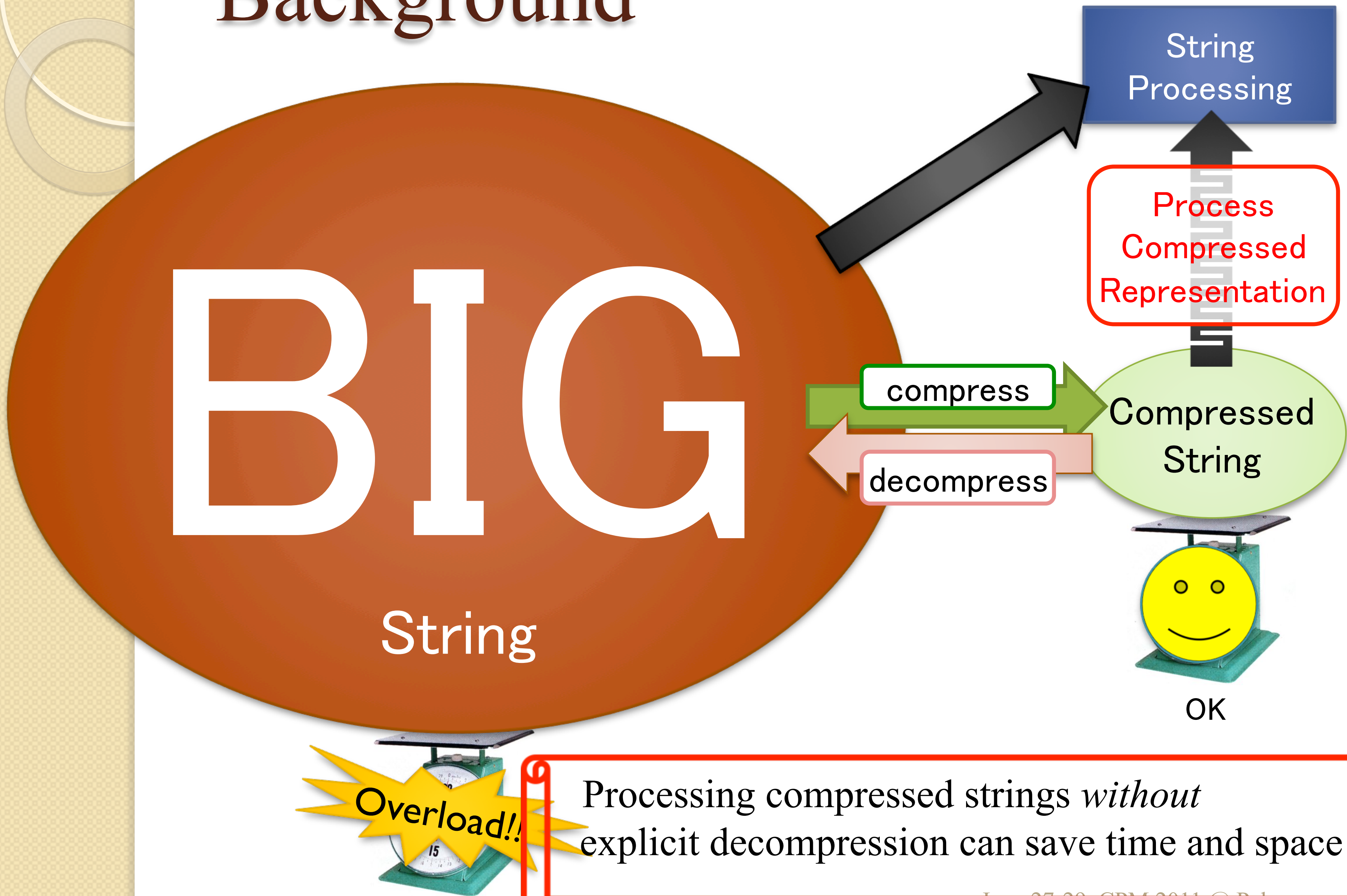
Faster Subsequence and Don’t-Care Pattern Matching on Compressed Texts

Takanori Yamamoto, Hideo Bannai,
Shunsuke Inenaga, Masayuki Takeda
Department of Informatics,
Kyushu University, JAPAN

Outline

- Background
- Preliminaries
- Algorithms
  - **Minimum Subsequence Occurrences on SLP**
  - Fixed Length Don’t Care Matching on SLP
  - Variable Length Don’t Care Matching on SLP
- Summary

Background



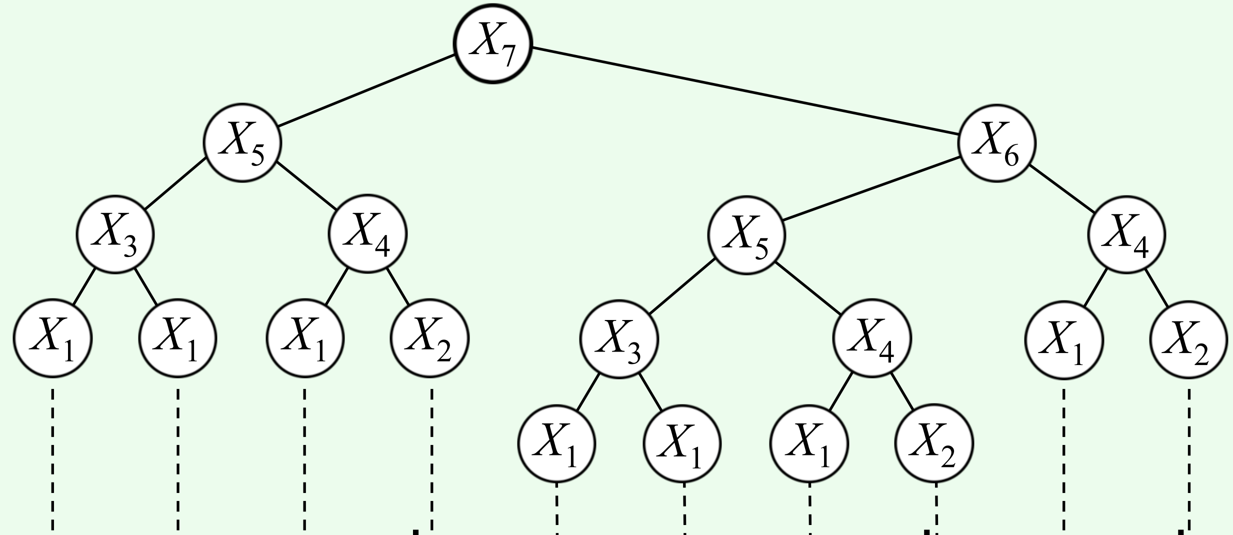
Processing compressed strings *without* explicit decompression can save time and space

Straight Line Program (SLP)

- Grammar in Chomsky Normal Form deriving single string
- Can model outputs of various compression algorithms

SLP $\mathcal{T}$, $n = 7$

$$X_1 = \mathrm{a} \qquad X_3 = X_1 X_1 \qquad X_5 = X_3 X_4 \qquad X_7 = X_5 X_6$$
$$X_2 = \mathrm{b} \qquad X_4 = X_1 X_2 \qquad X_6 = X_5 X_4$$



T = a a a b a a a b a b

$N = 10$

Subsequences

- String P is a ***subsequence*** of string T
  $\Leftrightarrow$ $\exists\, i_0, \ldots, i_{m-1}$ s.t.
  $$0 \le i_0 < \bullet\bullet\bullet < i_{m-1} \le |T|$$
  $$P[j] = T[i_j] \text{ for all } j = 0, \ldots, m-1$$
  (i_0, i_{m-1}) is called an ***occurrence*** of subsequence P in T

0 1 2 3 4 5 6 7 8 9 0 1 2

$T =$ `sicilypalermo`

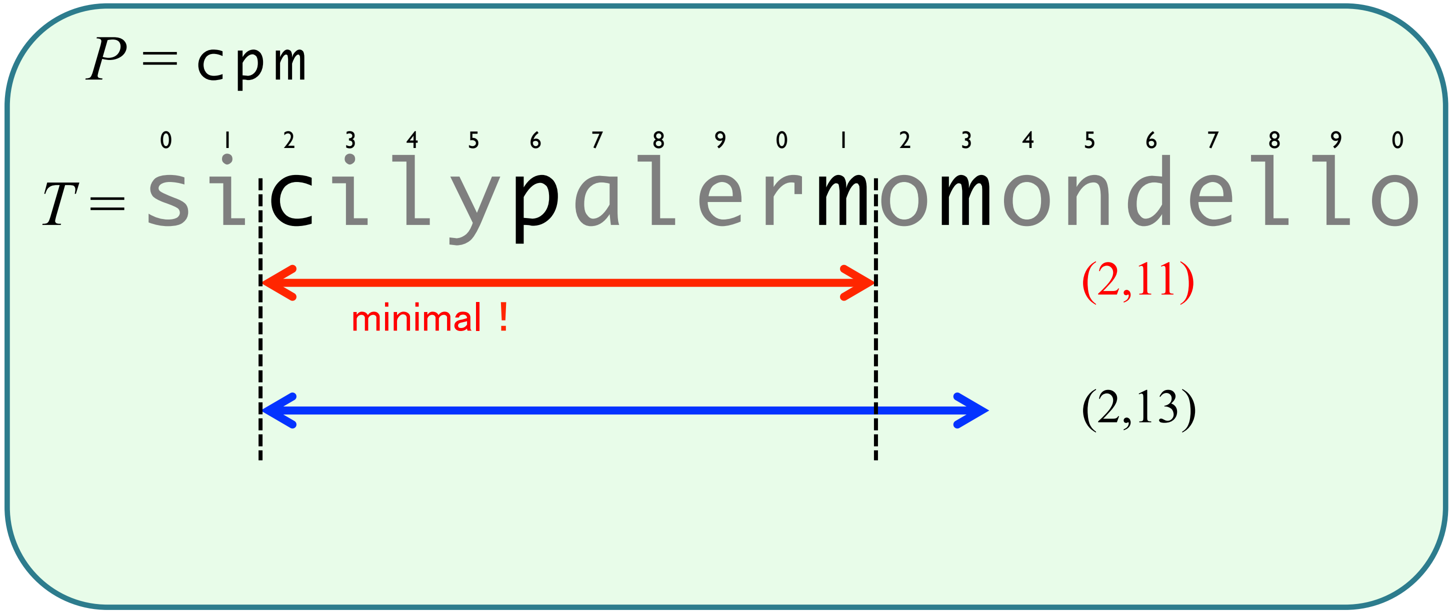
$P =$ `cpm`

$(i_0, i_1, i_2) = (2,6,11)$

$(i_0, i_2) = (2,11)$

Minimal Subsequence Occurrences

- An occurrence (i_0, i_{m-1}) of subsequence P in T is <u>*minimal*</u>, if there is *no* occurrence of P in $T[i_0 : i_{m-1}-1]$ or $T[i_0+1 : i_{m-1}]$



Window Subsequence Problems on SLP [Cégielski *et al*. 2006]

Minimal Subsequence Occurrences

Input : SLP of size n representing string T, string P
Output : # of minimal occurrences of subsequence P in T

Several variations, e.g.:

Bounded Minimal Subsequence Occurrences

Input : SLP of size n representing T, string P, integer w
Output : # of minimal occurrences (i_0, i_{m-1}) of subsequence P in T, where $i_{m-1} - i_0 < w$

Window Subsequence Problems on SLP

Decomp.&[Troníček 2001] ➔ $\boxed{O(Nm)}$

[Cégielski *et al*. 2006] ➔ $\boxed{O(nm^2 \log m)}$

[Tiskin 2009] ➔ $\boxed{O(nm^{1.5})}$

[Tiskin 2011] ➔ $\boxed{O(nm \log m)}$

This Work

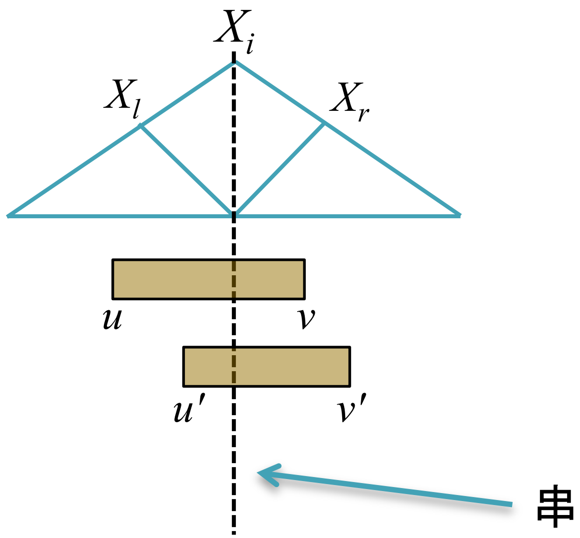
Window subsequence problems in $\boxed{O(nm)}$

Extensions to pattern matching with Fixed/Variable Length Don't Care Symbols in $\boxed{O(nm)}$

串：Crossing Occurrences

For $X_i = X_l X_r$, an occurrence (u, v) of P is a *crossing occurrence* in X_i when:

$$0 \le u < |X_l| \le v < |X_i|$$



串 is the Chinese character meaning “skewer”,
pronounced “KUSHI” in Japanese (Similar to the Greek letter ξ)

Counting Minimal Subsequence Occurrences on SLP

- M_i : # of minimal occurrences of P in X_i
- $M^{串}(l, r)$: # of *crossing* minimal occurrences of P in $X_i = X_l X_r$

M_n is the solution to our Problem

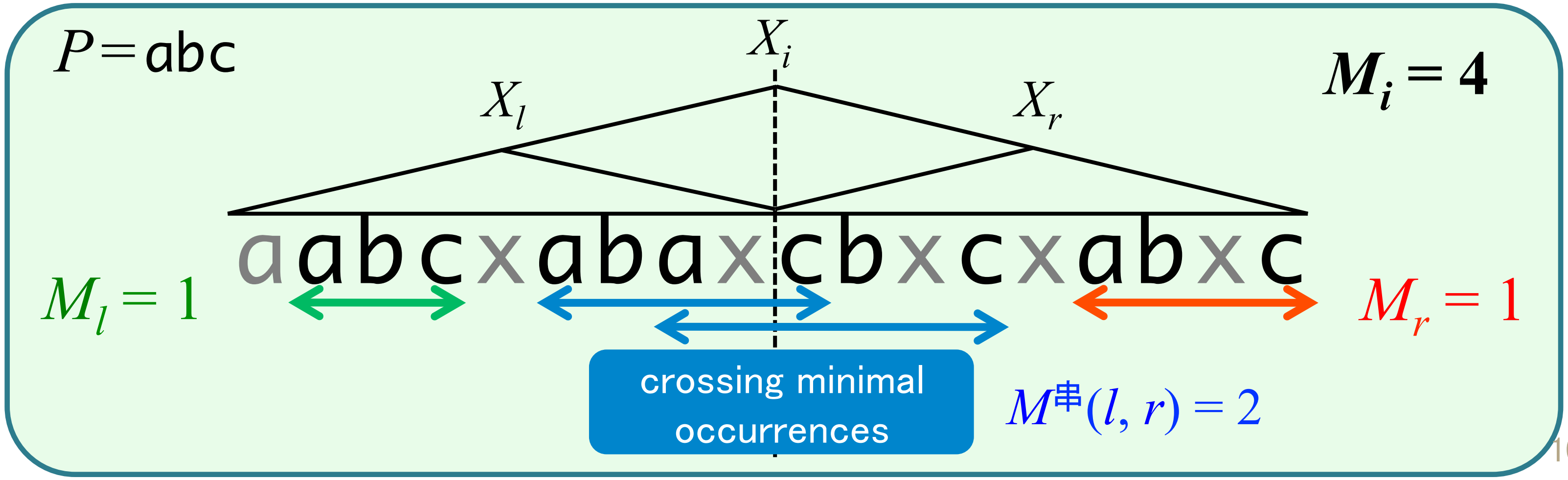
Computing M_i

- If $X_i = \mathrm{a}$ $(\mathrm{a} \in \Sigma)$

$$M_i = \begin{cases} 0 & \text{if } m \neq 1 \text{ or } P[j] \neq \mathrm{a} \\ 1 & \text{if } m = 1 \text{ and } P[j] = \mathrm{a} \end{cases}$$

- If $X_i = X_l X_r$ $(l, r < i)$

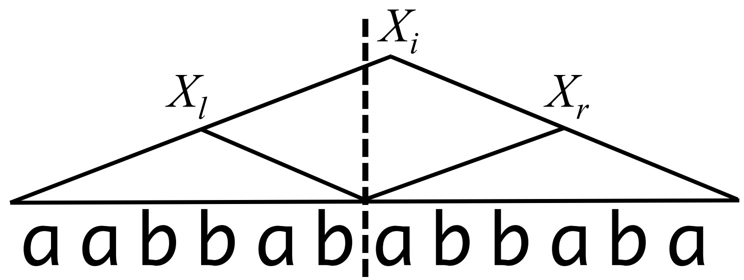
$$M_i = M_l + M_r + M^{串}(l, r)$$



Computing $M^{串}(l, r)$

there are at most $m-1$ crossing minimal occurrences

$P=$ abbba



k		shortest suffix of X_l containing $P[0:m\text{-}k\text{-}1]$	shortest prefix of X_r containing $P[m\text{-}k:m\text{-}1]$	
0	∞			-
1	5	a b b b	a	1
2	5	a b b	b a	4
3	2	a b	b b a	4
4	2	a	b b b a	6
5	-		a b b b a	6

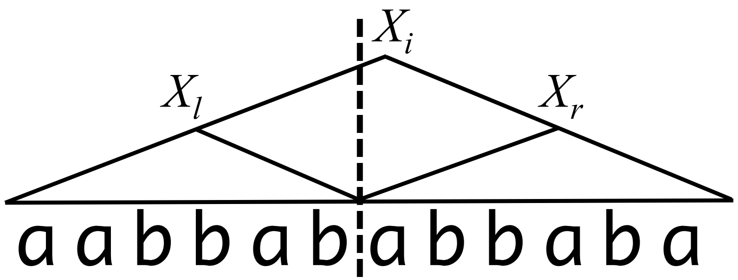
shortest suffix of X_l containing $P[0:m\text{-}k\text{-}1]$

shortest prefix of X_r containing $P[m\text{-}k:m\text{-}1]$

Computing $M^{串}(l, r)$

there are at most $m-1$ crossing minimal occurrences

$P =$ abbba



k				
0	∞			–
1	5	a b b b	a	1
2	5	a b b	b a	4
3	2	a b	b b a	4
4	2	a	b b b a	6
5	-		a b b b a	6

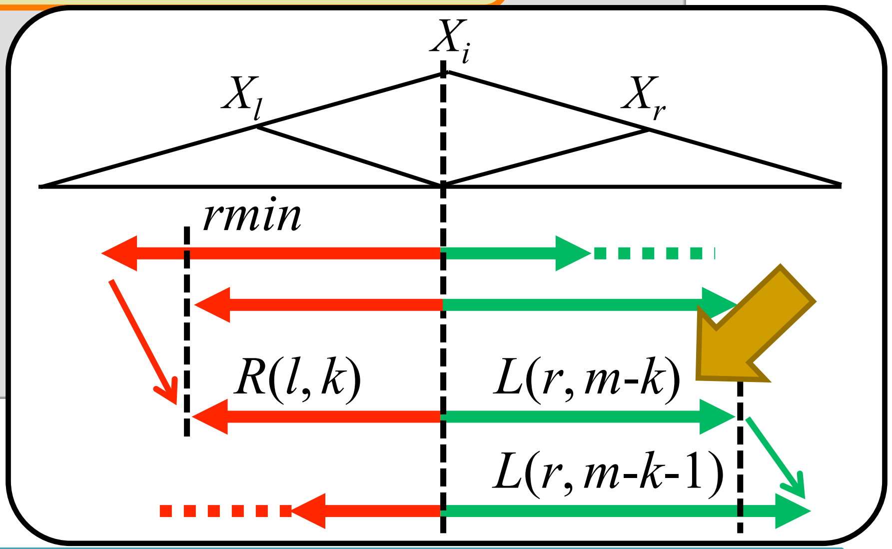
shortest suffix of X_l containing $P[0:m\text{-}k\text{-}1]$

shortest prefix of X_r containing $P[m\text{-}k:m\text{-}1]$

Computing $M^{串}(l, r)$

$M^{串}(l,r)$ for all $X_i = X_l X_r$ can be computed in total of $O(nm)$ time using L, R

$C := 0$, $rmin := R(l, 0)$
for $k := 1$ to $m - 1$
  if $rmin > R(l,k)$ and $L(r,m-k) < L(r,m-k-1)$ then
    $C := C + 1$
    $rmin := R(l, k)$
  end if
end for
$M^{串}(l, r) := C$



$L(i,j)$: Length of shortest prefix of X_i s.t. $P[j{:}m-1]$ is subsequence
$R(i,j)$: Length of shortest suffix of X_i s.t. $P[0{:}m-j-1]$ is subsequence

Computing Q (to compute L)

$Q(i,j)$: Length of longest prefix of $P[j:]$ that is a subsequence of X_i $(i = 1,\ldots,n,\ j = 0,\ldots,m)$

Computing $Q(i, j)$

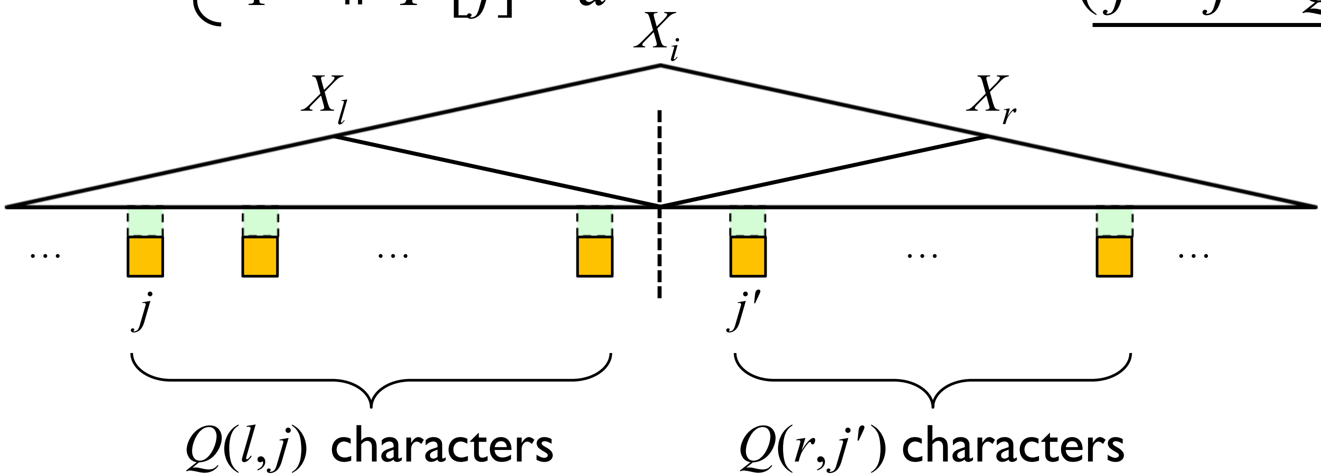
- If $X_i = \mathrm{a}\ \ (\mathrm{a} \in \Sigma)$

$$Q(i,j) = \begin{cases} 0 & \text{if } P[j] \neq \mathrm{a} \\ 1 & \text{if } P[j] = \mathrm{a} \end{cases}$$

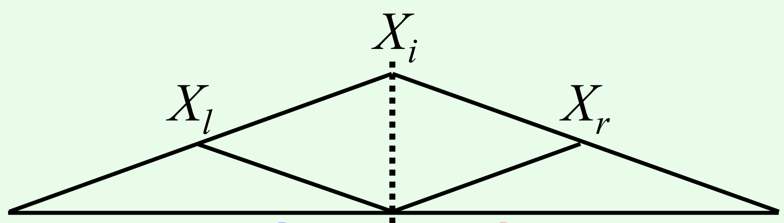
- If $X_i = X_l X_r\ \ (l, r < i)$

$$Q(i,j) = Q(l,j) + Q(r,j')$$

$$\underline{(j' = j + Q(l,j))}$$



Computing Q (to compute L)



x x a x b x c d x e x x

$P[j:]=$ abcdef

$P[j':]=$ cdef

$Q(l,j) = 2$

$Q(r,j') = 3$

$j' := j + Q(l,j)$

$Q(i,j) := Q(l,j) + Q(r,j')$
$= 2 + 3 = 5$

For all $i=1,\ldots,n$, $j=0,\ldots,m$

$Q(i, j)$ can be calculated in $O(nm)$ time using DP.

$Q(n,0)=m \iff P$ is a subsequence of $\mathcal{T}$

Computing L

$L(i, j)$: Length of shortest prefix of X_i s.t. $P[j:]$ is subsequence
$(i = 1,\ldots,n,\ j = 0,\ldots,m)$ (∞ if $P[j:]$ is not subsequence of X_i)

Computing $L(i, j)$

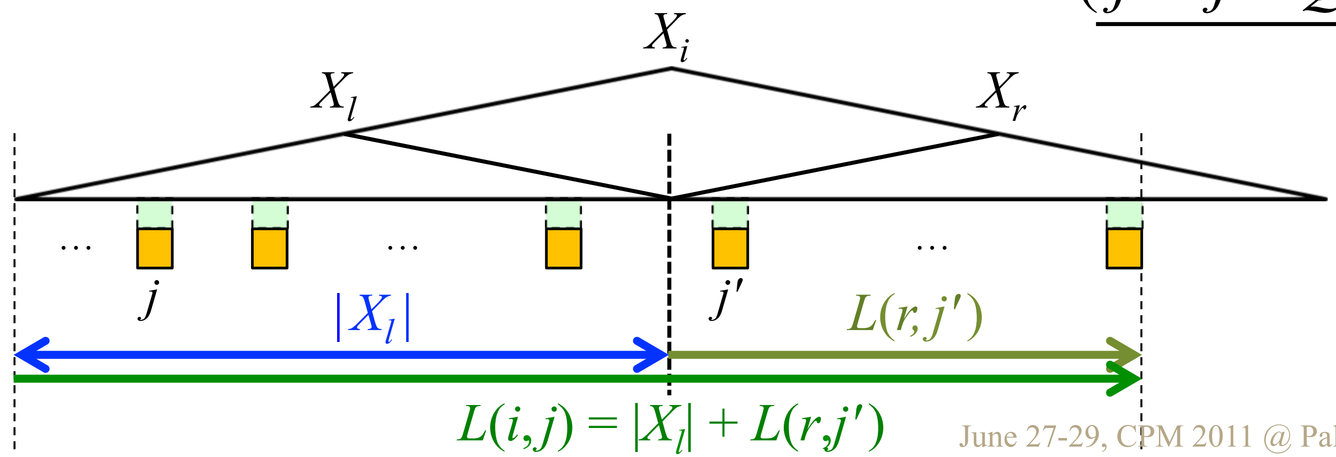
- If $X_i = \mathrm{a}$ ($\mathrm{a} \in \Sigma$)

$$L(i,j) = \begin{cases} 0 & \text{if } j = m \\ 1 & \text{if } P[j:] = \mathrm{a} \\ \infty & \text{if } P[j:] \neq \mathrm{a} \end{cases}$$

- If $X_i = X_l X_r$

$$L(i,j) = \begin{cases} L(l,j) & \text{if } j' = m \\ |X_l| + L(r, \underline{j'}) & \text{if } j' < m \end{cases}$$

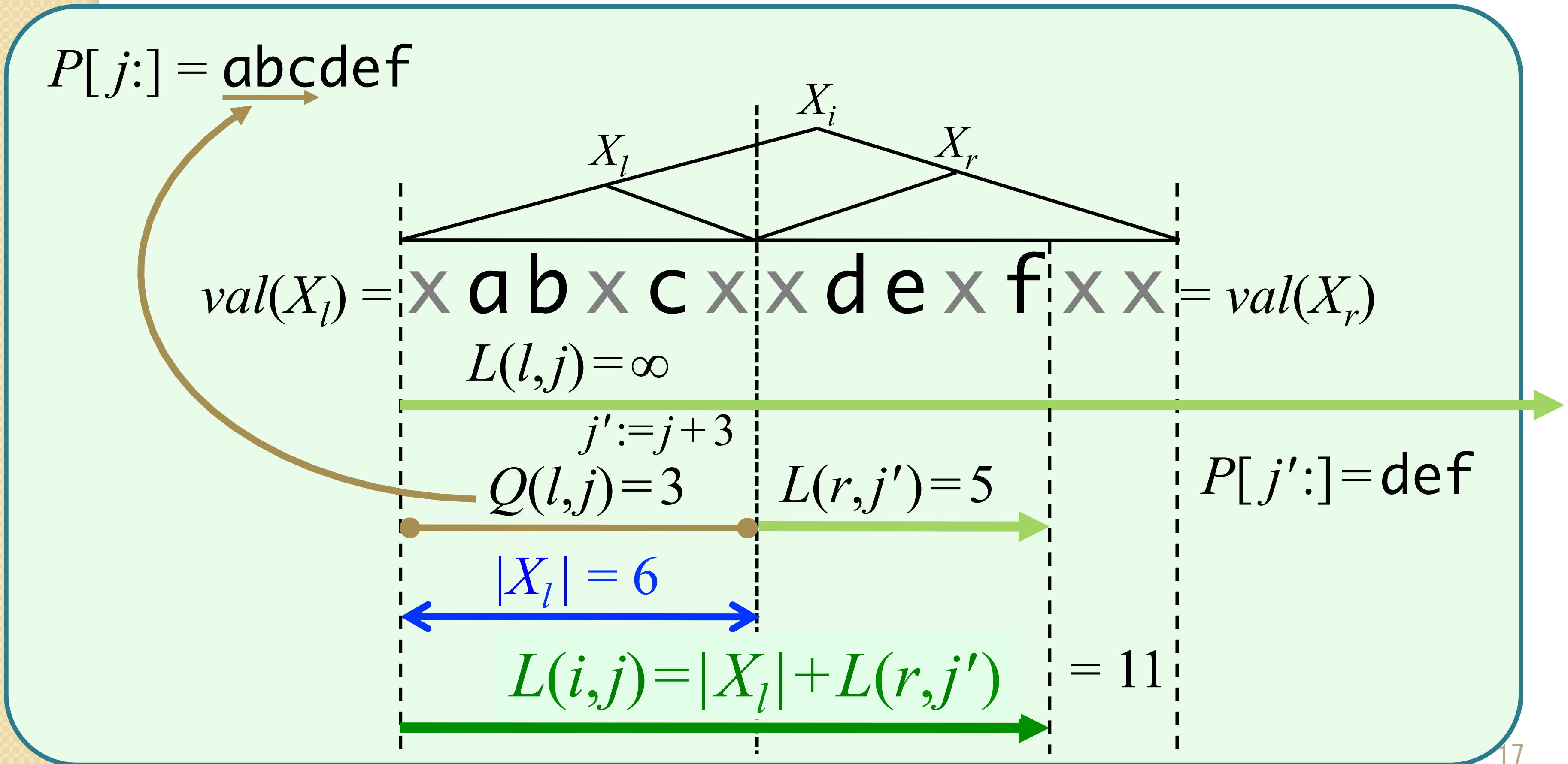
$$\underline{(j' = j + Q(l,j))}$$



[Cégielski *et al.*, 2007]
$O(nm^2 \log m)$

Computing L

$L(i, j)$ can be computed for all $i = 1,\ldots,n$, $j = 0,\ldots,m$, in total of $O(nm)$ time using $Q(i,j)$



Counting Minimal Subsequence Occurrences: Summary

Minimal Subsequence Occurrences

Input : SLP of size n representing string T, string P
Output : # of minimal occurrences of subsequence P in T

Theorem

Given SLP of size n and a subsequence pattern of size m, Minimal Subsequence Occurrences can be solved in $O(nm)$ time and space

$O(Nm)$ Decomp.&[Troníček 2001]

$O(nm^2 \log m)$ [Cégielski et al. 2007]

$O(nm^{1.5})$ [Tiskin 2009]

$O(nm \log m)$ [Tiskin 2011]

$\Rightarrow$ $O(nm)$

MATCHING WITH DON'T CARE SYMBOLS

Fixed Length Don't Care

P = ab○d○a ○: don't care symbol

T = xaabcdabddx

ab○d○b

Observation: Bounded Minimal Subsequence Occurrences Problem with $w = |P|$ ⇔ substring matching

Bounded Minimal Subsequence Occurrences

Input : SLP of size n representing T, string P, integer w

Output : # of minimal occurrences (i_0, i_{m-1}) of subsequence P in T, where $i_{m-1} - i_0 < w$

Fixed Length Don't Care

P = ab○d○a ○: don't care symbol

T = xaabcdabddx

ab○d○b

Observation: Bounded Minimal Subsequence Occurrences Problem with $w = |P|$ ⇔ substring matching

Solution

Extend algorithm to handle don't care symbol '○'

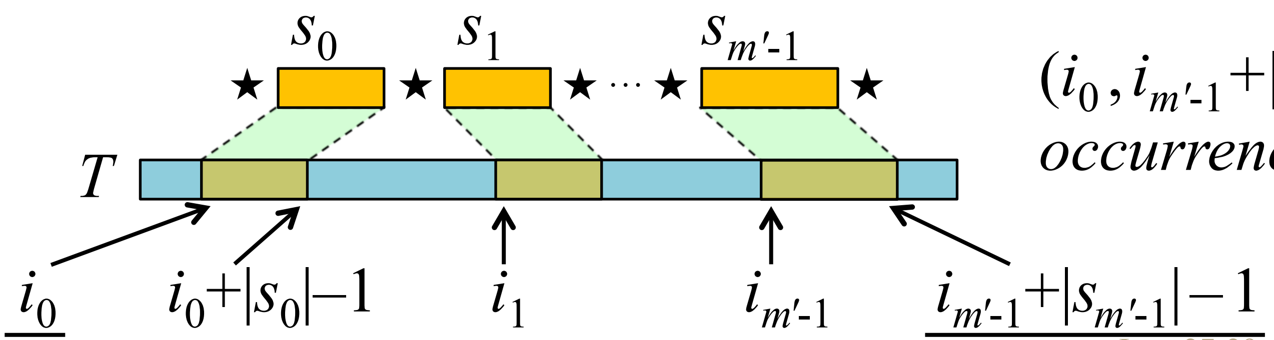
➔ Just modify base cases for Q and L, computation of M and $M^{\sharp}$ are the same

Variable Length Don't Care

P	VLDC Pattern ($\star\, s_0 \star s_1 \star \cdots \star s_{m'-1} \star$)
s_j	Segment ($s_j \in \Sigma^+, j = 0, \ldots, m'-1$)
m'	# of segments
m	pattern length ($m = |s_0| + |s_1| + \cdots + |s_{m'-1}|$)

P **occurs** in $T \in \Sigma^* \Leftrightarrow \exists\,(i_1, \ldots, i_{m'})$ s.t.

$$T[i_0] = s_0[0], \ldots, T[i_0 + |s_0| - 1] = s_0[|s_0| - 1], \cdots,$$
$$T[i_{m-1}] = s_{m'-1}[0], \ldots, T[i_{m'-1} + |s_{m'-1}| - 1] = s_{m'-1}[|s_{m'-1}| - 1],$$
$$i_0 < i_0 + |s_0| \le i_1 < \cdots \le i_{m'-2} < i_{m'-2} + |s_{m'-2}| \le i_{m'-1}$$



$(i_0, i_{m'-1} + |s_{m'-1}| - 1)$ is an *occurrence* of P in T

VLDC Pattern Matching

Let $Occ^{串}(X_i, s_j)$ denote the crossing occurrences of segment s_j in X_i $(i=1,\ldots,n,\ j=0,\ldots,m')$

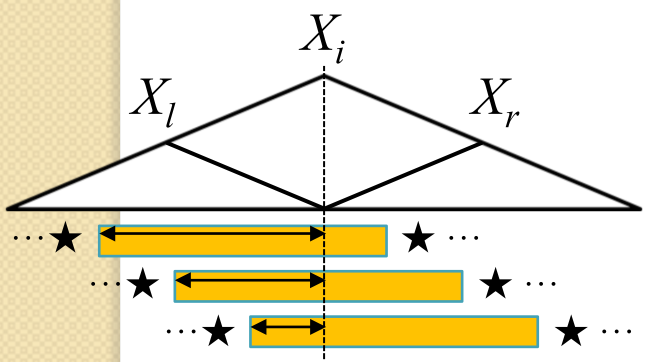
Length k suffix of X_l = Length k prefix of s_j

$$Occ^{串}(X_i, s_j) = \{k \mid X_l[|X_l|-k : |X_l|-1] = s_j[0 : k-1]$$

Length $|s_j| - k$ suffix of X_r = Length $|s_j| - k$ suffix s_j

$$\wedge\ X_r[0 : |s_j|-k-1] = s_j[k : |s_j|-1]$$

$$\wedge\ |X_l| \geq k\ \wedge\ |X_r| \geq |s_j|-k\ \}$$



[Kida *et al.*, 2003]

All $Occ^{串}(X_i, s_j)$ can be computed in total of $O(nm)$ time. Each $Occ^{串}(X_i, s_j)$ is an arithmetic progression, and can be represented in O(1) space

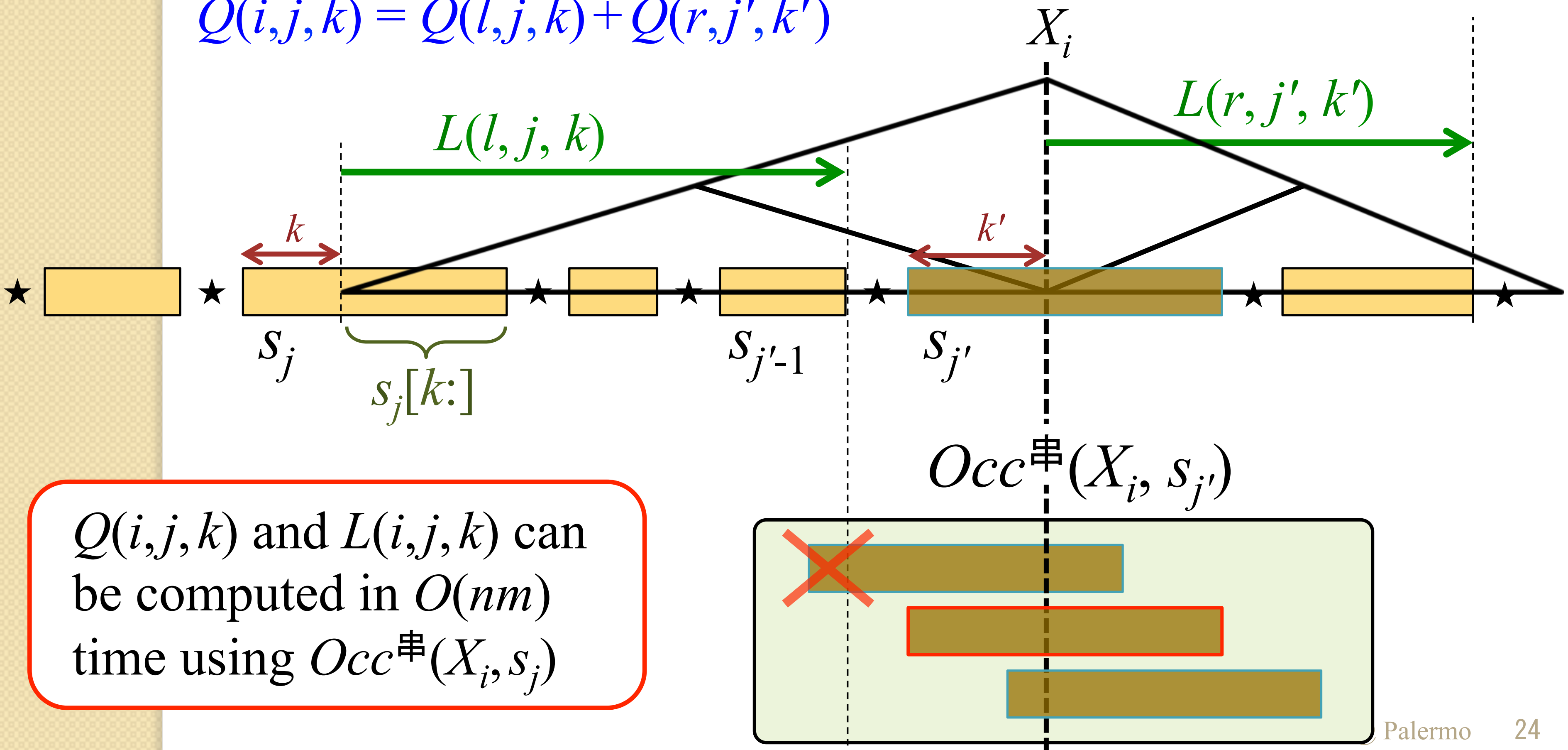
Computing Q and L for VLDC

case: $Q(l, j, k) \geq 1$ or $k = 0$

$j' := j + Q(l, j, k)$

$k' := \max\{x \mid x \in Occ^{串}(X_i, s_{j'}), x + L(l,j,k) \leq |X_l|\}$

$Q(i,j,k) = Q(l,j,k) + Q(r,j',k')$



$Q(i,j,k)$ and $L(i,j,k)$ can be computed in $O(nm)$ time using $Occ^{串}(X_i, s_j)$

Summary

Presented O(nm) algorithms on SLPs for:

- Window subsequence problems
- Fixed/Variable Length Don't Care Matching

Open Problems:

- Faster Longest Common Subsequence?
- Compressed Index for subsequence matching?