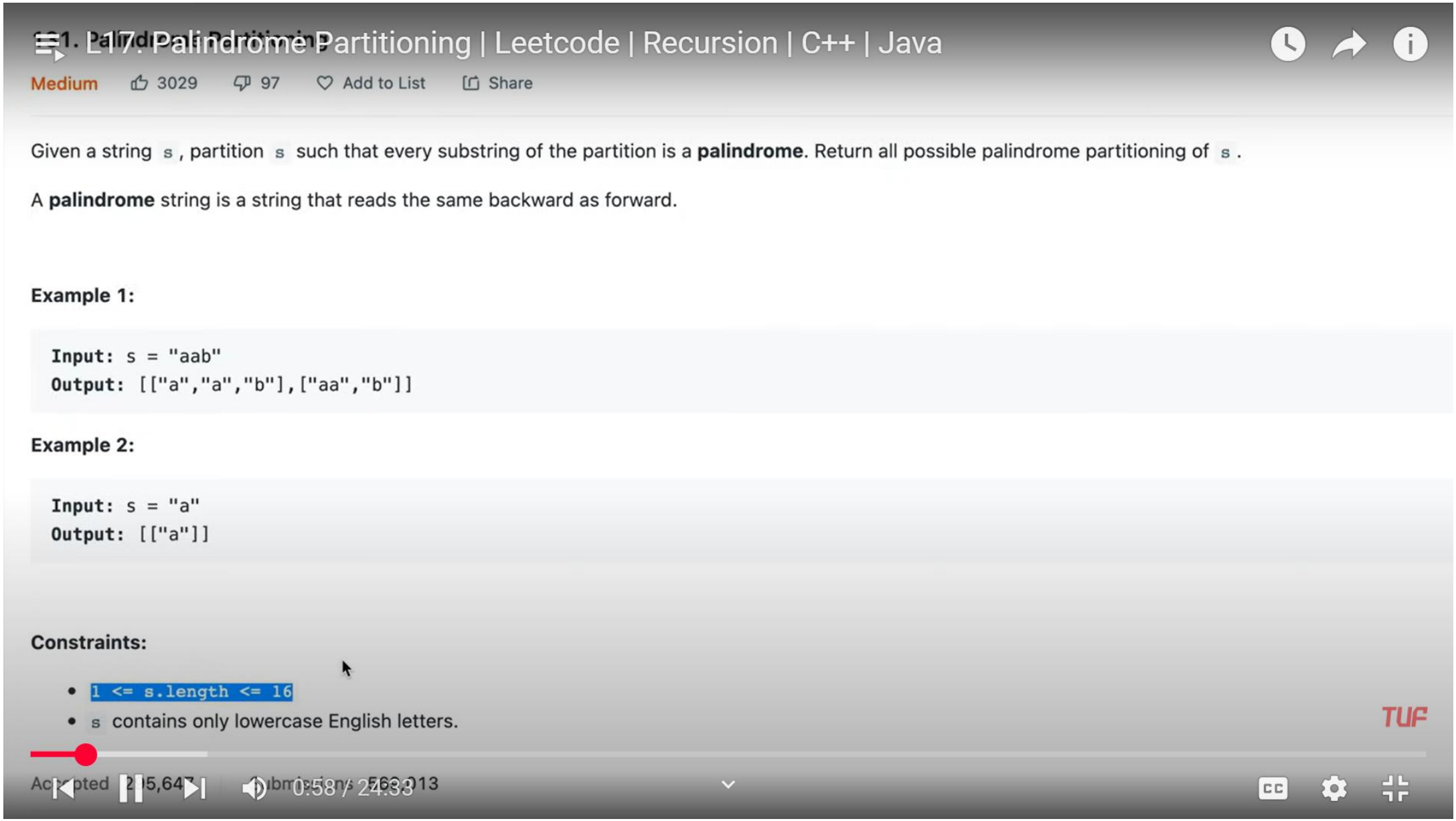
```
Input: s = "a"
Output: [["a"]]
```

Constraints:

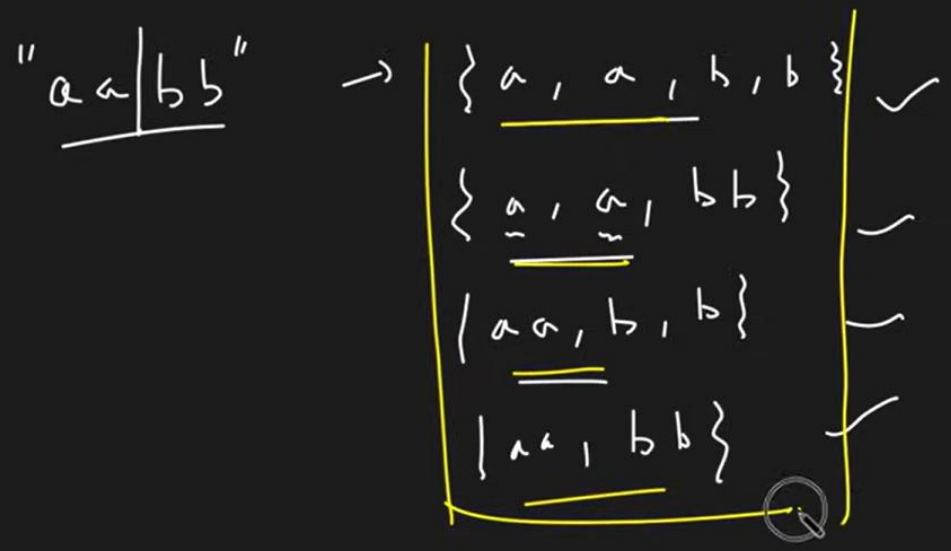
- `1 <= s.length <= 16`
- `s` contains only lowercase English letters.

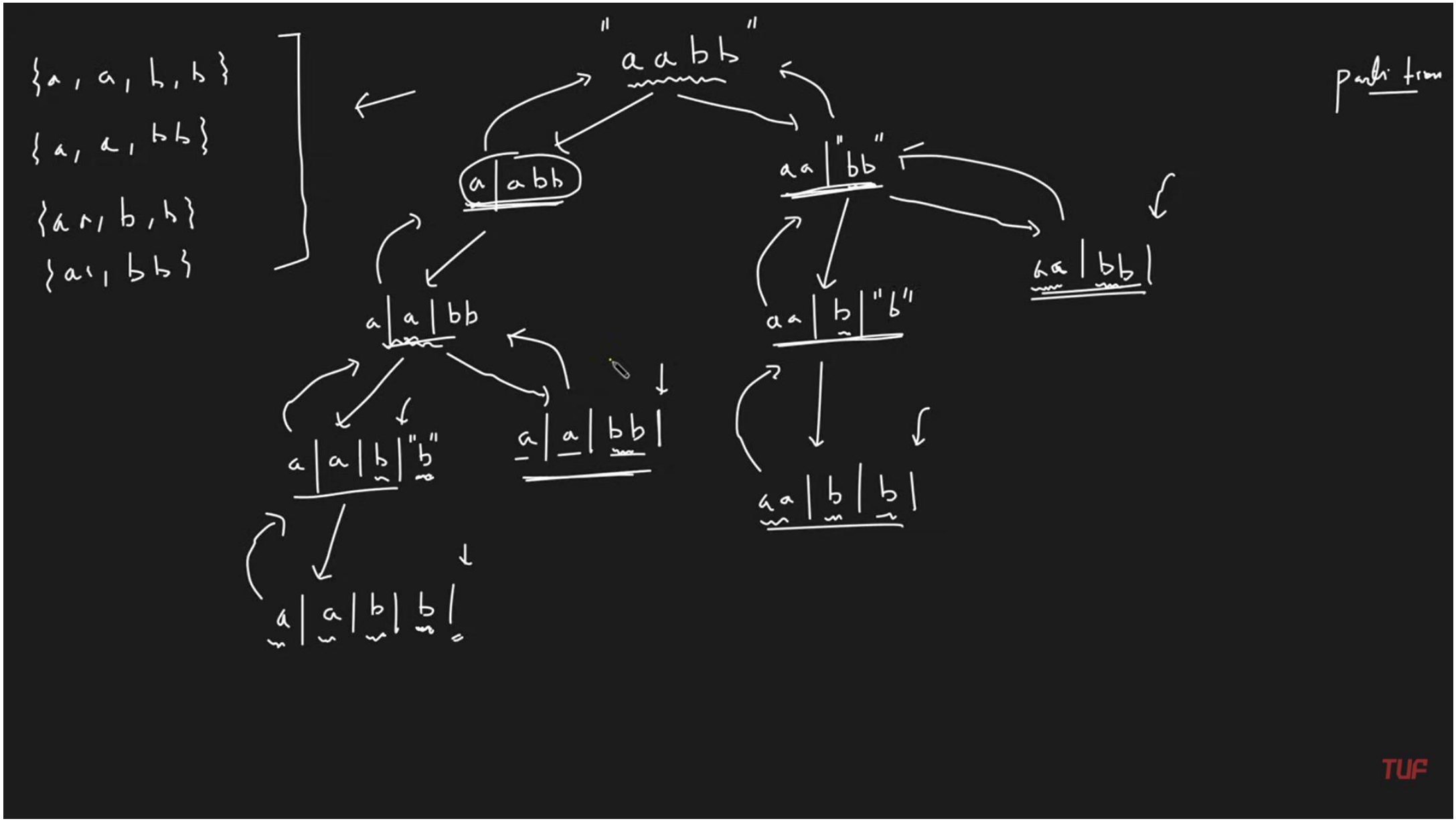
Accepted 295,647 Submissions 569,013

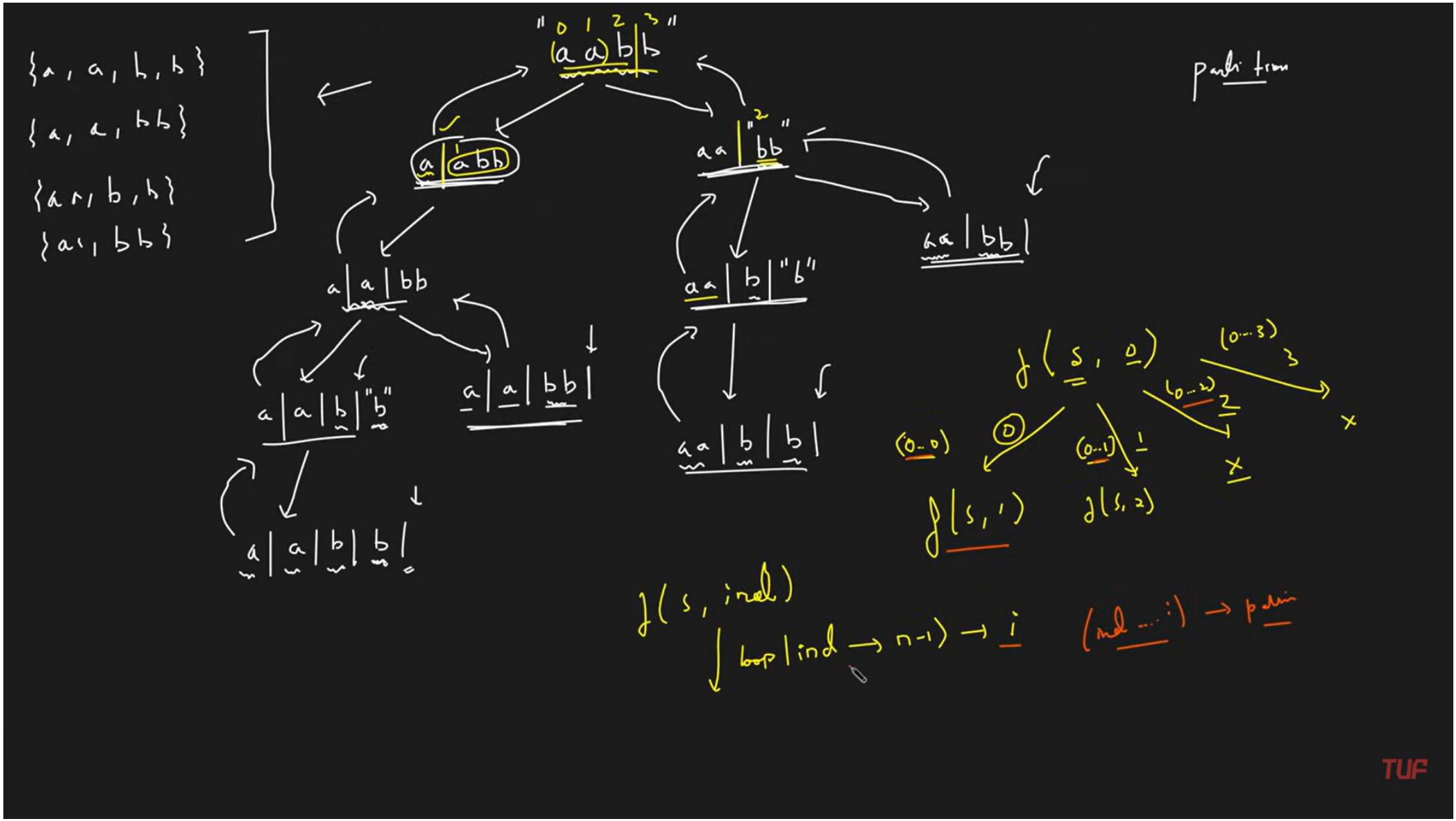
TUF

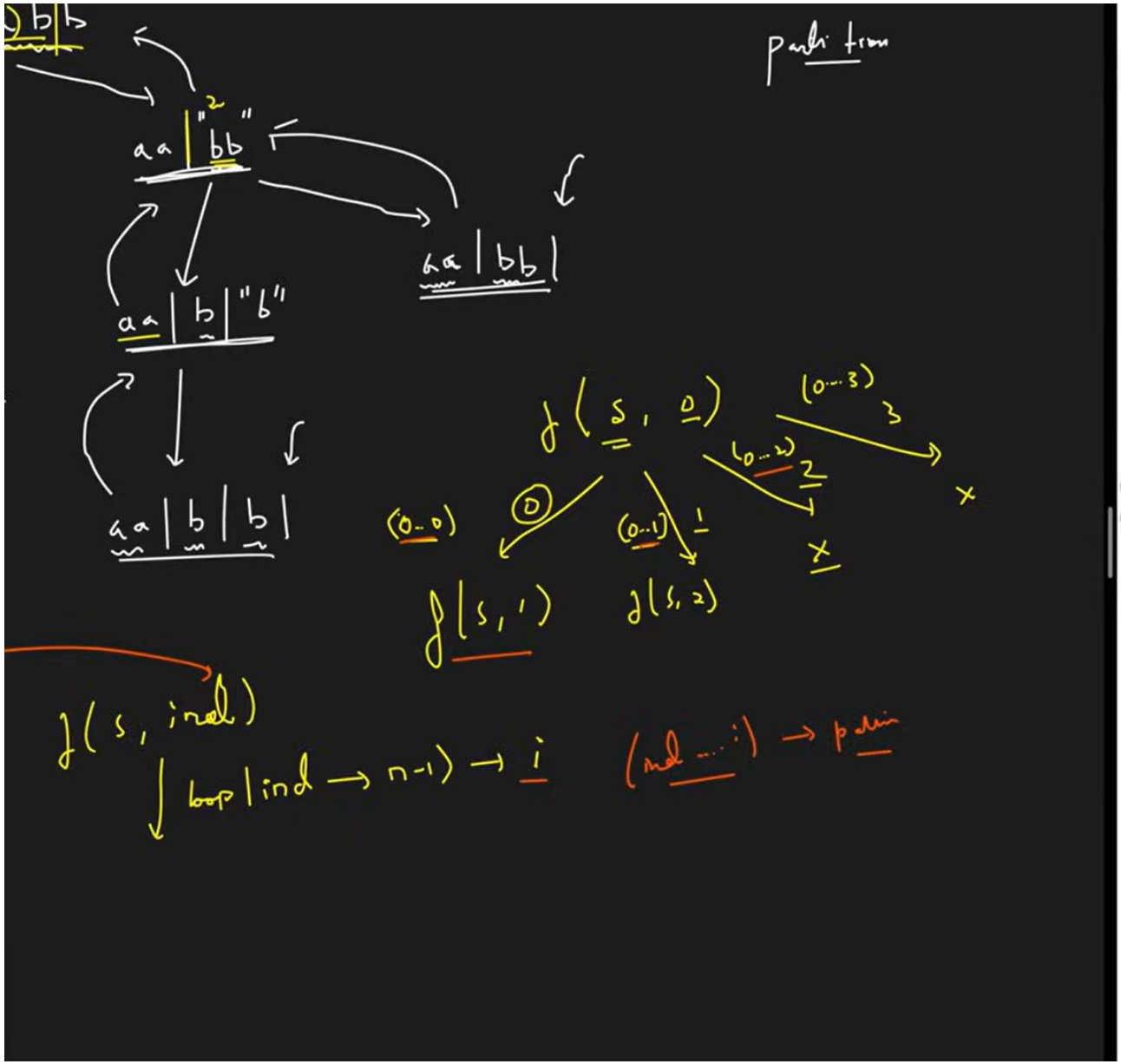

131. Palindrome Partitioning
L17. Palindrome Partitioning | Leetcode | Recursion | C++ | Java
Medium
3029
97
Add to List
Share
Given a string s , partition s such that every substring of the partition is a palindrome. Return all possible palindrome partitioning of s .
A palindrome string is a string that reads the same backward as forward.
Example 1:
Input: s = "aab"
Output: [["a","a","b"],["aa","b"]]
Example 2:
Input: s = "a"
Output: [["a"]]
Constraints:
1 <= s.length <= 16
s contains only lowercase English letters.
TUF
Accepted 205,647
Submissions 563,013
0:58 / 24:33

"a a b b" →




{a, a, b, b}
{a, a, bb}
{aa, b, b}
{aa, bb}
"aabb"
a | abb
aa | "bb"
aa | bb |
a | a | bb
aa | b | "b"
a | a | b | "b"
a | a | bb |
aa | b | b |
a | a | b | b |
partition


{a, a, b, b}
{a, a, bb}
{aa, b, b}
{aa, bb}
"0 1 2 3"
(a a) b | b
a | a bb
aa | "2" | bb
aa | bb |
a | a | bb
aa | b | "b"
a | a | b | "b"
a | a | bb |
aa | b | b |
a | a | b | b |
path from
f(s, 0)
(0...3)
3
(0...2)
2
x
x-1
(0...0)
0
(0...1)
1
f(s, 1)
f(s, 2)
f(s, ind)
bool | ind → n-1) → i
(ind ... i) → path
TUF



LeetCode Explore Problems Mock Contest Discuss

Limited time event to win giveaway!

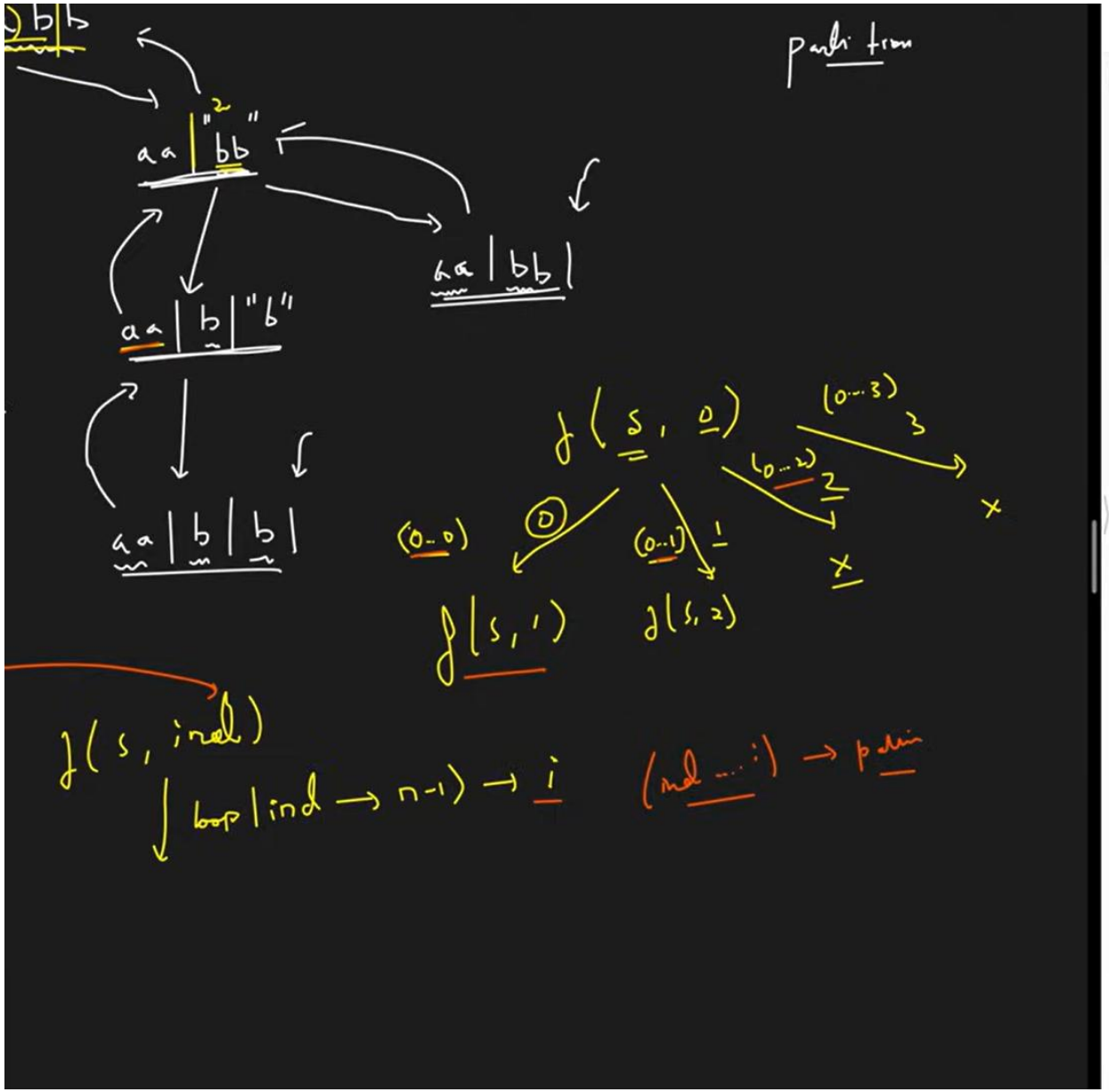
Java Autocomplete

```java
class Solution {
    public List<List<String>> partition(String s) {
        List<List<String>> res= new ArrayList<>();
        List<String> path = new ArrayList<>();
        func(0, s, path, res);
        return res;
    }

    void func(int index, String s, List<String> path, List<List<String>> res) {
        if(index == s.length()) {
            res.add(new ArrayList<>(path));
            return;
        }
        for(int i = index; i < s.length(); ++i) {
            if(isPalindrome(s, index, i)) {
                path.add(s.substring(index, i+1));
                func(i+1, s, path, res);
                path.remove(path.size()-1);
            }
        }
    }

    boolean isPalindrome(String s, int start, int end) {
        while(start <= end) {
            if(s.charAt(start++) != s.charAt(end--))
                return false;
        }
        return true;
    }
}
```

TUF



LeetCode Explore Problems Mock Contest Discuss Store

Limited time event to win giveaway!

Premium

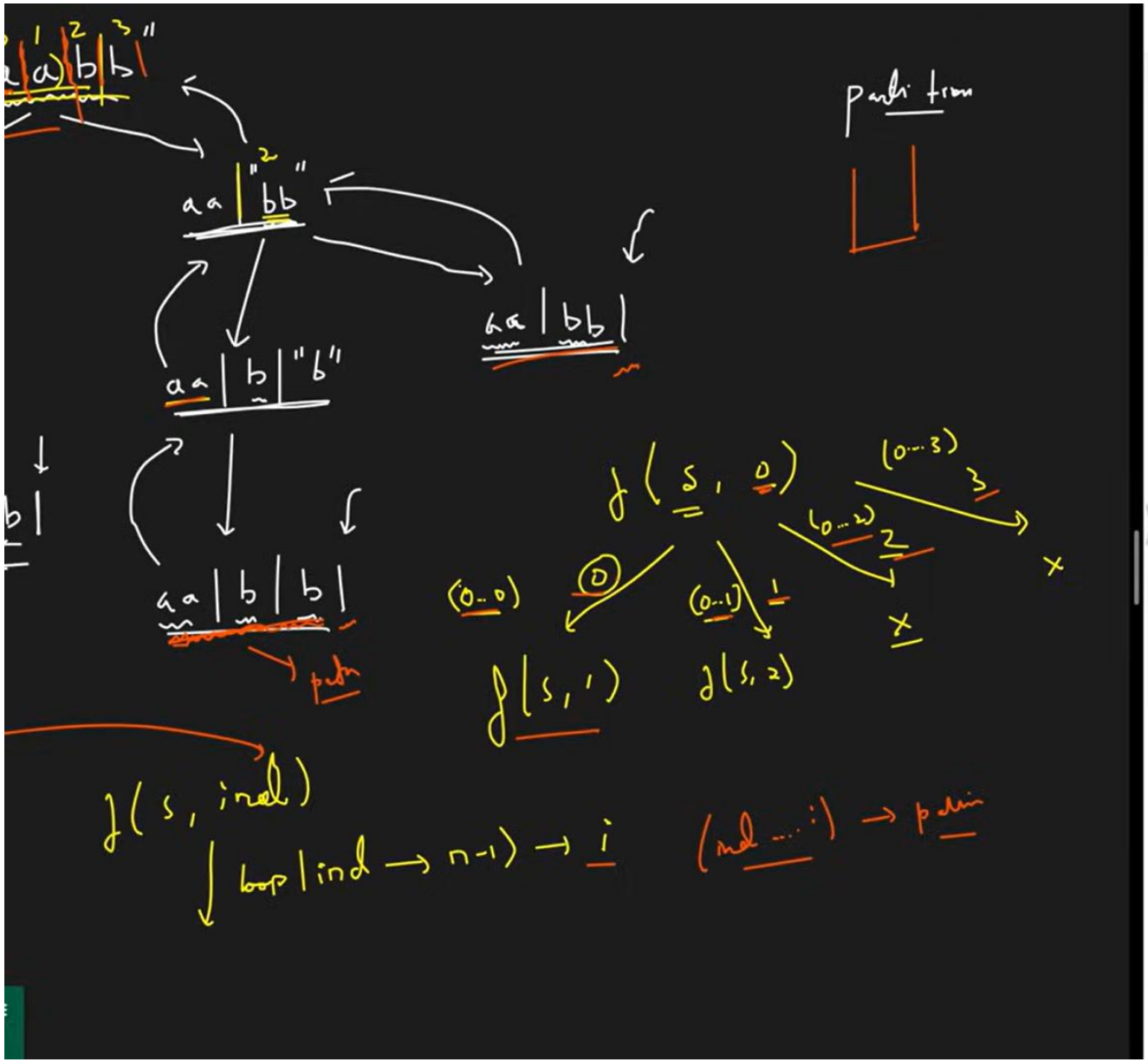
C++ Autocomplete

```cpp
class Solution {
public:
    vector<vector<string>> partition(string s) {
        vector<vector<string> > res;
        vector<string> path;
        func(0, s, path, res);
        return res;
    }

    void func(int index, string s, vector<string> &path,
              vector<vector<string> > &res) {
        if(index == s.size()) {
            res.push_back(path);
            return;
        }
        for(int i = index; i < s.size(); ++i) {
            if(isPalindrome(s, index, i)) {
                path.push_back(s.substr(index, i - index + 1));
                func(i+1, s, path, res);
                path.pop_back();
            }
        }
    }

    bool isPalindrome(string s, int start, int end) {
        while(start <= end) {
            if(s[start++] != s[end--])
                return false;
        }
        return true;
    }
};
```

TUF



LeetCode Explore Day 22 Problems Mock Contest Discuss Store

Limited time event to win giveaway!

C++ Autocomplete

```cpp
class Solution {
public:
    vector<vector<string>> partition(string s) {
        vector<vector<string> > res;
        vector<string> path;
        func(0, s, path, res);
        return res;
    }

    void func(int index, string s, vector<string> &path,
              vector<vector<string> > &res) {
        if(index == s.size()) {
            res.push_back(path);
            return;
        }
        for(int i = index; i < s.size(); ++i) {
            if(isPalindrome(s, index, i)) {
                path.push_back(s.substr(index, i - index + 1));
                func(i+1, s, path, res);
                path.pop_back();
            }
        }
    }

    bool isPalindrome(string s, int start, int end) {
        while(start <= end) {
            if(s[start++] != s[end--])
                return false;
        }
        return true;
    }
};
```

TUF