Topic 8: KMP, Automata

What we know about strings:

- Rolling hash
- Trie

Good for 90% of the problems!

What we will introduce today:

- KMP
- AC Automata

Review: Tip of your tounge (NAQ '23).

KMP

Motivation

Word search:

Find 'Bob' in 'Alice Bob --'

Naive algo: $O(|w| \cdot |S|)$

Rolling hash: probabilistic* $O(|S|)$

Goal: deterministic $O(|S|)$

(itself not as important,
but the research motivates other probs)

KMP: deterministic $O(|s|)$

Intuition: Word search is avg. fast

'Bob' in 'Alice Bob —'

B only appeared once!

Counterexample: 'a' ×500 in 'a' ×1000

vs.

'a' ×500 in 'a' ×499 + b

We only need to optimize for bad scenarios

Optimization 1: Throw away everything after a mismatch.

$w = aaa$ $\quad S = aabaaa$

aaa aabaaa

When we encounter b, we can throw away the matched prefix and start after b!

Complexity $O(|S|)$?

Optimization 1: Throw away everything after a mismatch.

$w = aaa$ $S = aabaaa$

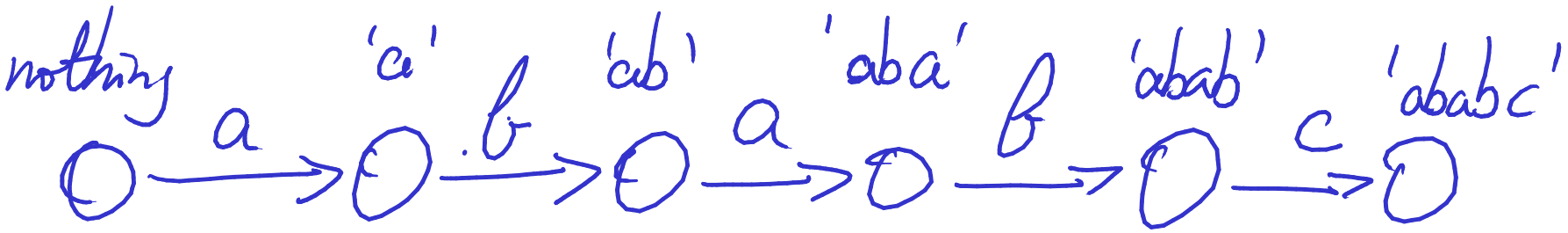
aaa aabaaa

When we encounter b, we can throw away the matched prefix and start after b!

Complexity $O(|S|)$?

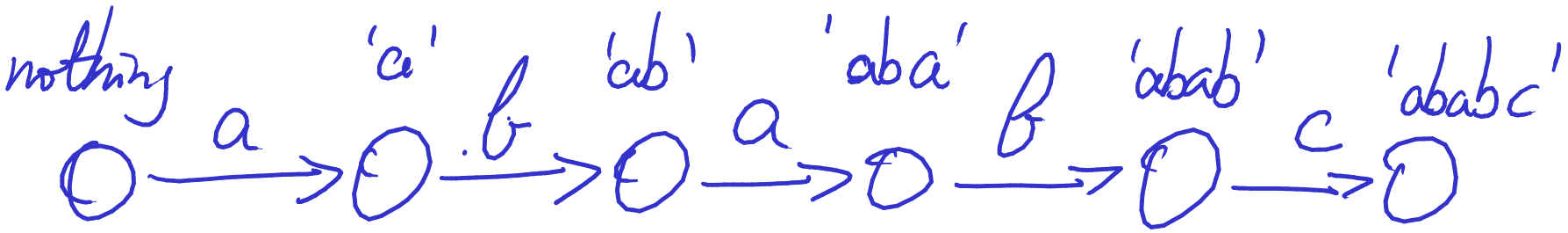
Counterexample 'abab' in 'ababab'

Optimization 2: Automata-based approach



$$S = abababc$$

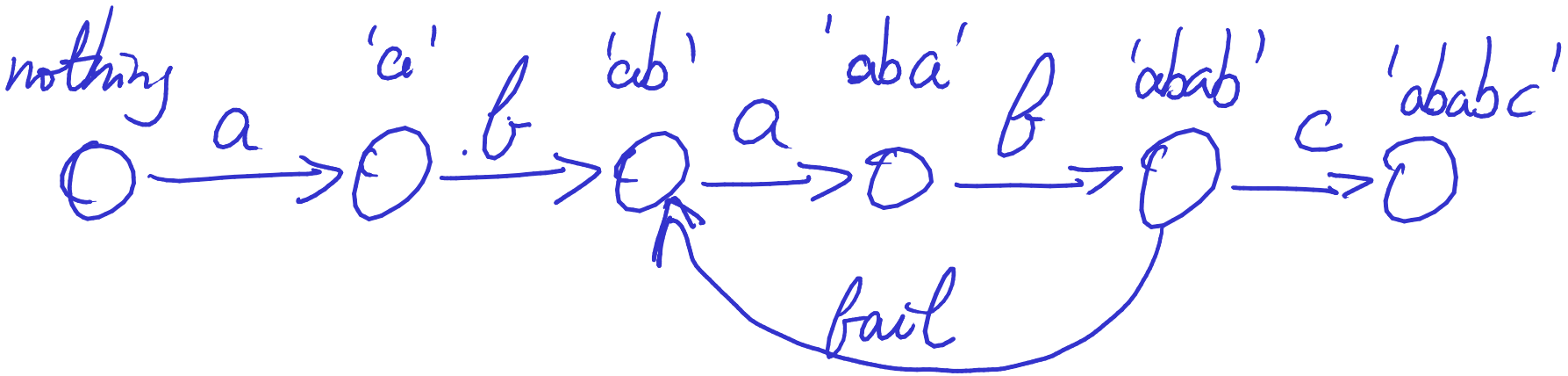
Optimization 2: Automata-based approach



Problem: need to figure out where to go after failing to advance.

Intuition: go to a 'weaker' node and try to advance again.

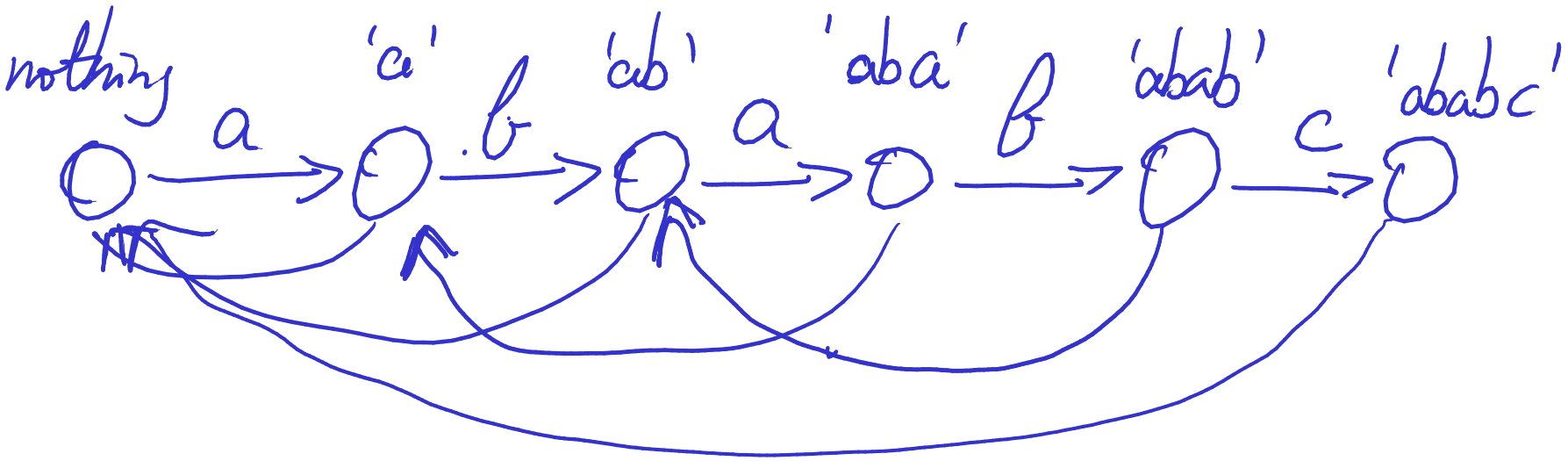
Optimization 2: Automata-based approach



* whatever ends with 'abab' also ends with 'ab'.

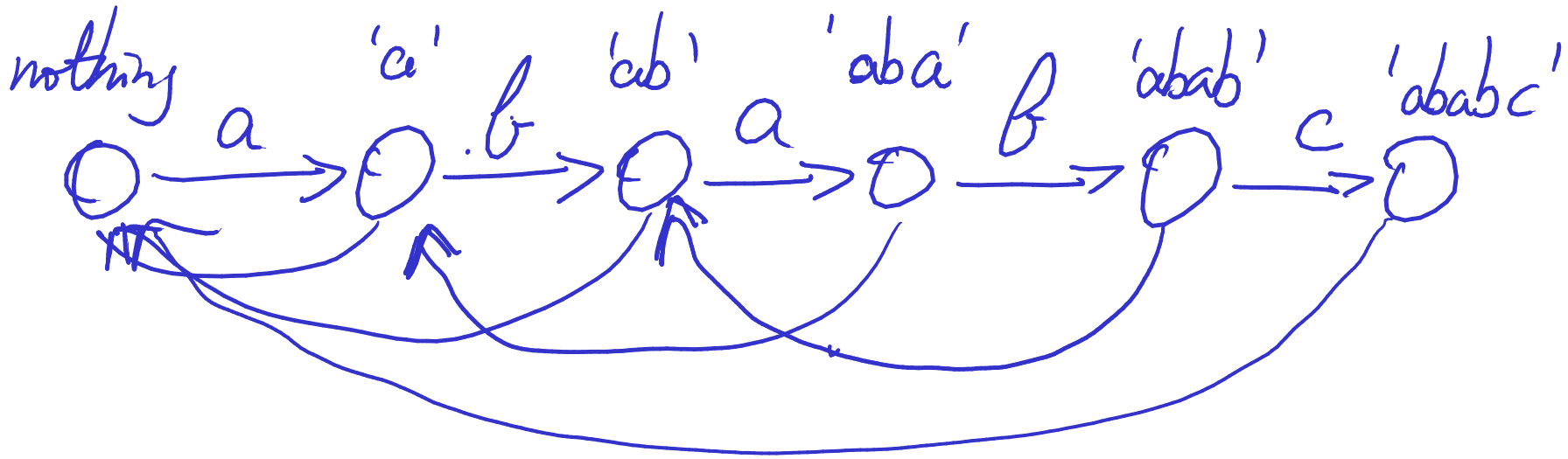
Intuition: go to a 'weaker' node and try to advance again.

Optimization 2: Automata-based approach



Intuition: go to a 'weaker' node and try to advance again.

Optimization 2: Automata-based approach

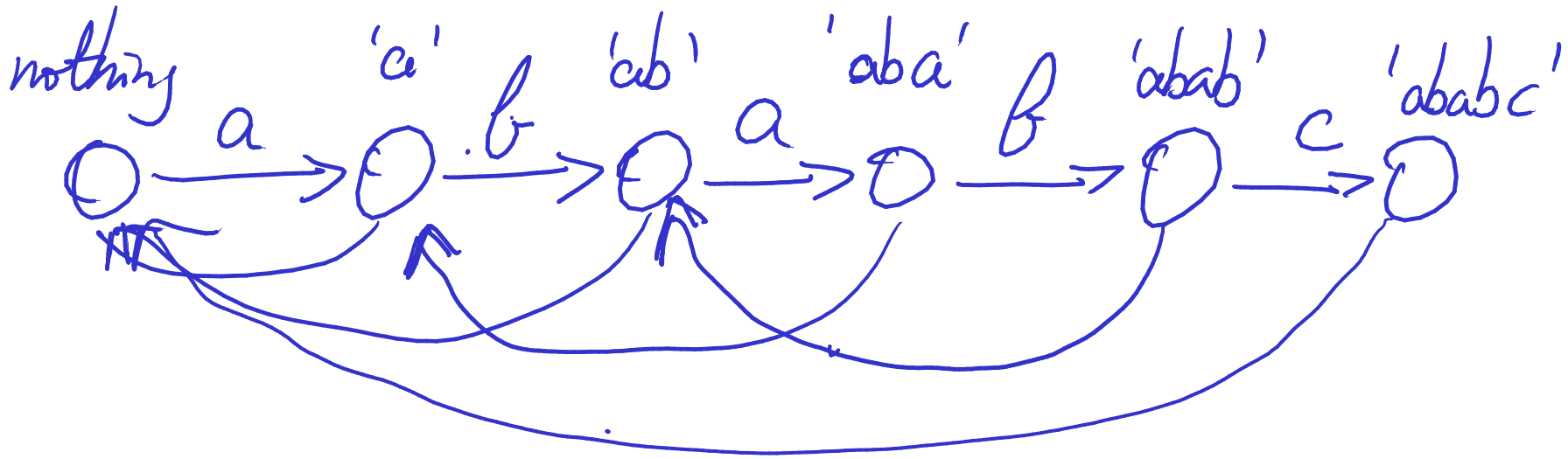


☆ fail[i] can be inferred from fail[i-1].

'aba' ⟶ 'a'

'aba'+'b' ⟶ 'a'+'b'

Optimization 2: Automata-based approach




```
f = fail[i-1], ch = w[i]
while (f is not start) {
    if (f advances with ch) {
        break;
    }
    f = fail[f];
}
if (f advances with ch) {
    fail[i] = f+1;
} else fail[i] = start;
```

Refer to codebase for an exact code !!!

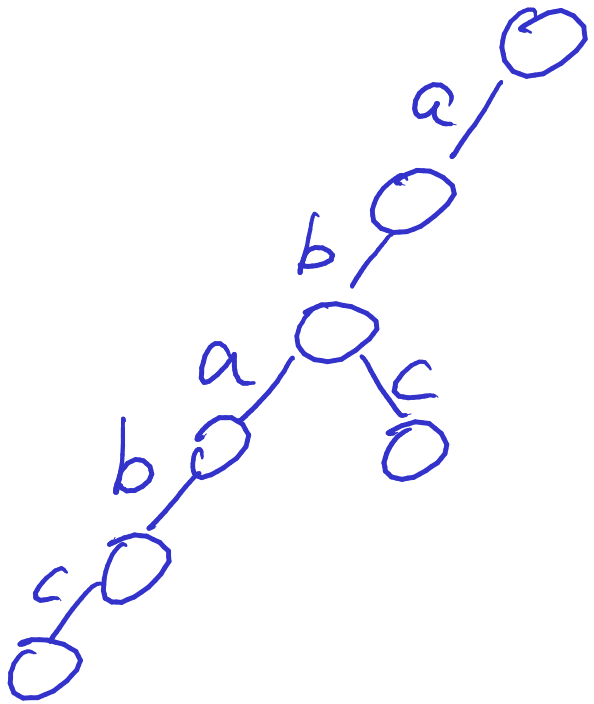
Compress Woods from CF 1200E

AC Automata

What if we want to match multiple strings simultaneously?

Sol: we build on a trie!

e.g. 'ababc' and 'abc'

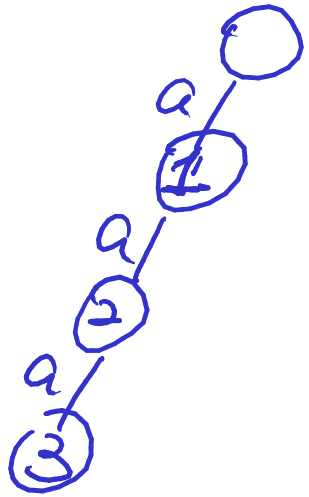


AC Automata

Caveat 1: a state may match multiple strings.

$w = \{'a' \quad 'aa' \quad 'aaa'\}$

$S = 'aaa'$




```
match_cnt[i] =
    match_cnt[fail[i]] +
    [i ∈ w];
```

Caveat 2: Jump to fail is slow in const.

Preprocess every jump as go[i]['a'], etc.

Indie Album on CF 1207G