Konstruktion von LCP-Arrays (ctd.) und Suche in Suffix Arrays

WS14/15

Johannes Fischer

Algorithm 1: Linear-Time Construction of the LCP-Array

for $i = 1, \ldots, n$ **do** $A^{-1}[A[i]] \leftarrow i$;
$h \leftarrow 0, H[1] \leftarrow 0$;
for $i = 1, \ldots, n$ **do**
    if $A^{-1}[i] \neq 1$ **then**
        $j \leftarrow A[A^{-1}[i] - 1]$;
        while $t_{i+h} = t_{j+h}$ **do** $h \leftarrow h + 1$;
        $H[A^{-1}[i]] \leftarrow h$;
        $h \leftarrow \max\{0, h - 1\}$;
    end
end

Algorithm 2: More Cache-Efficient Linear-Time Construction of the LCP-Array

$\Phi[n] \leftarrow A[n]$; `// assume that` T `is` $\$$`-terminated, so` $A[1] = n$

for $i = 2, \ldots, n$ **do** $\Phi[A[i]] \leftarrow A[i-1]$; `// "with whom I want to be compared"`

$h \leftarrow 0$;

for $i = 1, \ldots, n$ **do**

  $j \leftarrow \Phi[i]$;

  while $t_{i+h} = t_{j+h}$ **do** $h \leftarrow h + 1$;

  $H'[i] \leftarrow h$; `//` $\Phi[i]$ `can be overwritten by` H' `(saves space)`

  $h \leftarrow \max\{0, h-1\}$;

end

for $i = 1, \ldots, n$ **do** $H[i] \leftarrow H'[A[i]]$; `// put values back into suffix array order`

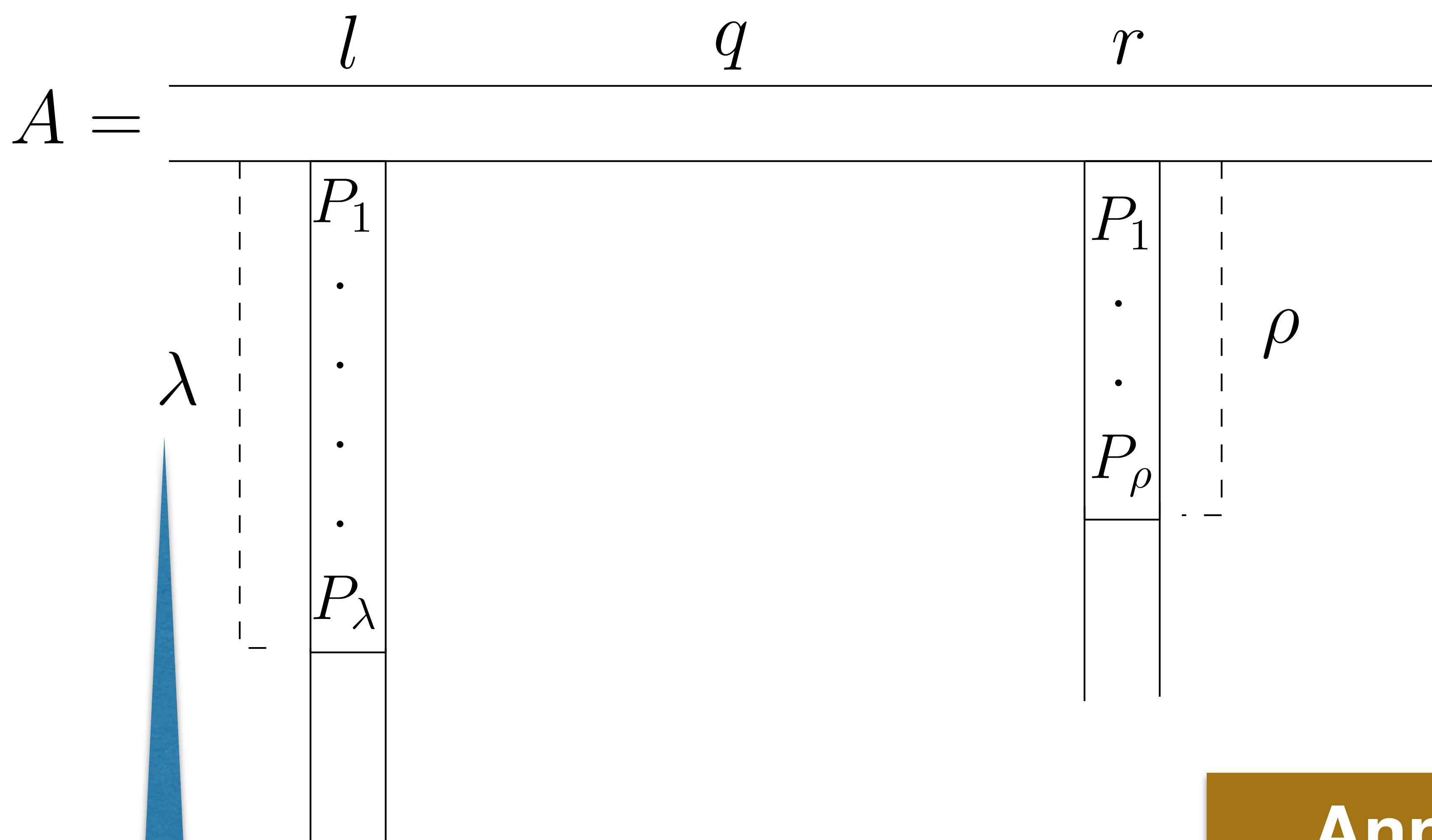
Suche in Suffix Arrays
(direkt - ohne Suffixbaum)

Algorithm 3: function `SAsearch`$(P_{1\ldots m})$

```
l ← 1; r ← n + 1;
while l < r do
    q ← ⌊(l+r)/2⌋;
    if P >_lex T_{A[q]... min{A[q]+m−1,n}} then
        l ← q + 1;
    else
        r ← q;
    end
end
s ← l; l−−; r ← n;
while l < r do
    q ← ⌈(l+r)/2⌉;
    if P =_lex T_{A[q]... min{A[q]+m−1,n}} then
        l ← q;
    else
        r ← q − 1;
    end
end
return [s, r];
```

Schnellere Suche:

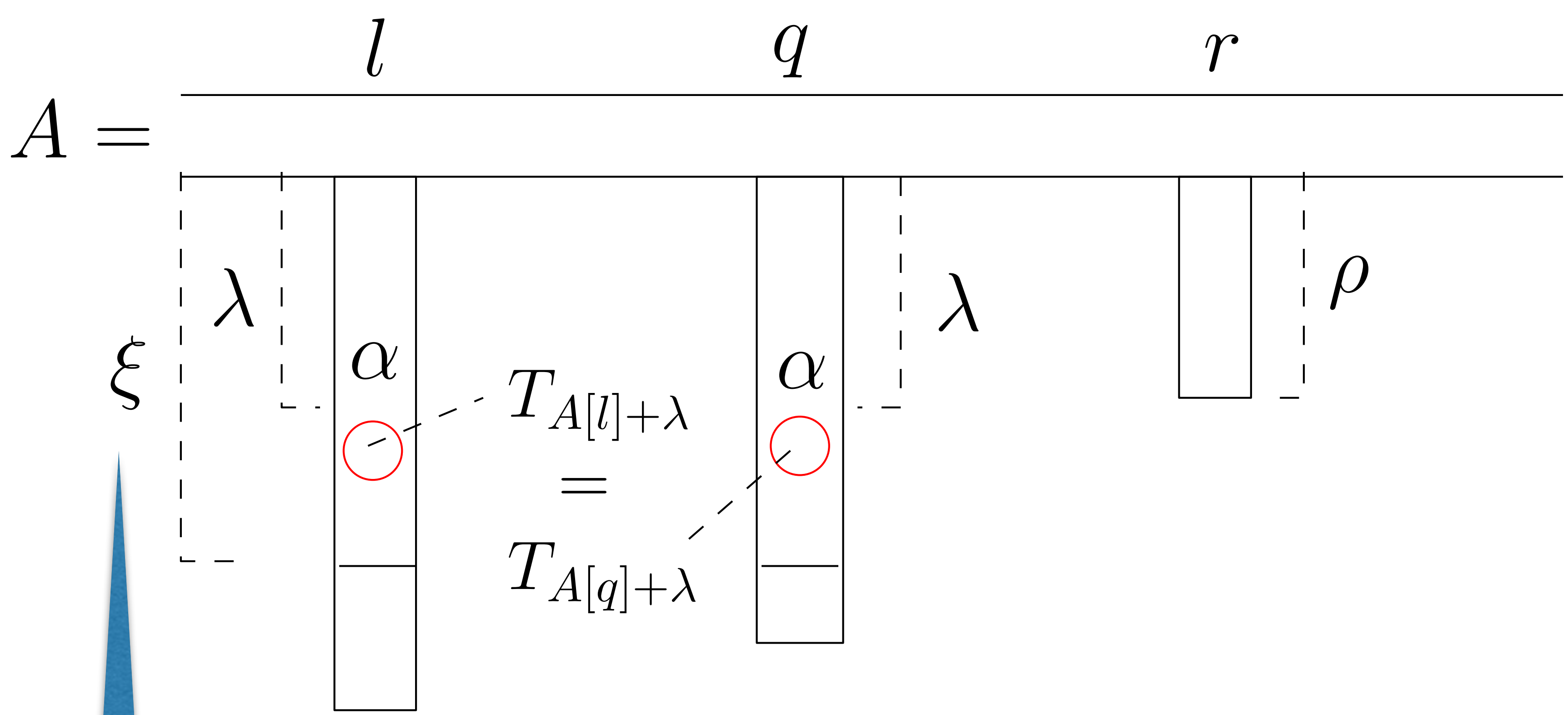
$$m \cdot \lg(n) \rightarrow m \mathbf{+} \lg(n)$$



Übereinstimmung zwischen Muster P und dem Suffix T[A[l]..n]

Annahme:
λ>ρ
(sonst vertausche)

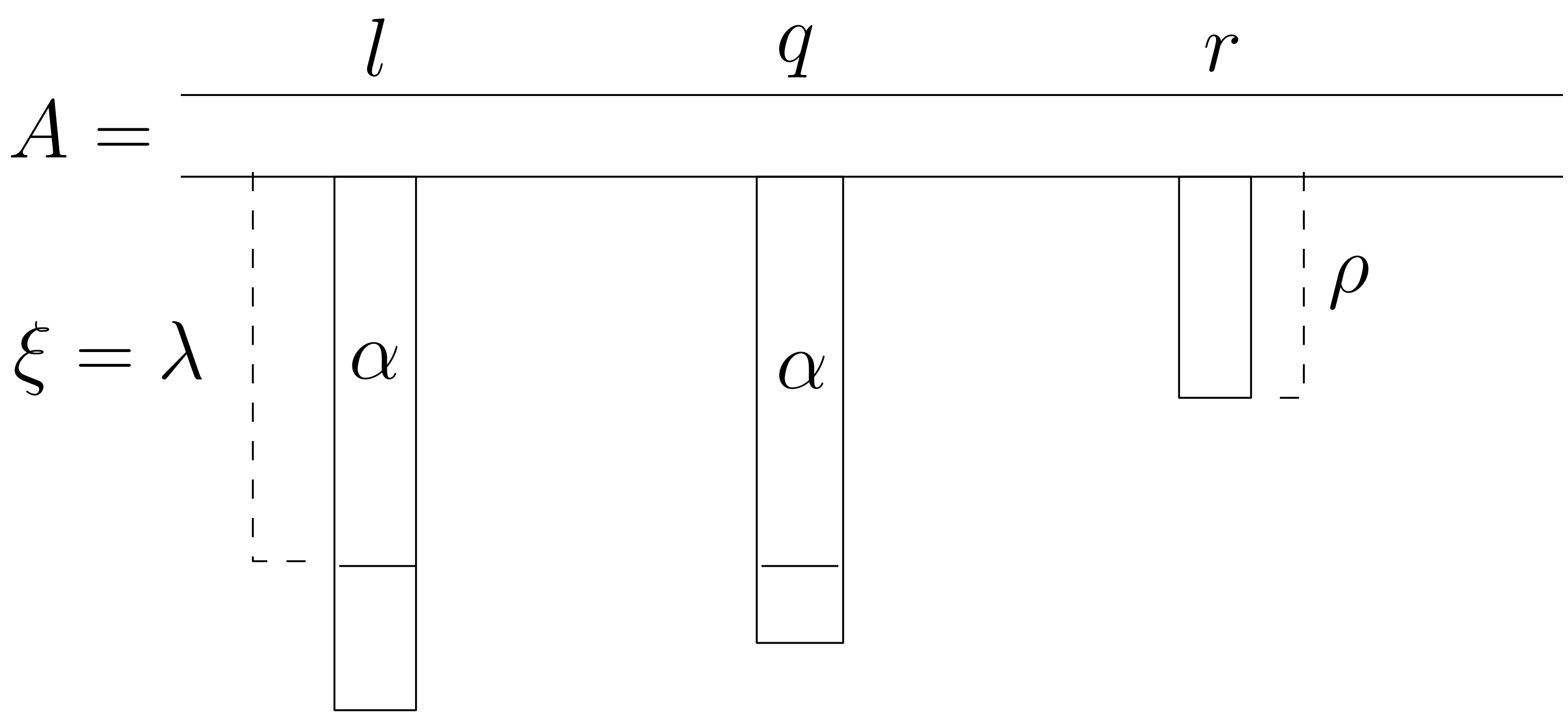
1. Fall: ξ>λ



LCP zwischen T[A[l]..n] und T[A[q]..n]

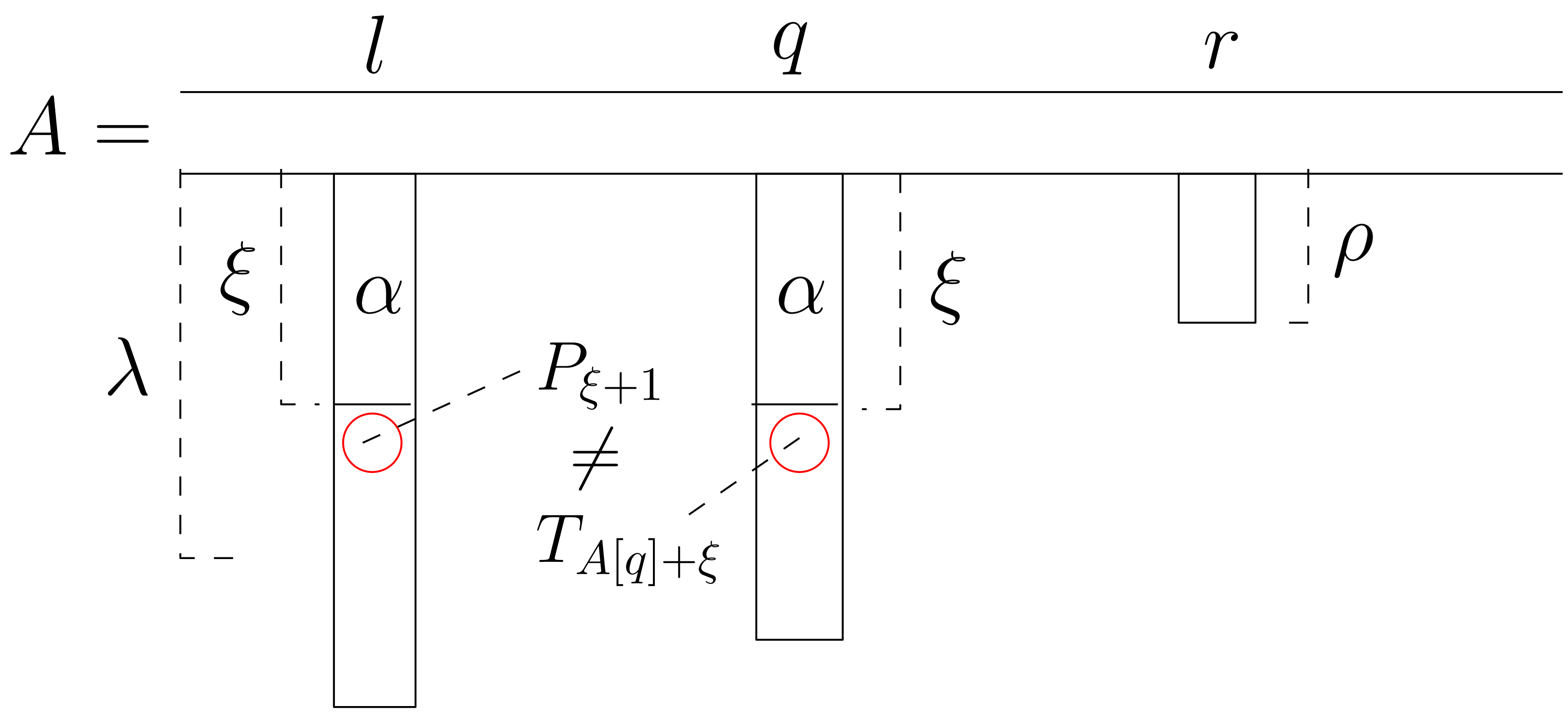
⇒ setze l ←q ohne weitere Vergleiche!

2. Fall: ξ=λ



Vergleiche wie bei der normalen binären Suche

3. Fall: ξ<λ



⇒ setze r ←q und ρ←ξ ohne weitere Vergleiche!