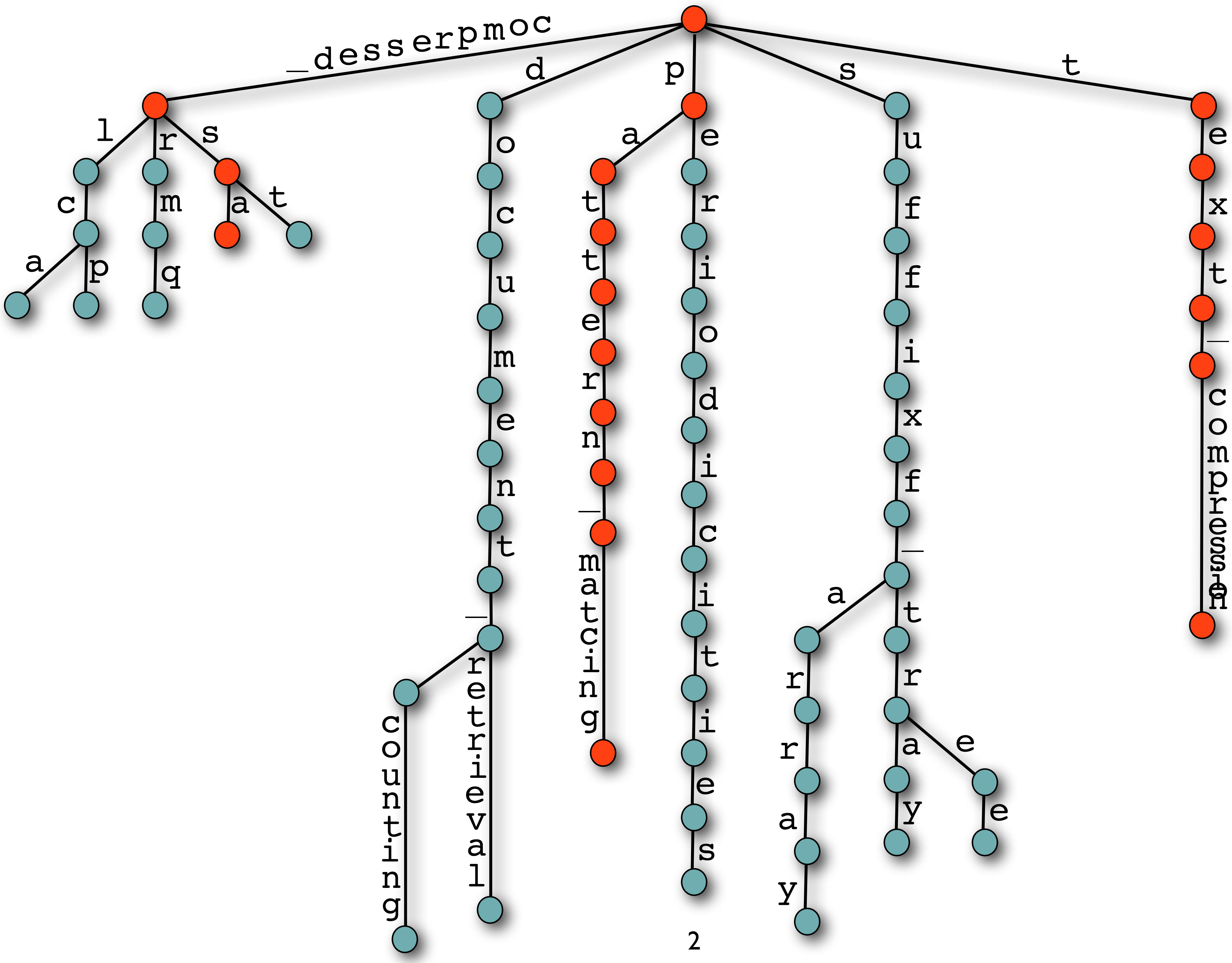
Lecture 8:
Backwards Search and FM Indices

Johannes Fischer

Topics



_desserpmoc
d
p
s
t
counting
retrieval
matching
compression

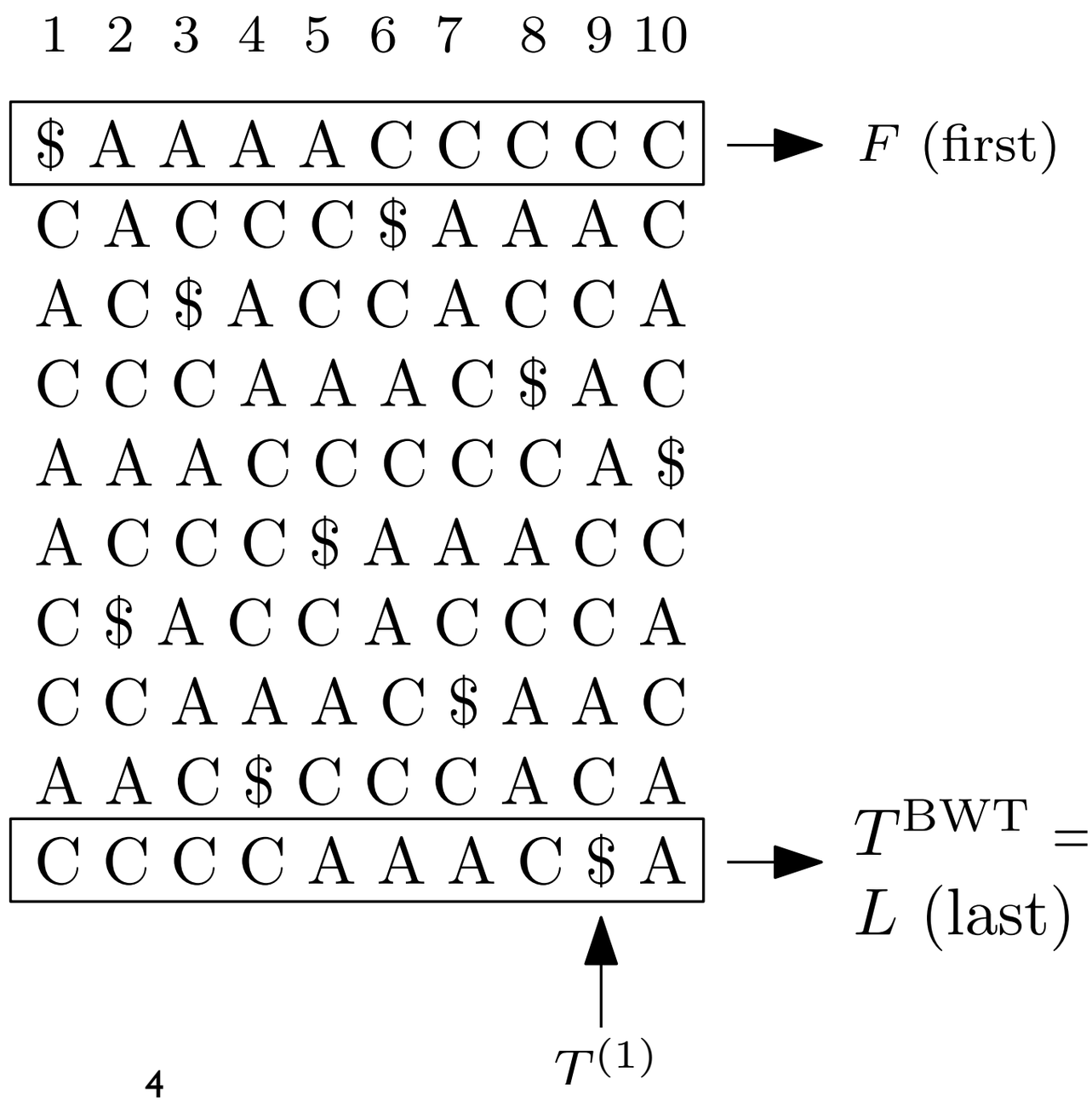
Burrows Wheeler Transform

T=CACAACCAC$

build cyclic rotations

Burrows Wheeler Transform

sort columns (=strings) lexicographically



Burrows Wheeler Transform

$T = \mathrm{CACAACCAC\$}$

	1	2	3	4	5	6	7	8	9	10
A=	10	4	8	2	5	9	3	7	1	6

$	**A**	**A**	**A**	**A**	**C**	**C**	**C**	**C**	**C**	→ F (first)
C	A	C	C	C	$	A	A	A	C	
A	C	$	A	C	C	A	C	C	A	
C	C	C	A	A	A	C	$	A	C	
A	A	A	C	C	C	C	C	A	$	
A	C	C	C	$	A	A	A	C	C	
C	$	A	C	C	A	C	C	C	A	
C	C	A	A	A	C	$	A	A	C	
A	A	C	$	C	C	C	A	C	A	
C	C	C	C	A	A	A	C	$	A	→ $T^{\mathrm{BWT}} = L$ (last)

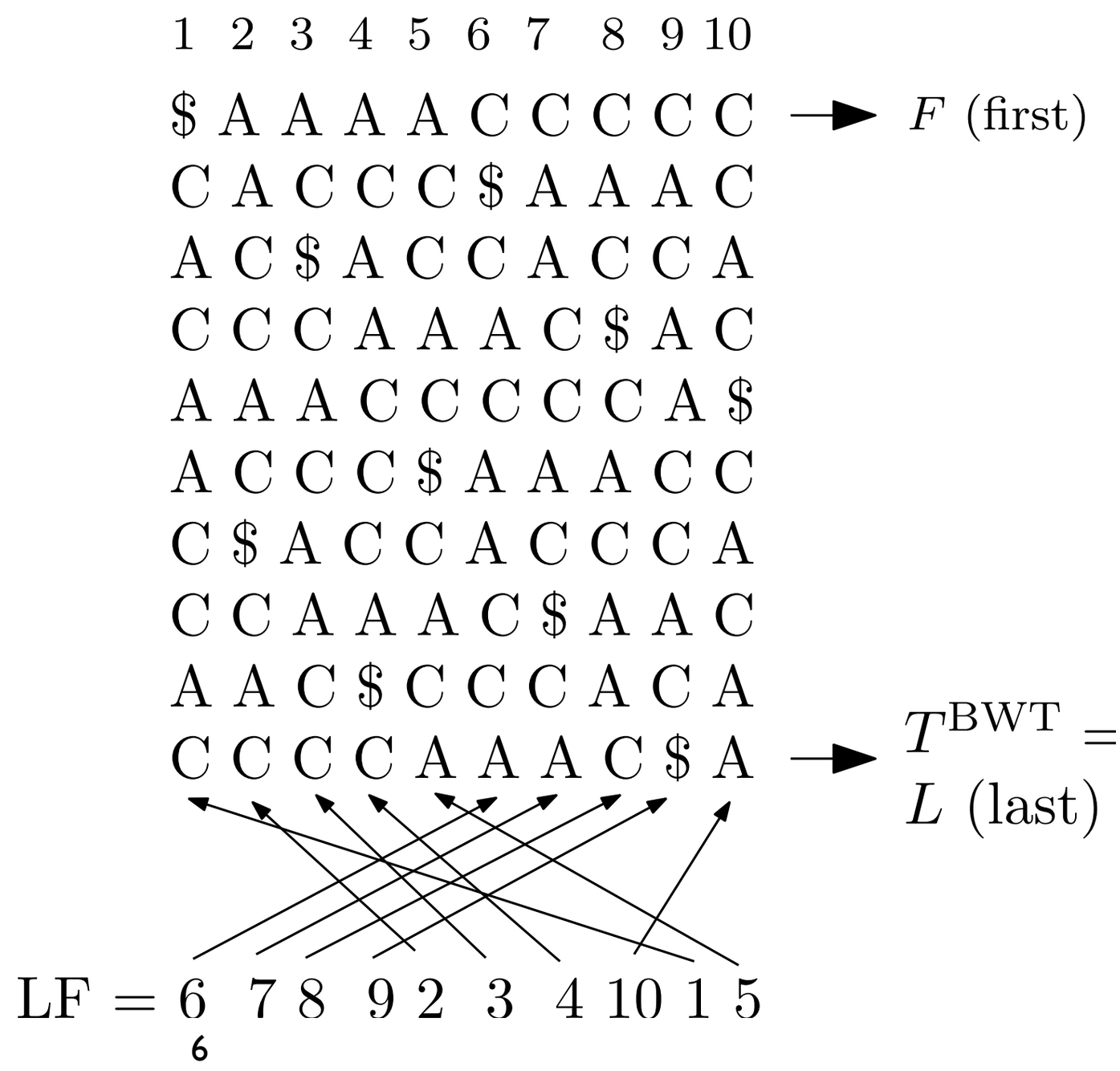
5

$L[i] = T[A[i]-1]$

Last to Front Mapping

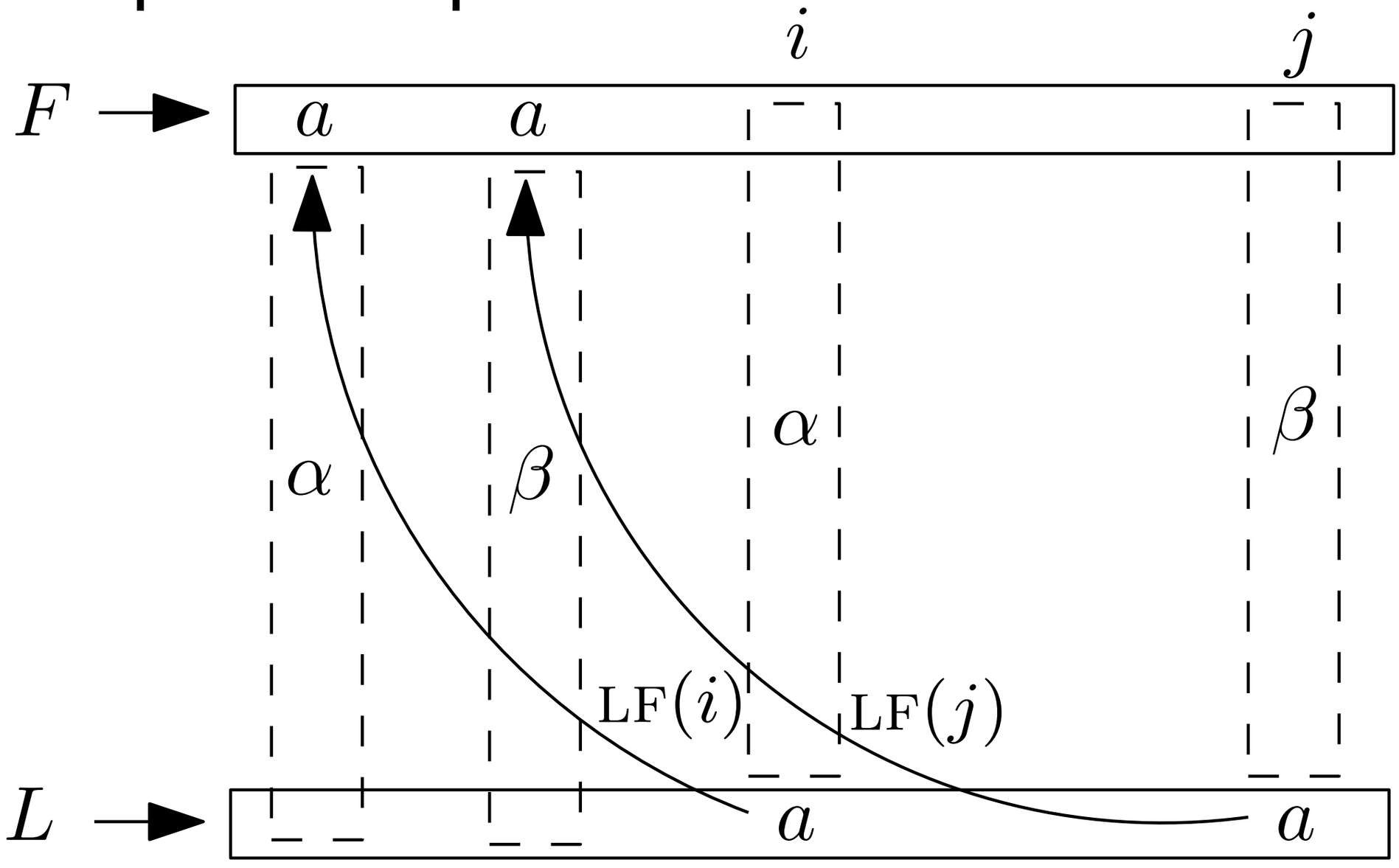
$T = \mathrm{CACAACCAC\$}$

LF[*i*]=*j* ⇔ *A*[*j*]=*A*[*i*]-1



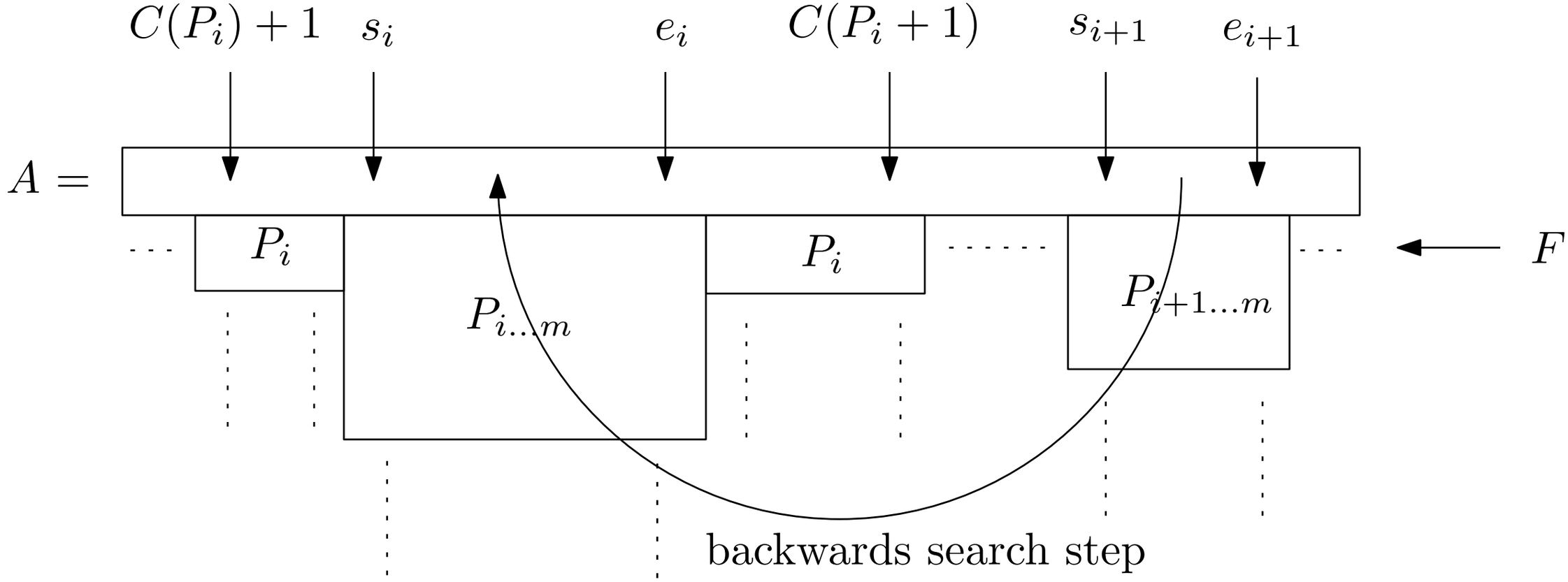
Last to Front Mapping

- equal chars preserve order in *F* and *L*

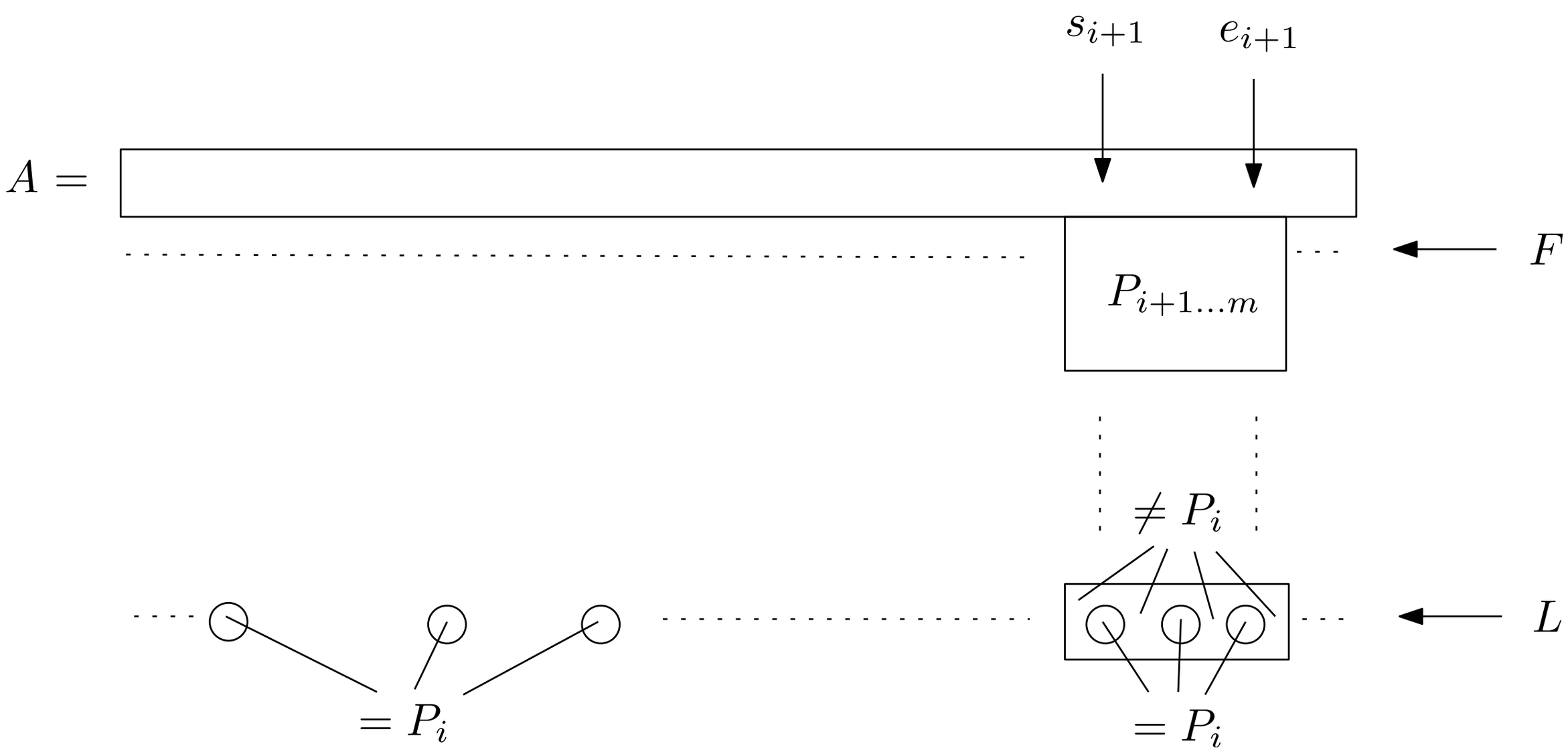


Backwards Search

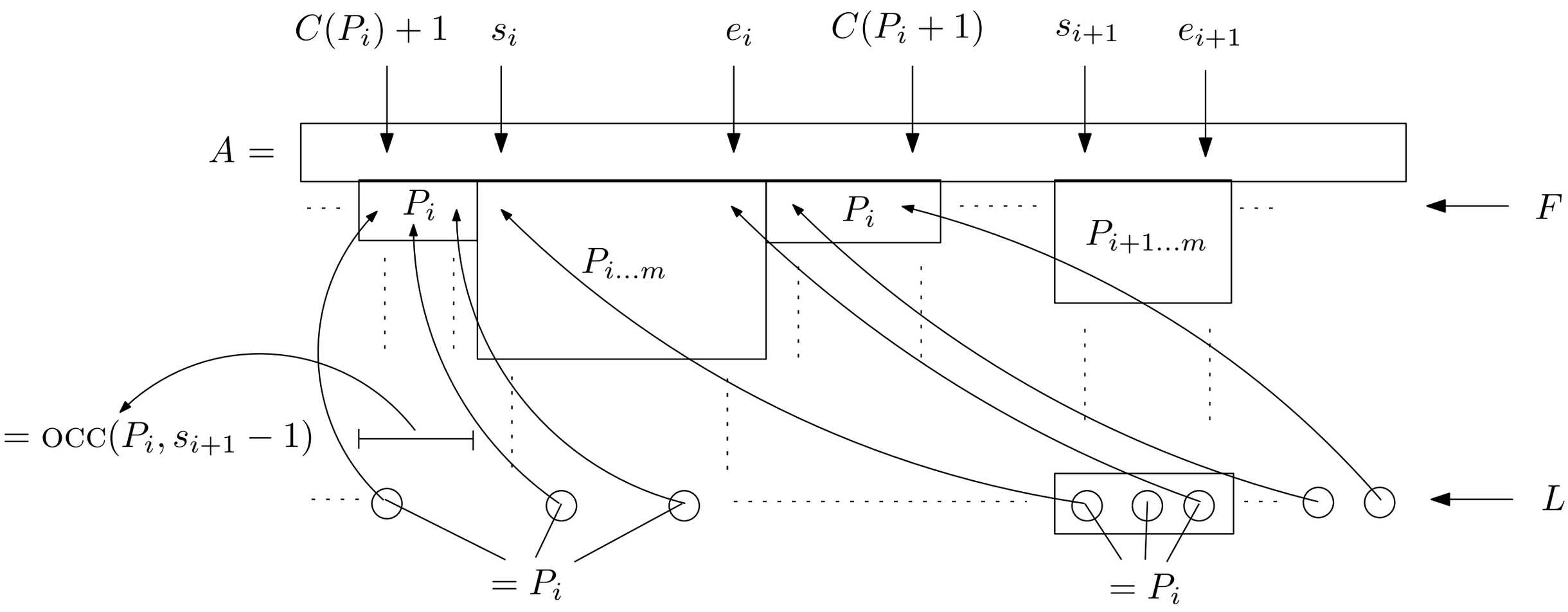
- $C[a]$:= # chars smaller than a in T f. $a \in \sum$
- $\mathrm{OCC}[a,i]$:= # a's in $L[1,i]$ for $a \in \sum$
- search for **interval** of $P[1,m]$ in A



Backwards Search



Backwards Search



Backwards Search

Algorithm 2: function `backwards-search`$(P_{1\ldots m})$

$s \leftarrow 1$; $e \leftarrow n$;
for $i = m \ldots 1$ **do**
  $s \leftarrow C(P_i) + \text{OCC}(P_i, s-1) + 1$;
  $e \leftarrow C(P_i) + \text{OCC}(P_i, e)$;
  if $s > e$ **then**
    return "no match";
  end
end
return $[s, e]$;

Backwards Search

- Note: **no use** of suffix array A
- Space:
  - C : $|\Sigma|$ lg n bits (small)
  - BWT L: n lg $|\Sigma|$ (same as text)
  - Occ: ??? (Q1)
- How to output text **positions**? (Q2)

Implementing Occ (Q1)

- $\text{rank}_1(B, i)$ = #1's in $B[1,i]$
- in ADS: $n+o(n)$ bits for rank in $O(1)$ time
- idea 1: bitmap B_a for every character $a \in \sum$

  $\Longrightarrow n\ |\Sigma|$ bits, $O(m)$ search time!
- idea 2: **wavelet trees**:

  $\Longrightarrow n \lg |\Sigma|$ bits, $O(m \lg |\Sigma|)$ search time

Finding Positions (Q2)

- $A[i] = j \Leftrightarrow A[\mathrm{LF}(i)] = j-1$
- sample every s'th position in T

  $\Longrightarrow O(n/s \lg n)$ extra bits, $O(s \cdot t_{\mathrm{OCC}})$ time
- say $s = \lg_{|\Sigma|} n \Rightarrow n \lg |\Sigma|$ bits, $O(\lg n)$ time
  - t_{OCC}: time for evaluating OCC (e.g. $O(\lg |\Sigma|)$)
- marking sampled positions: $n+o(n)$ bits

Summary

- **FM index** with
  - O(n lg |Σ|) space (using wavelet trees)
  - O(*m* lg |Σ|) search for # occurrences
  - O(*k* lg *n*) for outputting *k* occurrences (using sampled suffix array)
- can be improved, see e.g.
  - G. Navarro, V. Mäkinen: *Compressed Text Indices*. ACM Comput. Surv. **39**(1), 2007.