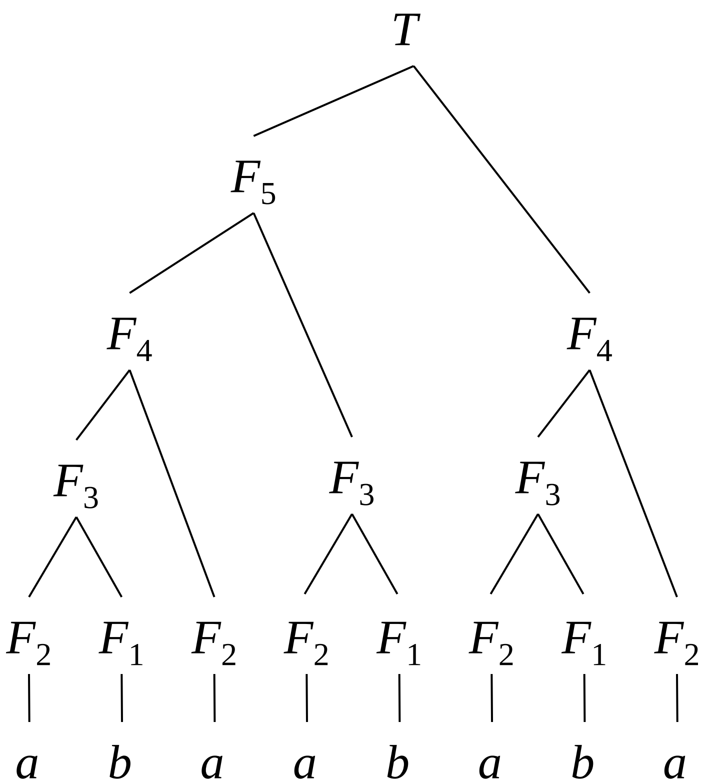
木構造の文法圧縮

国立情報学研究所
定兼　邦彦

文法圧縮

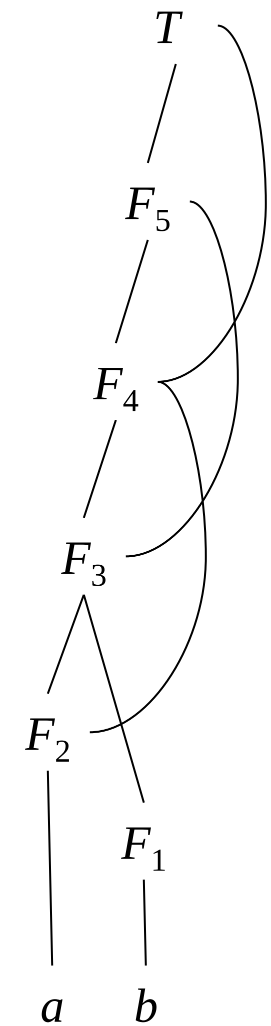
- 文字列をCFG (文脈自由文法) で表現
  - 入力文字列のみを生成する文法



文字列を表現する木

$$
\begin{aligned}
F_1 &\rightarrow b \\
F_2 &\rightarrow a \\
F_3 &\rightarrow F_2\ F_1 \\
F_4 &\rightarrow F_3\ F_2 \\
F_5 &\rightarrow F_4\ F_3 \\
T &\rightarrow F_5\ F_4
\end{aligned}
$$

文字列を表現する文法



DAG表現

Straight-Line Programs

- 本研究ではstraight-line program (SLP) に限定
  - $X_i \to c \ \ (c \in A)$
  - $X_i \to X_l\, X_r \ \ (l, r < i)$
  - 入力文字列 $T[1..N]$ は X_n で表わされる
- 1つの文字列を表わすCFGはSLPに変換可
- 最短のSLPを求める (n の最小化) のはNP完全
  - O(log N) 近似は線形時間 O(N) で求まる
- n 個の規則のSLPは，O(n log N) 個の規則，高さ O(log N) のSLPに変換できる
- 本研究では，SLPは与えられているとする
  - (高さの制限は無い)

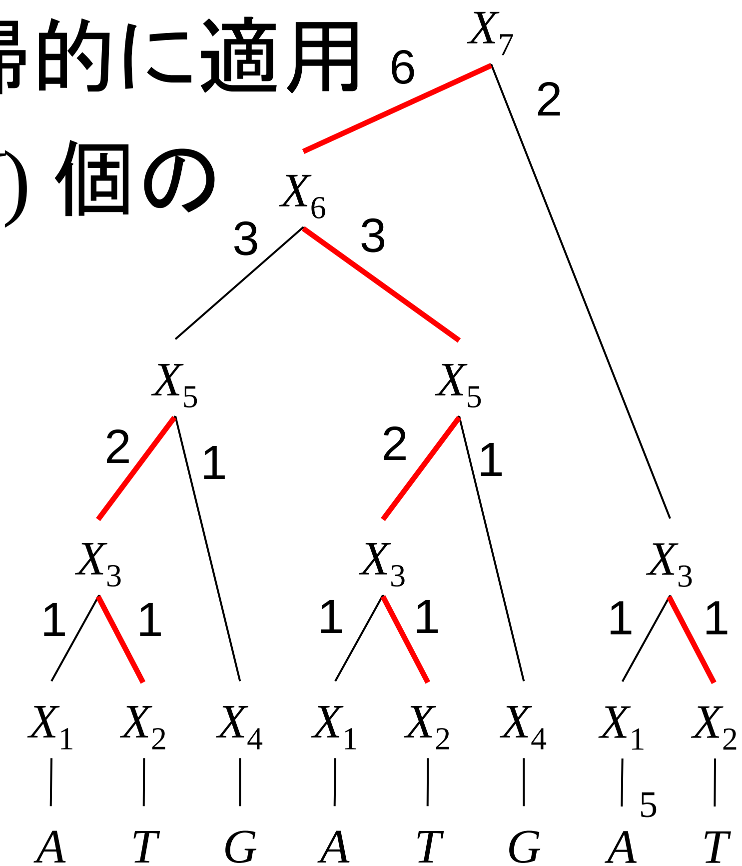
既存研究と本研究の結果

- Claude, Navarro MFCS09
  - 長さ m の部分文字列の復元: $O((m+h) \log n)$ 時間
  - パタン検索: $O((m(m+h)+h\ occ) \log n)$ 時間
  - (h: SLPの高さ, occ: パタンの出現回数)
  - サイズ: $O(n \log n) + n \log N$ bits
- 本研究
  - 長さ m の部分文字列の復元: $O(m + \log N)$ 時間
  - サイズ: $O(n \log N)$ bits
  - 編集距離kのパタン検索$O(n(\min\{mk, k^4+m\})+occ)$時間

Heavy Path Decomposition

[Harel, Tarjan 84]

- Heavy edge: 大きい部分木への枝
- Heavy path: 根からheavy edgeをたどるパス
- 木からheavy pathを除き，再帰的に適用
- 根から葉へのパスは O(log N) 個の heavy pathで表現される
- 各パスで2分探索
  → O(log $N \cdot$ log n) time
- 各パスを別々に格納
  → O(n^2 log N) bit space



Interval-Biased Search

- i 番目の葉を探す場合，heavy path上で左右の部分木のサイズを元に2分探索

左部分木のサイズ (右端の文字の位置)

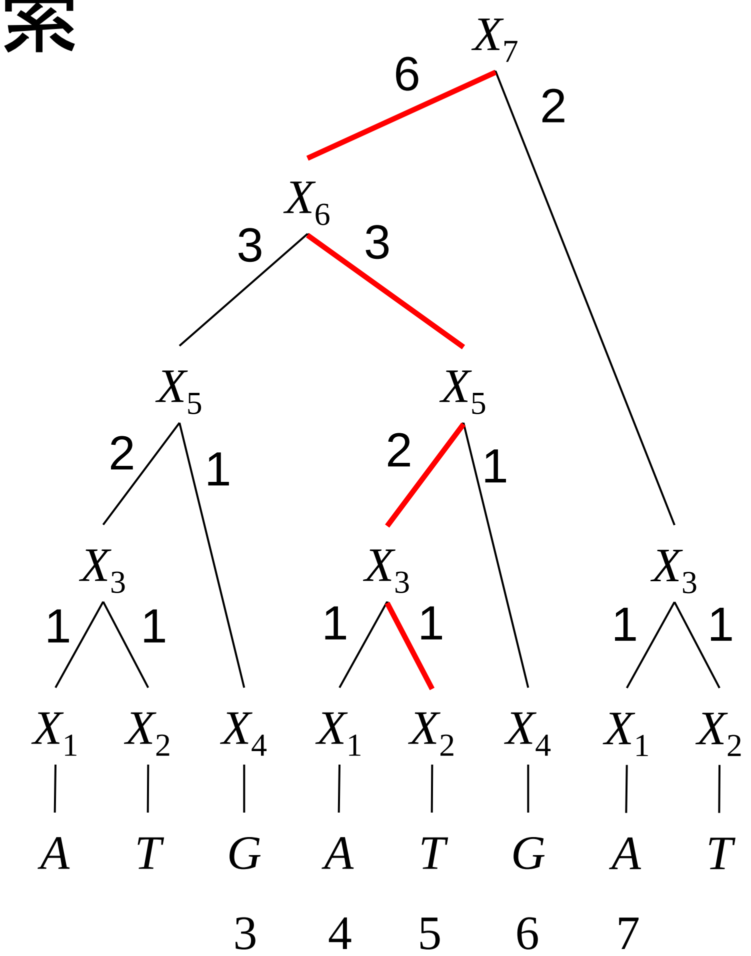
$X_7 - X_6 - X_3 - X_2$

3 4 5

右部分木のサイズ (左端の文字の位置)

$X_7 - X_5 - X_2$

7 6 5

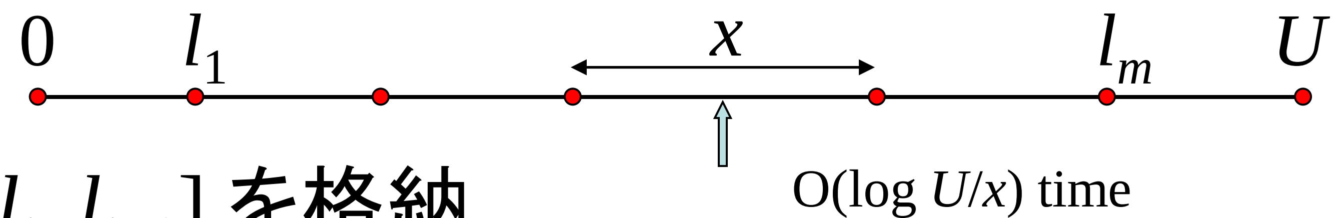


- 配列での2分探索
  $\rightarrow$ O(log n) time
- Interval-Biased search
  $\rightarrow$ O(log N/x) time
  (N は木のサイズ)
  (x は見つかった部分木のサイズ)

- 木全体のサイズ: N
- 1回目の探索でサイズ s_1 の部分木が見つかる
  - O(log N/s_1) time
- 2回目の探索でサイズ s_2 の部分木が見つかる
  - O(log s_1/s_2) time
- O(log N/s_1 + log s_1/s_2 + log s_2/s_3 + …)
  = O(log N) time

Interval-Biased Search Trees

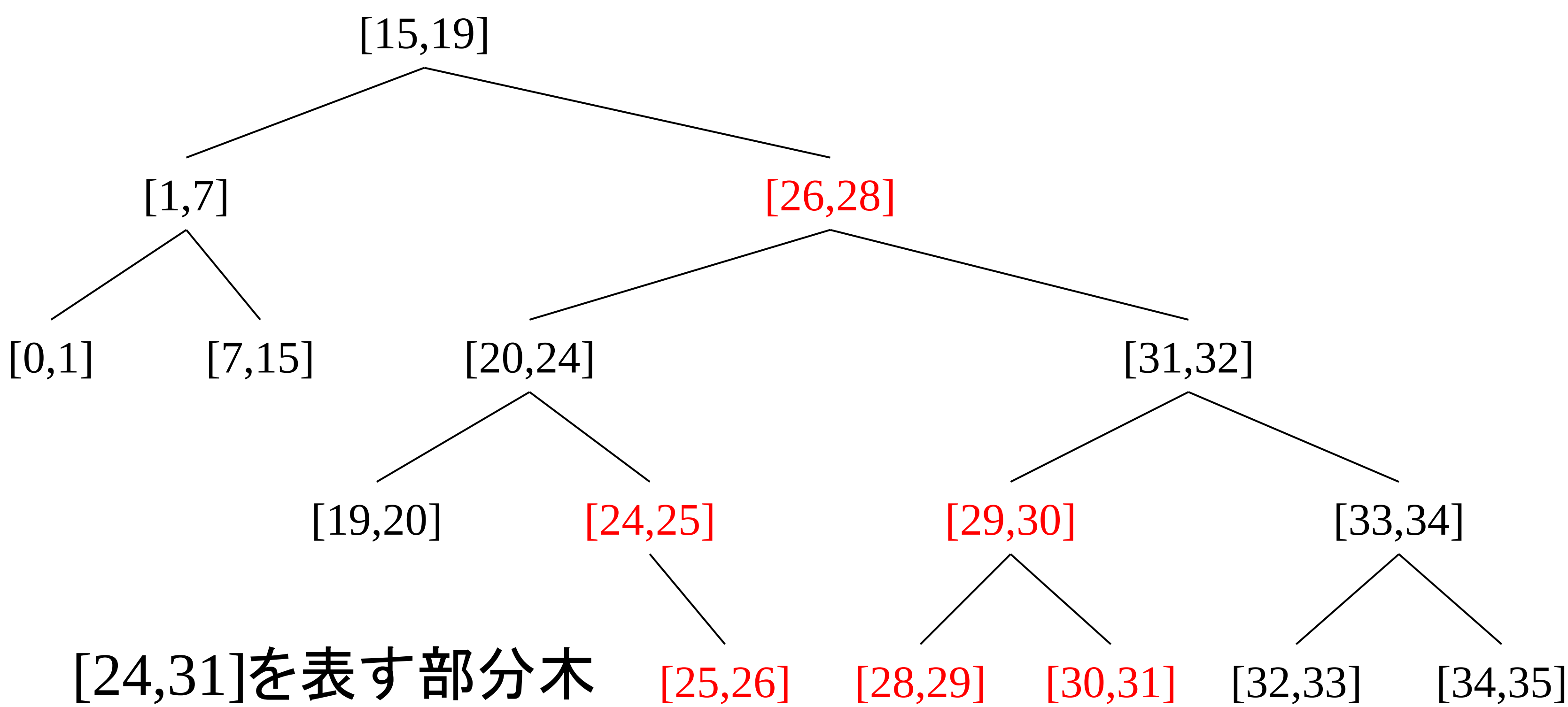
- $0 = l_0 \le l_1 \le l_2 \ldots \le\ l_m \le l_{m+1} = U$ のpredecessorを検索するための2分木



- 根は $[l_i, l_{i+1}]$ を格納
  (i は $(l_{m+1} - l_0)/2 \in [l_i, l_{i+1}]$ を満たすもの)
- 根の左部分木は $l_0 \le l_1 \le l_1\ \ldots \le\ l_i$ に対する木，右部分木は $l_{i+1}\ \ldots \le\ l_m \le l_{m+1}$に対する木
- 区間 $[l_i, l_{i+1}]$ の長さ $x = l_{i+1} - l_i$ が $\frac{U}{2^{j-1}} \le x \le \frac{U}{2^j}$ のとき，その区間は深さ j 以下の節点に格納される
  →検索時間は O(log U/x)

- $l_s \leq \ldots \leq l_t$ でのpredecessorを求める場合
長さ $x = l_t - l_s$ の区間は高さ $1 + \log x$ 以下の(2つの)部分木で表現されている
- 2つの部分木の根とそれらの最近共通祖先(lca)は定数時間で求まる (word RAMモデル)

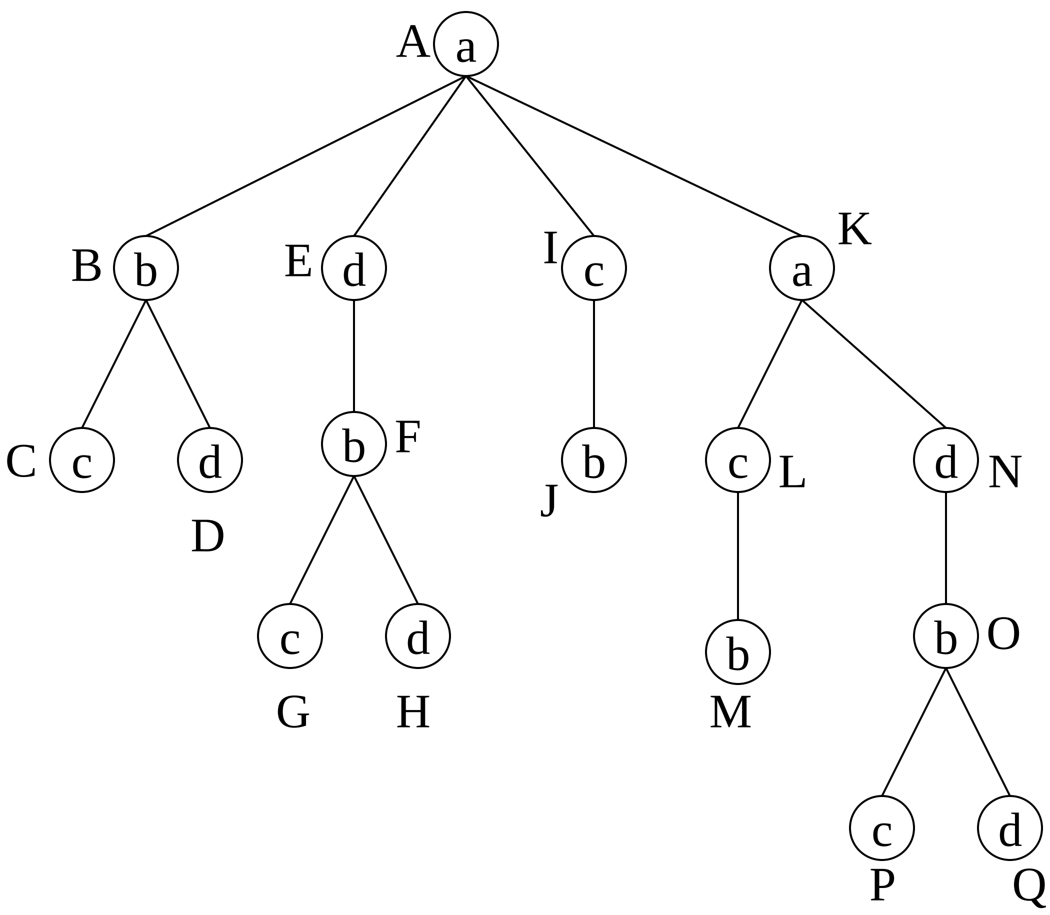
0 1 7 15 19 20 24 25 26 28 29 30 31 32 33 34 35

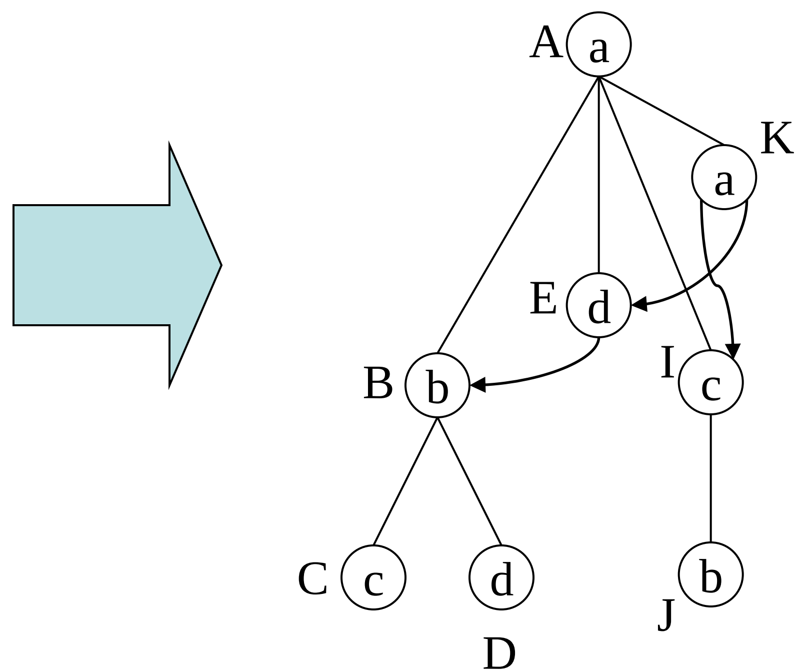


[24,31]を表す部分木

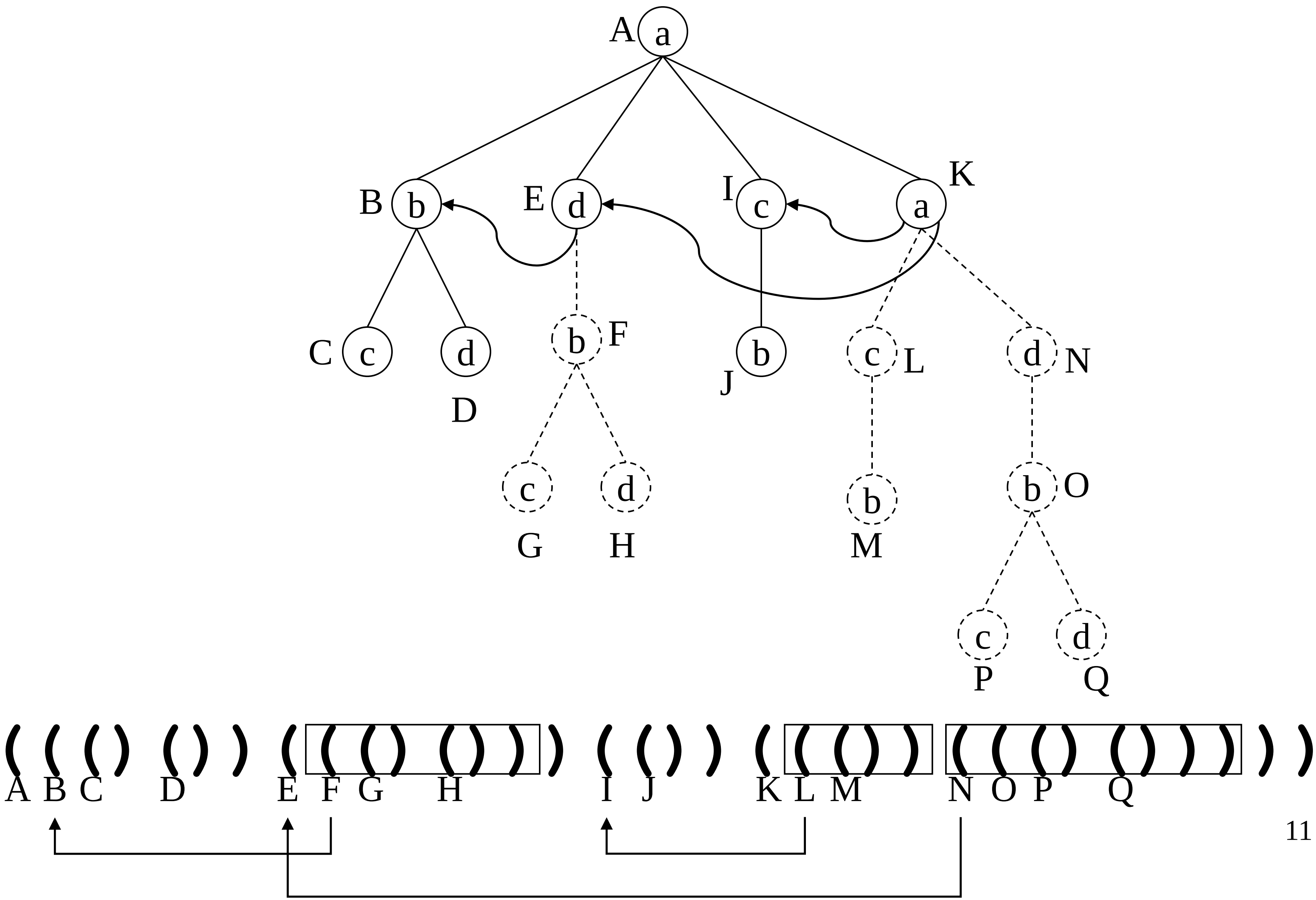
木構造の文法圧縮

- 順序木を，同じ部分木を共有することで圧縮する
- N 節点→ n 節点
- 木に対する各種問い合わせを O(log N) 時間





部分木を共有する場合のBP表現



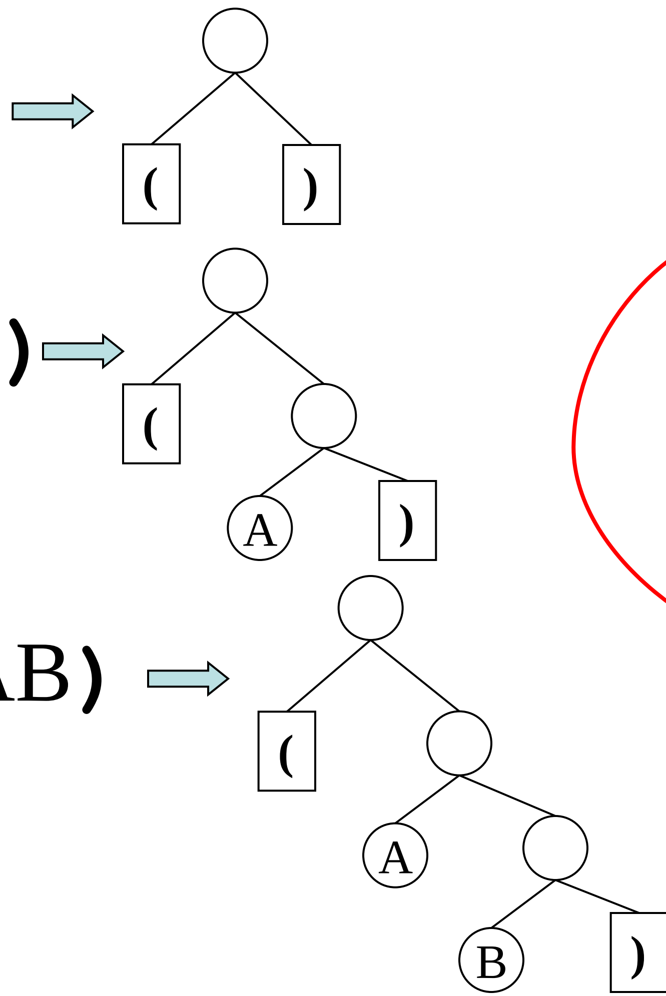
BPを表現するSLP

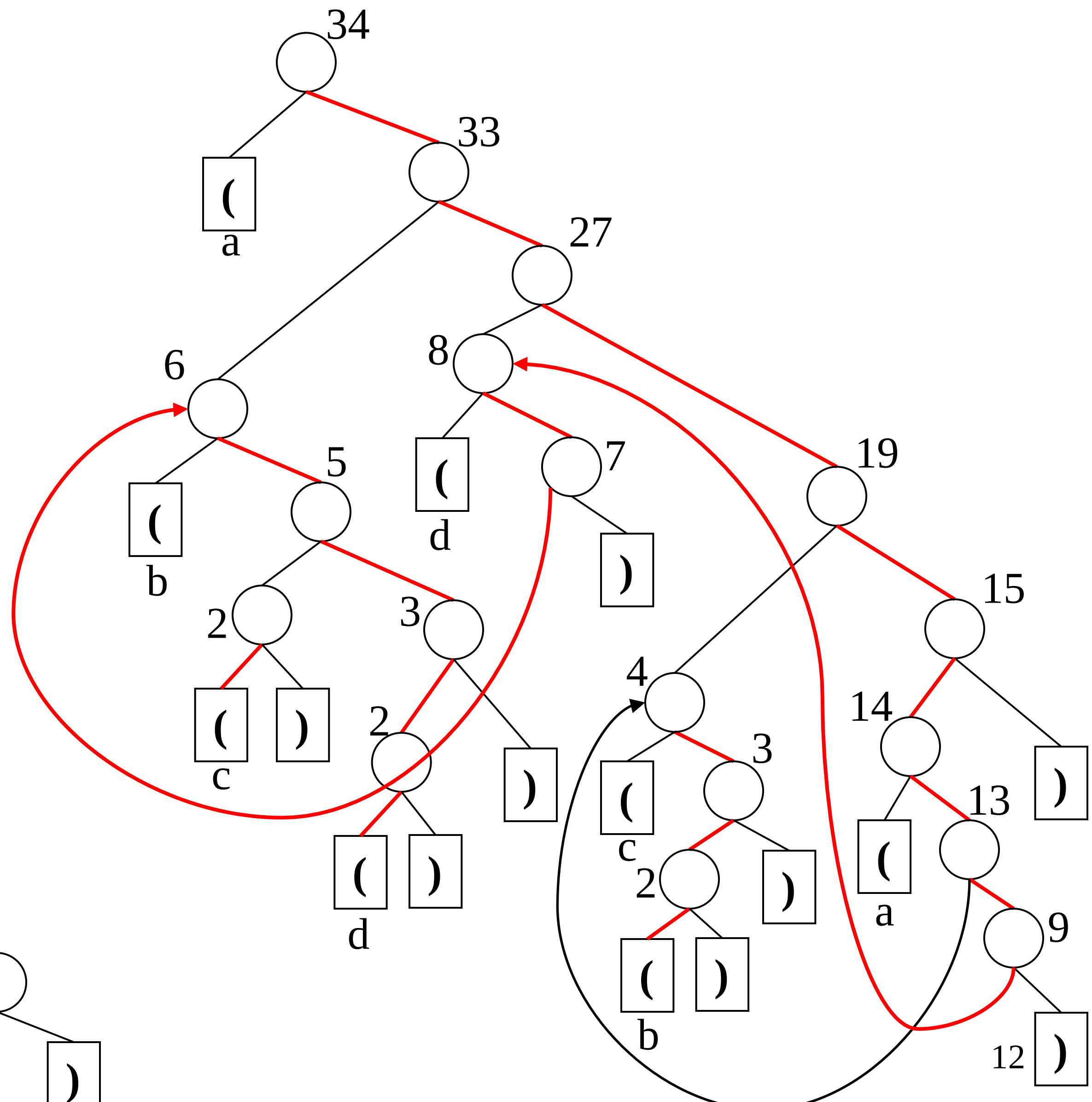
変換規則

()

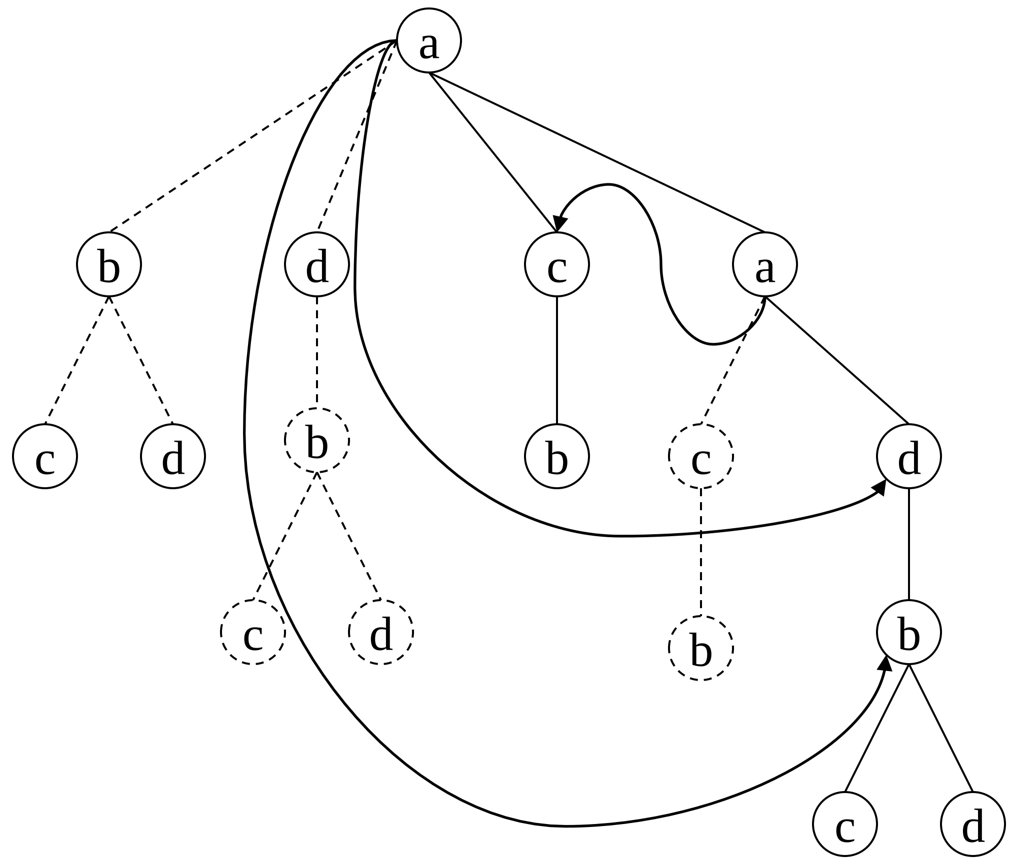
(A)

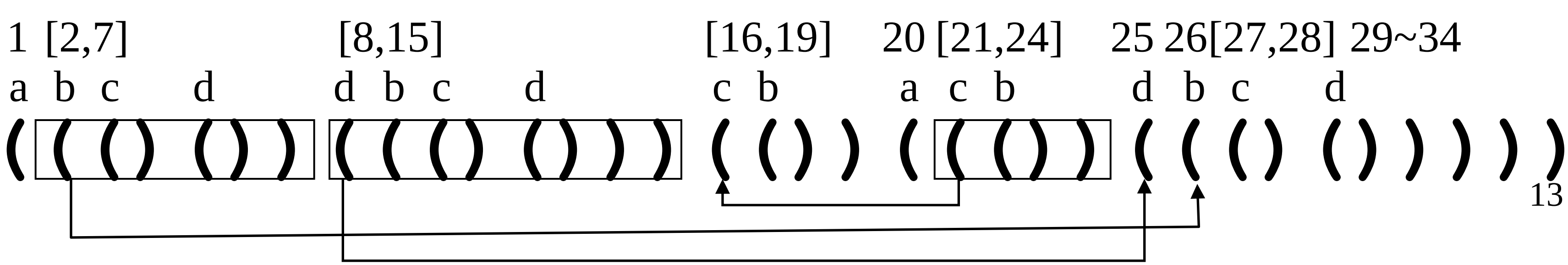
(AB)





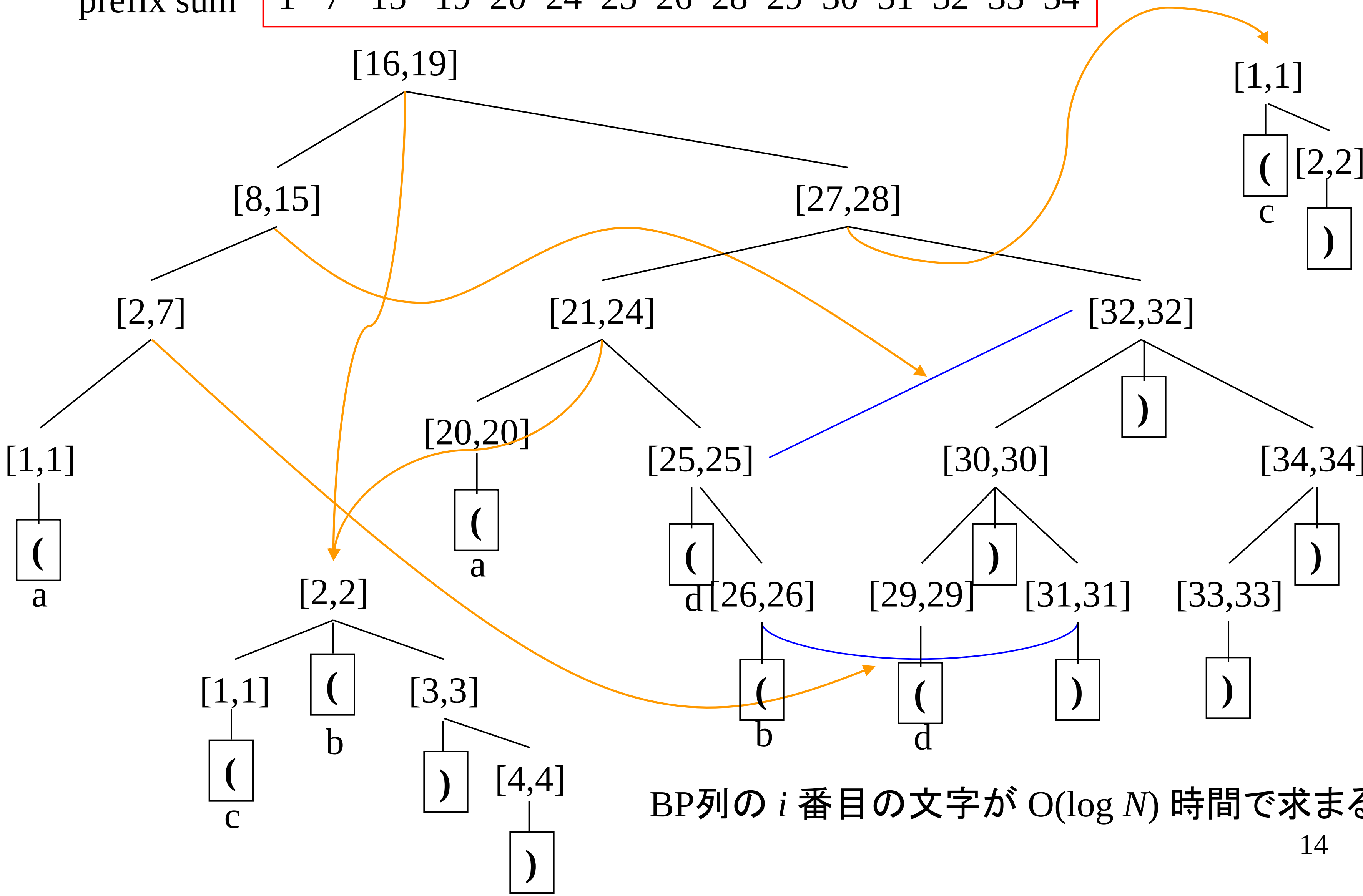
Heavy path decompositionに従いSLPを変更





Interval-Biased Search Tree

size	1	6	8	4	1	4	1	1	2	1	1	1	1	1	1
prefix sum	1	7	15	19	20	24	25	26	28	29	30	31	32	33	34



BP列の i 番目の文字が O(log N) 時間で求まる

Interval-Biased Search Treeを用いた区間最大最小木

- Interval-Biased Search Treeの構造のまま，区間最大最小木を構築
- 木の高さ: O(log N)
- *fwd_excess* の時間: O(log N/x)
  (x は見つかった部分木のサイズ)

((( ) ( )) ((( ) ( ))) (( )) ((( )) ((( ) ( )))))

1232321234343212321234323454543210