Aho-Corasick Algorithm – Multiple Patterns matching

ChaoHui Zheng
University of Tennessee
CS594 Class
October 5 2021

Problem

- Given Patterns P1, P2, P2, P3, ….
- Given a text string T
- Find all occurrences of P1, P2, P3 … in text T

Applications

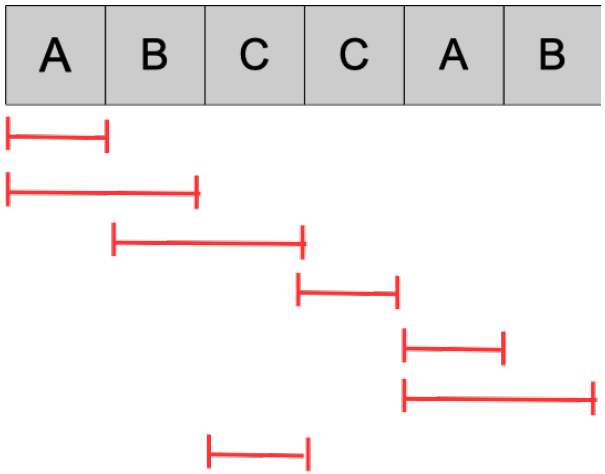
- Regular expression
- Plagiarism detection
- Spell checking
- Popular Interview question

Input/Output example

Patterns

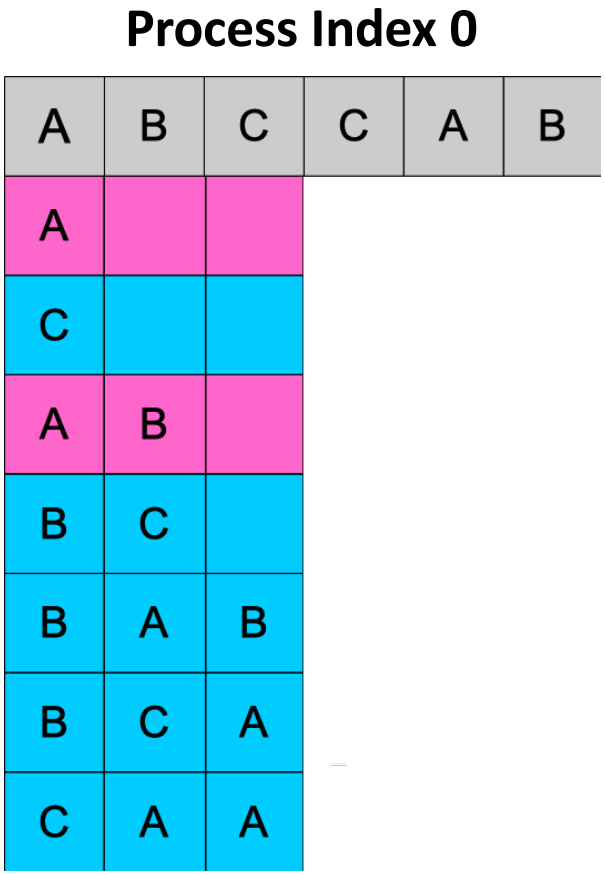
A		
C		
A	B	
B	C	
B	A	B
B	C	A
C	A	A

Text



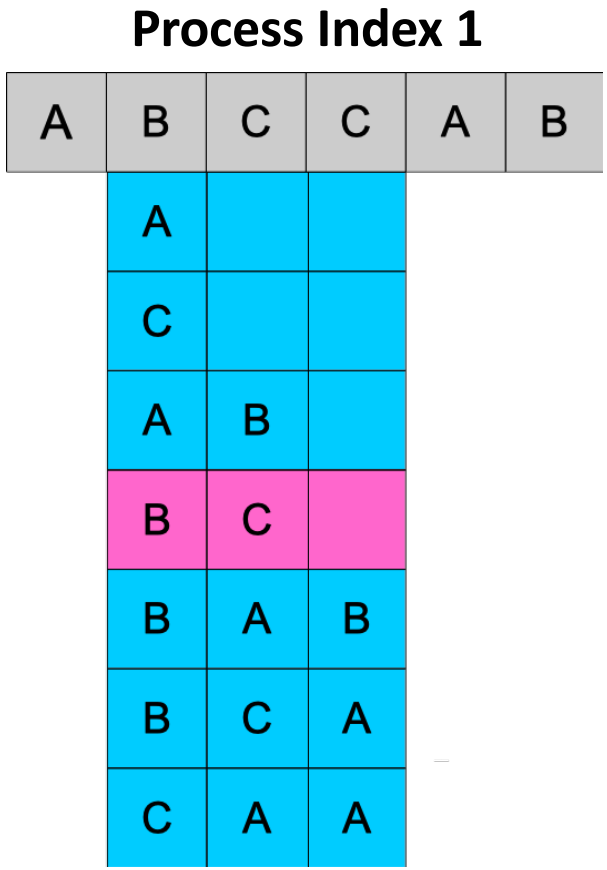
Naïve approach

- For every index i in text T, we check all patterns



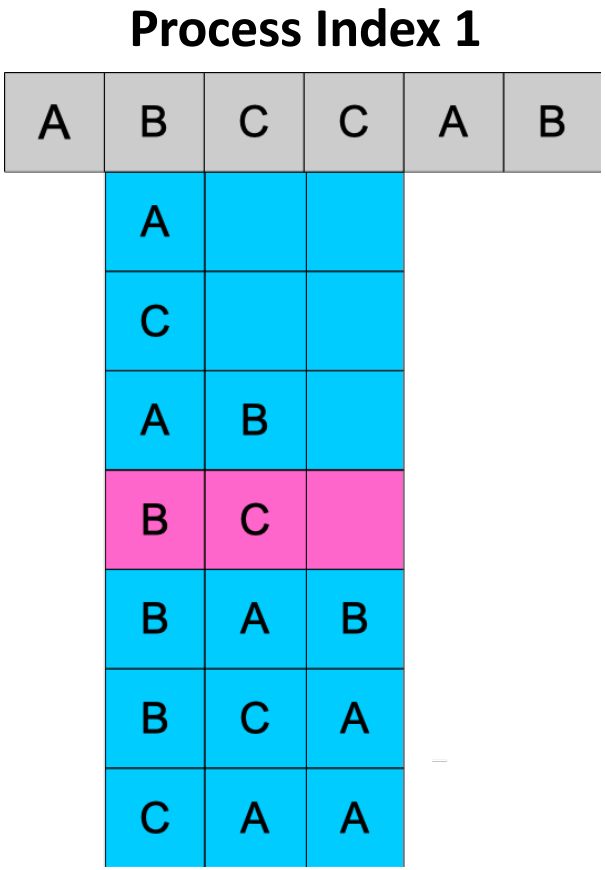
Naïve approach

- For every index i in text T, we check all patterns



Naïve approach

- For every index i in text T, we check all patterns
- Big O: O(L(T) * (L(P1) + L(P2) + L(P3) …)

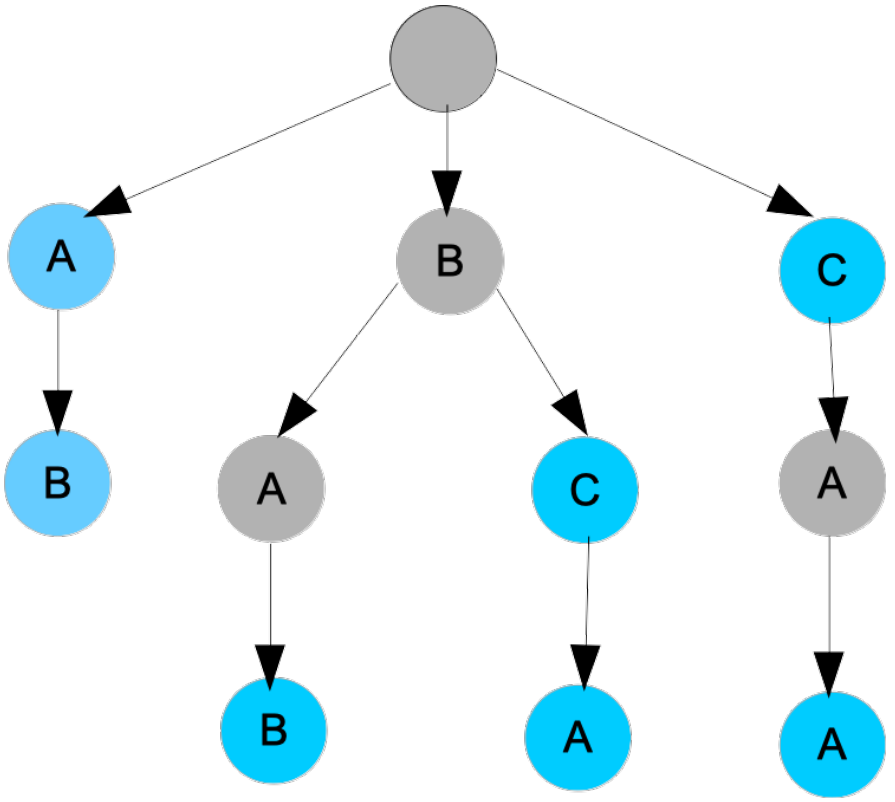


Can we do better?

Aho-Corasick

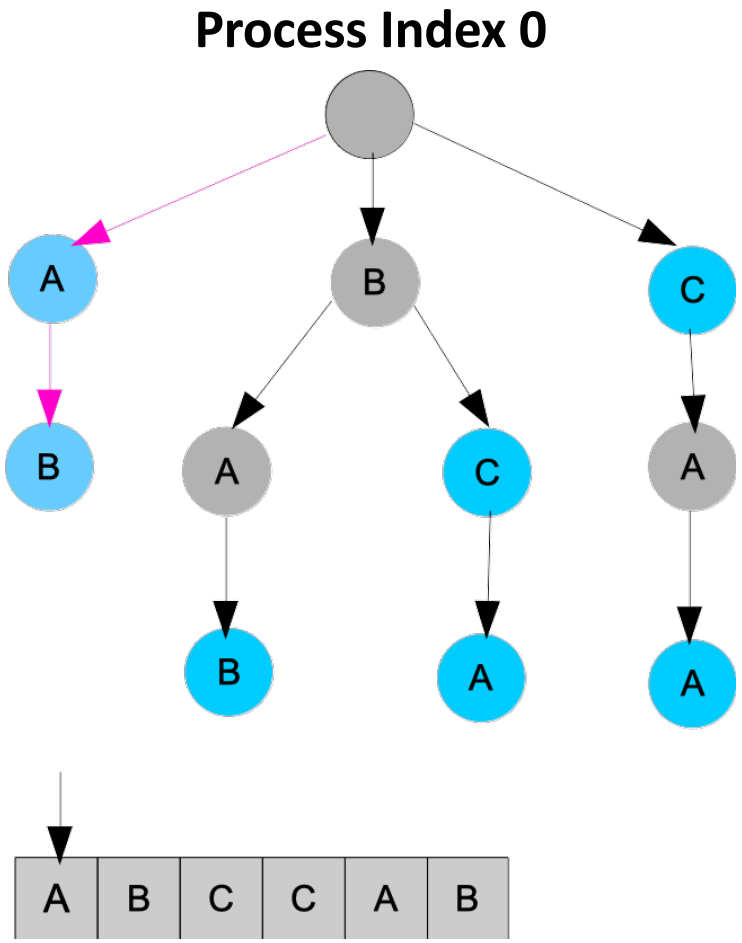
1. Build the graph representation for patterns (Trie)

A		
C		
A	B	
B	C	
B	A	B
B	C	A
C	A	A



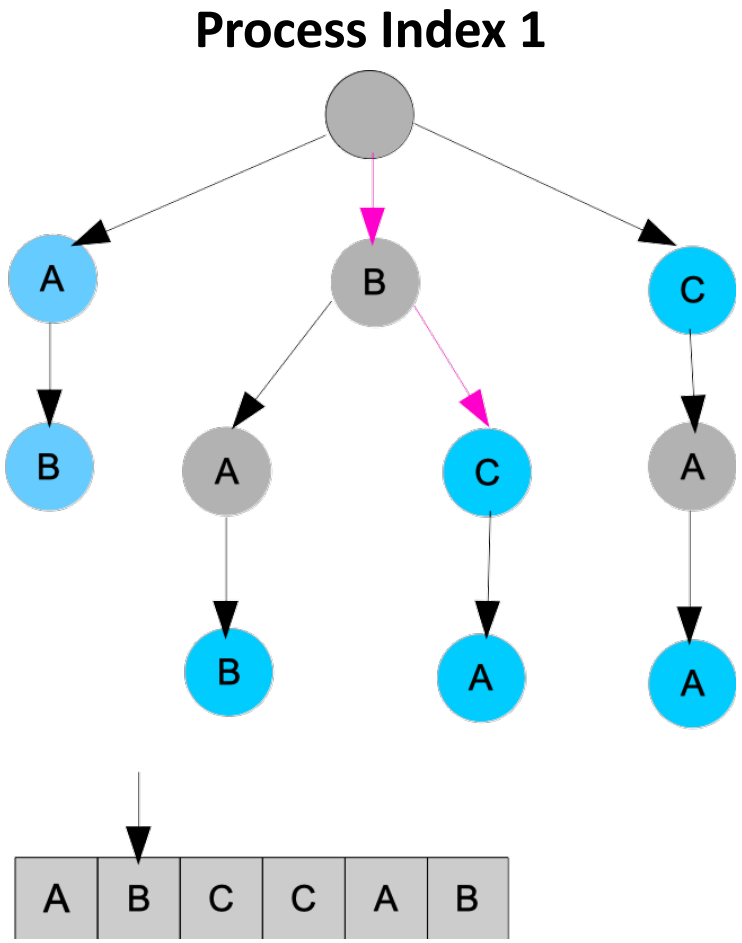
Aho-Corasick

1. Build the graph representation for patterns (Trie)



Aho-Corasick

1. Build the graph representation for patterns (Trie)

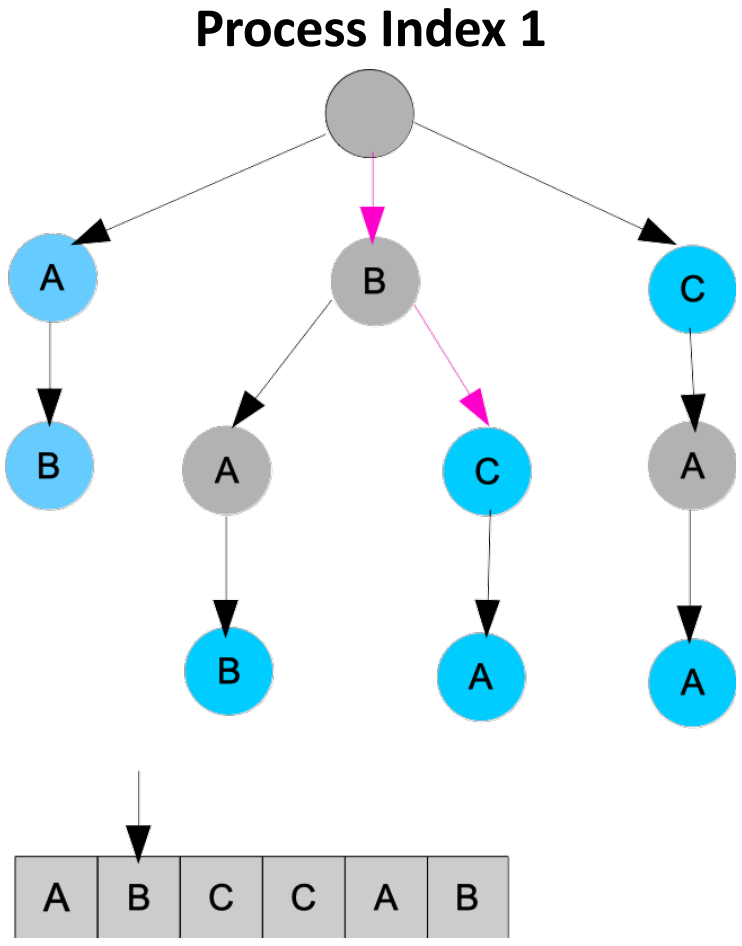


Aho-Corasick

1. Build the graph representation for patterns (Trie)

O(L(T) * L_max(P1, P2, P3, …))

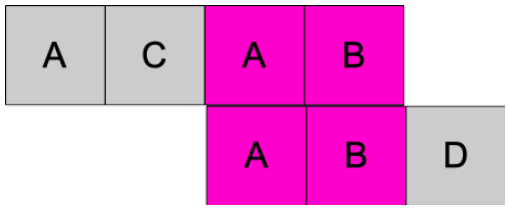
Can we do better?



Aho-Corasick

1. Build the graph representation for patterns (Trie)

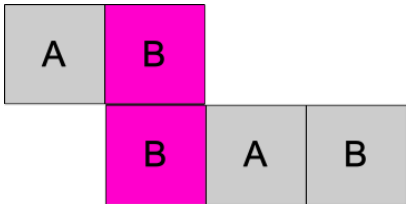
Path overlap – The ending of one path overlaps the beginning of the other path.

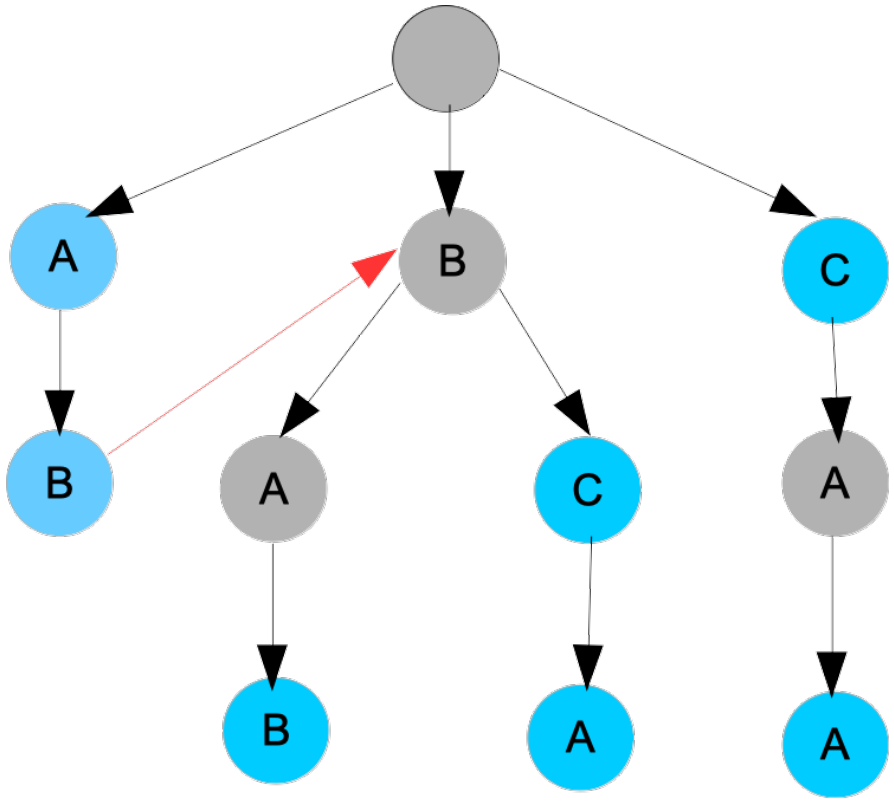


Aho-Corasick

1. Build the Trie
2. Add failure links.

Overlap B

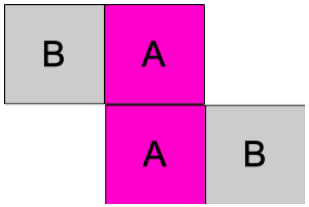


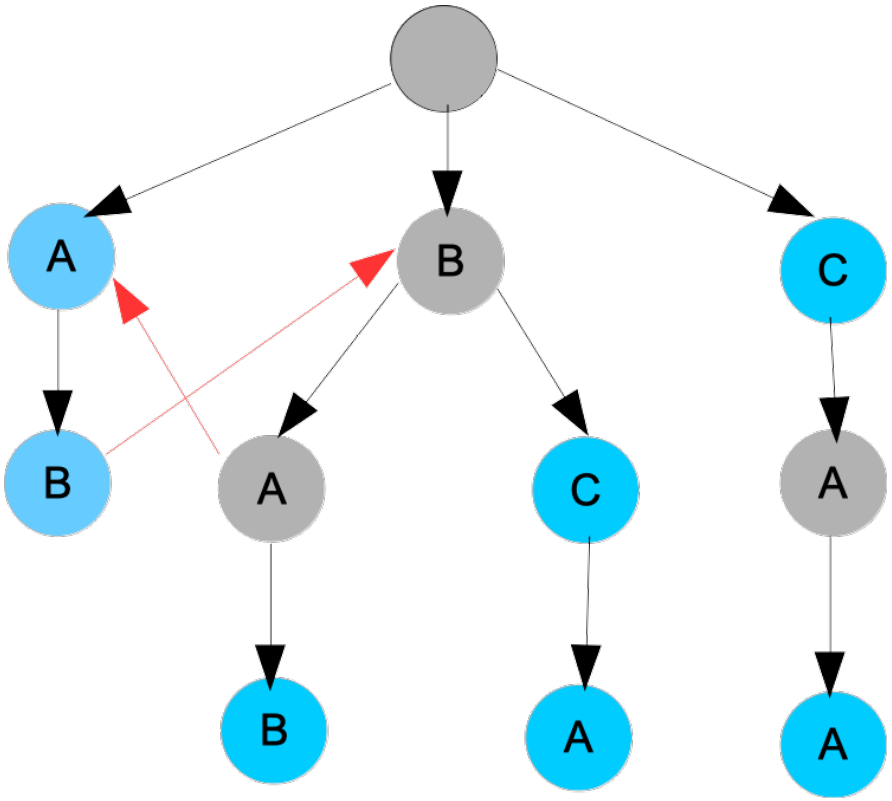


Aho-Corasick

1. Build the Trie
2. Add failure links.

Overlap A

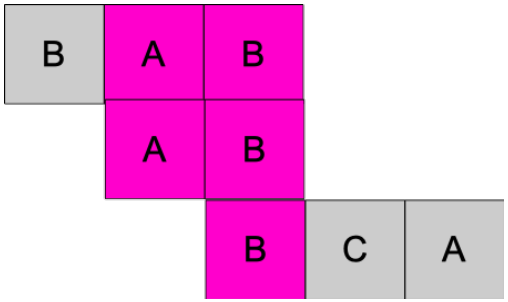


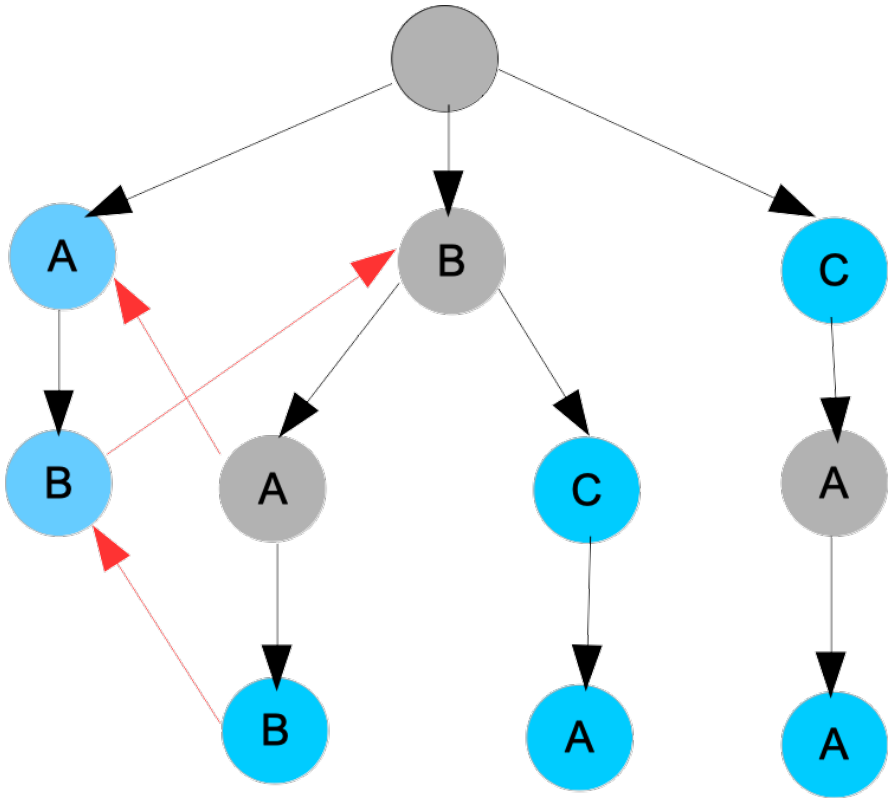


Aho-Corasick

1. Build the Trie
2. Add failure links.

Largest overlap AB

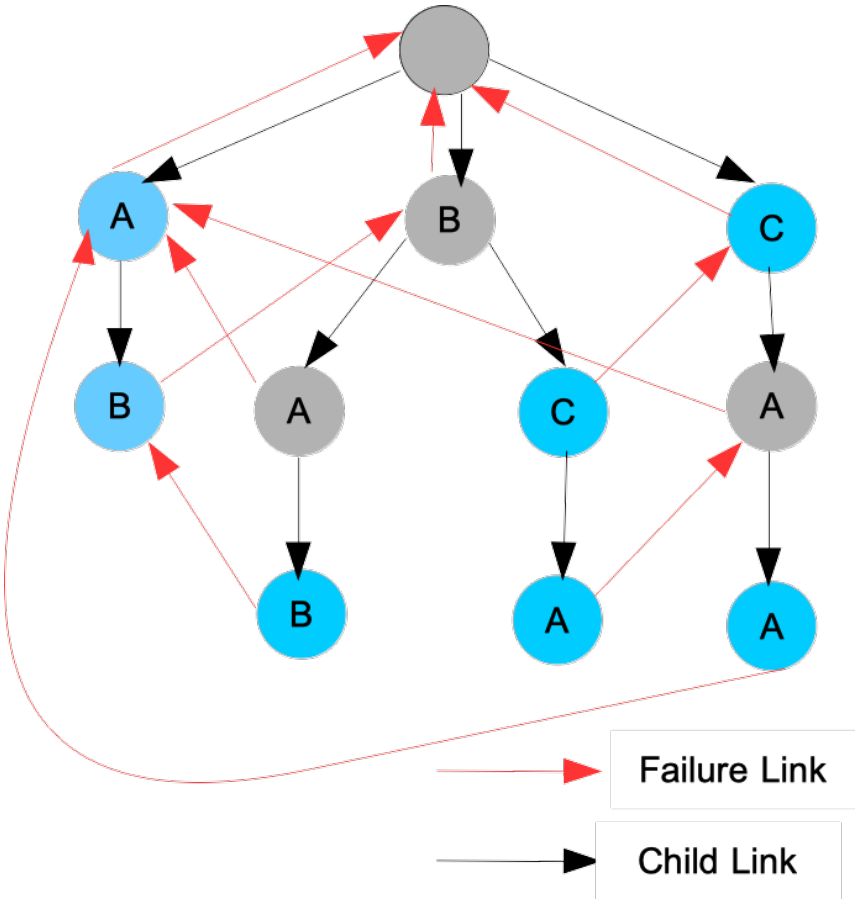




Aho-Corasick

1. Build the Trie
2. Add failure links.

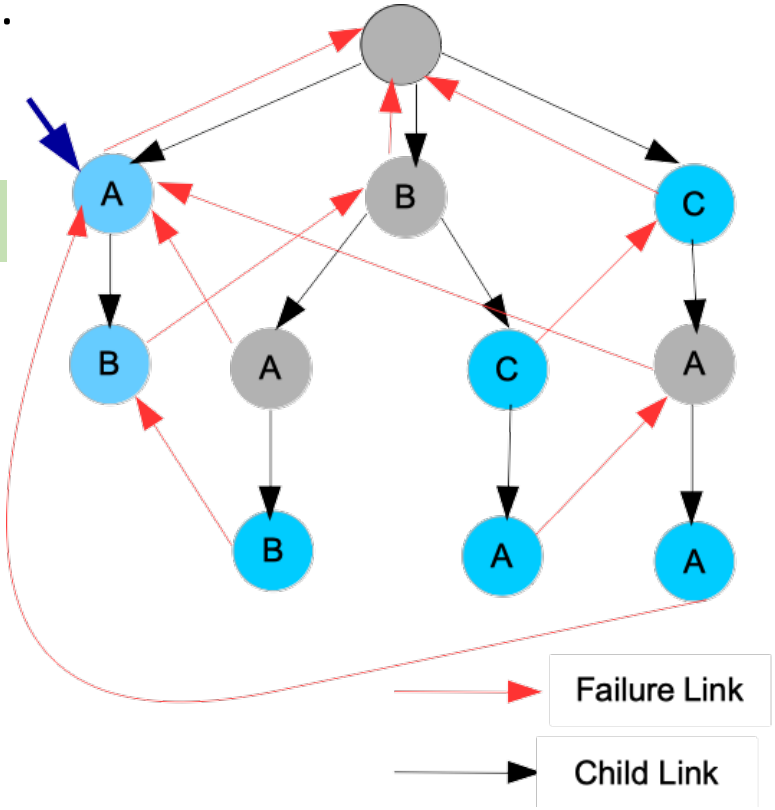
Follow the failure links if it fails to find the match

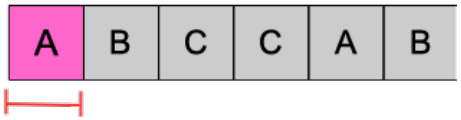


Aho-Corasick

1. Build the Trie
2. Add failure links.

Match A

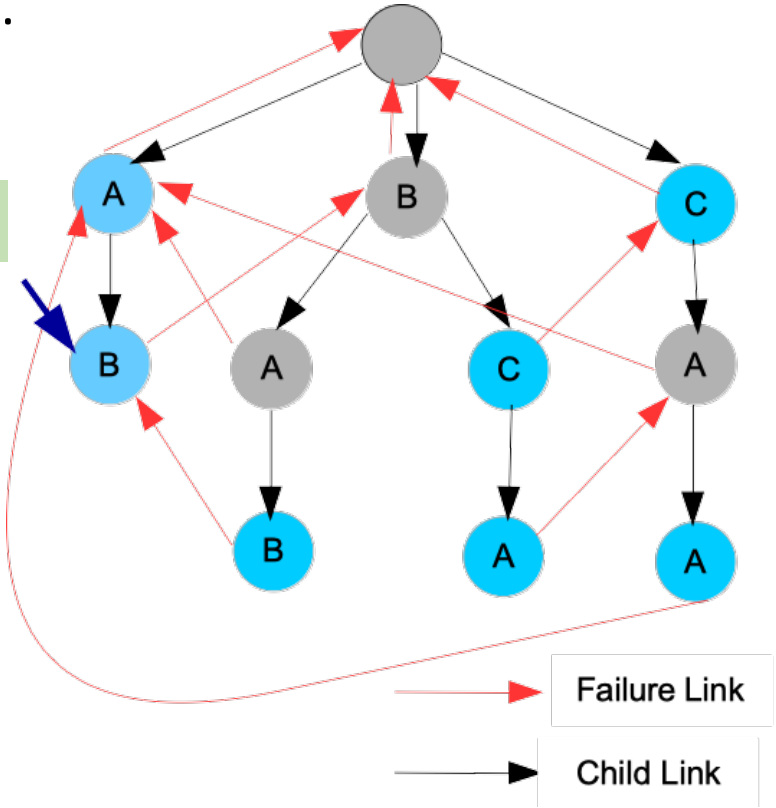


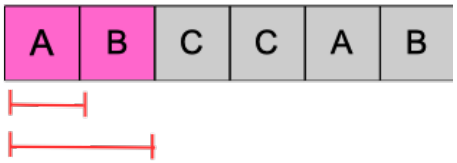


Aho-Corasick

1. Build the Trie
2. Add failure links.

Match AB



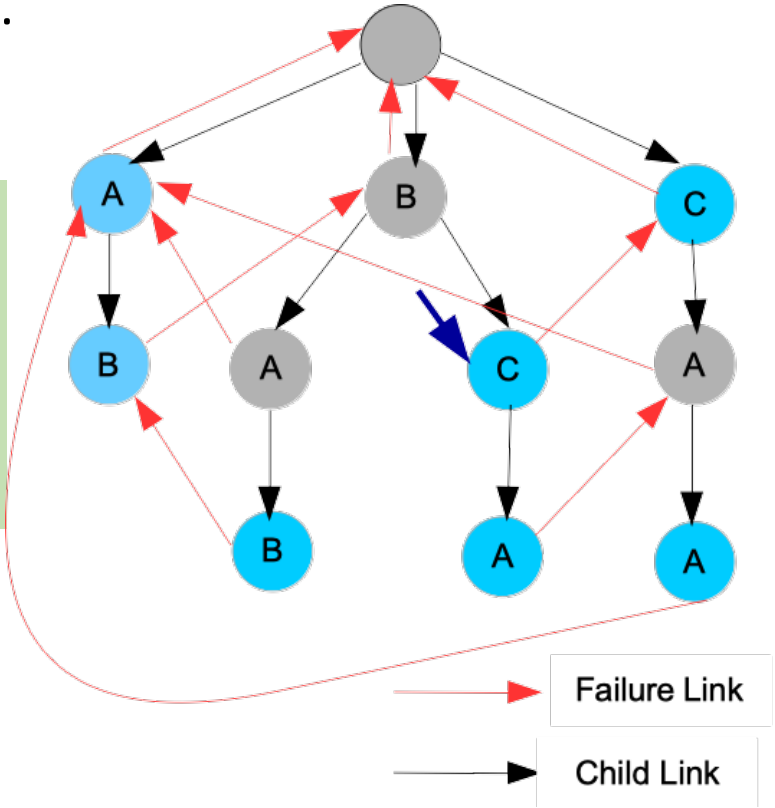


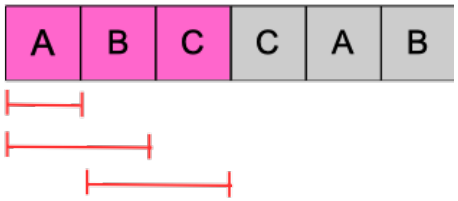
Aho-Corasick

1. Build the Trie
2. Add failure links.

Follow failure link to middle B, and then move to C

Match BC



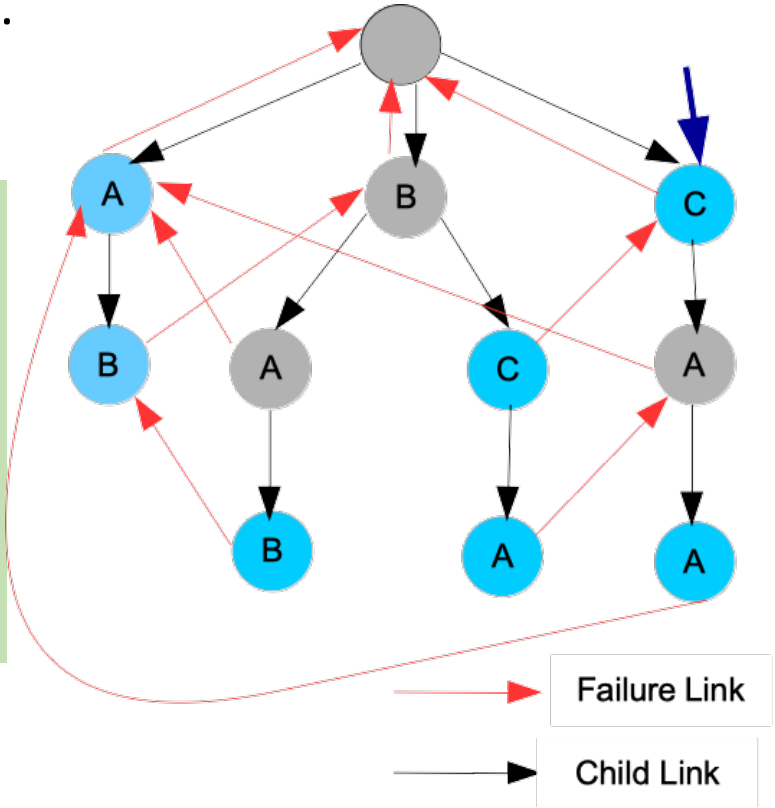


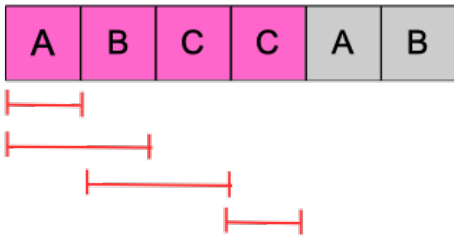
Aho-Corasick

1. Build the Trie
2. Add failure links.

Follow failure link to C, and then follow failure link again to root, and then move to C

Match C

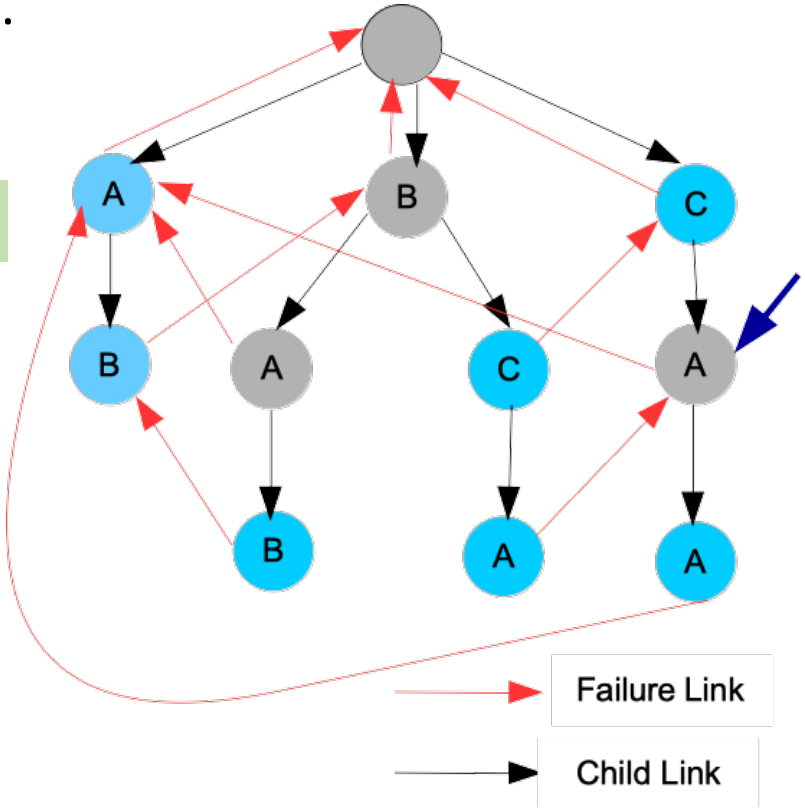


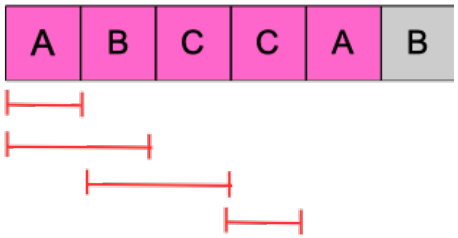


Aho-Corasick

1. Build the Trie
2. Add failure links.

Match nothing

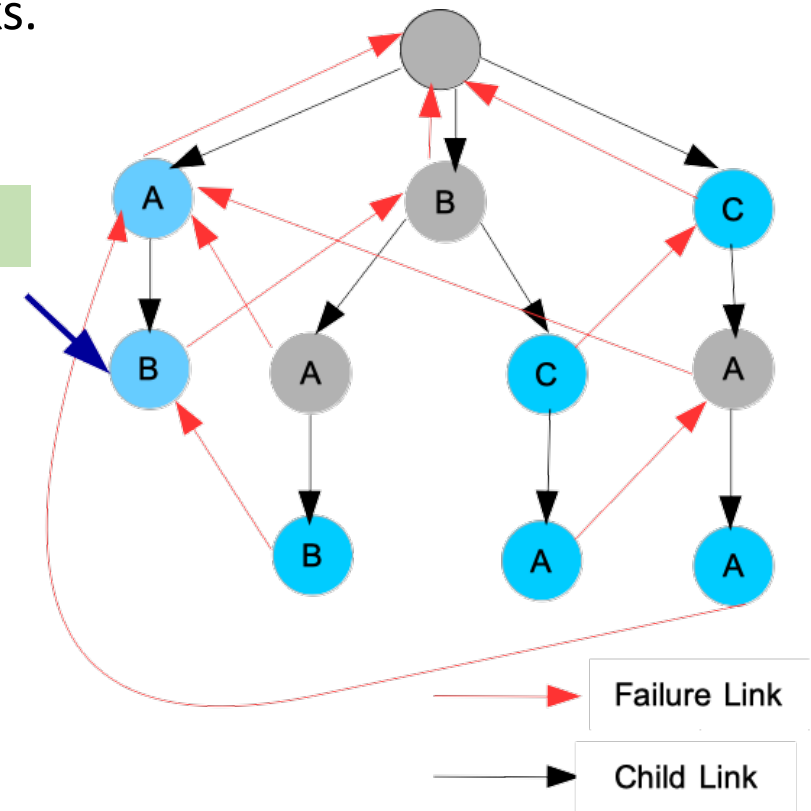


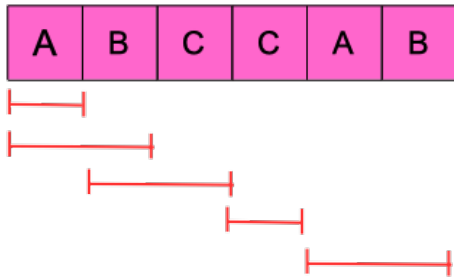


Aho-Corasick

1. Build the Trie
2. Add failure links.

Match AB

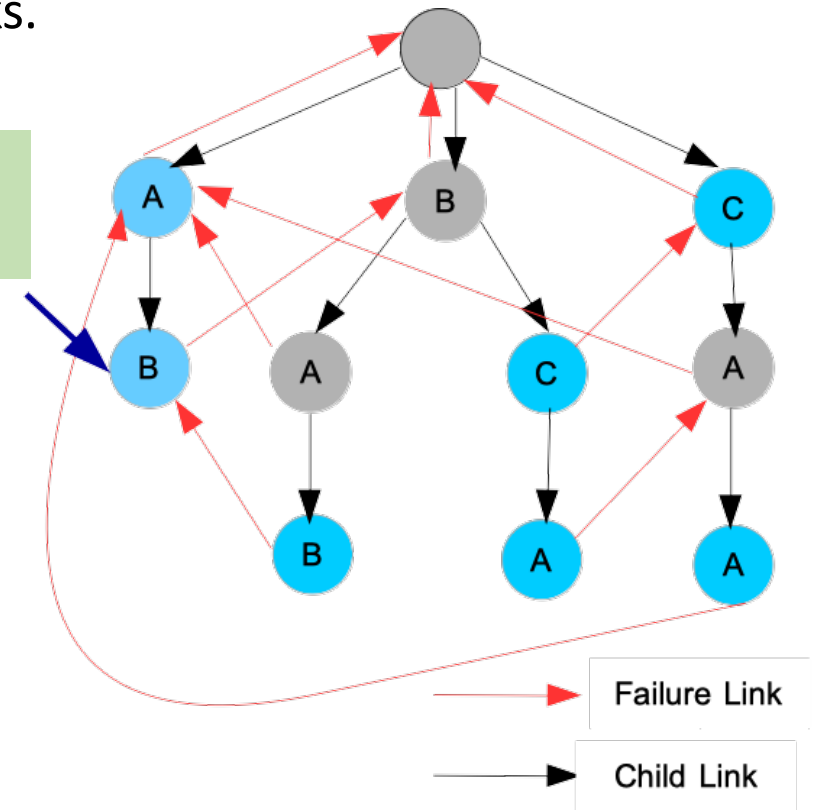


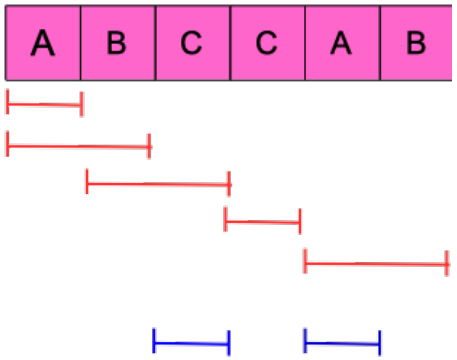


Aho-Corasick

1. Build the Trie
2. Add failure links.

We are missing C and A



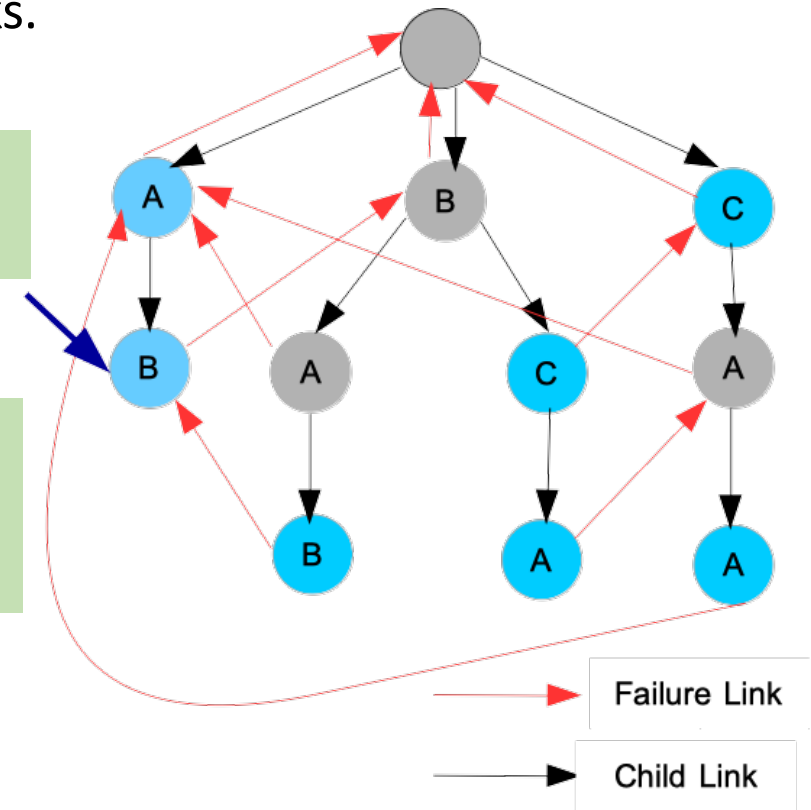


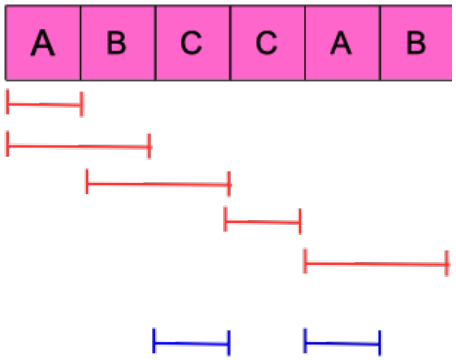
Aho-Corasick

1. Build the Trie
2. Add failure links.

We are missing C and A

One pattern may be a substring of other patterns

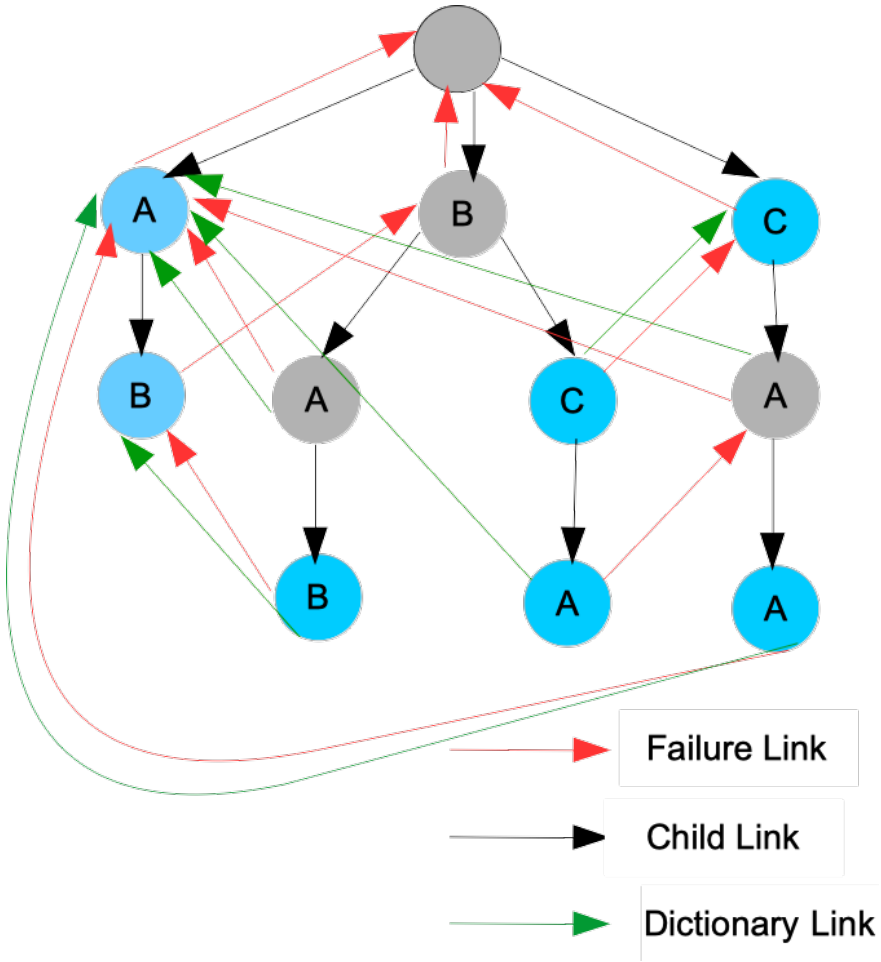




Aho-Corasick

1. Build the Trie
2. Add failure links.
3. Add dictionary links.

- To create dictionary link, for each node, follow failure links until we hit root node or blue node.
- During search, follow the dictionary links.

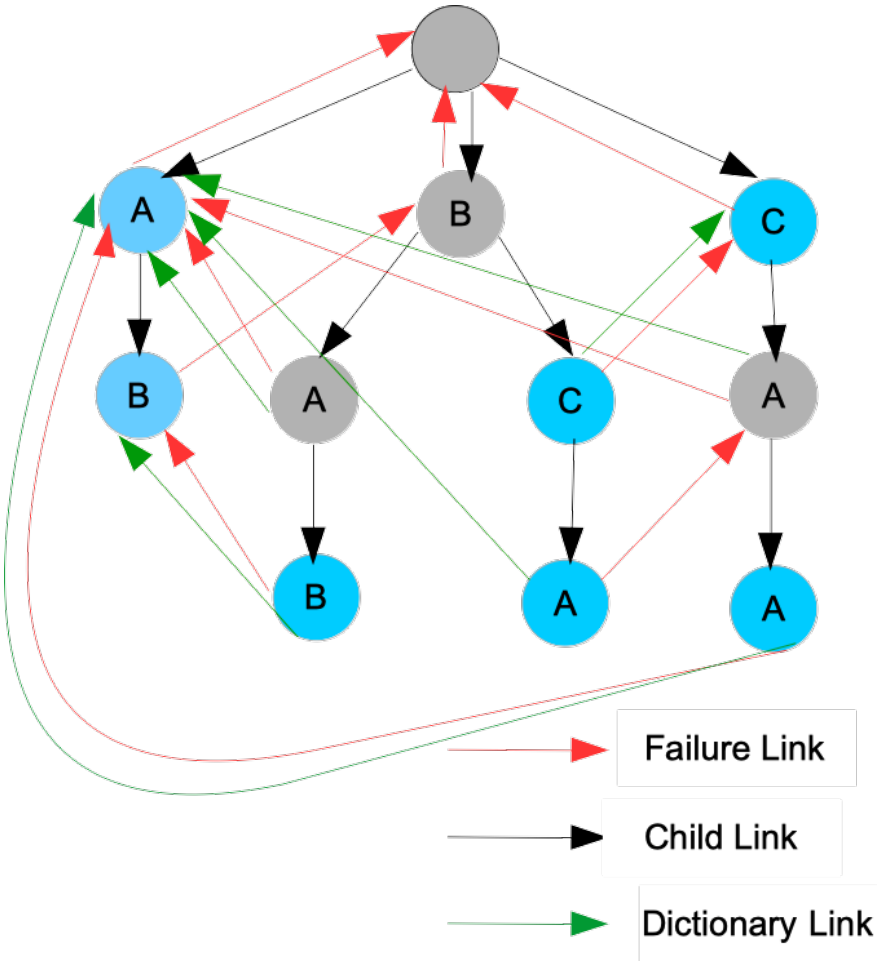


Aho-Corasick

1. Build the Trie
2. Add failure links.
3. Add dictionary links.

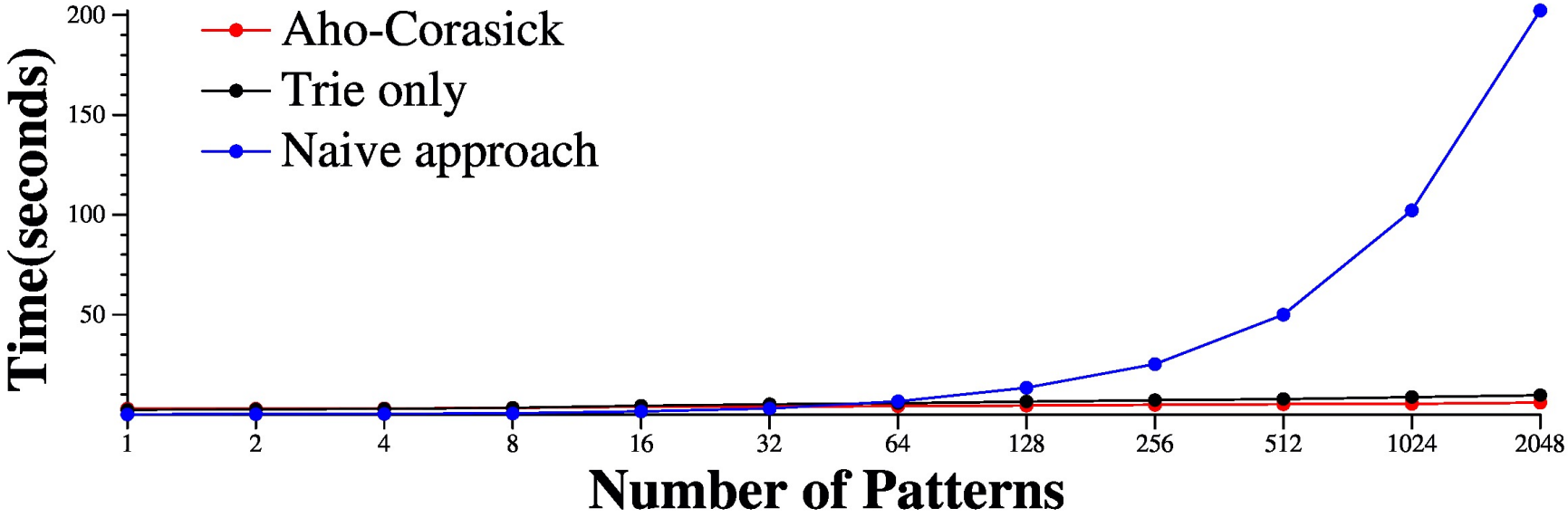
- To create dictionary link, for each node, follow failure links until we hit root node or blue node.
- During search, follow the dictionary links.

O(L(T) + M) where M is the number of matches



Experiments

- Randomly choose n patterns from roughly 10,000 most common English words
- Randomly choose 10,000,000 words from all English words bank.



Thanks!
Implementations:
https://github.com/czheng4/Aho-Corasick