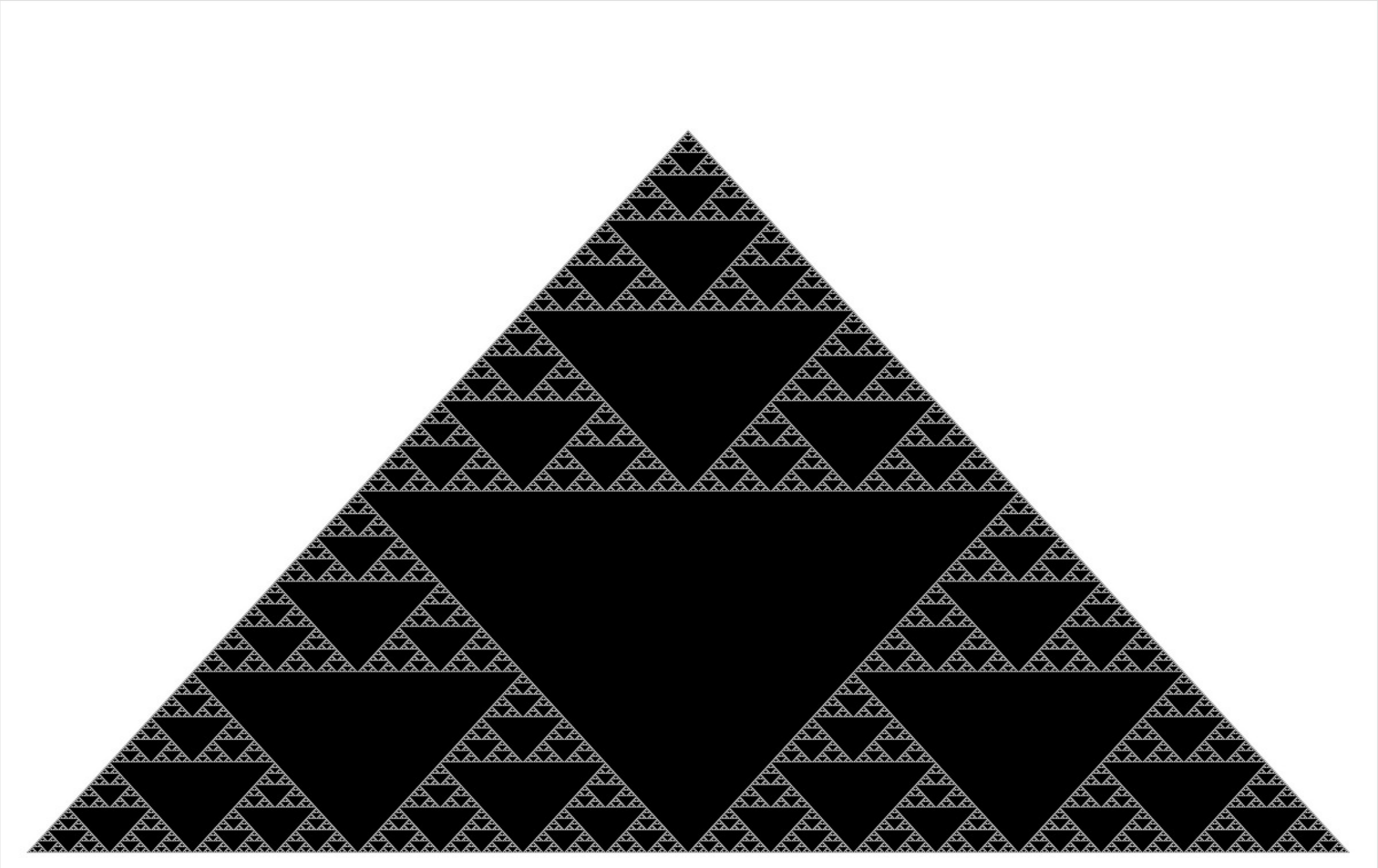
String matching



Knuth-Morris-Pratt

Faster computation by using pattern symmetries within itself (vs transitions for each char/state)

The function π does this, namely

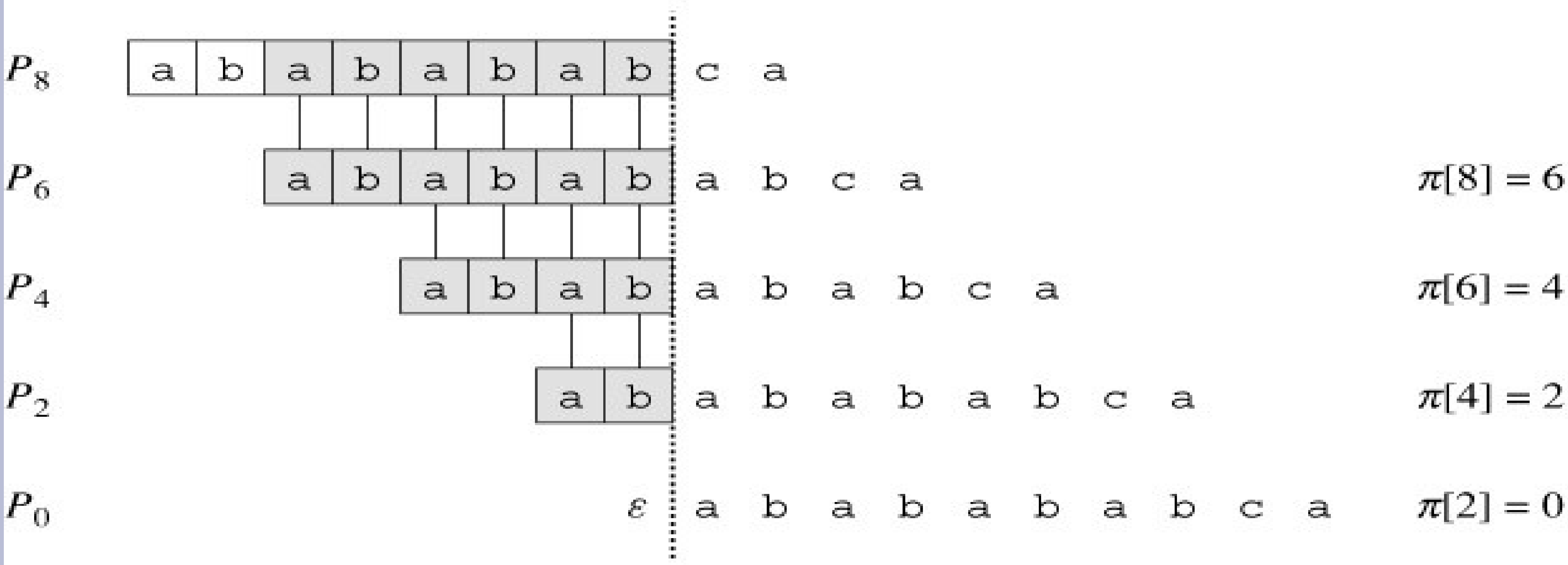
$\pi(q) = \max(k : k < q \text{ and } P_k \;] \; P_q)$

Namely, π finds shifts of P on itself

Knuth-Morris-Pratt

i	1	2	3	4	5	6	7	8	9	10
$P[i]$	a	b	a	b	a	b	a	b	c	a
$\pi[i]$	0	0	1	2	3	4	5	6	0	1

(a)



(b)

Knuth-Morris-Pratt

T = “abcabaabcaca”

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

Start q=0, see T[1]='a'=P[q+1]=P[1]

At q=1, see T[2]='b'=P[q+1]=P[2]

At q=2, see T[3]='c'... not P[q+1]

π(q) = π(2) = 0. At 0, stop follow π

At q=0, see T[4]='a'=P[q+1]=P[1]

At q=1, see T[5]='b'=P[q+1]=P[2]

Knuth-Morris-Pratt

T = “abcabaabcaca”

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

At q=1, see T[5]='b'=P[q+1]=P[2]

At q=2, see T[6]='a'=P[q+1]=P[3]

At q=3, see T[7]='a'=P[q+1]=P[4]

At q=4, see T[8]='b'=P[q+1]=P[5]

At q=5, see T[9]='c'=P[q+1]=P[6]

At q=6, see T[10]='a'=P[q+1]=p[7]

Knuth-Morris-Pratt

T = “abcabaabcaca”

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

At q=6, see T[10]='a'=P[q+1]=p[7]

Match! Set q=π(q)=π(7)=1

At q=1, see T[11]='c'... not P[2]

π(q) = π(1) = 0. At 0, stop follow π

At q=0, see T[12]='a'=P[q+1]=P[1]

At q=1, but no more T, so done

Knuth-Morris-Pratt

```
Compute-Prefix-Function(P)
k = 0, π[1] = 0
for q = 2 to |P|
  while k > 0 and P[k+1] ≠ P[q]
    k = π[k]
  if P[k+1] == P[q]
    k = k+1
  π[q]=k            // Runtime = ???
```

Knuth-Morris-Pratt

```
Compute-Prefix-Function(P)
k = 0, π[1] = 0
for q = 2 to |P|
  while k > 0 and P[k+1] ≠ P[q]
    k = π[k]
  if P[k+1] == P[q]
    k = k+1
  π[q]=k              // Runtime = O(|P|)
```

Knuth-Morris-Pratt

```
KMP-Matcher(T,P,π) // runtime?
q = 0
for i = 1 to |T|
   while q > 0 and P[q+1] ≠ T[ i ]
      q = π[q]
   if P[q+1] == T[ i ], then q = q+1
   if q == |P|
      match found, and set q = π[q]
```

Knuth-Morris-Pratt

The while loop decreases q, so it can only run as many times as q increases

q increases only if match in T, so at most |T| times

O(|T| + |T|) = O(|T|)
(why not |T|*|T|?)

Knuth-Morris-Pratt

You try it!

P={a, b, a, a}
S={a, a, b, a, c, a, a, b, a, a, b, a, a, a}

What are π's?
Where are matches?

KMP correctness

Let $\pi^*[q] = \{\pi[q], \pi[\pi[q]], \ldots 0\}$

Lemma 32.5: $\pi^*[q] = \{k : k < q$ and $P_k] P_q \}$

Remember:

$\pi(q) = \max(k : k < q$ and $P_k] P_q)$,

so fairly obvious (see next slide)

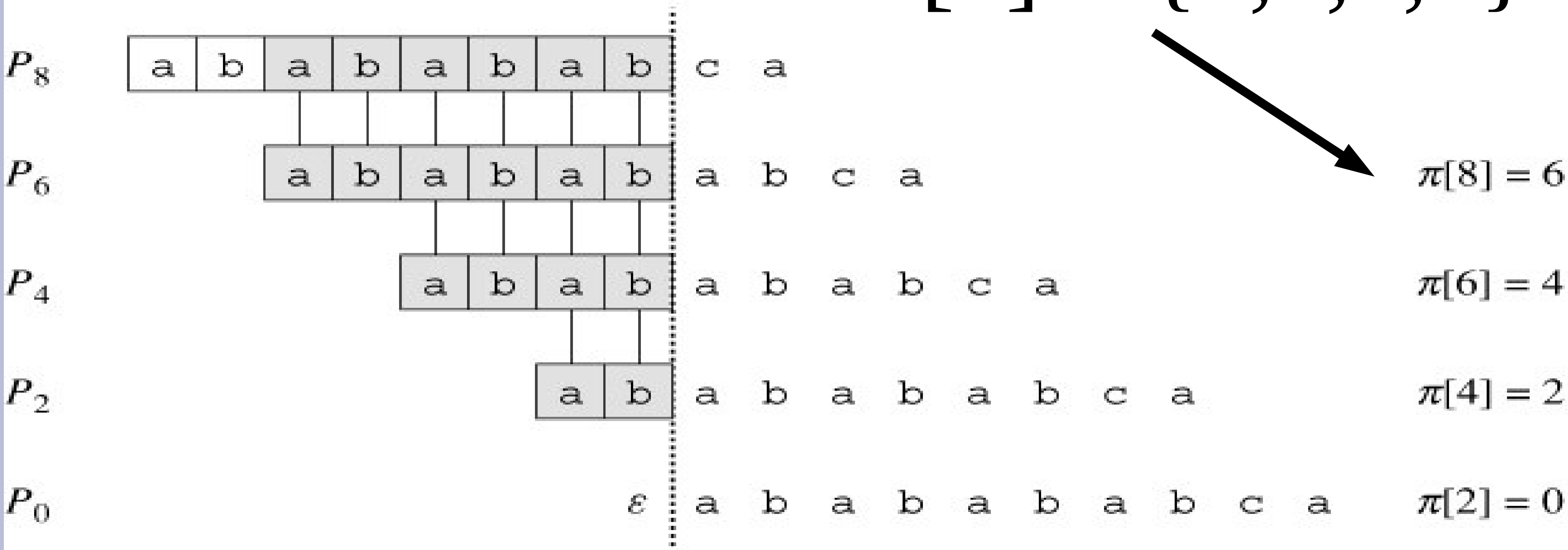
(Tip: prove 2 sets equal by showing A subset B and B subset A)

KMP correctness

i	1	2	3	4	5	6	7	8	9	10
$P[i]$	a	b	a	b	a	b	a	b	c	a
$\pi[i]$	0	0	1	2	3	4	5	6	0	1

(a)

$\pi^*[8] = \{6,4,2,0\}$



(b)

KMP correctness

Lemma 32.6: if π[q] > 0, then π[q]-1 in π*[q-1]

Proof: π[q] < q and $P_{\pi[q]}$] P_q, so π[q] – 1 < q – 1 and $P_{\pi[q]-1}$] P_{q-1} (we know π[q] > 0, so we can drop a char)

Previous lemma says: π*[q] = {k : k < q and P_k] P_q }, above let q=q-1, k=π[q]-1, then done

KMP correctness

Let E_{q-1}={k in π*[q-1] : P[k+1]=P[q]}

Corollary 32.7: π[q] = {0 or 1+max{k in E_{q-1}} if E_{q-1} not empty}

Proof:

Case 1: E_{q-1} empty, no match, so 0

Case 2: By def of E_{q-1}, k+1 < q and P_{k+1}]P_q implies π[q]≥1+max{k in E_{q-1}}

KMP correctness

(E_{q-1}={k in π*[q-1] : P[k+1]=P[q]})

Case 2 (cont): π[q]≥1+max{k in E_{q-1}}

Let r = π[q] – 1, then P_{r+1}] P_q so

P[r+1] = P[q]. Lemma 32.6 says

r in π*[q-1], so r in E_{q-1}.

Thus π[q]≤1+max{k in E_{q-1}}

Thus π[q]=1+max{k in E_{q-1}}

KMP correctness

$k=\pi[q-1]$ at the start of the for loop in Compute-Prefix-Function alg

The while loop finds max{k in E_{q-1}} and adds one for Corollary 32.7

If there k=0, then either the max was 0 and it will be incremented to 1 or no match and will stay 0

KMP correctness

KMP alg correctness (map to FA alg):

Base: both start with q=0

Step ($q'=\sigma(T_{i-1})$):

Case $\sigma(T_i)=0$: q=0 and same

Case $\sigma(T_i)=q'+1$: while does not run, then increases q, so $q=q'+1=\sigma(T_i)$

(continued)

KMP correctness

Step: $q'=\sigma(T_{i-1})$, Case $0<\sigma(T_i)\leq q'$:

while loop terminates when

$P[q+1]=T[i]$, so $q+1 = \sigma(P_{q'}T[i])$

$=\sigma(T_{i-1}T[i])$

$=\sigma(T_i)$, then q is incremented so...

$q=\sigma(T_i)$