Pattern Matching on Compressed Texts II

Shunsuke Inenaga
Kyushu University, Japan

Agenda

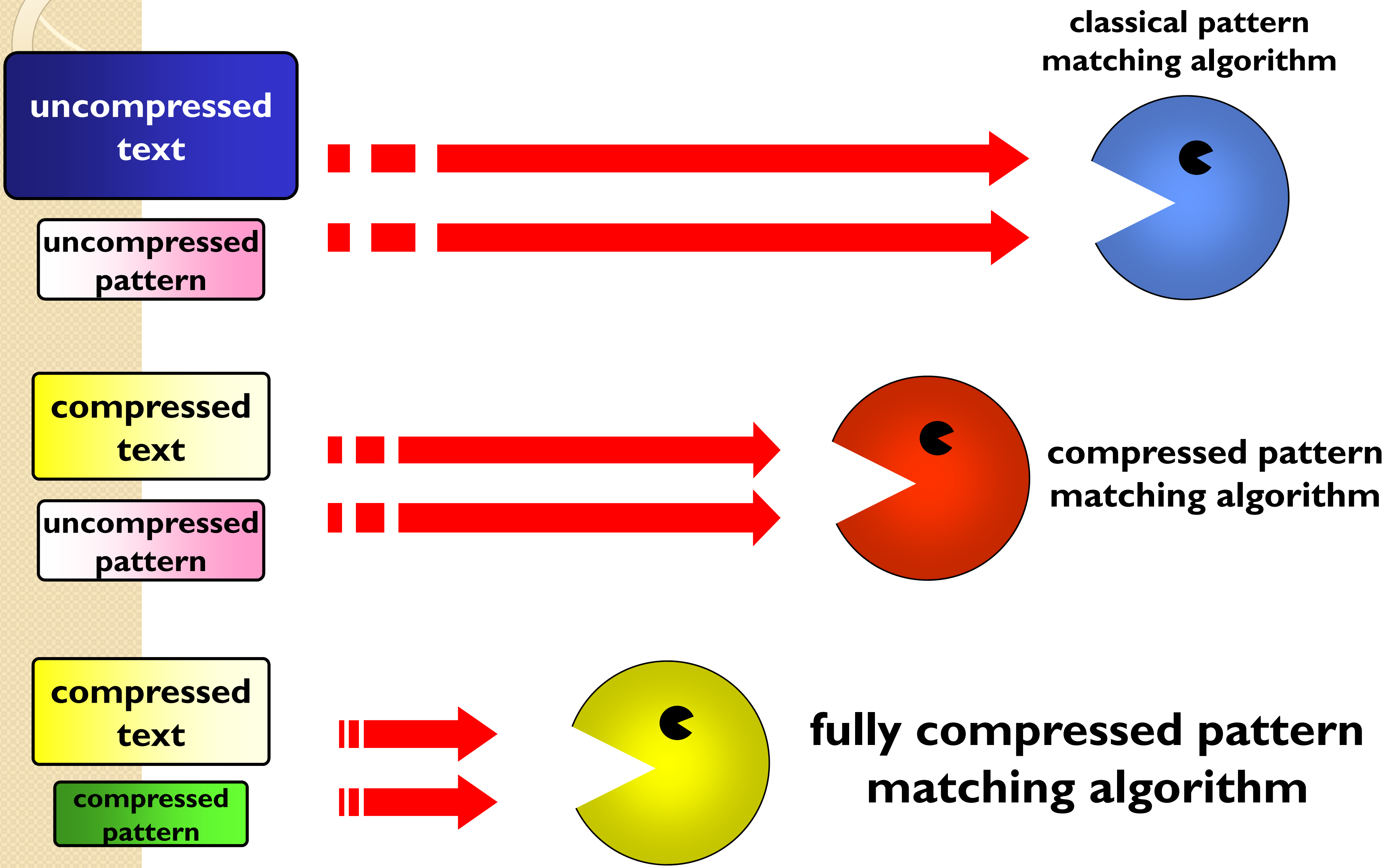
- Fully Compressed Pattern Matching
- Straight Line Program
- Compressed String Comparison
- Period of Compressed String
- Pattern Discovery from Compressed String (Palindrome and Square)
- FCPM for 2D SLP
- Open Problems

Fully Compressed Pattern Matching [1/3]

compressed pattern: &(aG

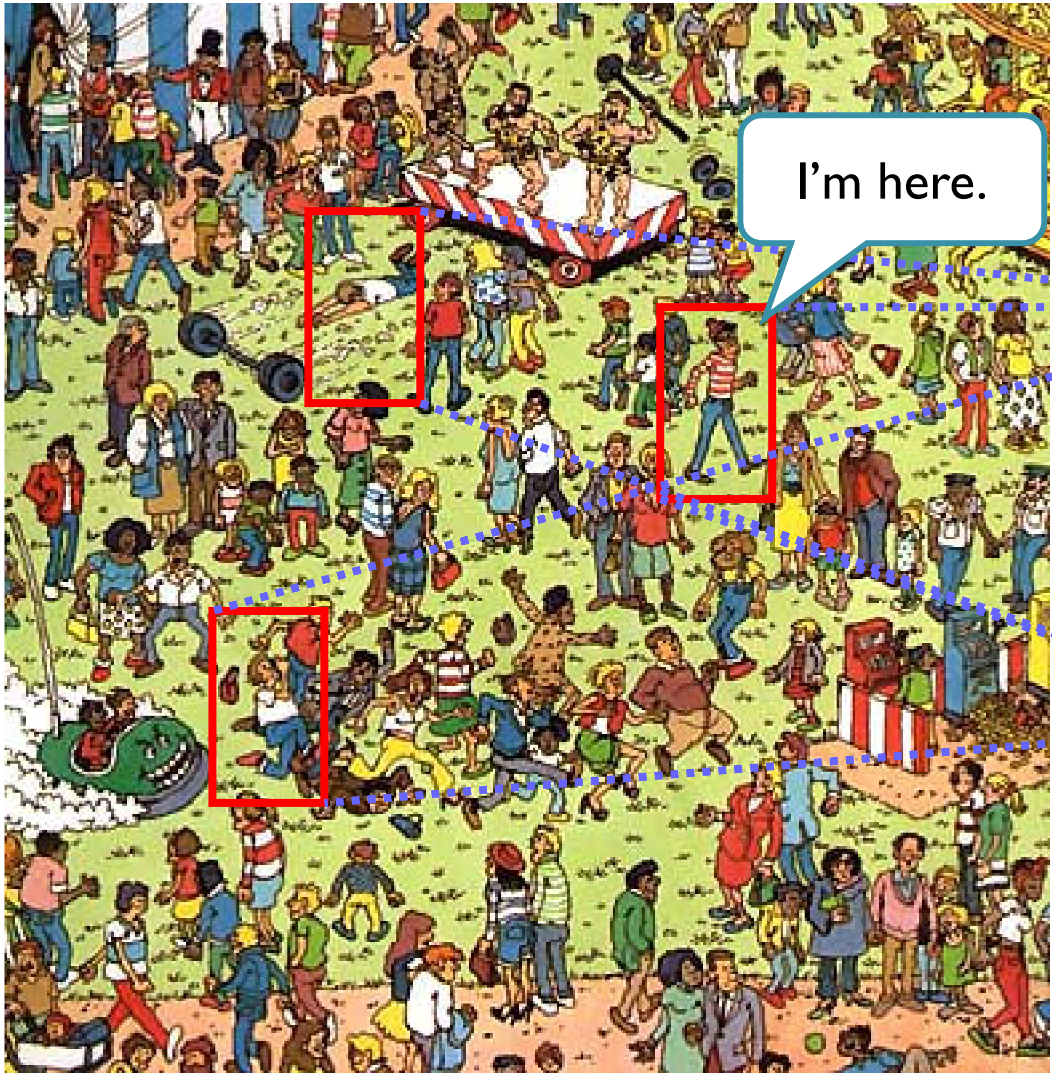
compressed text: geoiy083qa0gj(#*gpfomo)#(JGWRE$(U)%ARY)(JPED(A%RJG)ER%U)JGODAAQWT$JGWRE)$RJ)REWJFDOPIJKSeoiy083qa0gj(#*gpfomo)#(JGWRE$(U)%ARY)(JPED(A%RJG)ER%U)JGODAAQWT$JGWRE)$geoiy083qa0gj(#*gpfomo)#(JGWRE$(U)%ARY)(JPED(A%RJG)ER%U)JGODAAQWT$JGWRE)$geoiy083qa0gj(#*gpfomo)#(JGWRE$(U)%ARY)(

Fully Compressed Pattern Matching [2/3]



Possible Application of FCPM

compressed text



compressed pattern

Fully Compressed Pattern Matching [3/3]

FCPM Problem

Input : $\mathcal{T} = \mathbf{compress}(T)$ and $\mathcal{P} = \mathbf{compress}(P)$.

Output : Set $Occ(T, P)$ of substring occurrences of pattern P in text T.

- $Occ(T,P) = \left\{ |u|+1 : T = uPw,\ u, w \in \Sigma^* \right\}$

Straight Line Program [1/2]

SLP $\boldsymbol{T}$: sequence of assignments

$$X_1 = expr_1\ ;\ X_2 = expr_2;\ \ldots\ ;\ X_n = expr_n;$$

X_k : variable,

$$expr_k : \begin{cases} a & (\ a \in \Sigma\ ) \\ X_i X_j & (\ i, j < k\ ). \end{cases}$$

SLP $\boldsymbol{T}$ for string T is a CFG in Chomsky normal form s.t. $L(\boldsymbol{T}) = \{T\}$.

Straight Line Program [2/2]

SLP T

n

$$X_1 = a$$
$$X_2 = b$$
$$X_3 = X_1X_2$$
$$X_4 = X_3X_1$$
$$X_5 = X_3X_4$$
$$X_6 = X_5X_5$$
$$X_7 = X_4X_6$$
$$X_8 = X_7X_5$$

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

N

$$N = O(2^n)$$

Straight Line Program [2/2]

SLP T

n

$X_1 = a$
$X_2 = b$
$X_3 = X_1X_2$
$X_4 = X_3X_1$
$X_5 = X_3X_4$
$X_6 = X_5X_5$
$X_7 = X_4X_6$
$X_8 = X_7X_5$

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

N

$N = O(2^n)$

From LZ77 to SLP

For any string $\boldsymbol{T}$ given in LZ77-compressed form of size k, an SLP generating $\boldsymbol{T}$ of size $O(k^2)$ can be constructed in $O(k^2)$ time.

[Rytter '00, '03, '04]

FCPM for SLP

FCPM Problem for SLP

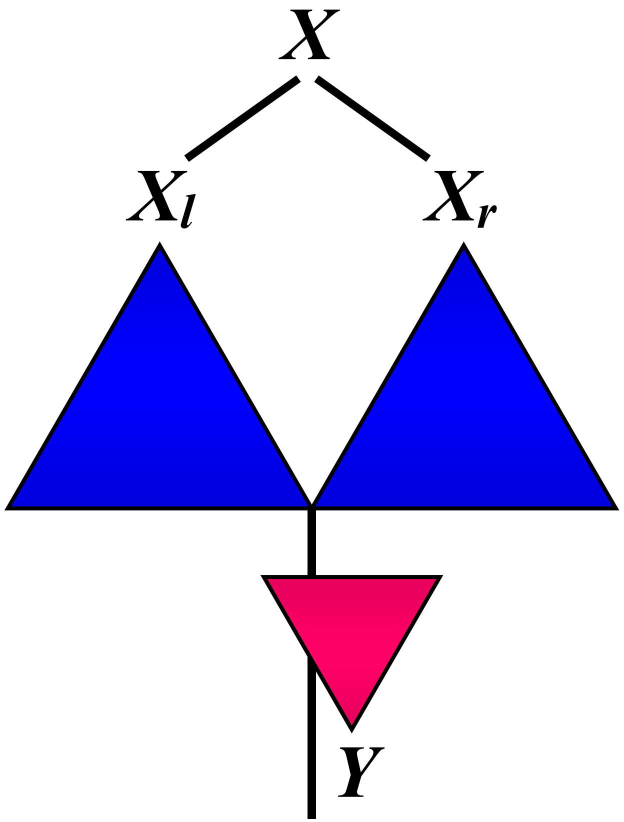
Input : SLP $\boldsymbol{T}$ for text T and SLP $\boldsymbol{P}$ for pattern P.

Output : Compact representation of set $Occ(T, P)$ of substring occurrences of P in T.

- We want to solve the problem <u>efficiently</u> (i.e., <u>polynomial</u> time & space in n and m).
  - n = the size of SLP $\boldsymbol{T}$, m = the size of SLP $\boldsymbol{P}$
- $|T| = O(2^n) \Longrightarrow$ $\boldsymbol{T}$ (also $\boldsymbol{P}$) cannot be decompressed
- $|Occ(T,P)| = O(2^n) \Longrightarrow$ compact representation

Key Definition

$$Occ^{\triangle}(X, Y) = \{\, i \in Occ(X, Y) \mid |X_l| - |Y| \le i \le |X_l| \}$$



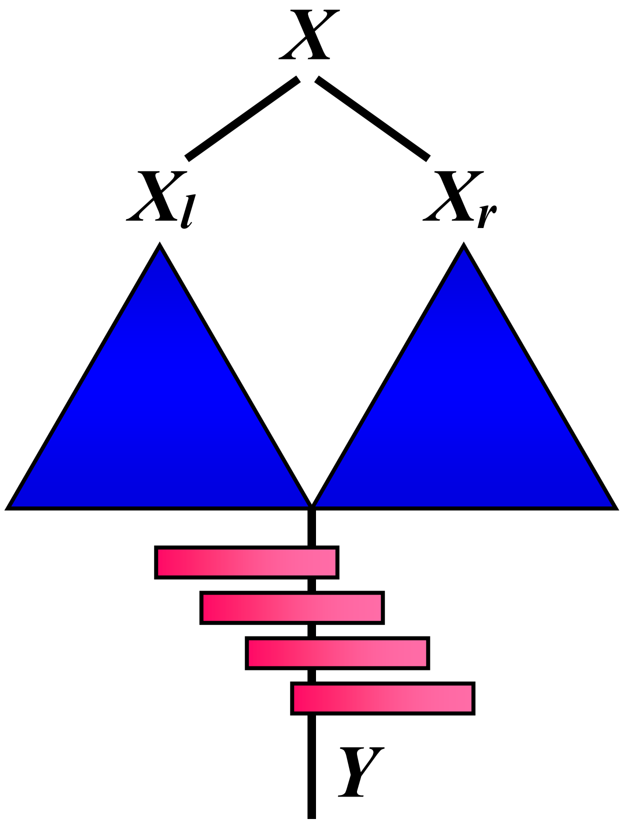
set of occurrences of Y that cover or touch the boundary of X_l and X_r.

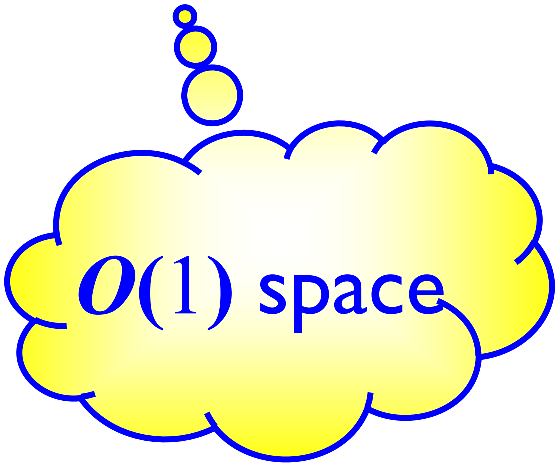
X: variable of T
Y: variable of P

Key Lemma

[Miyazaki et al. '97]

$Occ^{\triangle}(X, Y)$ forms **a single arithmetic progression**.

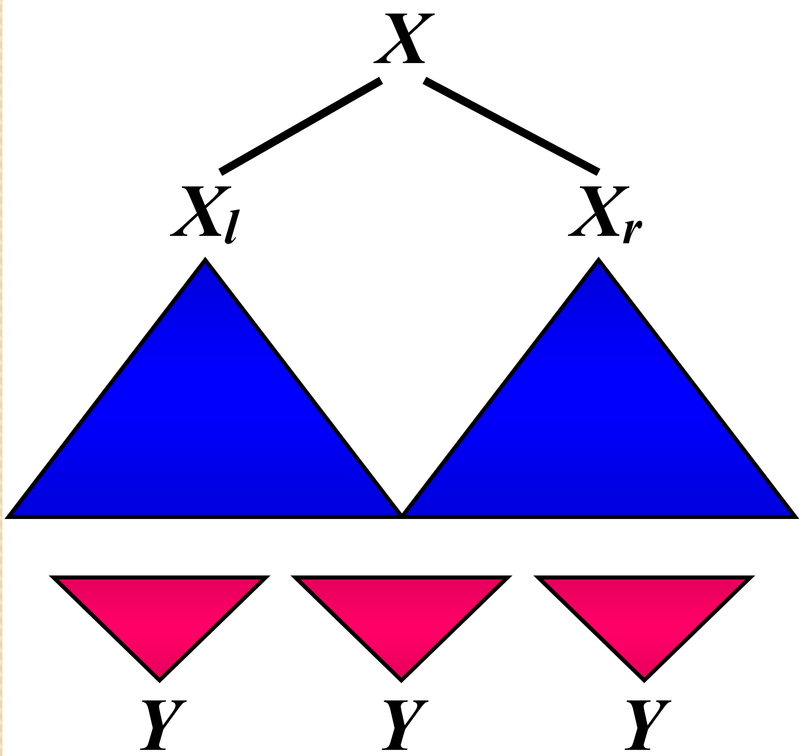




Key Observation

$$Occ(X,Y) = Occ(X_l,Y) \cup Occ^{\Delta}(X,Y) \cup Occ(X_r,Y) \oplus |X_l|$$

[Miyazaki et al. '97]



Computing $Occ(X, Y)$ is reduced to computing $Occ^{\Delta}(X, Y)$.

DP for $Occ^{\triangle}(X_i, Y_j)$

$Occ^{\triangle}(T, P)$

X_n	$Occ^{\triangle}(X_n, Y_1)$	....	$Occ^{\triangle}(X_n, Y_j)$	....	$Occ^{\triangle}(X_n, Y_m)$
⋮					
X_i	$Occ^{\triangle}(X_i, Y_1)$		$Occ^{\triangle}(X_i, Y_j)$		$Occ^{\triangle}(X_i, Y_m)$
⋮					
X_1	$Occ^{\triangle}(X_1, Y_1)$		$Occ^{\triangle}(X_1, Y_j)$		$Occ^{\triangle}(X_1, Y_m)$
	Y_1		Y_j		Y_m

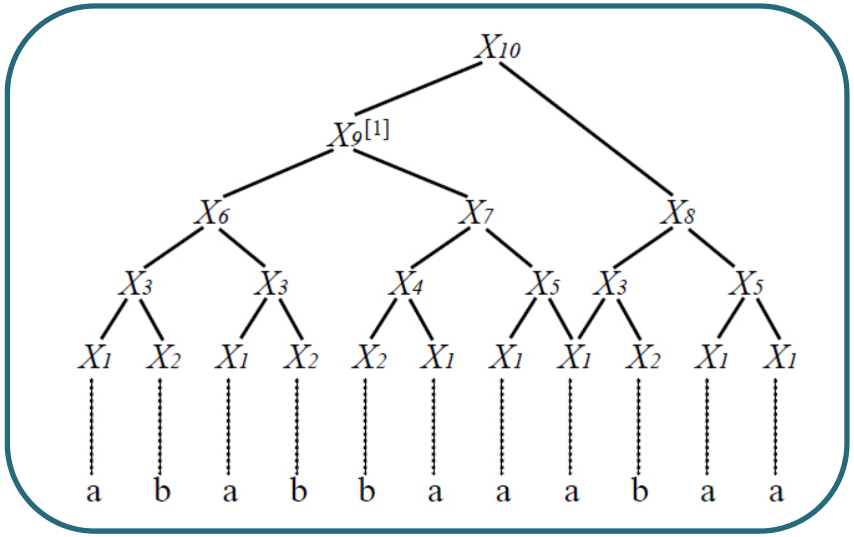
$O(1)$ space

Compact representation of $Occ(T, P)$ which answers a membership query to $Occ(T, P)$ in $O(n)$ time.

Known Results

	Time	Space	Compression
Miyazaki et al. '97	$O(m^2n^2)$	$O(mn)$	SLP
Lifshits '07	$O(mn^2)$	$O(mn)$	SLP
Hirao et al. '00	$O(mn)$	$O(mn)$	Balanced SLP

Balanced SLP



Fully Compressed Subsequence Pattern Matching [1/2]

FC Subsequence PM Problem

Input : SLP $\boldsymbol{T}$ for text T and SLP $\boldsymbol{P}$ for pattern P.

Output : Find whether P is a subsequence of T.

- P is said to be a subsequence of T, if P can be obtained by removing zero or more characters from T.

Fully Compressed Subsequence Pattern Matching [2/2]

The Fully Compressed Subsequence Pattern Matching Problem on SLP compressed strings is NP-hard.

[Lifshits & Lohrey '06]

Compressed String Comparison [1/2]

CSC Problem

Input : SLPs ***T*** and ***S*** for strings T and S, resp.

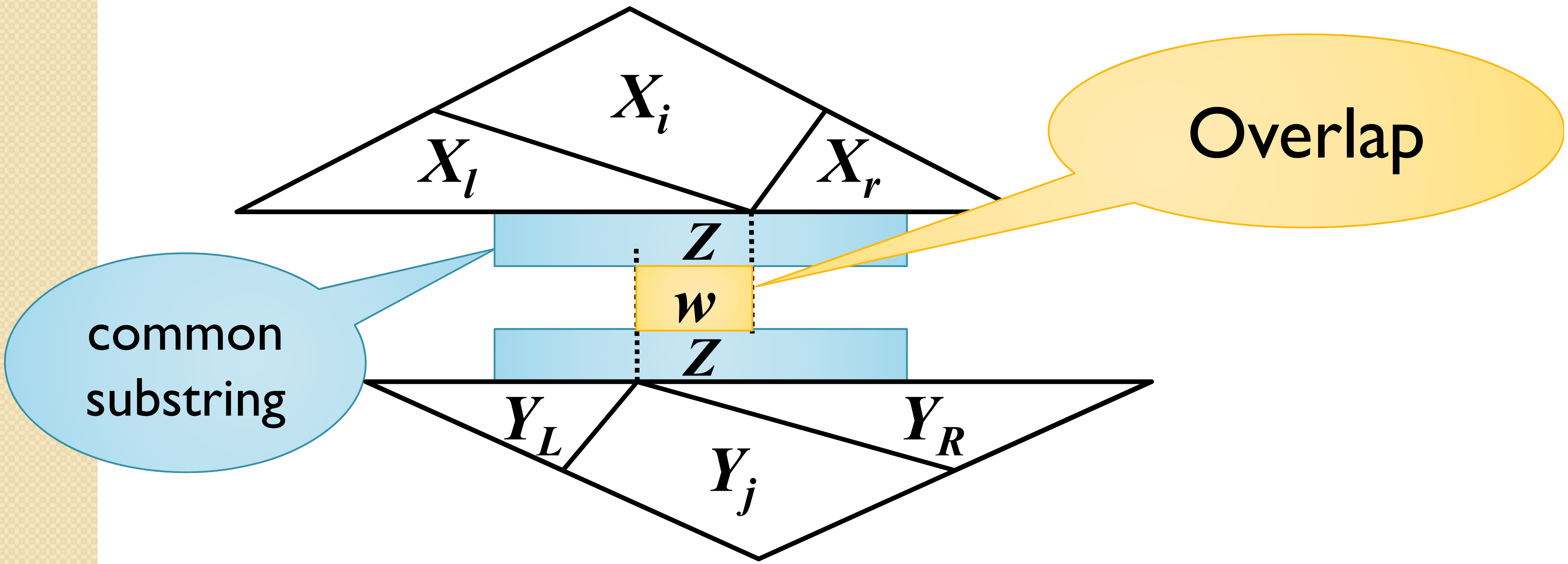
Output : Dis(similarity) of T and S.

Compressed String Comparison [2/2]

Measure	Time	Space	Reference
Equality	$O(mn^2)$	$O(mn)$	Lifshits '07
Hamming Distance	#P-complete	PSPACE	Lifshits '07
Longest Common Substring	$O((m+n)^4\log(m+n))$	$O((m+n)^3)$	Matsubara et al. '08
Longest Common Subsequence	NP-hard	PSPACE	Lifshits & Lohrey '06

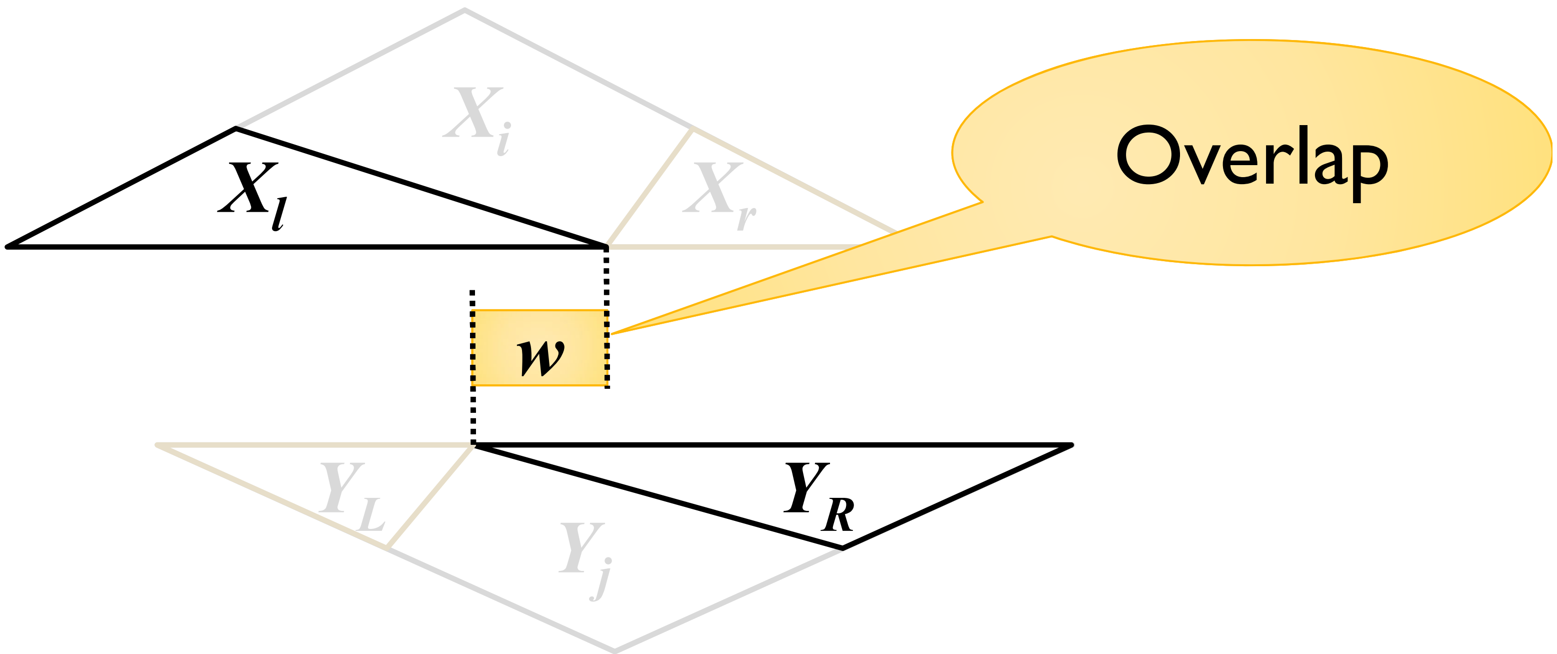
Property of common substrings [1/3]

- For each common substring Z of string S and T, there always exists a variable $X_i = X_l X_r$ and $Y_j = Y_L Y_R$ such that:
  - Z is a common substring of X_i and Y_j
  - Z contains an overlap between X_l and Y_R



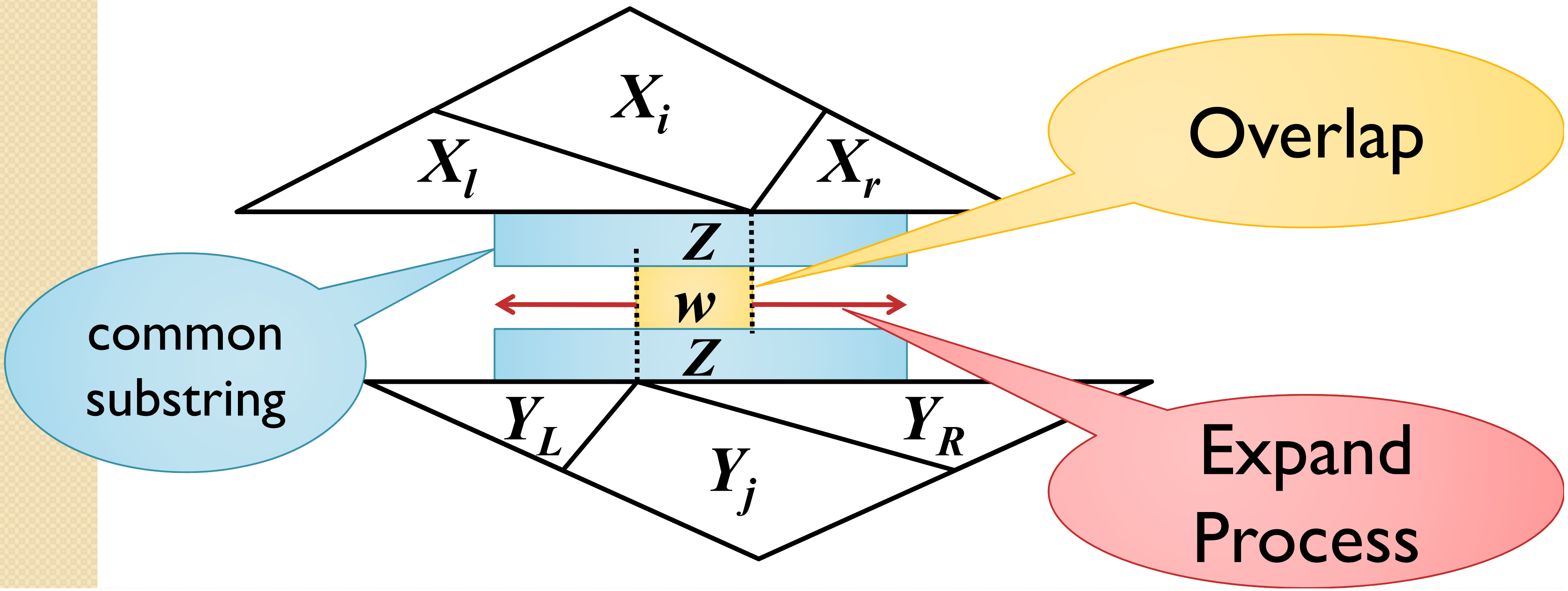
Property of common substrings [2/3]

- For each common substring Z of string S and T, there always exists a string w such that:
  - w is a substring of Z
  - w is an overlap of variables of S and T



Property of common substrings [1/3]

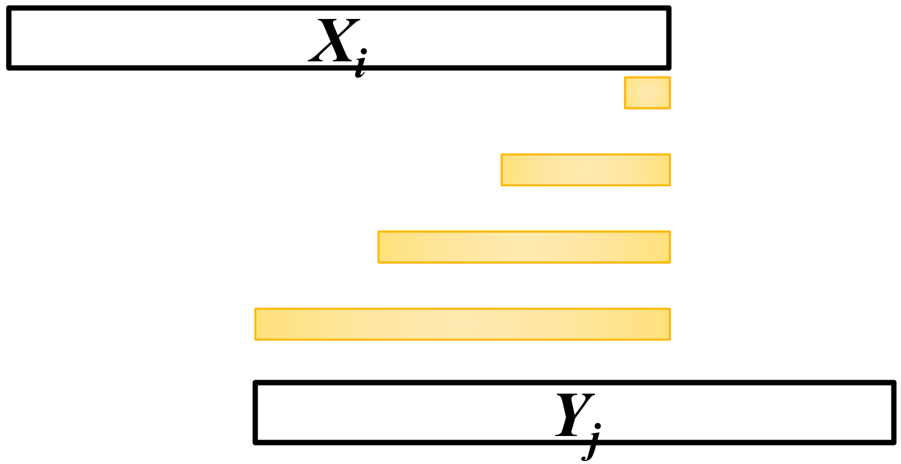
- For each common substring Z of string S and T, there always exists a string w such that:
  - Z can be calculated by expanding w



Computing Overlaps

Lemma [Karpinski et al. '97]

For any variables X_i and X_j of SLP T, $OL(X_i, X_j)$ can be represented by $O(n)$ arithmetic progressions.



Theorem [Karpinski et ai. '97]

For any SLP T, $OL(X_i, X_j)$ can be computed in total of $O(n^4 \log n)$ time and $O(n^3)$ space for each i, j.

Periods of Compressed String [1/2]

Compressed Period Problem

Input : SLP $\boldsymbol{T}$ for string T.

Output : Compact representation of set $\boldsymbol{Period(T)}$ of periods of T.

- $\boldsymbol{Period(T) = \left\{ |T| - |u| : T = uv = wu,\ v, w \in \Sigma^{+} \right\}}$

Periods of Compressed String [2/2]

An $\boldsymbol{O(n)}$-size representation of $\boldsymbol{Period(T)}$ can be computed in $\boldsymbol{O(n^4)}$ time with $\boldsymbol{O(n^3)}$ space.

[Lifshits '06, '07]

Compressed Palindrome Discovery [1/2]

Compressed Palindrome Discovery Problem

Input : SLP $\boldsymbol{T}$ for string T.

Output : Compact representation of set $\boldsymbol{Pal(T)}$ of maximal palindromes of T.

- $\boldsymbol{Pal(T)} = \left\{ \begin{array}{l} (p,q) : T[p:q] \text{ is the maximal palindrome} \\ \text{centered at } \lfloor (p+q)/2 \rfloor. \end{array} \right\}$
- ex. $T = baabbaa$

Compressed Palindrome Discovery [2/2]

An $O(n^2)$-size representation of $Pal(T)$ can be computed in $O(n^4)$ time with $O(n^2)$ space.

[Matsubara et al. '08]

Composition System

CS $\boldsymbol{T}$: sequence of assignments

$$X_1 = expr_1 \,;\, X_2 = expr_2;\; \ldots\; ;\, X_n = expr_n;$$

X_k : variable,

$$expr_k : \begin{cases} a & (\, a \in \Sigma \,), \\ X_i X_j & (\, i, j < k \,), \\ {}^{[p]}X_i X_j{}^{[q]} & (\, i, j < k \,). \end{cases}$$

- ${}^{[p]}X = X[1{:}p]$
- $X^{[q]} = X[|X|\text{-}q{+}1{:}|X|]$

From LZ77 to CS

For any string $\boldsymbol{T}$ given in LZ77-compressed form of size k, a CS generating $\boldsymbol{T}$ of size $\boldsymbol{O(k\log k)}$ can be constructed in polynomial time.

[Gasieniec et al. '96]

Compressed Square Discovery [1/2]

Compressed Square Problem

Input : CS $\boldsymbol{T}$ for string T.

Output : Check the square freeness of T
(whether T contains a square or not).

- A square is any non-empty string of the form xx.

Compressed Square Discovery [2/2]

We can test square freeness of $\boldsymbol{T}$ in polynomial time in the size of given composition system $\boldsymbol{T}$.

[Gasieniec et al. '96, Rytter'00]

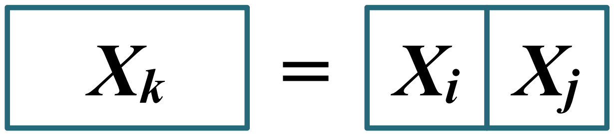
2D SLP

2D SLP $\boldsymbol{T}$: sequence of assignments

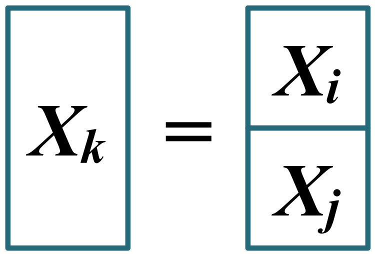
$$X_1 = expr_1 \ ; X_2 = expr_2; \ \ldots \ ; X_n = expr_n;$$

X_k : variable,

$$expr_k : \begin{cases} a & (\ a \in \Sigma\ ), \\ X_i \oplus X_j & (\ i, j < k,\ height(X_i) = height(X_j)\ ), \\ X_i \square\ X_j & (\ i, j < k,\ width(X_i) = width(X_j)\ ), \end{cases}$$



horizontal concatenation $\oplus$



vertical concatenation $\square$

FCPM for 2D SLP

The Fully Compressed Pattern Matching Problem for 2D SLP is Σ_2^P-complete.

[Berman et al. '97, Rytter'00]

Open Problems [1/2]

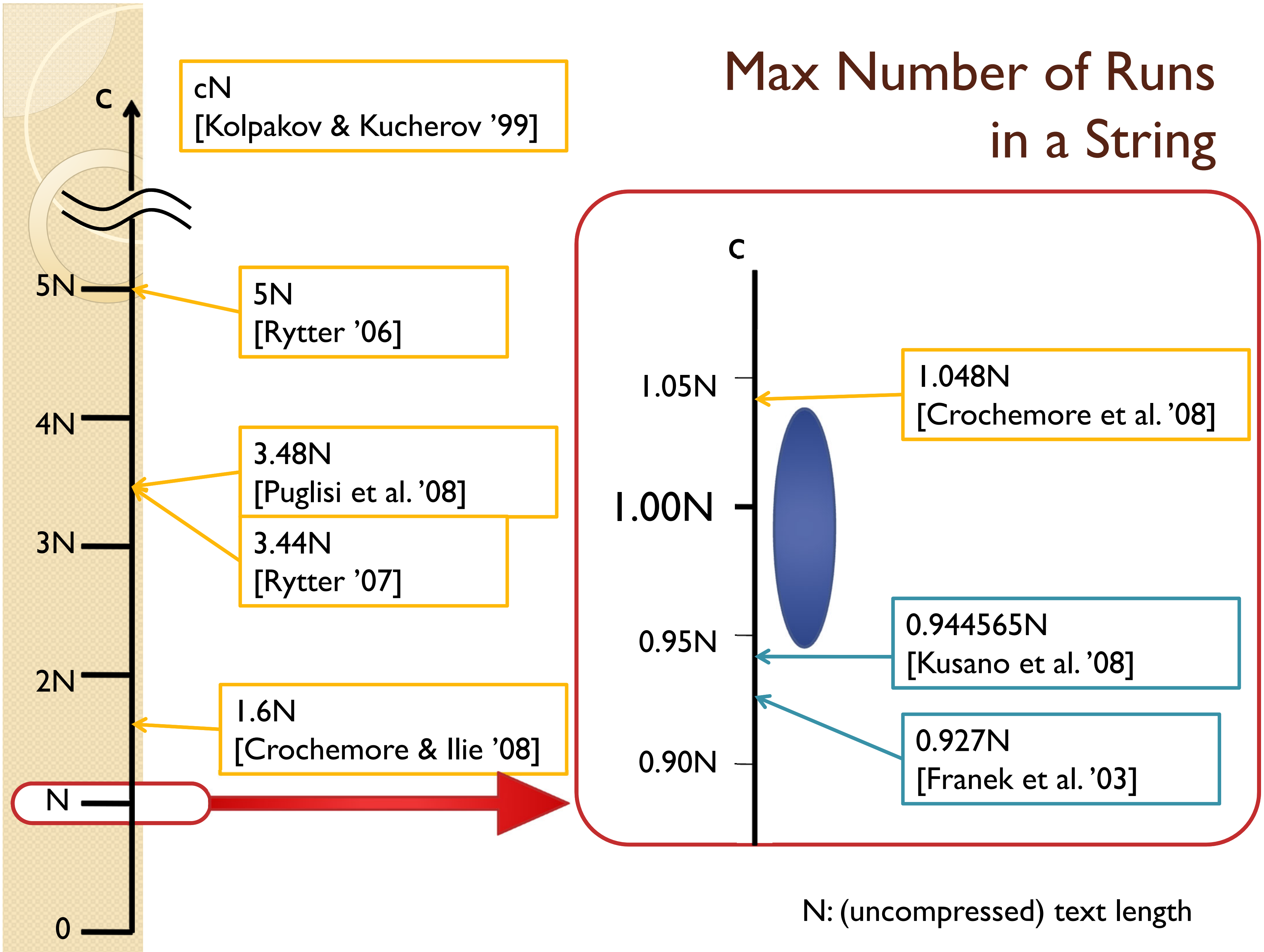
- Edit distance of two SLP-compressed strings.

- Compact representation of all maximal runs of an SLP-compressed string.

  - A run is any string x whose minimal period p satisfies $p \leq |x|/2$.

  - ex. $(aab)^{\frac{8}{3}} = aabaabaa$

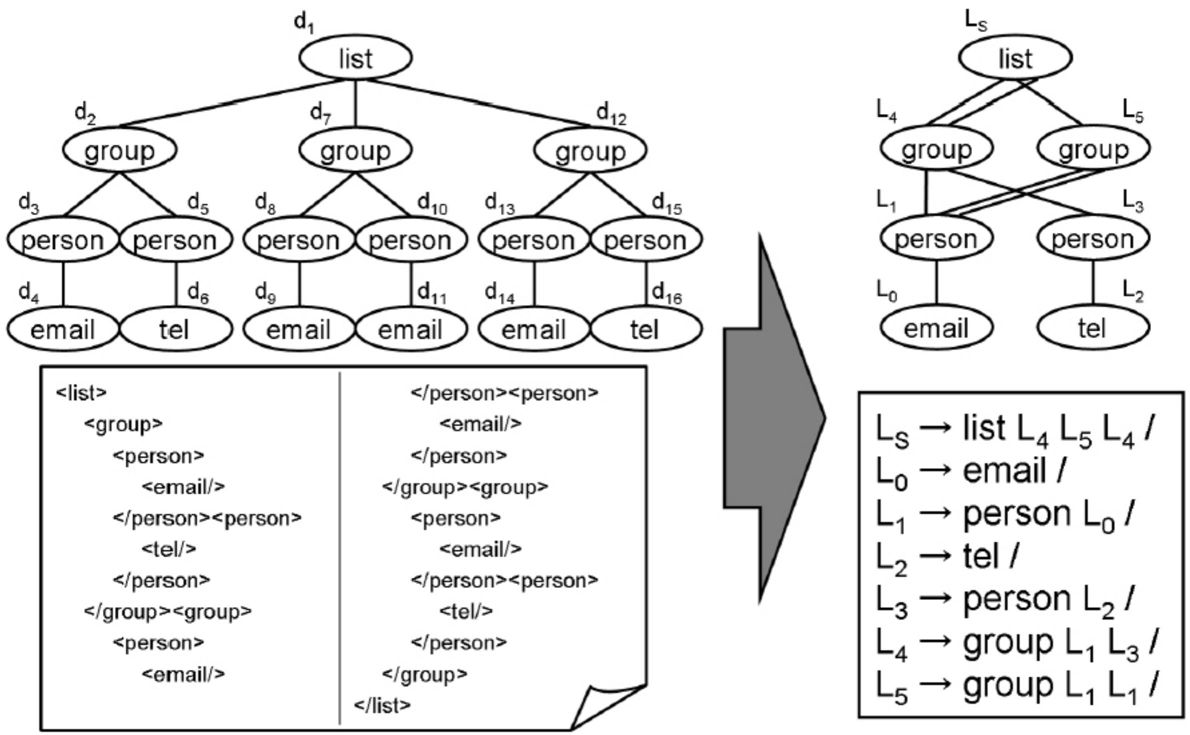
Max Number of Runs in a String



N: (uncompressed) text length

Open Problems [2/2]

- Fully Compressed Tree Pattern Matching for grammar based XML compression.
  - TGCA (Tree Grammar Compression Algorithm) [Onuma et al. '06]



References [1/5]

- [Matsubara et al. '08] W. Matsubara, S. Inenaga, A. Ishino, A. Shinohara, T. Nakamura, and K. Hashimoto, **Computing longest common substring and all palindromes from compressed strings**, Proc. SOFSEM'08, LNCS4910, pp. 364-375, 2008

- [Lifshits '07] Y. Lifshits, **Processing compressed texts: A tractability border** , Proc. CPM'07, LNCS 4580, pp 228-240, 2007

- [Lifshits '06] Y. Lifshits, **Solving Classical String Problems an Compressed Texts**, Dagstuhl Seminar Proceedings 06201, Schloss Dagstuhl, 2006

- [Hirao et ail. '00] M. Hirao, A. Shinohara, M. Takeda, and S. Arikawa, **Faster fully compressed pattern matching algorithm for balanced straight-line programs**, Proc. of SPIRE2000, pp. 132-138, IEEE Computer Society, 2000

References [2/5]

- [Miyazaki et al. '97] M. Miyazaki, A. Shinohara, and M. Takeda, **An improved pattern matching algorithm for strings in terms of straight-line programs**, Proc. CPM'97, LNCS1264, pp.1-11, 1997

- [Gasieniec '96] L. Gasieniec, M. Karpinski, W. Plandowski, W. Rytter, **Efficient Algorithms for Lempel-Zip Encoding (Extended Abstract)**, Proc. SWAT'96, LNCS1097, pp. 392-403, 1996

- [Lifsthis & Lohrey '06] Y. Lifshits and M. Lohrey, **Querying and Embedding Compressed Texts**, Proc. MFCS'06, LNCS4162, pp. 681-692, 2006

- [Rytter '04] W. Rytter, **Grammar Compression, LZ-Encodings, and String Algorithms with Implicit Input**, Proc. ICALP 2004, LNCS 3142, pp. 15-27, 2004

References [3/5]

- [Rytter '03] W. Rytter, **Application of Lempel-Ziv factorization to the approximation of grammar-based compression**, TCS, Volume 302, Number 1-3, pp. 211-222, 2003

- [Rytter '00] W. Rytter, **Compressed and fully compressed pattern matching in One and Two Dimensions**, Proceedings of IEEE, Volume 88, Number 11, pp. 1769-1778, 2000

- [Berman et al. '97] P. Berman, M. Karpinski, L. L. Larmore, W. Plandowski, W. Rytter, **On the Complexity of Pattern Matching for Highly Compressed Two-Dimensional Texts**, Proc. CPM'97, LNCS1264, pp. 40-51 1997

- [Onuma et al. '06] J. Onuma, K. Doi, and A. Yamamoto, **Data compression and anti-unification for semi-structured documents with tree grammars (in Japanese)**, IEICE Technical Report AI2006-9, pages 45–50, 2006.

References [4/5]

- [Kusano et al. '08] K. Kusano, W. Matsubara, A. Ishino, H. Bannai, A. Shinohara, **New Lower Bounds for the Maximum Number of Runs in a String**, http://arxiv.org/abs/0804.1214

- [Franek et al. '03] F. Franek, R. Simpson, W. Smyth, **The maximum number of runs in a string**, Proc. AWOCA'03, pp. 26–35, 2003.

- [Kolpakov & Kucherov '99] R. Kolpakov and G. Kucherov, **Finding maximal repetitions in a word in linear time**, Proc. FOCS'99, pp. 596–604, 1999.

- [Rytter '06] W. Rytter, **The number of runs in a string: Improved analysis of the linear upper bound**, Proc. STACS'06, LNCS3884, pp. 184–195, 2006.

References [5/5]

- [Rytter '07] W. Rytter, **The number of runs in a string**, Inf. Comput., Volume 205, Number 9, pp. 1459–1469, 2007.
- [Crochemore & Ilie '08] M. Crochemore and L. Ilie, **Maximal repetitions in strings**, J. Comput. Syst. Sci., Volume 74, Number 5, pp. 796-807, 2008.
- [Crochremore et al. '08] M. Crochemore, L. Ilie, and L. Tinta, **Towards a Solution to the "Runs" Conjecture**, Proc. CPM'08, LNCS5029, pp. 290-302, 2008.
- [Puglisi et al. '08] S. Puglisi, J. Simpson, W. F. Smyth, **How many runs can a string contain?**, TCS, Volume 401, Issues 1-3, pp.165-171, 2008.
- [Kaprinski et al. '97] M. Karpinski, W. Rytter, A. Shinohara, **An efficient pattern-matching algorithm for strings with short descriptions**, Nordic Journal of Computing, Number 4, pp.172–186, 1997.