CPM2013 2013/6/19

A Succinct Grammar Compression

Yasuo Tabei[1],
Yoshimasa Takabatake[2],
Hiroshi Sakamoto[2]

1. Japan Science and Technology Agency
2. Kyusyu Institute of Technology

Straight Line Program (SLP)

- Canonical form of a CFG deriving a single string
- Every production rule satisfies
  - Right hand side of a digram
  - Subscript of the left symbol is larger than subscrips of the right symbols, i.e., $X_k \rightarrow X_i X_j$ (k>i,j)

Example:

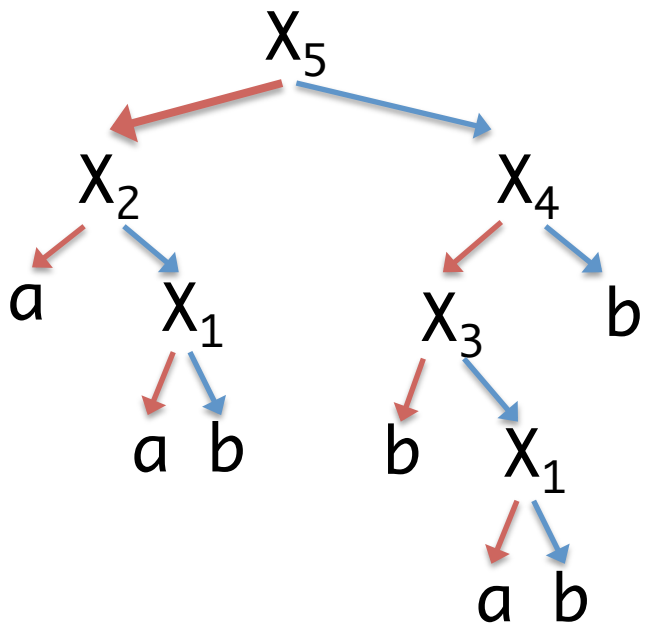
$X_1 \rightarrow ab$

$X_2 \rightarrow aX_1$

$X_3 \rightarrow bX_1$

$X_4 \rightarrow X_3 b$

$X_5 \rightarrow X_2 X_4$



Grammar compression

- Builds up an SLP from a given string
- Two crucial data structures to access a production rule

  $X_k \rightarrow X_iX_j$

  1. Dictionary (Array) : Given X_k, return X_iX_j
  2. Reverse dictionary (Hash table) : Given X_iX_j, return X_k if $X_k \rightarrow X_iX_j$ is registered in the dictionary

$X_1 \rightarrow ab$

$X_2 \rightarrow aX_1$

$X_3 \rightarrow bX_1$

$X_4 \rightarrow X_3b$

$X_5 \rightarrow X_2X_4$

	1	2	3	4	5	6	7	8	9	10
A	a	b	a	X_1	b	X_1	X_3	b	X_1	X_3

Access $X_k \rightarrow$ A[2k-2]A[2k-1]

Space : 2nlogn bits (n : #variables)

Three open problems about an optimal encoding of an SLP

1. An nontrivial information theoretic lower bound for encoding an SLP
2. Optimal encoding of an SLP
   - Standard array : 2nlogn bits (n:#variables)
   - Present an encoding asymptotically equivalent to the lower bound, while supporting fast random access
3. Space-efficient data structure for reverse dictionary
   - Hash table uses O(nlog(n)) bits
   - Present a data structure of 2nlogn(1+o(1)) bits

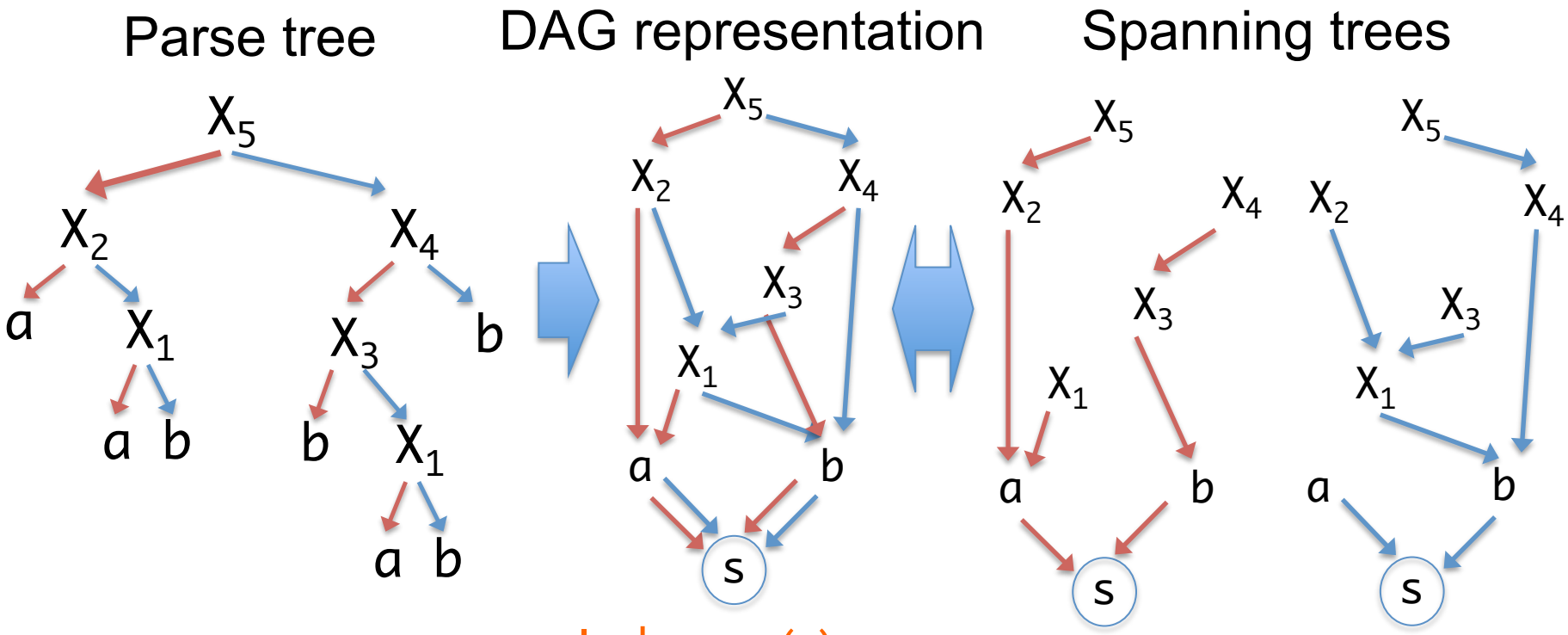
An information theoretic lower bound for representing an SLP

An information theoretic lower bound for representing an SLP of n variables : logn! + 2n + o(n) bits

- Use two techniques for the proof
  1. Spanning tree decomposition for representing an SLP as two ordered trees
  2. Right most expansion for completely enumerating ordered trees
- First introduce these two techniques, and then show a sketch of the proof

Spanning tree decomposition [SPIRE11]

- Any SLP can be represented as left and right spanning trees.

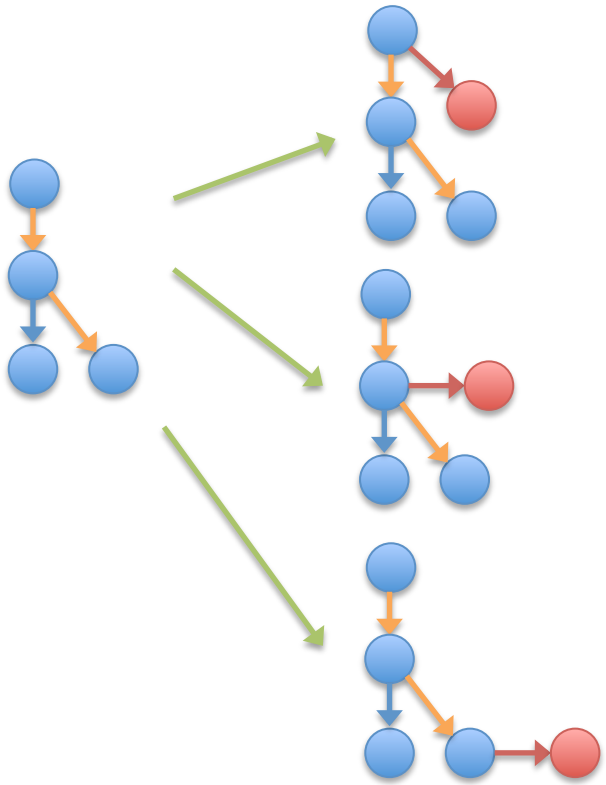


Right most expansion [KDD02,SDM02]

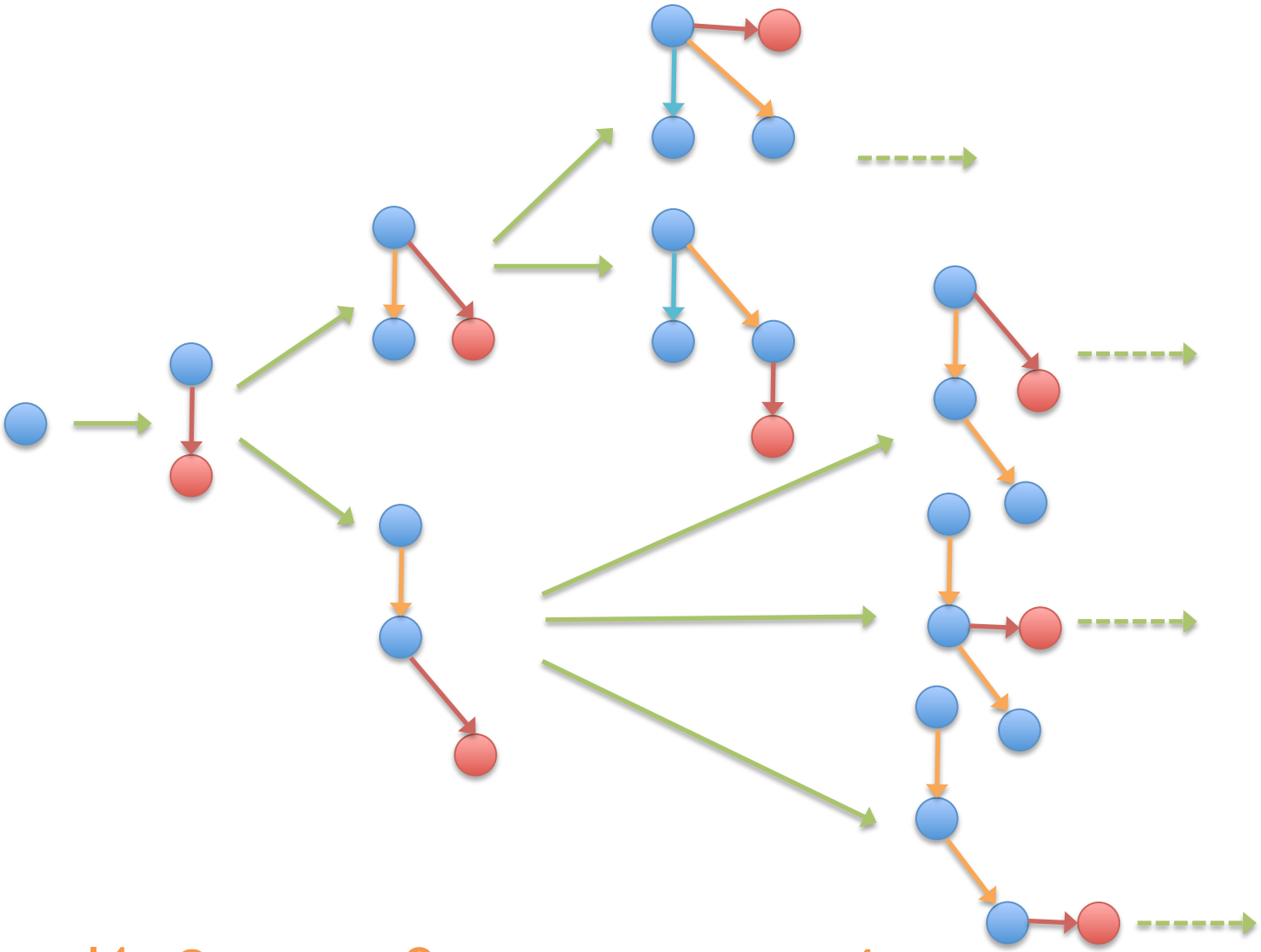
- Build trees of (m+1) nodes from a tree of m nodes
  - Add a node to the nodes on the right most path

Example :

→ : right most path



Search space : All trees can be enumerated by applying the right most expansion, recursively



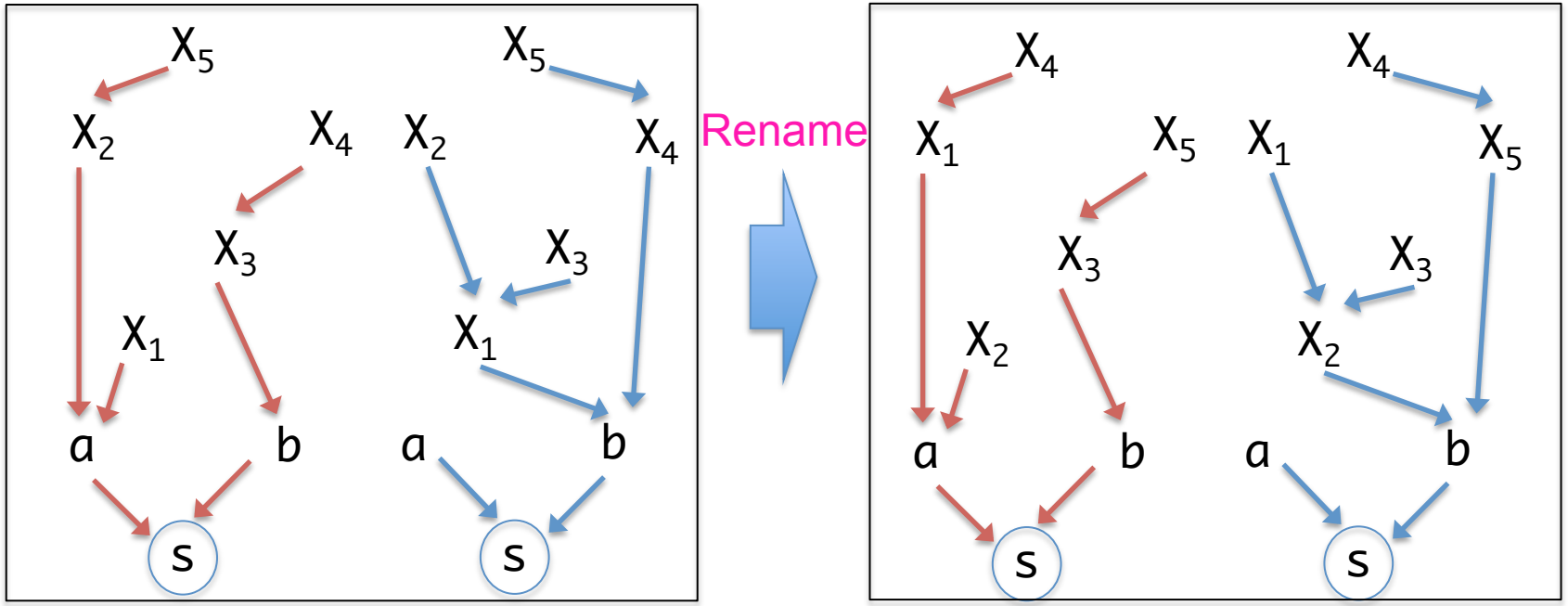
Sketch of the proof (detail)

- Basic idea: Consider a super set S(n) of DAG(n) without the restriction of the in-degree 2σ of the sink, and count |S(n)| by the induction
  - Get $|S(n)|/n^{\sigma} \leq |DAG(n)| \leq |S(n)|$
- Decompose S(n) into the left and right trees by the spanning tree decomposition
- Count the number of left trees and the right trees of (n +1) nodes by induction
  - Apply the right most expansion to the left tree
  - $|S(n)| = C_n(n-1)!$ where $C_n \simeq 2^{2n}n^{-3/2}$
- Get the information-theoretic minimum bits for representing G∈DAG(n) : logn!+2n+o(n) bits

An optimal encoding of an SLP

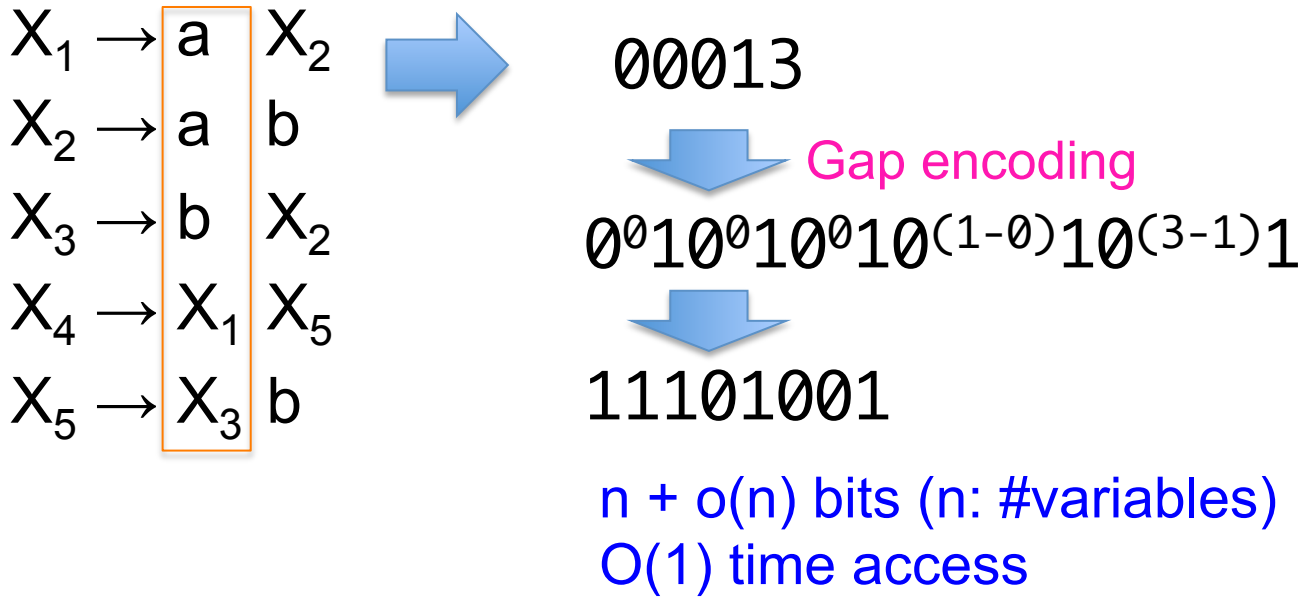
An optimal encoding of an SLP

- Basic idea : Encode the left and right symbols of the right hand side of the production rules, respectively
- Rename the variables by traversing the left tree in the breadth first manner



Encoding the left symbols

- Left symbols are monotonically increasing
- Apply gap encoding to the left symbols
- Use rank/select dictionary for O(1)-time access



n + o(n) bits (n: #variables)
O(1) time access

Encoding the right symbols D (detail)

- Extract subarrays s_i of monotonically increasing and decreasing elements from D
- Use two integer arrays D_ρ, D_π and two bit arrays B,b
- $2n \log \rho(1 + o(1))$ $\rho << \sqrt{n}$ bits, and $O(\log \log \rho)$ access time

$$s_1 = \{2, 3, 4\}, s_2 = \{1, 5\}$$

index	1	2	3	4	5
D	2	0	2	5	0
D_ρ	2	1	1	1	2
D_π	1	2	2	1	1

B = 110010010001

b = 01

- s_i:indices of increasing/decreasing elements
- $D_\rho[i]$ indicates which s_j contains D[i]
- D_π is the sorted D_ρ w.r.t. $D[i]$ of the pairs $(D[i], D_\rho[i])$
- B is the gap encoding of the sorted D
- b[i] indicates s_i is increasing or decreasing

Space-efficient data structure for reverse dictionary

Space-efficient data structure for reverse dictionary

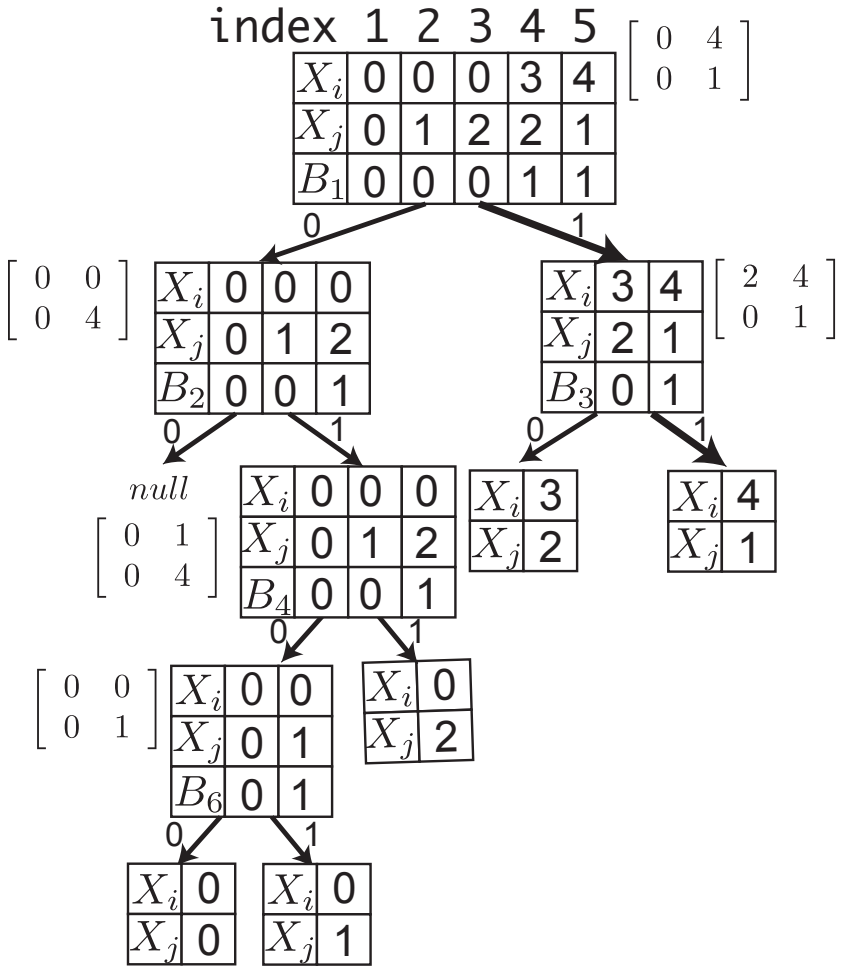
- Recap : Reverse dictionary D^{-1}

$$D^{-1}(X_i, X_j) = \begin{cases} X_k, & \text{if } X_k \rightarrow X_i X_j \text{ is in the dictionary D} \\ X_{n+1}, & \text{otherwise.} \end{cases}$$

- Basic idea: Build a wavelet tree (WT) consisting of right symbols X_iX_j, and simulate reverse dictionary on the WT.
- Access and update time: O(logn), Space: 2nlogn(1+o(1)) bits

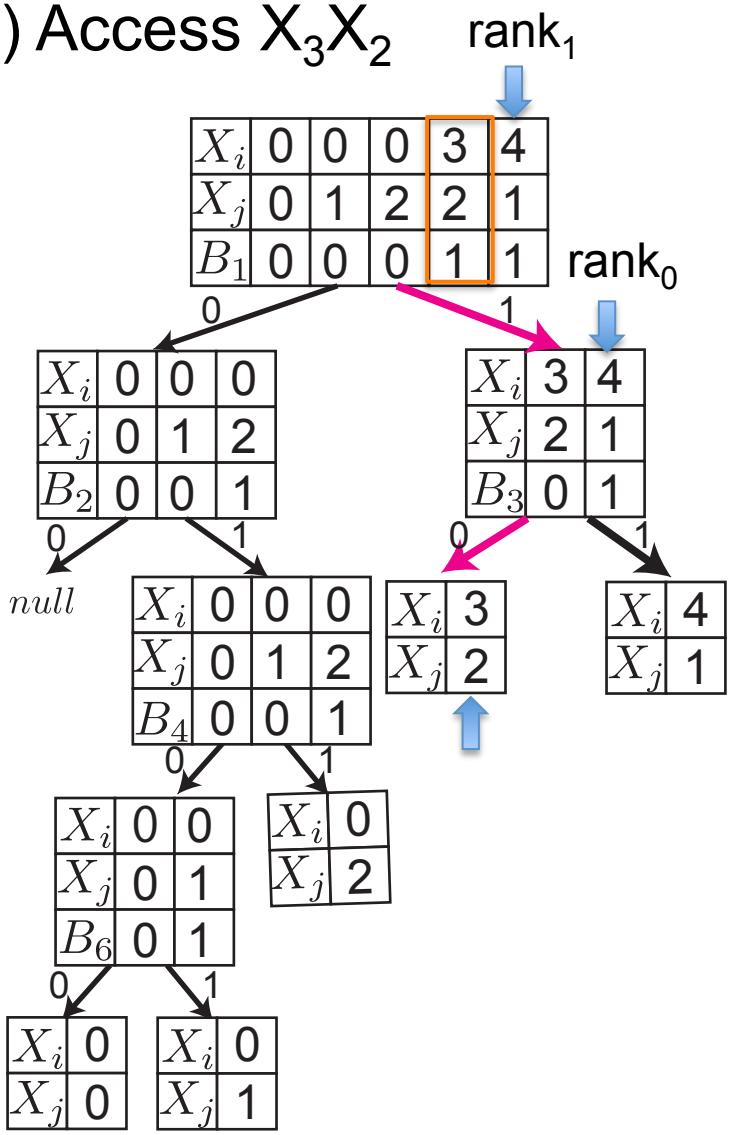
Build WT from digrams : The range of a digram is split into the higher half (right) and the lower half (left)

$$X_1 \rightarrow ab$$
$$X_2 \rightarrow aX_1$$
$$X_3 \rightarrow bX_2$$
$$X_4 \rightarrow X_3X_2$$
$$X_5 \rightarrow X_4X_1$$



Accessing $X_k \rightarrow XiXj$

EX) Access X_3X_2



- Start from the root B_1 as i = n (#variables)
- Apply $rank_1(B_j,i)$ for the right child and $rank_0(B_j,i)$ for the left child
- After reaching a leaf, go up to the root by applying select operation
- Solution: $X_k = select_{0/1}(B_1,i)$

Conclusion

- Three open problems related to an optimal encoding of an SLP
  1. an information theoretic lower bound
  2. an optimal encoding
  3. a dynamic data structure for reverse dictionary
- Novel challenges : Developing succinct data structures of an SLP for various applications

  e.g., self-index, pattern mining, q-gram mining etc