Aho–Corasick Algorithm: Report
Turganbek Nurhan SE-2435

1. Overview

The Aho–Corasick algorithm efficiently searches multiple patterns within a single long text. It constructs a finite automaton from the given patterns and processes the text in a single pass, reporting all pattern matches as they occur.

2. Algorithm Structure

The algorithm is built on three key structures:
• A **Trie** (prefix tree) for all patterns.
• **Failure links**, similar to the KMP fallback mechanism, for skipping mismatched branches.
• **Output links** that track which patterns end at each node.

3. Construction Steps

1. Insert all patterns into a Trie.
2. Build failure links via BFS to handle mismatches.
3. Merge output links to ensure all suffix matches are found.

4. Time and Space Complexity (Corrected)

- **Building phase**: $O(\Sigma \mid \text{pattern} \mid + \Sigma \text{ alphabet}) = O(\text{total pattern length})$.
- **Search phase**: $O([\text{text}] + \text{number of matches})$.

This means the algorithm processes each character of the text exactly once, and each match is reported in constant time.

Space complexity: $O(\text{total\_trie\_nodes} \times \text{alphabet\_size})$, which depends on the number of distinct characters and total pattern nodes.

5. Comparison with Other Algorithms

Compared to naive search and KMP:

- Naive search: $O([\text{text}] \times \mid \text{patterns}])$
- KMP: $O([\text{text}] + \mid \text{pattern}])$, but only for one pattern
- Aho–Corasick: $O([\text{text}] + \Sigma \mid \text{patterns}] + \text{number\_of\_matches})$, efficient for multiple patterns

6. Practical Results

In this Java implementation, the automaton was tested with Formula 1–related pattern sets (drivers and teams). The algorithm correctly found all pattern occurrences in a single text pass. This confirms both its correctness and performance efficiency.

7. Conclusion

The Aho–Corasick algorithm combines Trie-based structure and KMP-like backtracking to achieve linear-time multi-pattern search. Its efficiency makes it ideal for text analytics, intrusion detection, and real-time log scanning.