Program to Find the Count of Distinct Substrings

Mr. Akash Yadav

Assistant Professor

Artificial Intelligence & Data Science

Lesson Plan

Subject/Course	Competitive Coding
Lesson Title	Program to Find the Count of Distinct Substrings

Lesson Objectives
Understand what substrings are and how to count them.
Learn how to find all distinct substrings in a given string.
Explore efficient methods using hashing or Trie.
Analyze time and space complexity of substring counting

Problem Statement:

Write a program to count the number of distinct substrings in a given string.

The program should:

1. Take a string as input.
2. Find all unique substrings.
3. Return the count of those distinct substrings.

Concept

What is a Substring?

- A **substring** is a continuous sequence of characters within a string.
  Example: For "abc", substrings are
  a, b, c, ab, bc, abc.
- Total substrings = $n \times (n + 1) / 2$ for a string of length n.
- But we only want **distinct substrings** (no duplicates).

Concept

⚙️ Approaches

1. **Brute Force:** Generate all substrings and use a set — O(n^2) substrings, O(n^3) time (inefficient).

1. **Trie-based Approach:**
   - Insert all suffixes of the string into a Trie.
   - Each new branch in the Trie represents a **new distinct substring**.
   - Count total Trie nodes = total distinct substrings + 1 (for empty string).

Algorithm/Logic

1. Initialize an empty HashSet.

2. Generate all substrings using two loops:

   - Outer loop for start index.

   - Inner loop for end index.

3. Add each substring to the HashSet.

4. The size of the HashSet gives the distinct substring count.

Visualization

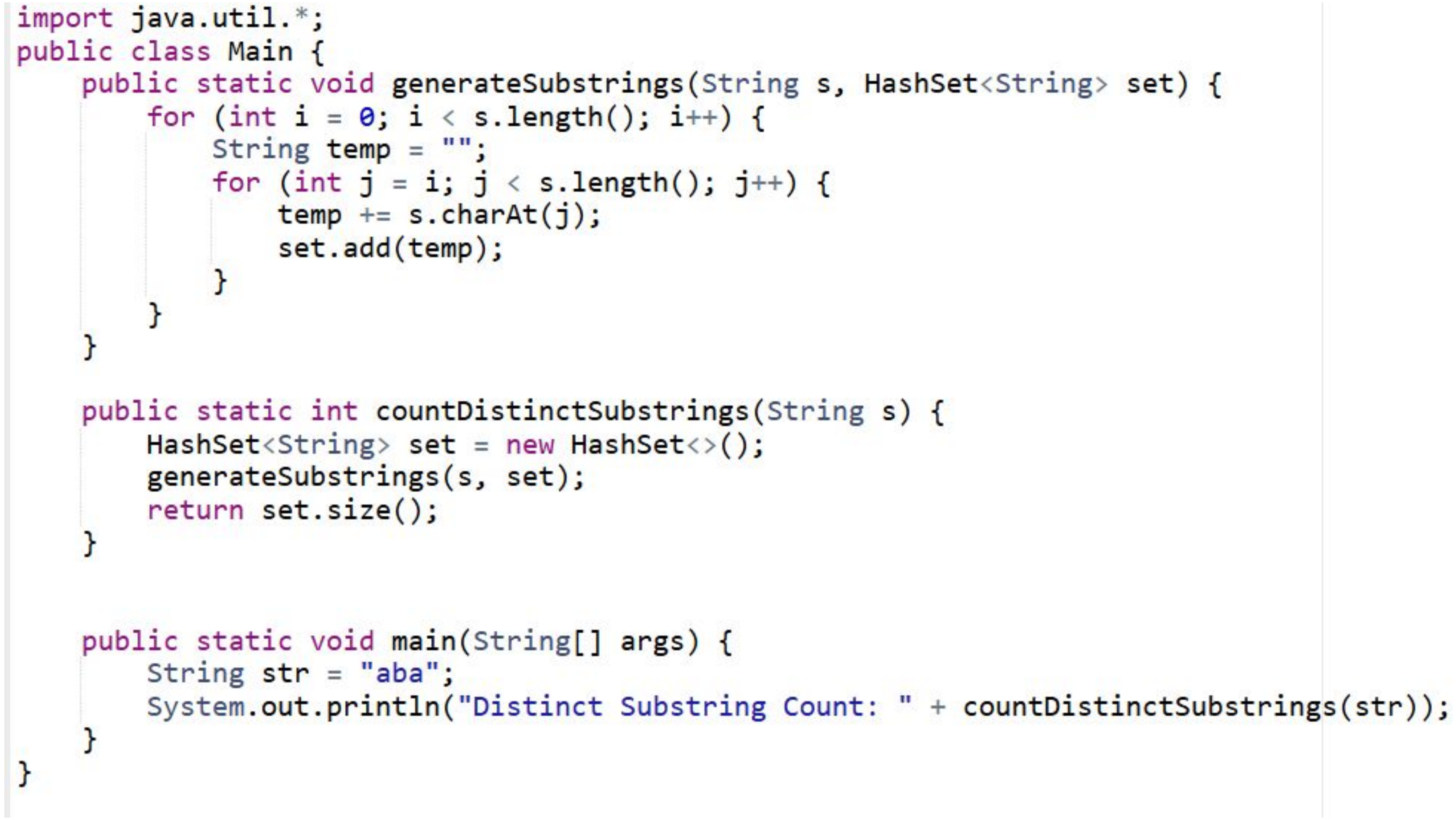
Example:
Input: "aba"

All substrings:
a, b, a, ab, ba, aba

Distinct substrings:
a, b, ab, ba, aba

Count = 5

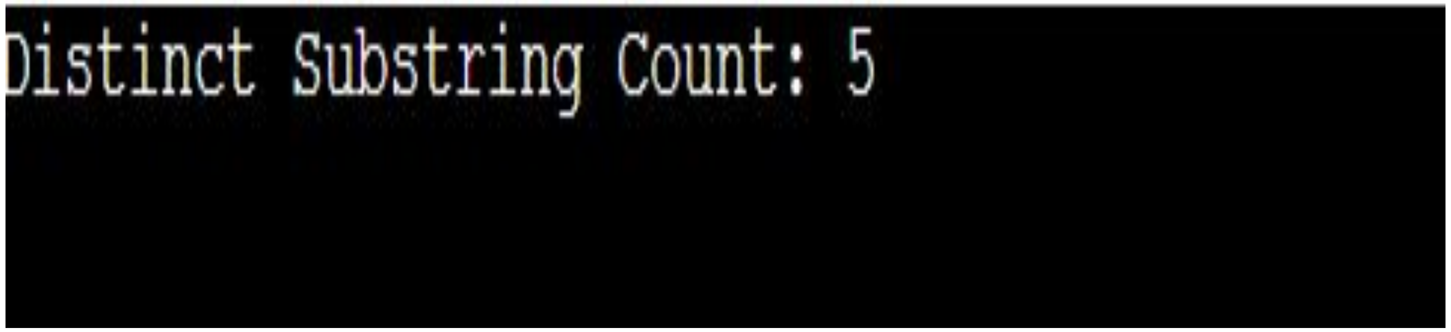
Code Implementation


```java
import java.util.*;
public class Main {
    public static void generateSubstrings(String s, HashSet<String> set) {
        for (int i = 0; i < s.length(); i++) {
            String temp = "";
            for (int j = i; j < s.length(); j++) {
                temp += s.charAt(j);
                set.add(temp);
            }
        }
    }

    public static int countDistinctSubstrings(String s) {
        HashSet<String> set = new HashSet<>();
        generateSubstrings(s, set);
        return set.size();
    }


    public static void main(String[] args) {
        String str = "aba";
        System.out.println("Distinct Substring Count: " + countDistinctSubstrings(str));
    }
}
```

Output



```
istinct Substring Count: 5
```

Example Walkthrough

1. Start = 0 → Substrings: a, ab, aba
2. Start = 1 → Substrings: b, ba
3. Start = 2 → Substrings: a
4. Unique ones stored in HashSet.
5. Final count = 5.

Time & Space Complexity

Time Complexity:

$O(n^2)$ – two nested loops for substring generation.

Space Complexity:

$O(n^2)$ – storing all substrings in the HashSet.

Summary

1. Substrings are continuous character sequences.
2. Use HashSet to remove duplicates automatically.
3. Total time complexity is $O(n^2)$.
4. Simple and effective approach for small strings.

Practice Questions:

1. Repeated Substring Pattern — LeetCode #459

🔗 https://leetcode.com/problems/repeated-substring-pattern/

Concept: Check if the given string can be formed by repeating a substring.

Why Practice: Related to substring structure and repetition detection.

Practice Questions:

2. Longest Substring Without Repeating Characters — LeetCode #3

🔗https://leetcode.com/problems/longest-substring-without-repeating-characters/

Concept: Sliding window and unique substring logic.

Why Practice: Reinforces unique substring concepts.

Practice Questions:

3. Count Unique Substrings – LeetCode #1698

🔗 https://leetcode.com/problems/number-of-distinct-substrings-in-a-string/

Concept:
Use Trie (Prefix Tree) or Suffix Array to count the number of distinct substrings efficiently.

Why Practice:
This problem is a **direct application** of the practical — it strengthens understanding of **Trie-based substring counting**, **suffix generation**, and **string manipulation**.

Thanks