Finding Characteristic Substrings from Compressed Texts

○稲永俊介　坂内英夫（九州大学）

テキストマイニングとテキスト圧縮

- テキストマイニング：与えられたテキストデータから，未知の**規則**や**知識**を半自動的に抽出する技術
- テキスト圧縮：テキストの冗長性を排除して，データの**記述長を短縮**する技術

圧縮

展開

本研究

- 与えられた圧縮テキストに特徴的な部分文字列（パタン）を効率よく発見するアルゴリズムを提案する.
  - 最長繰り返し部分文字列
  - 最長重複なし繰り返し部分文字列
  - 最頻出部分文字列
  - 最頻出重複なし部分文字列
  - 入力圧縮パタンの最大拡張パタン

Straight Line Programによるテキスト圧縮

SLP 7

$X_1 = \mathrm{a}$

$X_2 = \mathrm{b}$

$X_3 = X_1X_2$

$X_4 = X_3X_1$

$X_5 = X_3X_4$

$X_6 = X_5X_5$

$X_7 = X_4X_6$

$X_8 = X_7X_5$

$T = \mathrm{abaababaabaababaaba}$

SLP 7は言語$\{T\}$を生成するChomsky標準形の文脈自由文法.

Straight Line Programによるテキスト圧縮

SLP 7

$X_1 = \mathrm{a}$

$X_2 = \mathrm{b}$

$X_3 = X_1X_2$

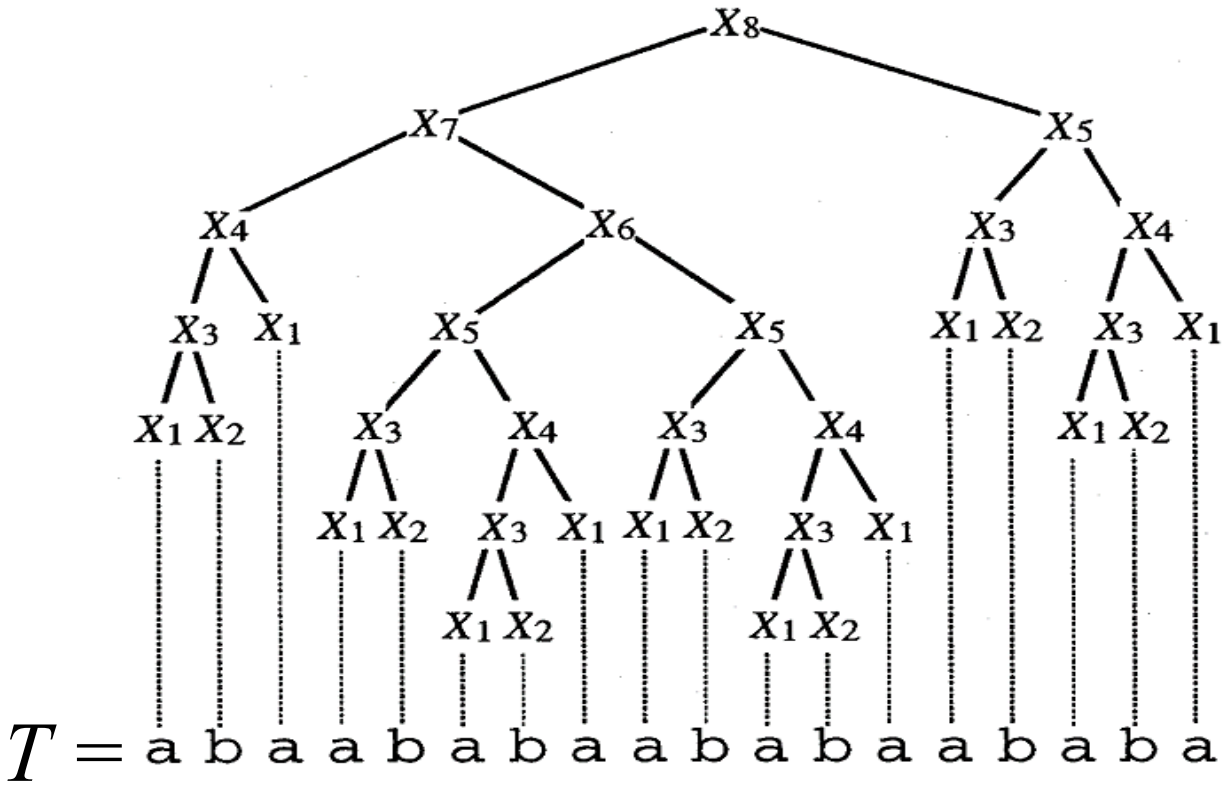
$X_4 = X_3X_1$

$X_5 = X_3X_4$

$X_6 = X_5X_5$

$X_7 = X_4X_6$

$X_8 = X_7X_5$



LZ系圧縮，文法圧縮，連長圧縮などで圧縮された文字列は高速にSLPに変換できる．

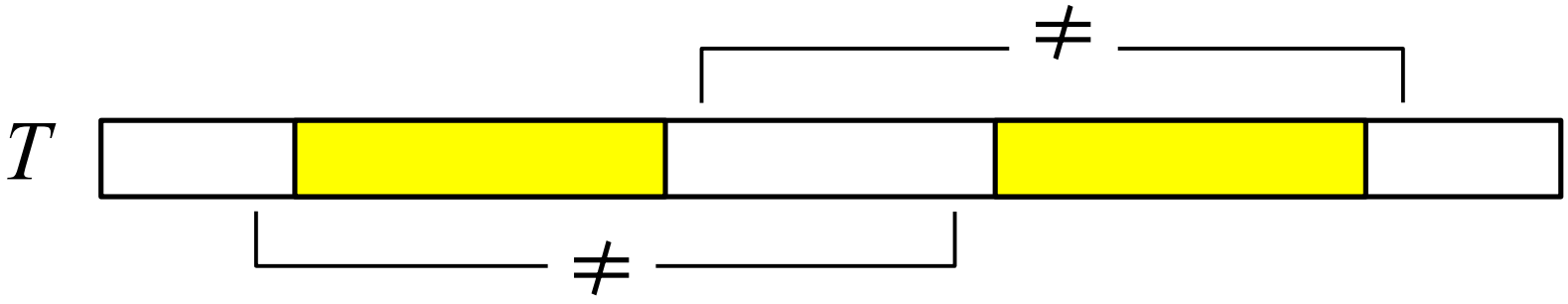
SLPの指数的圧縮能力

- 元テキストは，SLP圧縮されたテキストに対して指数的に大きくなりうる.
- テキスト T = abababab…ab　（abのN回繰り返し）
- SLP：$X_1 = \mathrm{a},\ X_2 = \mathrm{b},\ X_3 = X_1X_2,\ X_4 = X_3X_3,$
  $X_5 = X_4X_4,\ \ldots\ ,\ X_n = X_{n\text{-}1}X_{n\text{-}1}$
- $N = O(2^n)$
- 与えられた圧縮テキストを展開してから処理するアルゴリズムは，最悪時には指数時間を要する.

⇒ 本研究：圧縮テキストを陽に展開することなく，しかも圧縮サイズの多項式時間で動作する手法を提案

最長繰り返し部分文字列発見

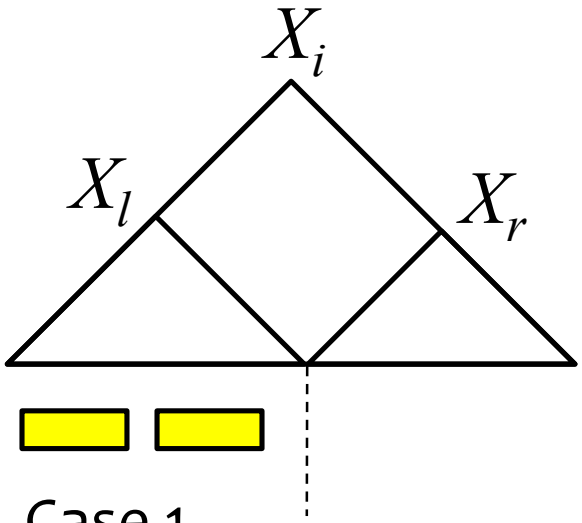
- 入力: テキスト T を生成するSLP $\mathcal{T}$
- 出力: T の**最長繰り返し部分文字列**



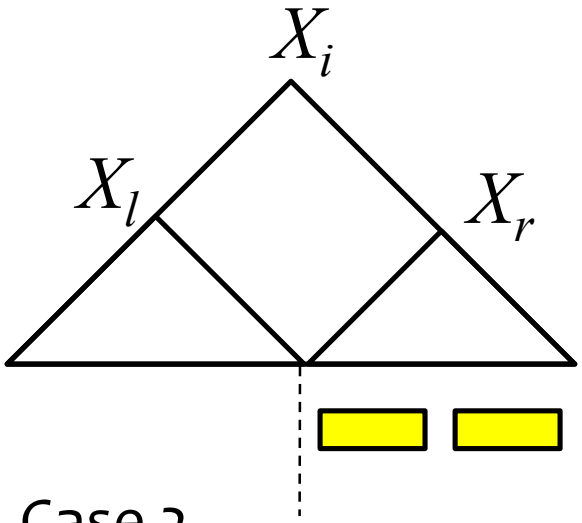
例

T = aabaabcabaabb

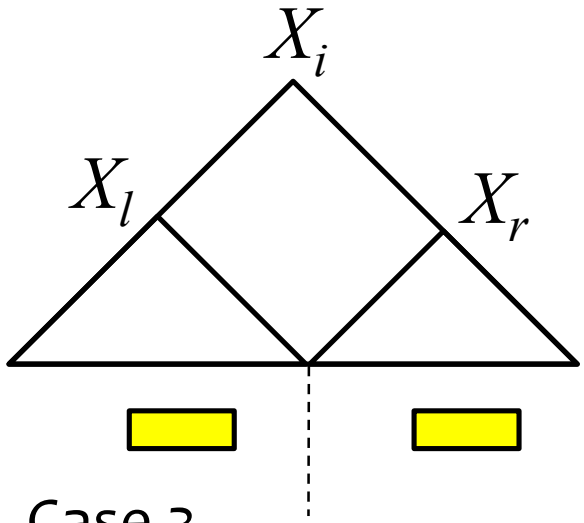
Key Observation – ６つの場合分け



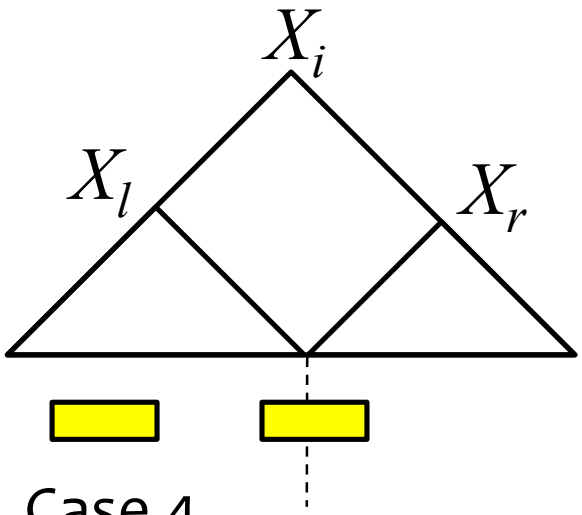
Case 1



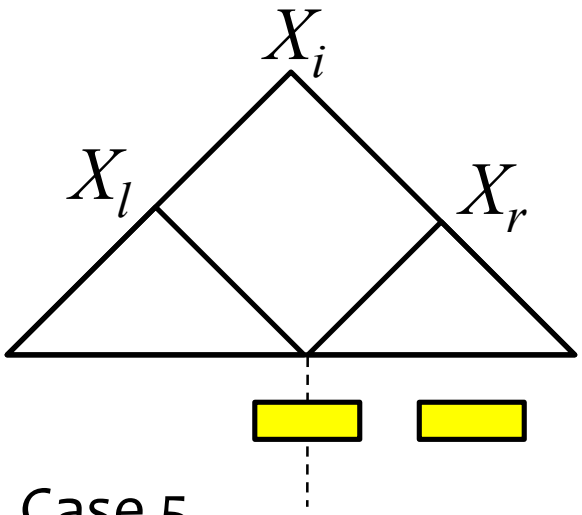
Case 2



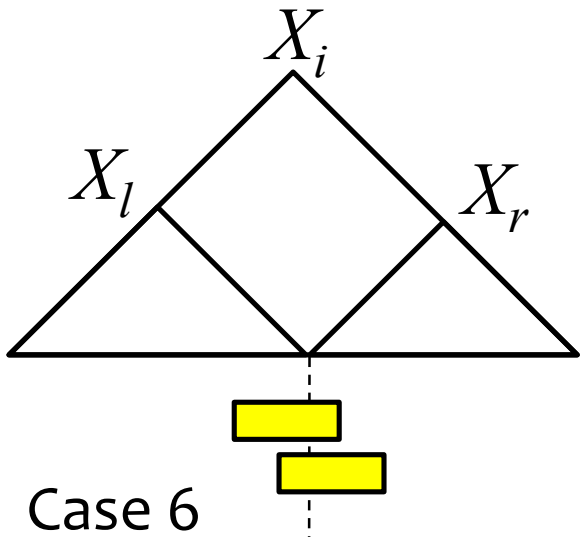
Case 3



Case 4



Case 5



Case 6

最長繰り返し部分文字列発見アルゴリズム

Input: SLP $\mathcal{T}$

Output: LRS of text T

foreach variable X_i of SLP $\mathcal{T}$ **do**

compute LRS of Case 1;
compute LRS of Case 2;
compute LRS of Case 3;
compute LRS of Case 4;
compute LRS of Case 5;
compute LRS of Case 6;

return two positions and the length of the "longest" LRS above;

最長繰り返し部分文字列発見アルゴリズム

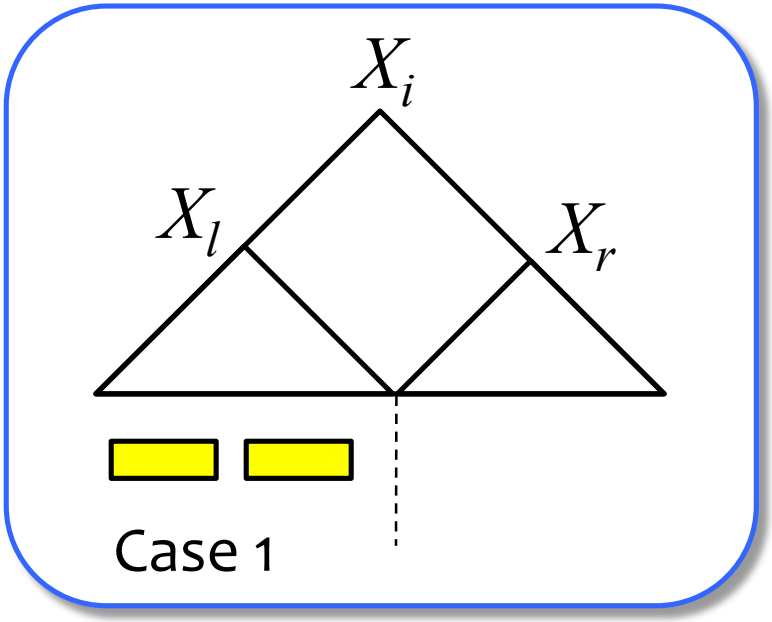
Input: SLP $\mathcal{T}$

Output: LRS of text T

foreach variable X_i of SLP $\mathcal{T}$ **do**

compute LRS of Case 1;
compute LRS of Case 2;
compute LRS of Case 3;
compute LRS of Case 4;
compute LRS of Case 5;
compute LRS of Case 6;

return two positions and the length of the "longest" LRS above;



最長繰り返し部分文字列発見アルゴリズム

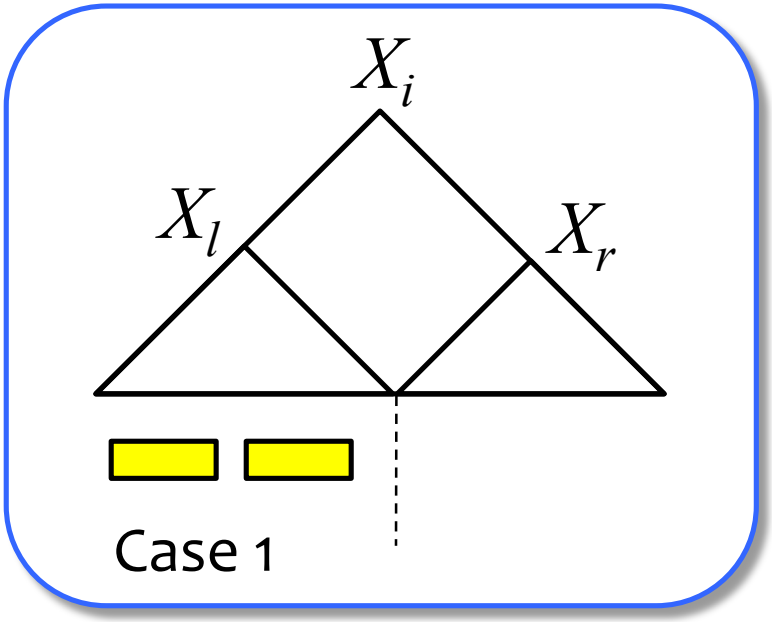
Input: SLP $\mathcal{T}$

Output: LRS of text T

foreach variable X_i of SLP $\mathcal{T}$ **do**

compute LRS of Case 1;
compute LRS of Case 2;
compute LRS of Case 3;
compute LRS of Case 4;
compute LRS of Case 5;
compute LRS of Case 6;

return two positions and the length of the "longest" LRS above;



最長繰り返し部分文字列発見アルゴリズム

Input: SLP $\mathcal{T}$

Output: LRS of text T

foreach variable X_i of SLP $\mathcal{T}$ **do**

compute LRS of Case 2;
compute LRS of Case 3;
compute LRS of Case 4;
compute LRS of Case 5;
compute LRS of Case 6;

return two positions and the length of the "longest" LRS above;

最長繰り返し部分文字列発見アルゴリズム

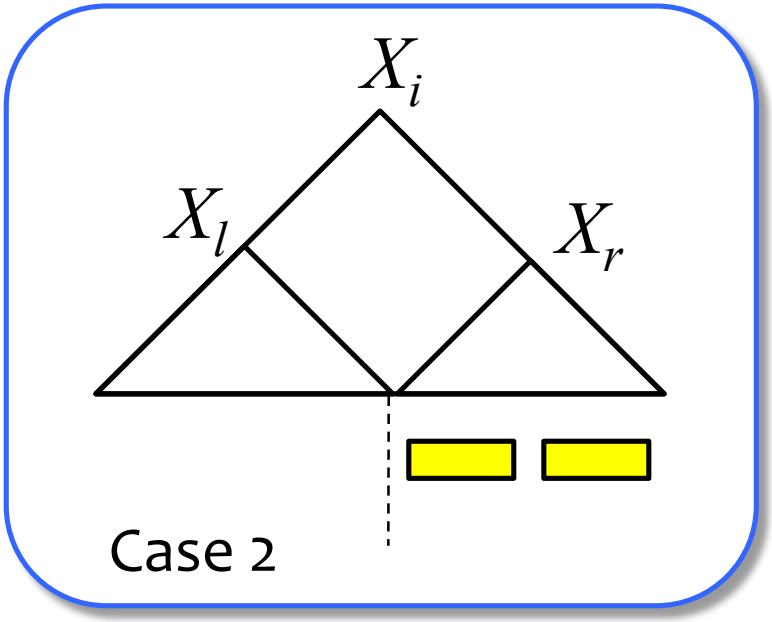
Input: SLP $\mathcal{T}$

Output: LRS of text T

foreach variable X_i of SLP $\mathcal{T}$ **do**

compute LRS of Case 2;
compute LRS of Case 3;
compute LRS of Case 4;
compute LRS of Case 5;
compute LRS of Case 6;

return two positions and the length of the "longest" LRS above;



最長繰り返し部分文字列発見アルゴリズム

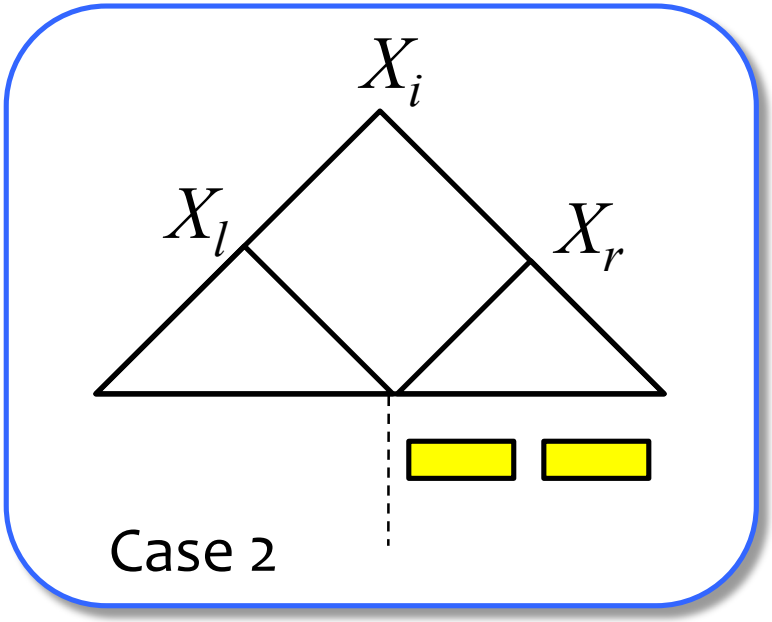
Input: SLP $\mathcal{T}$

Output: LRS of text T

foreach variable X_i of SLP $\mathcal{T}$ **do**

compute LRS of Case 2;
compute LRS of Case 3;
compute LRS of Case 4;
compute LRS of Case 5;
compute LRS of Case 6;

return two positions and the length of the "longest" LRS above;



最長繰り返し部分文字列発見アルゴリズム

Input: SLP $\mathcal{T}$

Output: LRS of text T

foreach variable X_i of SLP $\mathcal{T}$ **do**

compute LRS of Case 3;
compute LRS of Case 4;
compute LRS of Case 5;
compute LRS of Case 6;

return two positions and the length of the "longest" LRS above;

最長繰り返し部分文字列発見アルゴリズム

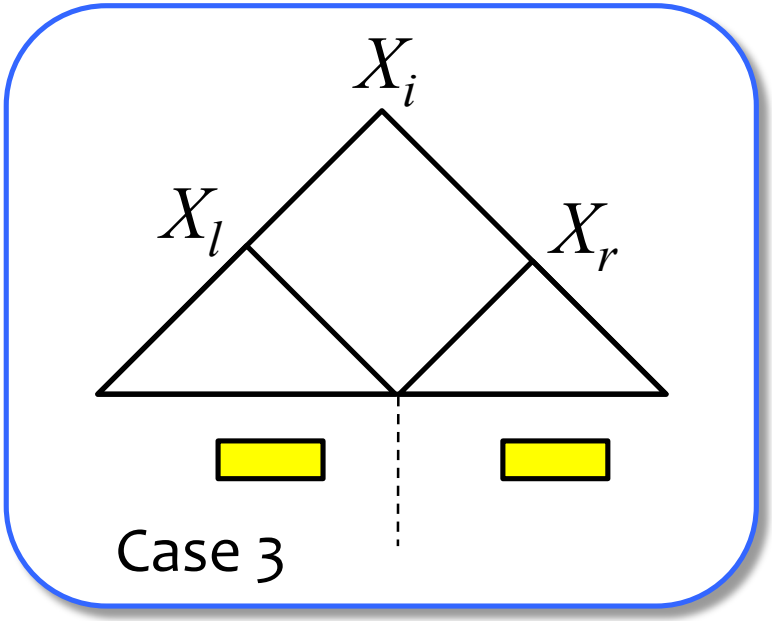
Input: SLP $\mathcal{T}$

Output: LRS of text T

foreach variable X_i of SLP $\mathcal{T}$ **do**

compute LRS of Case 3;
compute LRS of Case 4;
compute LRS of Case 5;
compute LRS of Cas

return two positions a
the "longest"



Case 3 の最長繰り返し部分文字列は，
X_l と X_r の最長共通部分文字列である

SLP圧縮テキストの最長共通部分文字列

定理１ [Matsubara et al. 2009]

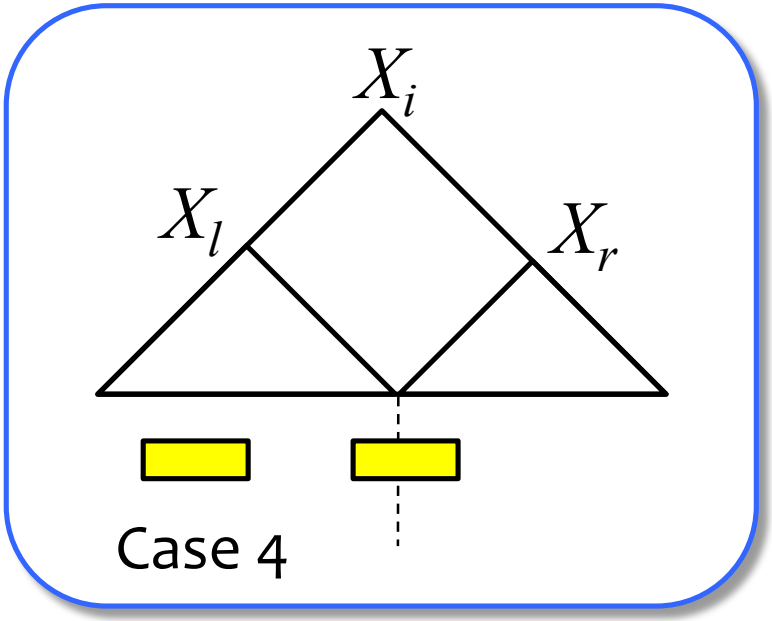
すべてのSLP 変数 X_l と X_rの最長共通部分文字列を$O(n^4 \log n)$時間で計算できる.

最長繰り返し部分文字列発見アルゴリズム

Input: SLP $\mathcal{T}$

Output: LRS of text T

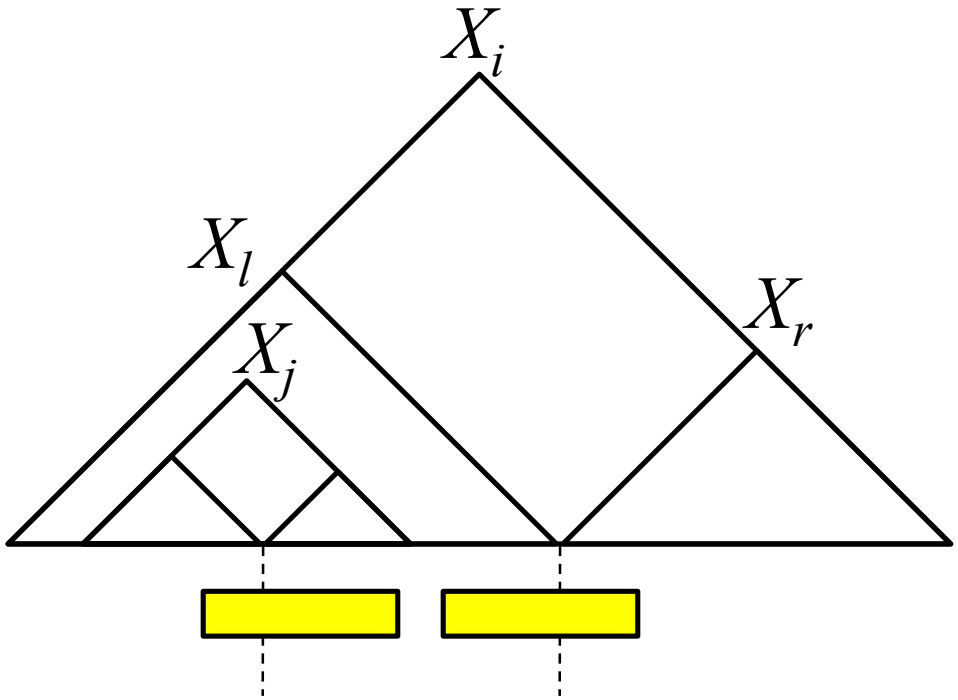
foreach variable X_i of SLP $\mathcal{T}$ **do**



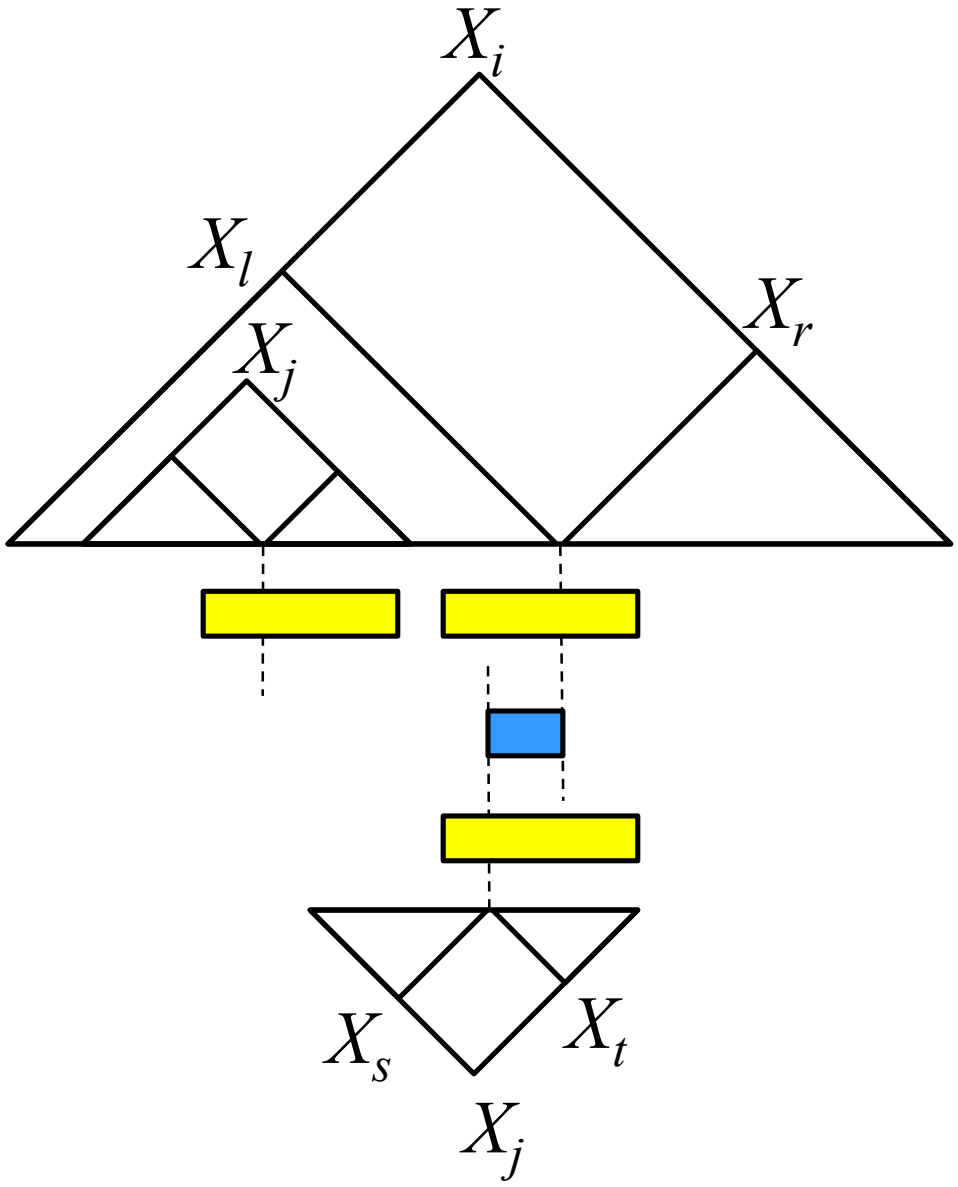
compute LRS of Case 3;
compute LRS of Case 4;
compute LRS of Case 5;
compute LRS of Case 6;

return two positions and the length of the "longest" LRS above;

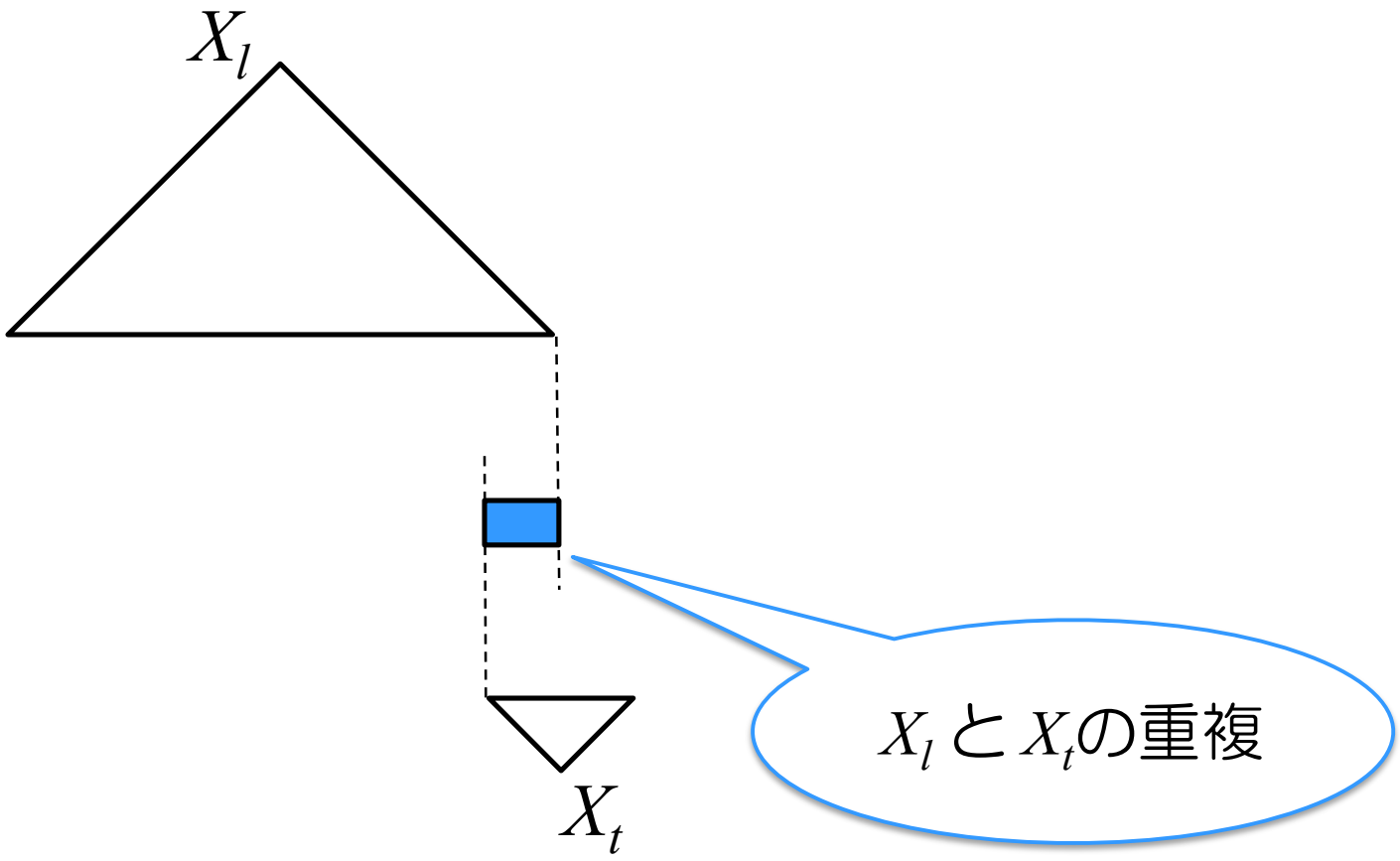
Case 4



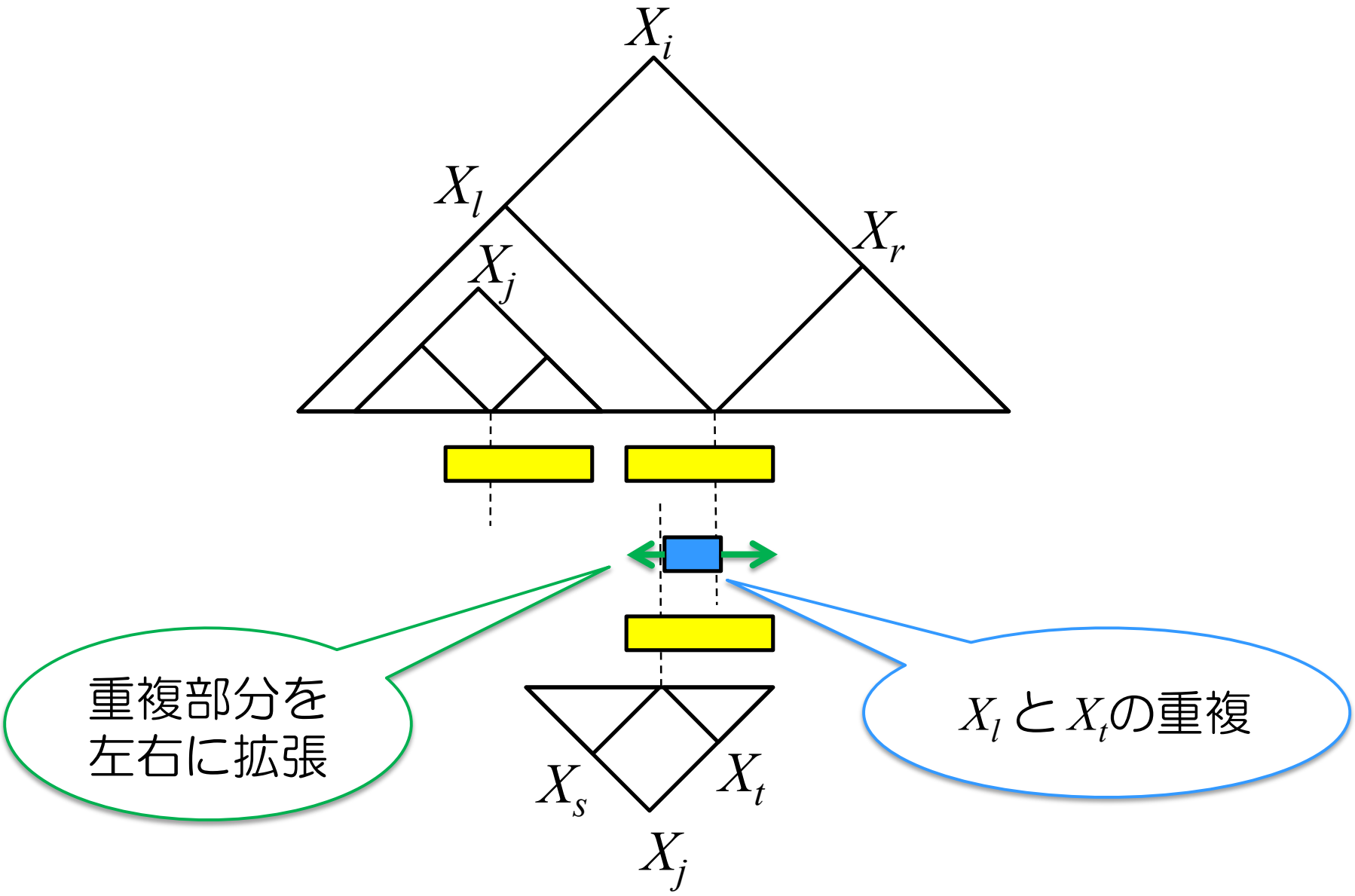
Case 4-1



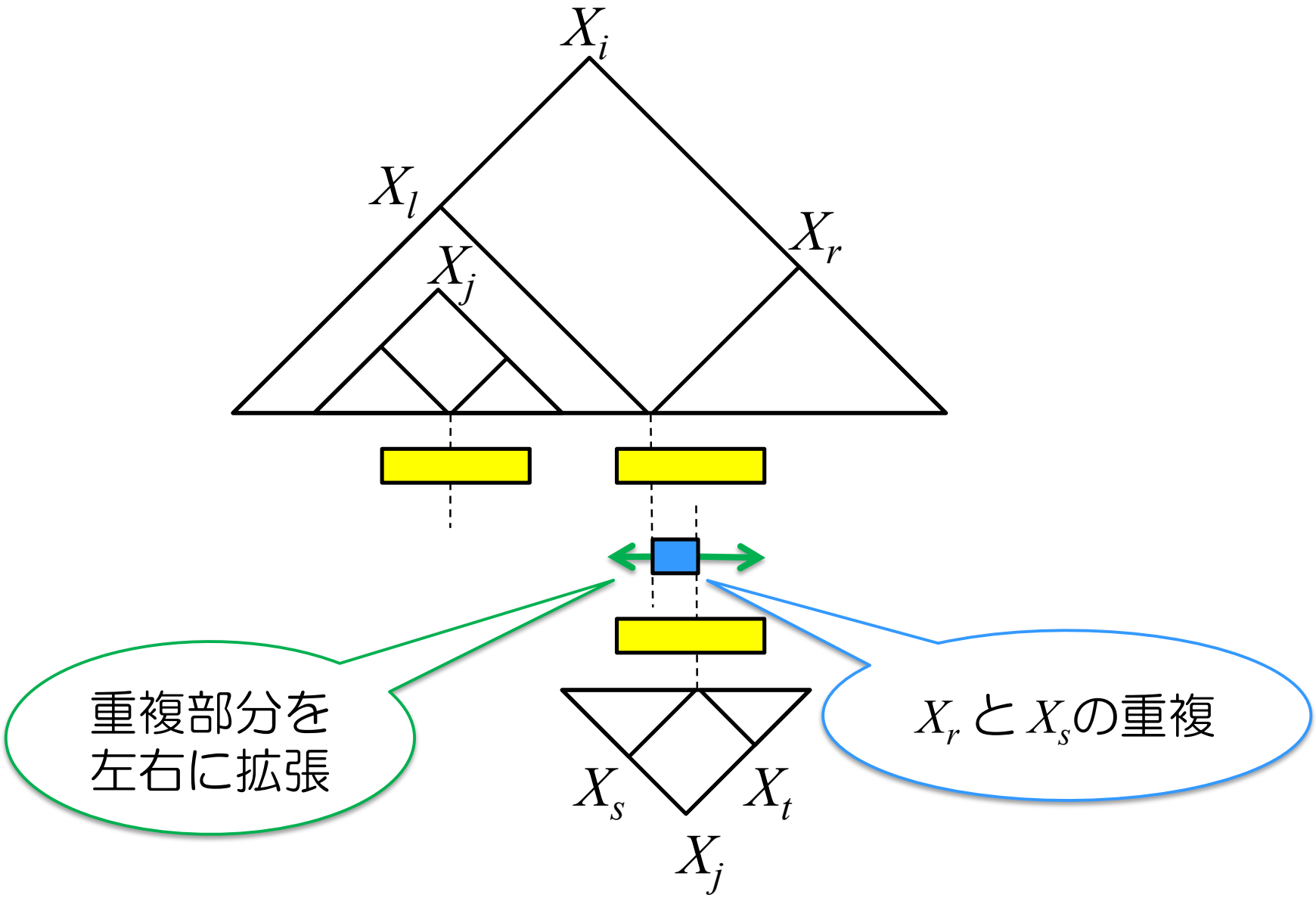
Case 4-1



Case 4-1

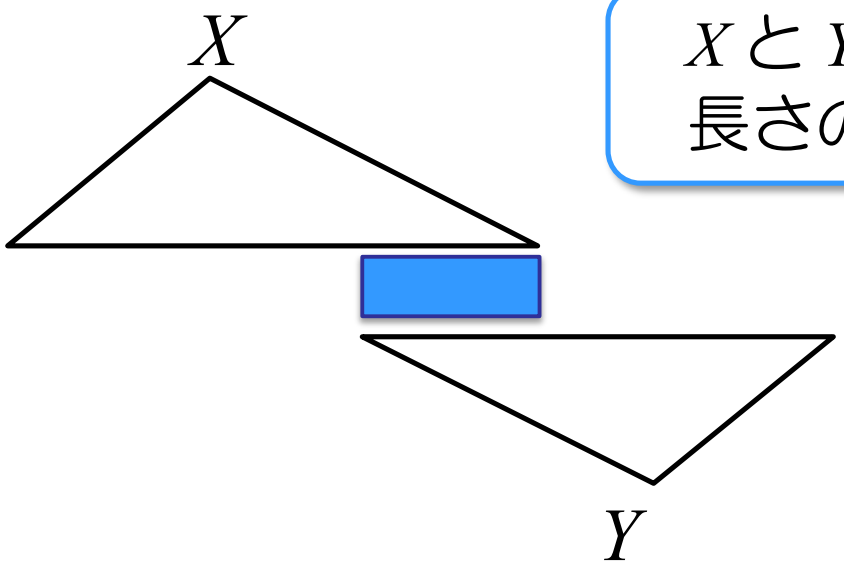


Case 4-2



重複の集合

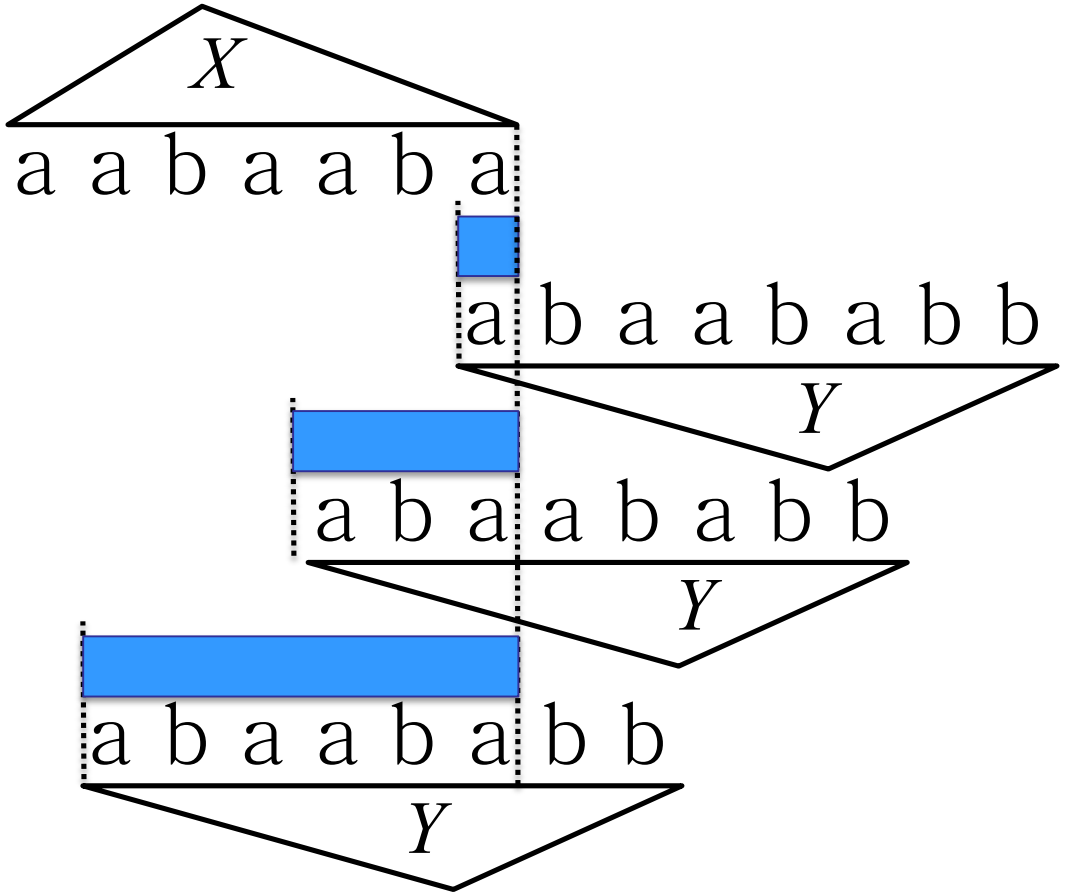
$$OL(X,Y) = \{k > 0 \mid X[|X| - k + 1 : |X|] = Y[1 : k]\}$$



X と Y の重複の
長さの集合

重複の集合

$OL(\mathrm{aabaaba}, \mathrm{abaababb}) = \{1, 3, 6\}$



重複の集合

補題１ [Kaprinski et al. 1997]

任意の２つの変数X_i と Y_jに対して，
$OL(X_i, Y_j)$ は $O(n)$ 個の等差数列をなす．

補題２ [Kaprinski et al. 1997]

すべての変数 X_i と Y_jに対して，
$OL(X_i, Y_j)$ は $O(n^4 \log n)$ 時間で計算できる．

Case 4

補題 3

任意の変数 X_iに対して，Case 4 の最長繰り返し部分文字列は$O(n^3 \log n)$時間で計算できる.

[証明のスケッチ]

- $OL(X_i, X_j)$の各等差数列の全要素を$O(n \log n)$時間で左右に拡張できる.
- $OL(Xi, X_j)$ のサイズは $O(n)$（補題 1 より）.
- X_iの子孫の数は高々n-1個.

最長繰り返し部分文字列発見アルゴリズム

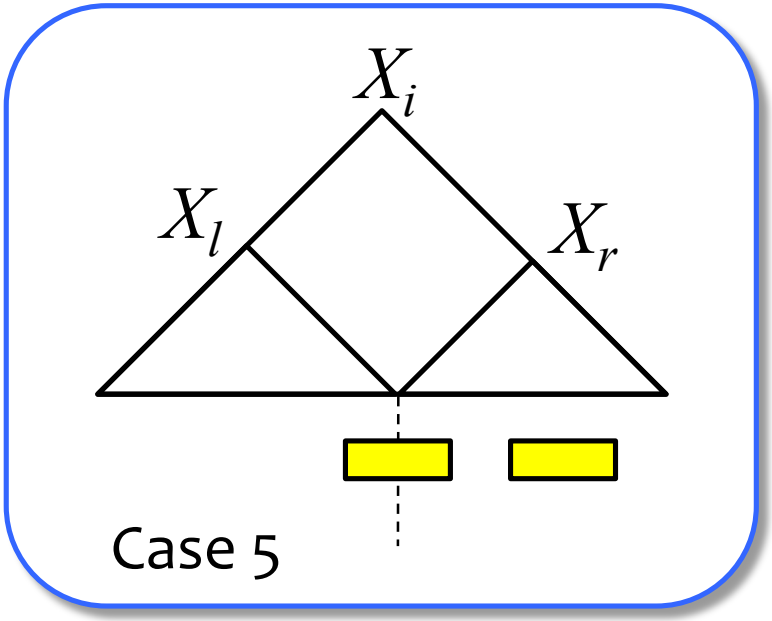
Input: SLP $\mathcal{T}$

Output: LRS of text T

foreach variable X_i of SLP $\mathcal{T}$ **do**

compute LRS of Case 3;
compute LRS of Case 4;
compute LRS of Case 5;
compute LRS of Case 6;

return two positions and the length of the "longest" LRS above;



Case 4 と対称

最長繰り返し部分文字列発見アルゴリズム

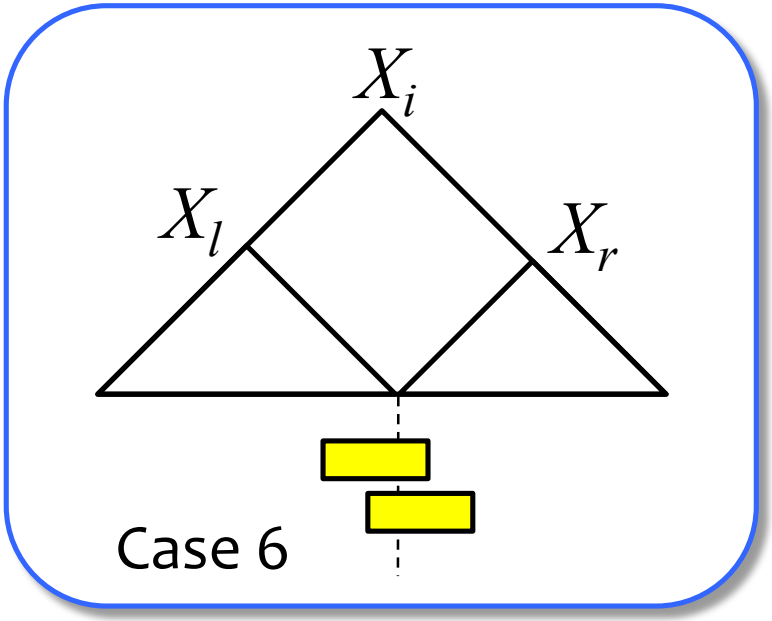
Input: SLP $\mathcal{T}$

Output: LRS of text T

foreach variable X_i of SLP $\mathcal{T}$ **do**

compute LRS of Case 3;
compute LRS of Case 4;
compute LRS of Case 5;
compute LRS of Case 6;

return two positions and the length of the "longest" LRS above;



Case 4 と同様

最長繰り返し部分文字列発見アルゴリズム

定理２

テキスト T を生成するSLP $\mathcal{T}$ に対して，Tの最長繰り返し部分文字列を$O(n^4 \log n)$時間で計算できる．

最長重複なし繰り返し部分文字列発見

- 入力: テキスト T を生成するSLP $\mathcal{T}$
- 出力: T の**最長重複なし繰り返し部分文字列**

例

$T =$ abababababab

最長繰り返し部分文字列は
abababab

最長重複なし繰り返し
部分文字列は abab

最長重複なし繰り返し部分文字列発見

定理３

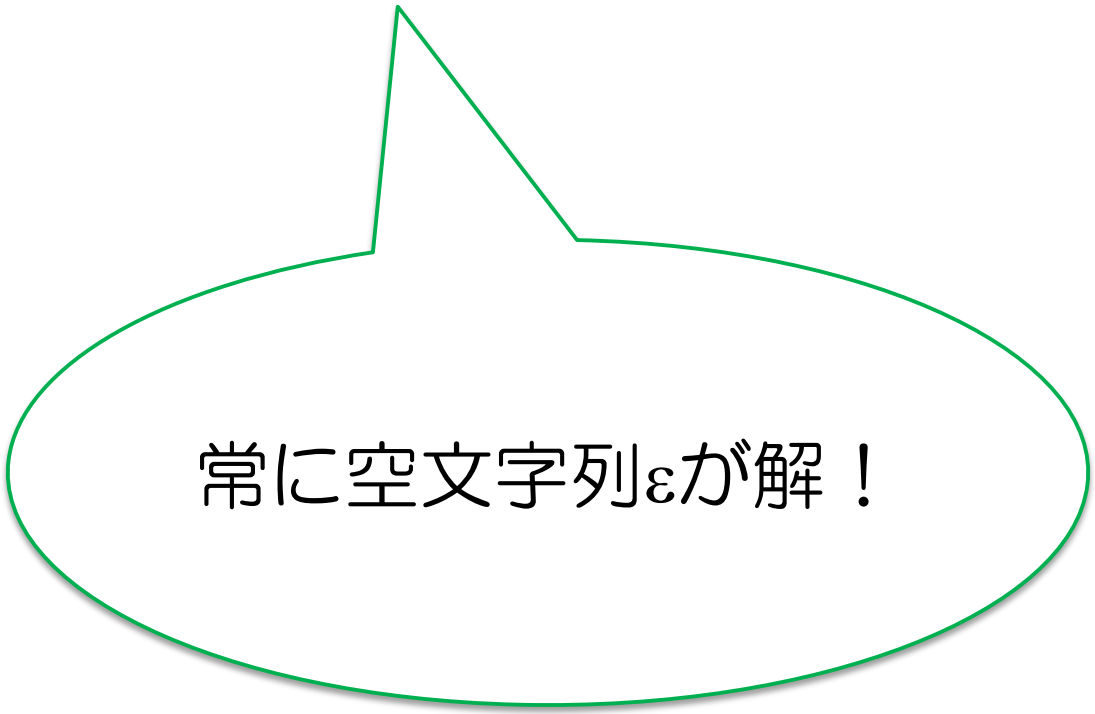
テキスト T を生成するSLP $\mathbf{7}$ に対して, Tの最長重複なし繰り返し部分文字列を $O(n^6 \log n)$時間で計算できる.

まとめと今後の課題

- 与えられたSLP圧縮テキストから，特徴的な部分文字列を多項式時間で発見するアルゴリズムを提案した.
  - 圧縮テキストを陽に展開するあらゆるアルゴリズムは指数時間を要する.
- 異なるタイプの特徴的な部分文字列を効率よく発見できるだろうか？
  - Squares (xx という形の部分文字列)
  - Cubes (xxx という形の部分文字列)
  - Runs (x^k という形の部分文字列．ただし $k \geq 2$)

最頻出部分文字列発見

- 入力: テキスト T を生成するSLP $\mathcal{T}$
- 出力: Tの**最頻出部分文字列**

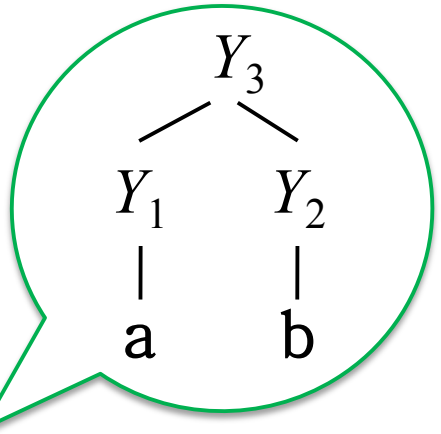


最頻出部分文字列発見

- 入力: テキスト T を生成するSLP $\mathcal{T}$
- 出力: Tの長さ２の**最頻出部分文字列**

最頻出部分文字列発見アルゴリズム

$|\Sigma|^2$ 個の長さ2の部分文字列



Input: SLP $\mathcal{T}$

Output: MFS of text T

foreach substring P of T of length 2 **do**

construct an SLP $\mathcal{P}$ which generates substring P;

compute num. of occurrences of P in T;

return substring of maximum num. of occurrences;

補題４

任意の２変数 X_i と Y_j に対して，X_i 中の Y_j の出現回数を $O(n)$ 時間で計算できる.

ただし，$O(n^2)$ 時間の前処理が必要.

最頻出部分文字列発見

定理４

テキスト T を生成するSLP $\mathcal{T}$ に対して, Tの長さ２の最頻出部分文字列を$O(|\Sigma|^2 n^2)$時間で計算できる.

最頻出重複なし部分文字列発見

- 入力: テキスト T を生成するSLP $\mathcal{T}$
- 出力: Tの長さ２の**最頻出重複なし部分文字列**

例

$T =$ aaaaababab

長さ２の最頻出
部分文字列はaa

長さ２の最頻出重複なし
部分文字列はab

最頻出重複なし部分文字列発見

定理５

テキスト T を生成するSLP $\mathbf{7}$ に対して, Tの長さ２の最頻出部分文字列を$O(n^4 \log n)$時間で計算できる.

入力圧縮パタンの最大伸張パタン発見

- 入力: テキスト T を生成するSLP $\mathcal{T}$ と
  パタン P を導出するSLP $\mathcal{P}$
- 出力: 以下を満たす T の部分文字列 $\alpha P \beta$
  - テキスト T 中において，α は常に P の直前に出現し，β は常に P の直後に出現する．
  - α と β は最長．

例

$T = \mathrm{bbaabaabbbaabb}$

$P = \mathrm{ab}$

$\alpha = \mathrm{ba}$

$\beta = \varepsilon$

入力圧縮パタンの最大伸張パタン発見

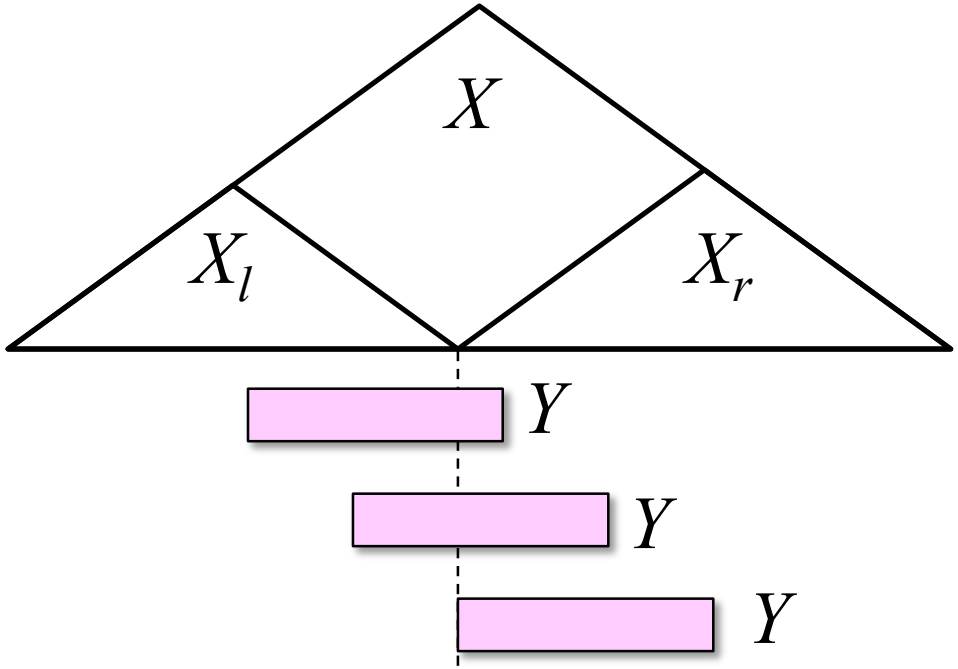
- 最大伸張パタンの応用例
  - ブログスパム発見 [Narisawa et al. 2007]
  - 最頻出部分文字列の最大伸張パタン発見

入力圧縮パタンの最大伸張パタン発見

定理６

テキスト T を生成するSLP $\mathcal{T}$ とパタン P を導出するSLP $\mathcal{P}$ に対して，T における P の最大伸張パタンを $O(n^4 \log n)$ 時間で計算できる.

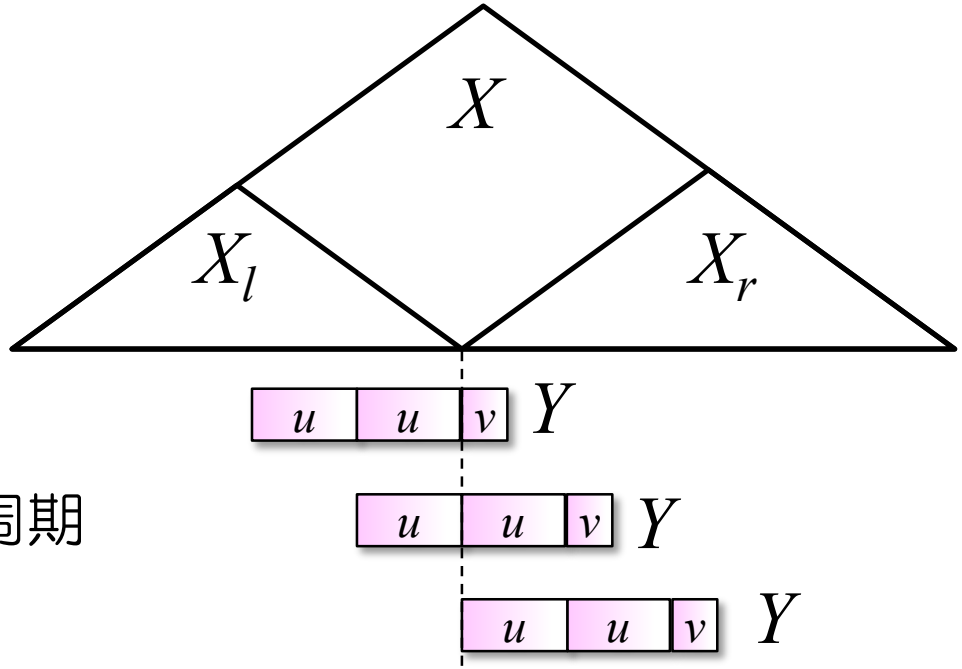
“串刺し”補題 [1/2]



補題５ [Miyazaki et al. 1997]

任意の変数 $X = X_l X_r$ と Y に対して，X の境界に触れる Y の出現は１つの等差数列をなす．

“串刺し”補題 [2/2]

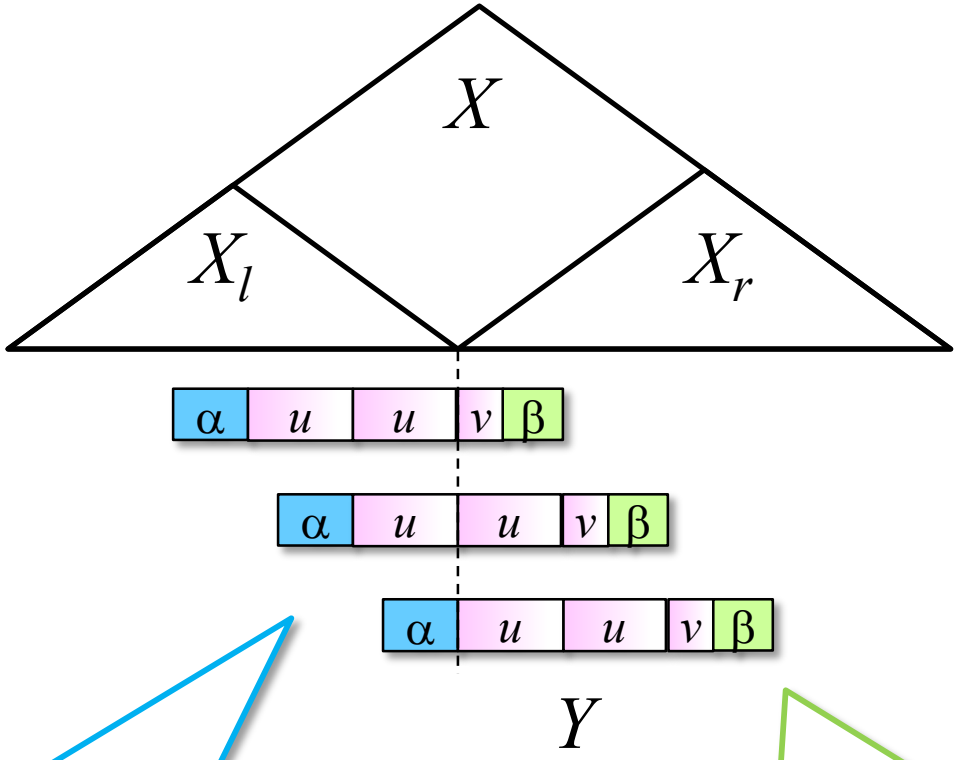


$|u|$ は Y の最小周期
v は u の接尾辞

補題５ [Miyazaki et al. 1997]

（つづき）
等差数列の要素数が３以上のとき，
等差数列の交差は Y の最小周期である．

左右の最大伸張



左側の最大伸張 α は u の接尾辞である.

右側の最大伸張 β は $uv[|v|: |uv|]$ の接頭辞である.