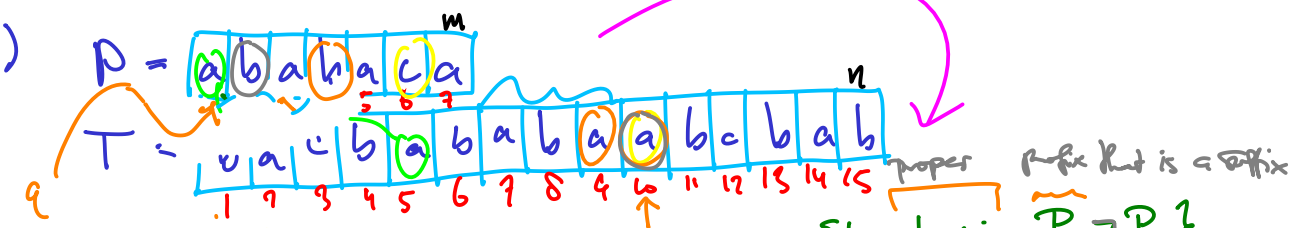
CS 445 02/26/19

1. Knuth-Morris-Pratt : trace
2. Aho-Corasick

find all occurrences of P in T (possibly overlapping)

1)



i	1	2	3	4	5	6	7
P[i]	a	b	a	b	a	c	a
π[i]	0	0	1	2	3	0	1

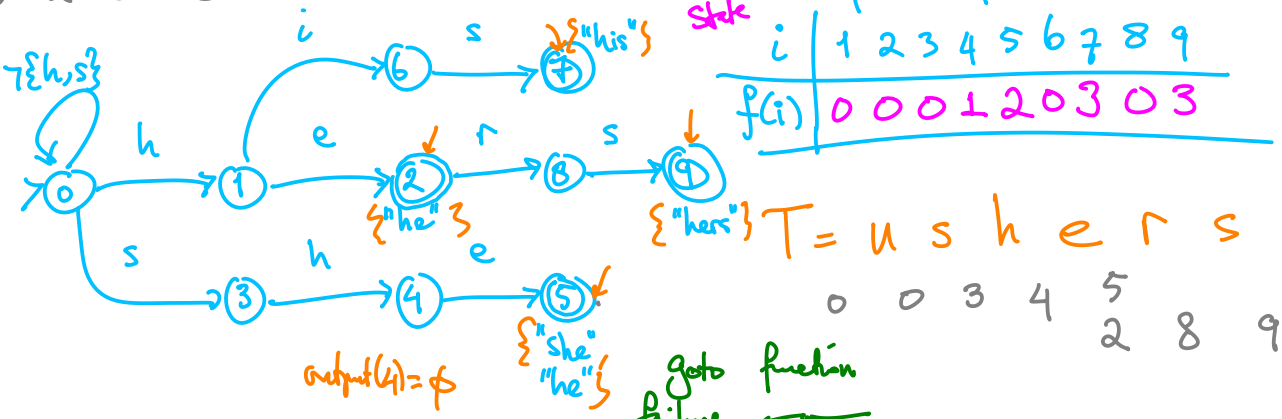
$\pi[i] = \max\{k : k < i, P_k \sqsupset P_i\}$

```
q ← 0
for i ← 1 to n
    while (q > 0) and P[q+1] ≠ T[i]:        (P[2] = b, T[3] = c)
        [ q ← π[q] ]
    if P[q+1] == T[i]:
        [ q ← q+1 ]
    if q == m:
        [ print "Match"
          q = π[q] ]
```

Aho-Corasick: {he, she, his, hers} ✓ → T

P_1 P_2 P_3 P_4

build a trie



state i	1	2	3	4	5	6	7	8	9
f(i)	0	0	0	1	2	0	3	0	3

T = u s h e r s

0 0 3 4 5 2 8 9

goto function

failure

output

"she" "he" "hers"

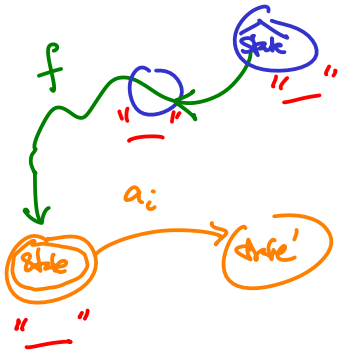
Pattern Matcher (Aho-Corasick)

Input: $x = a_1 a_2 \dots a_n$

g = goto

f = failure

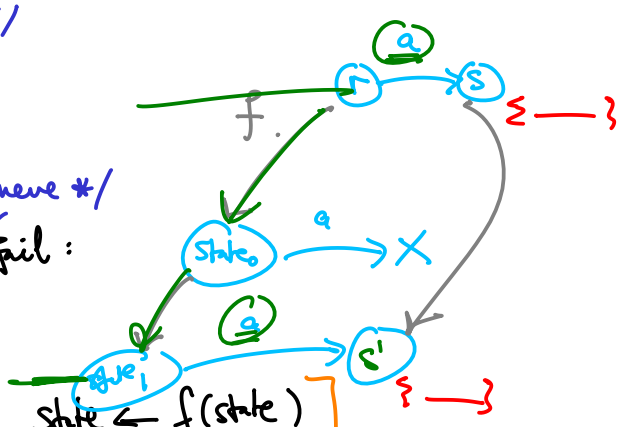
1. state ← 0
2. for i ← 1 to n:
3. [while g(state, a_i) = fail:
4. state ← f(state)]
5. state ← g(state, a_i)
6. if output(state) ≠ ϕ:
7. print i, output(state)



Algorithm 3: Construction of the failure function

Q = queue

1. Q ← ϕ
2. for each a s.t. g(0, a) = s ≠ 0:
3. Q ← Q ∪ {s} /* enqueue */
4. f(s) ← 0
5. while Q ≠ ϕ:
6. r ← next state in Q /* dequeue */
7. for each a s.t. g(r, a) = s ≠ fail:
8. Q ← Q ∪ {s}
9. state ← f(r)
10. [while g(state, a) = fail do state ← f(state)]
11. f(s) ← g(state, a)
12. output(s) ← output(s) ∪ output(f(s))



Next topic: Regular Expression ⟶ DFA