

Number of Distinct Substrings

Suffix Trie — count unique substrings in $O(n^2)$ Σ

● HARD

■ Hook

Count the number of distinct substrings of a string. Brute force: $O(n^3)$ generate all substrings, put in HashSet. Suffix Trie: insert all suffixes, count total nodes — each node = one distinct substring.

■ Core Intuition — Suffix Trie

Insert every suffix of the string into a Trie. Each path from root to any node represents a unique substring. Total distinct substrings = total nodes in the Trie (excluding root).

■ Visual — 'abc'

```
Suffixes of 'abc': 'abc', 'bc', 'c'
Trie after inserting all:
root → a → b → c*
      → b → c*
      → c*

Nodes (excluding root): a, ab, abc, b, bc, c = 6
Distinct substrings = 6 ✓
(They are: a,b,c,ab,bc,abc)
```

■ Code

```

public int countDistinctSubstrings(String s) {
    TrieNode root = new TrieNode();
    int count = 0;
    // Insert all suffixes
    for (int i=0; i<s.length(); i++) {
        TrieNode node = root;
        for (int j=i; j<s.length(); j++) {
            int idx = s.charAt(j) - 'a';
            if (node.children[idx] == null) {
                node.children[idx] = new TrieNode();
                count++; // new node = new distinct substring
            }
            node = node.children[idx];
        }
    }
    return count;
}

```

■ TIME

$O(n^2)$

■ SPACE

$O(n^2)$

n suffixes, each of average length $n/2$. Optimal: Suffix Array $O(n \log n)$ or Suffix Automaton $O(n)$.

- ✓ Insert every suffix into Trie
- ✓ Each new node created = 1 new distinct substring
- ✓ Total new nodes = answer
- ✓ Time: $O(n^2)$ | Space: $O(n^2)$
- ✓ Optimal: Suffix Automaton $O(n)$ time and space

■ MENTAL MODEL — Remember This Forever

Every path from root to any node in a Suffix Trie spells out exactly one distinct substring. New nodes only get created when a new unique prefix is seen. So counting node creations counts distinct substrings directly.