Practical No -24

Aim:- Write a Program to find Distinct substrings in a string

Example: String "apple"

1. Generating Substrings

The substrings of "apple" can be generated as follows:

- From index 0:
  - "a"
  - "ap"
  - "app"
  - "appl"
  - "apple"
- From index 1:
  - "p"
  - "pp"
  - "ppl"
  - "pple"
- From index 2:
  - "p"
  - "pl"
  - "ple"

- From index 3:
    - "l"
    - "le"
- From index 4:
    - "e"

So, the complete list of substrings for "apple" is:

- From index 0: "a", "ap", "app", "appl", "apple"
- From index 1: "p", "pp", "ppl", "pple"
- From index 2: "p", "pl", "ple"
- From index 3: "l", "le"
- From index 4: "e"

2. Inserting into Trie

Now, we will insert each of these substrings into the Trie. The Trie structure will ensure that only distinct substrings are stored. Here’s how the insertion would look:

- Insert "a":
    - Create a node for 'a'.
- Insert "ap":
    - Create a node for 'p' under 'a'.
- Insert "app":
    - Create a node for the second 'p' under 'ap'.
- Insert "appl":

    - Create a node for 'l' under 'app'.
- Insert "apple":
    - Create a node for 'e' under 'appl'.
- Insert "p":
    - Create a node for 'p' at the root level.
- Insert "pp":
    - Create a node for the second 'p' under the first 'p'.
- Insert "ppl":
    - Create a node for 'l' under 'pp'.
- Insert "pple":
    - Create a node for 'e' under 'ppl'.
- Insert "pl":
    - Create a node for 'l' under 'p'.
- Insert "ple":
    - Create a node for 'e' under 'pl'.
- Insert "l":
    - Create a node for 'l' at the root level.
- Insert "le":
    - Create a node for 'e' under 'l'.
- Insert "e":
    - Create a node for 'e' at the root level.

3. Counting Distinct Substrings

After inserting all the substrings into the Trie, we can count the distinct substrings. The countDistinctSubstrings method traverses the Trie and counts the number of distinct substrings.

Distinct Substrings of "apple"

The distinct substrings of "apple" are:

1. "a"
2. "ap"
3. "app"
4. "appl"
5. "apple"
6. "p"
7. "pp"
8. "ppl"
9. "pple"
10. "pl"
11. "ple"
12. "l"
13. "le"
14. "e"

Final Count

The total number of distinct substrings for the string "apple" is 14.

Program :- .

```
import java.util.HashSet;

class TrieNode {
    HashSet<TrieNode> children; // Using HashSet to avoid duplicates
    boolean isEndOfWord;

    public TrieNode() {
        children = new HashSet<>();
        isEndOfWord = false;
    }
}

class Trie {
    private TrieNode root;

    public Trie() {
        root = new TrieNode();
    }

    // Insert a substring into the Trie
    public void insert(String substring) {
```

```
        TrieNode node = root;
        for (char ch : substring.toCharArray()) {
            TrieNode childNode = new TrieNode();
            if (!node.children.contains(childNode)) {
                node.children.add(childNode);
            }
            node = childNode;
        }
        node.isEndOfWord = true; // Mark the end of the substring
    }

    // Count distinct substrings
    public int countDistinctSubstrings() {
        return countDistinctSubstrings(root) - 1; // Subtract 1 to exclude
the root
    }

    private int countDistinctSubstrings(TrieNode node) {
        int count = 1; // Count this node
        for (TrieNode child : node.children) {
            count += countDistinctSubstrings(child); // Count all children
        }
        return count;
    }
```

```
}

public class DistinctSubstrings {
    public static void main(String[] args) {
        String input = "banana";
        Trie trie = new Trie();

        // Generate all substrings and insert them into the Trie
        for (int i = 0; i < input.length(); i++) {
            for (int j = i + 1; j <= input.length(); j++) {
                String substring = input.substring(i, j);
                trie.insert(substring);
            }
        }

        // Count distinct substrings
        int distinctCount = trie.countDistinctSubstrings();
        System.out.println("Number of distinct substrings: " +
distinctCount);
    }
}
```

Output:-

```
Number of distinct substrings: 15
```