頻出部分文字列のマイニング

坪井祐太
日本IBM(株) 東京基礎研究所(TRL)
yutat@jp.ibm.com

概要

- 問題設定（頻出部分文字列問題）
- 3分割法によるアルゴリズム（提案手法）
- N-gram PrefixSpan
- 接尾辞木 による方法
- 計算機実験

背景

- N-gramモデル

$$P(w_N \mid w_1, \ldots, w_{N-1}) = \frac{C(w_1, \ldots, w_{N-1}, w_N)}{C(w_1, \ldots, w_{N-1})}$$

$C(w_1, \ldots, w_{N-1})$は単語列$w_1, \ldots, w_{N-1}$の頻度

- 可変長N-gramモデル (Ron 1996)
- 低頻度のN-gramは無視されることが多い

頻出部分文字列問題

- 頻出部分文字列問題とは、
  - 文字列 **s** 中で ある閾値 ξ 回以上出現する部分文字列を全てを並べ挙げる問題。
- 例:
  文字列 s=sakurasaku の頻出部分文字列(ξ =2)は以下の通り(カッコ内は頻度)。
  - a(3), ak(2), aku(2),
  - k(2), ku(2),
  - s(2), sa(2), sak(2), saku(2),
  - u(2)

頻出パターンマイニング

- 頻出する部分集合や部分構造をパターンとして高速に列挙。
- 頻出部分集合 マイニング
  - Apriori アルゴリズム(Agrawalら 1994)
- 部分構造マイニング
  - 部分系列マイニング(Agrawalら 1995, Peiら 2001)
  - 部分木マイニング（浅井ら 2002, Zaki 2002)
  - 部分グラフマイニング(猪口ら 2000, Yanら2002)

頻出部分系列マイニング問題（Sequential Pattern Mining Problem)

- 系列パターン問題(Agrawalら 1995)は、系列データベース中からある閾値以上出現する順序が保たれた要素列を見付ける問題
- 要素が連接している場合としていない場合を区別しない。
- 例:
  - sakurasaku
  - shibaraku

提案手法

- 頻出部分文字列の列挙アルゴリズム
  - 分割統治法 (3分割法)
  - 高速で、メモリ使用量が少ない
  - コンテクスト・ブラウジングが容易

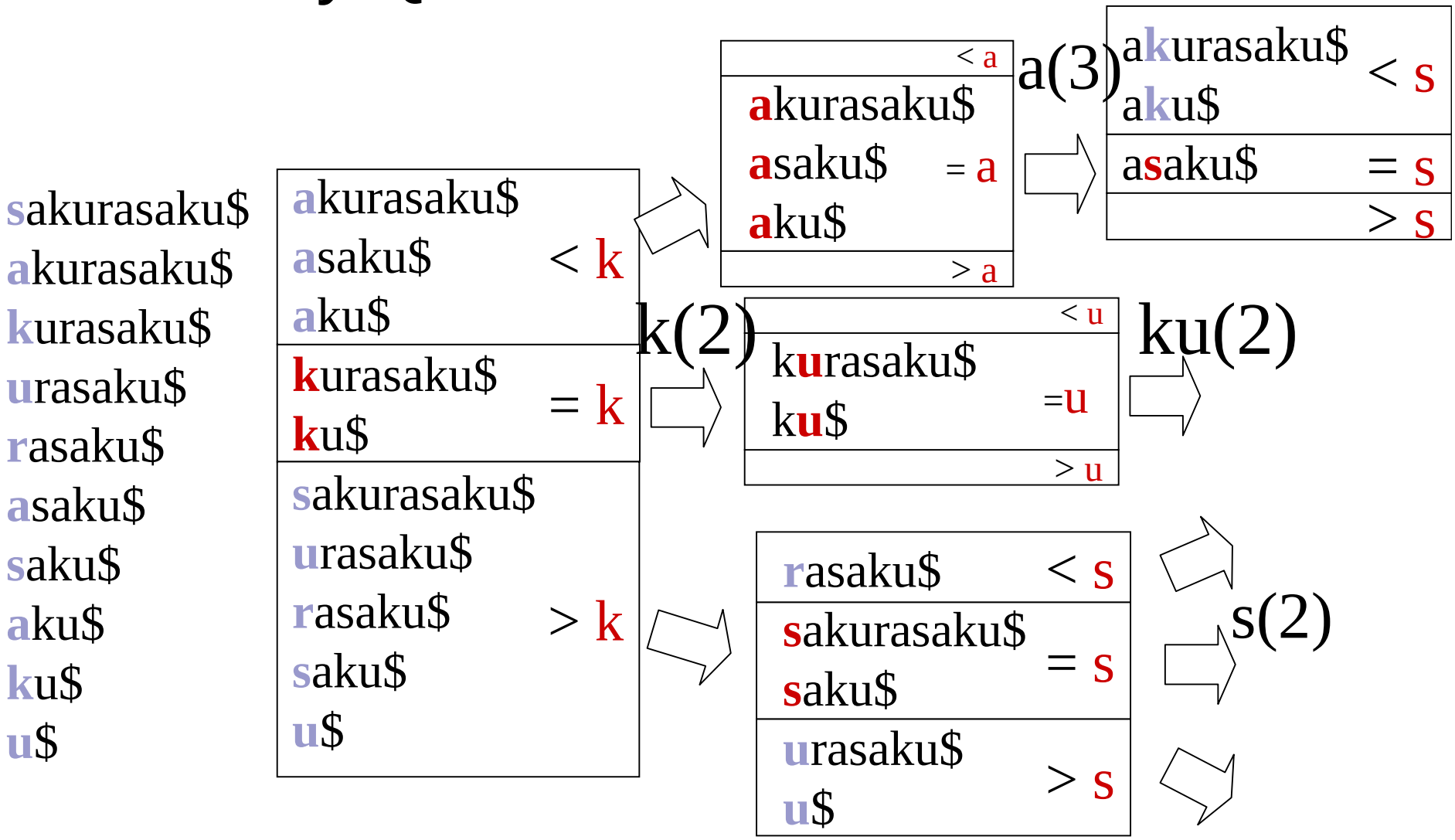
アルゴリズムの概要

- 入力文字列の全ての接尾辞を整列
  - 整列には3分割法を用いる
  - 頻出しない部分文字列を整列しない (枝刈り)

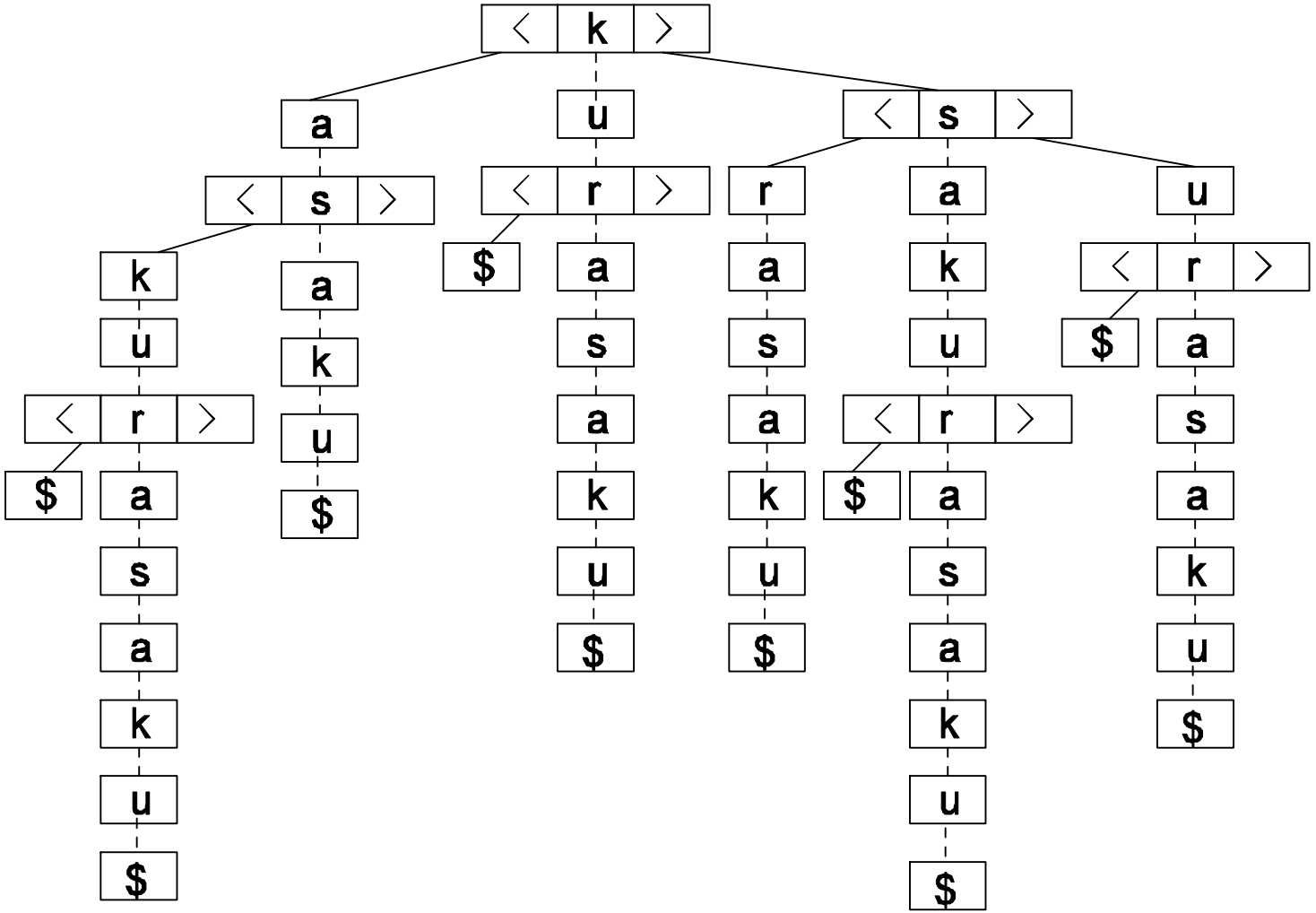
3分割法で整列(Multi-key Quicksort)

- Bentley and Sedgewick 1997
- 文字列のように複数要素をもつアイテムを効率的に整列するアルゴリズム
- 一文字目、二文字目、三文字目…と階層的に整列する
- 基準要素(pivot)より小さい、大きい、等しいアイテム集合に3分割する。

Multi-key Quicksortによる接尾辞の整列



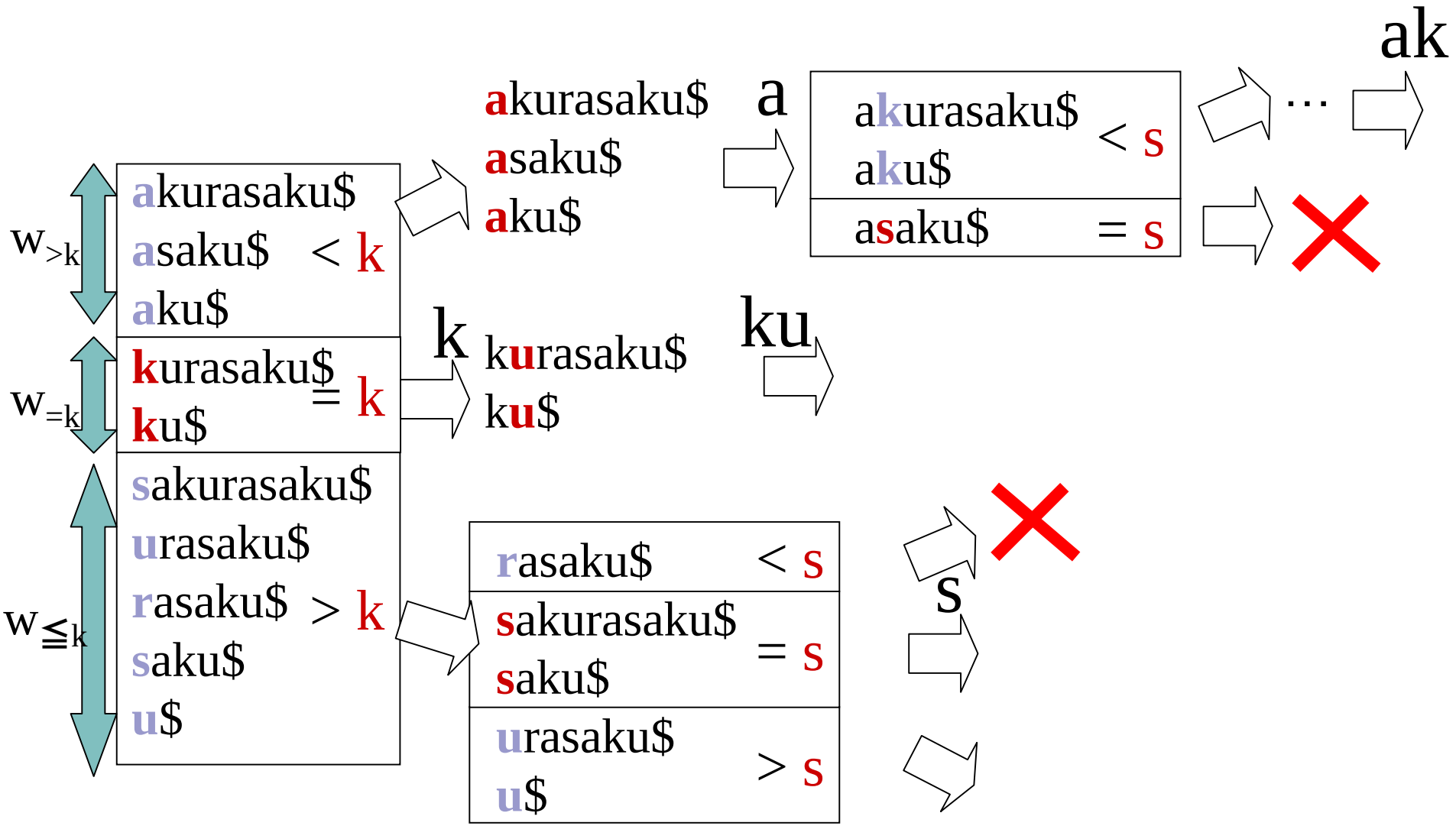
Multi-key Quicksortの探索木 (3分木)



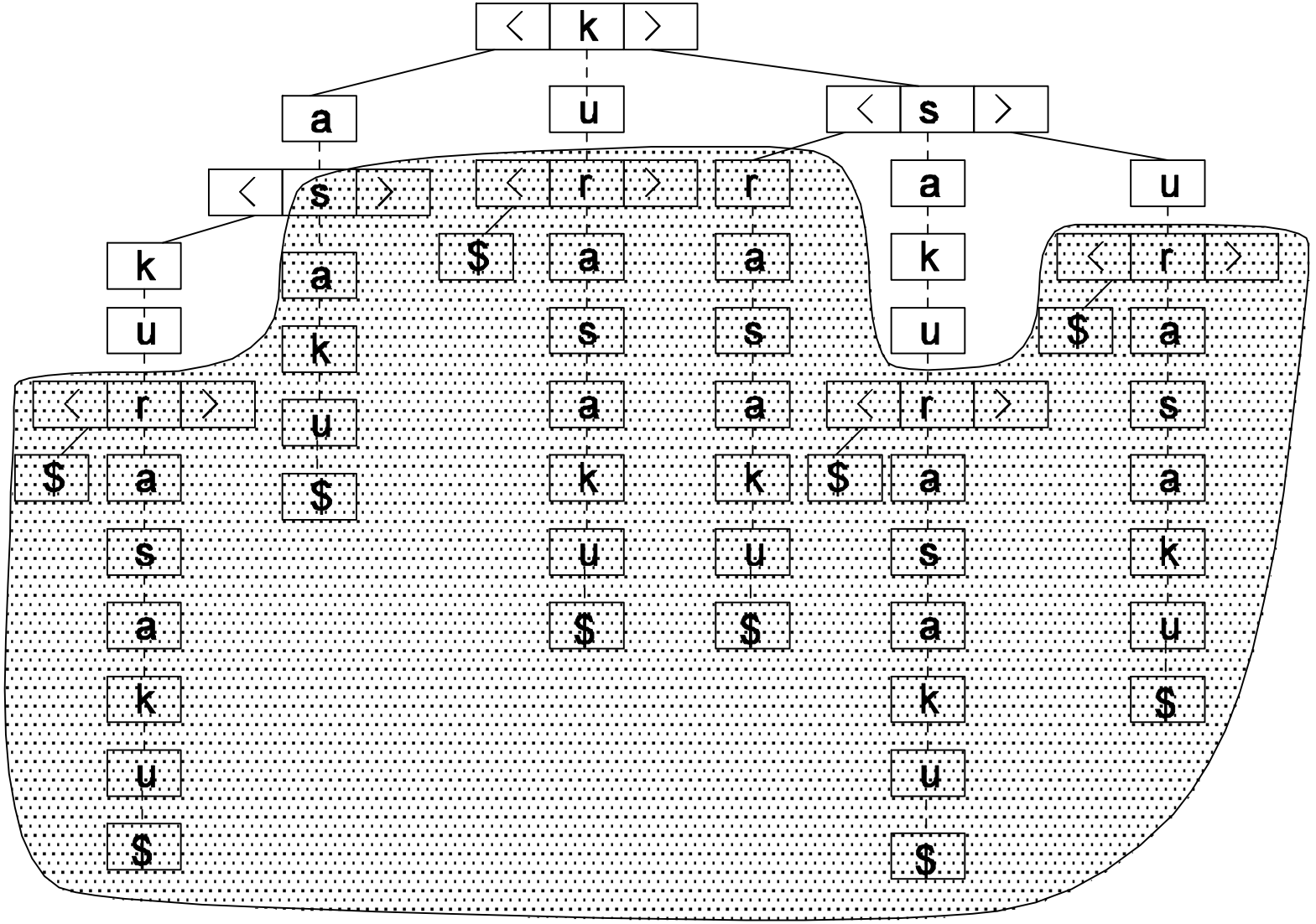
非頻出部分文字列の枝刈り

- ある部分文字列が頻出でないときその文字列を含む上位部分文字列も頻出ではない。

非頻出部分文字列の枝刈り (ξ =2)



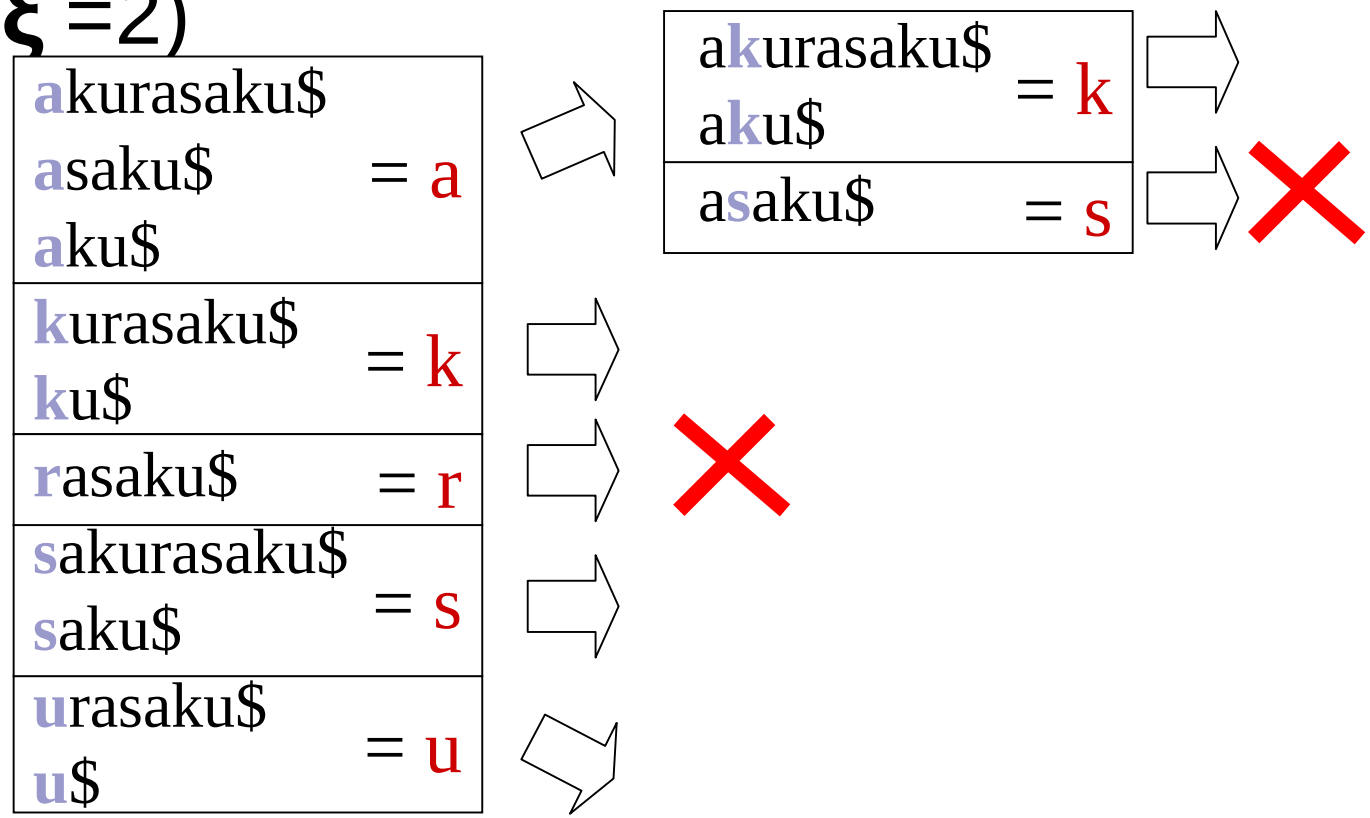
頻出部分文字列のみの列挙(枝刈り)



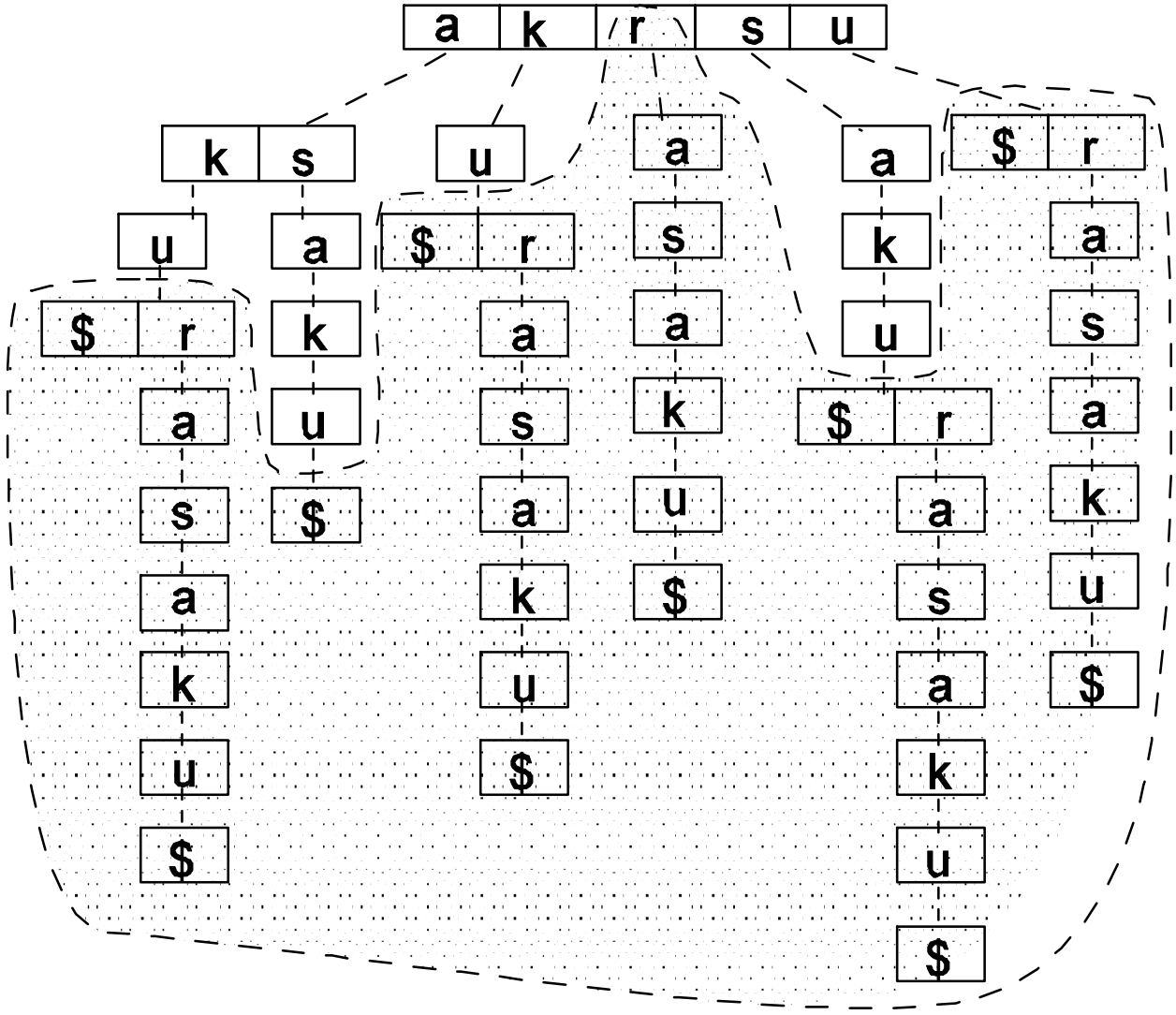
関連研究

- N-gram PrefixSpan（工藤ら 2002)

(ξ=2)

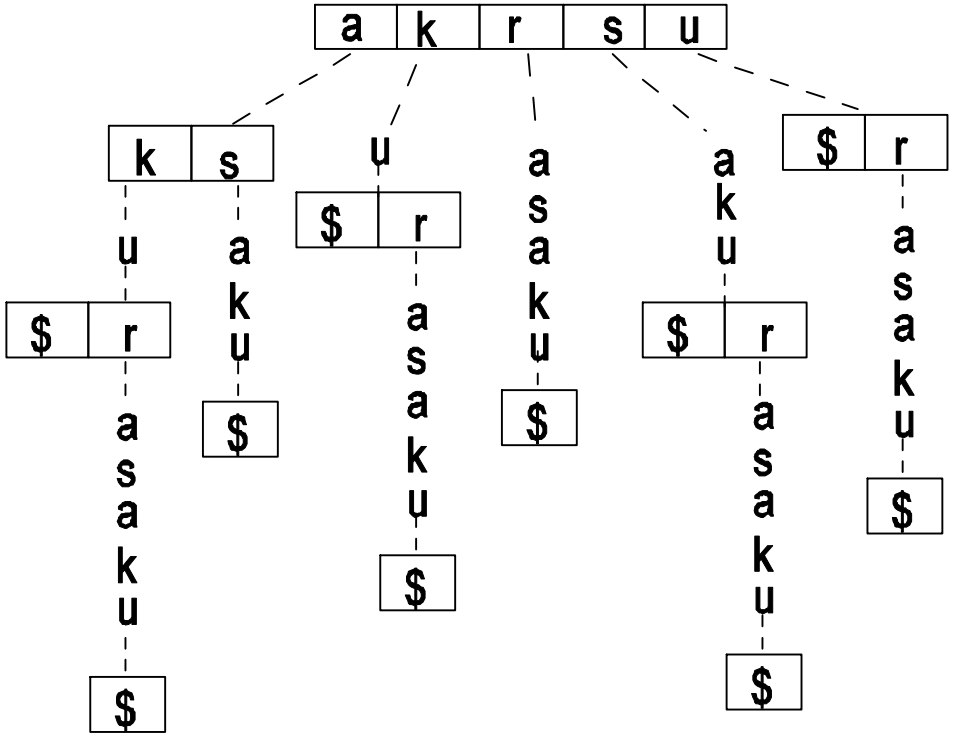


N-gram PrefixSpan の探索空間



接尾辞木(Suffix Tree)による方法

- 枝分かれの無いノードを圧縮したトライ（ノード数は最悪2n)
- 終端ノード数＝部分文字列の頻度
- 全てのノードをたどる事で頻出部分文字列を列挙できる
  → O(n)
- 線形時間で生成するオンライン・アルゴリズム（Ukkonen 1995)



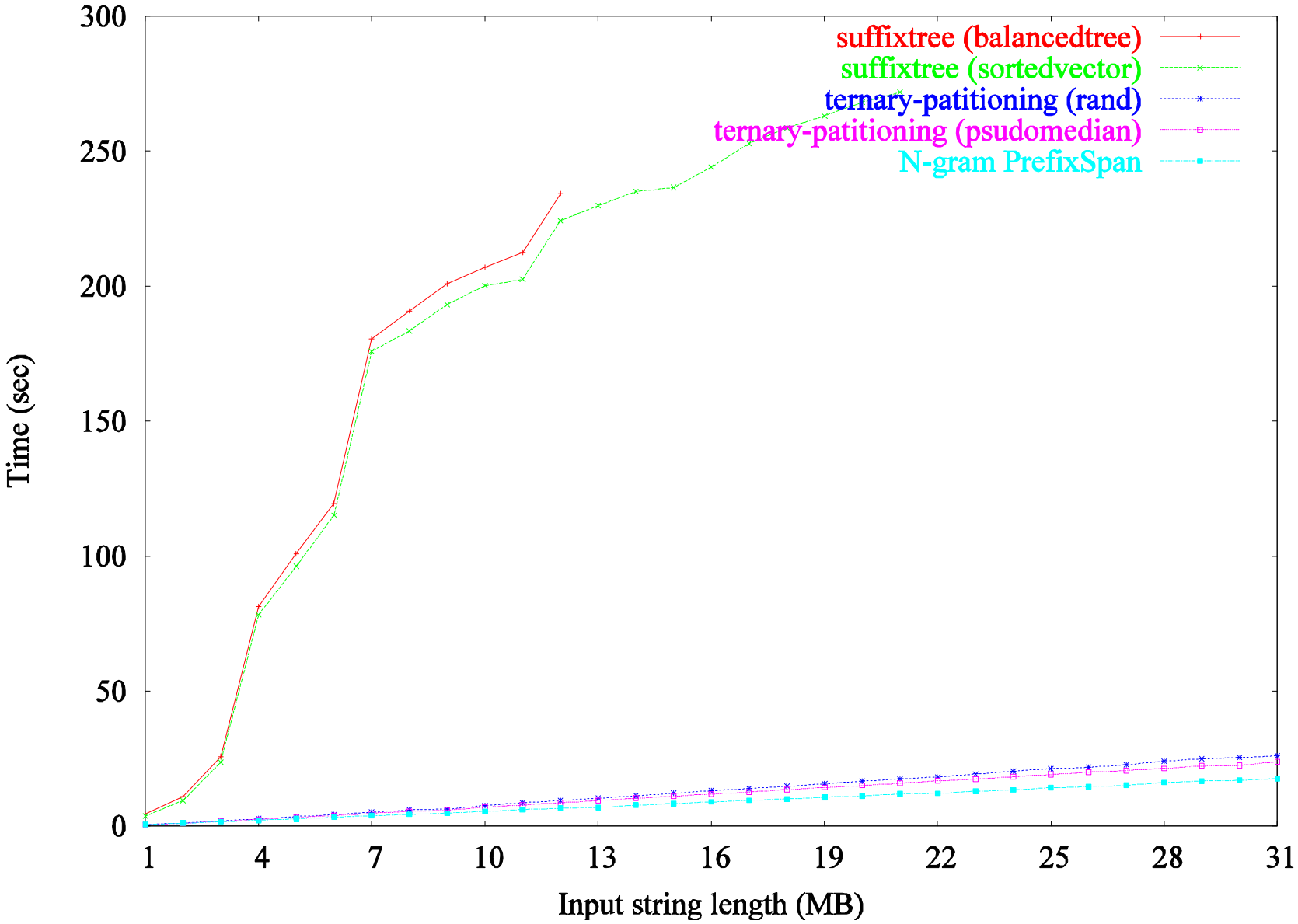
計算機実験

- 実験1: 入力サイズ（Scalability）
  - パラメータ: data size (1 – 31MB)
  - データ:　英語Newsgroups記事 (20 Usenet)
  - 固定値:　閾値(ξ): 0.001% (5MB の時 ξ =52回)
- 実験2: 閾値 (Threshold)
  - パラメータ: 閾値 : 0.001 - 0.0001%
    (5MB の時 ξ = 52 – 5回)
  - データ:　英語Newsgroups記事 (5MB),
    DNA配列（大腸菌, 4.4MB）
  - 固定値:　Data size (5MB, 4.4MB)

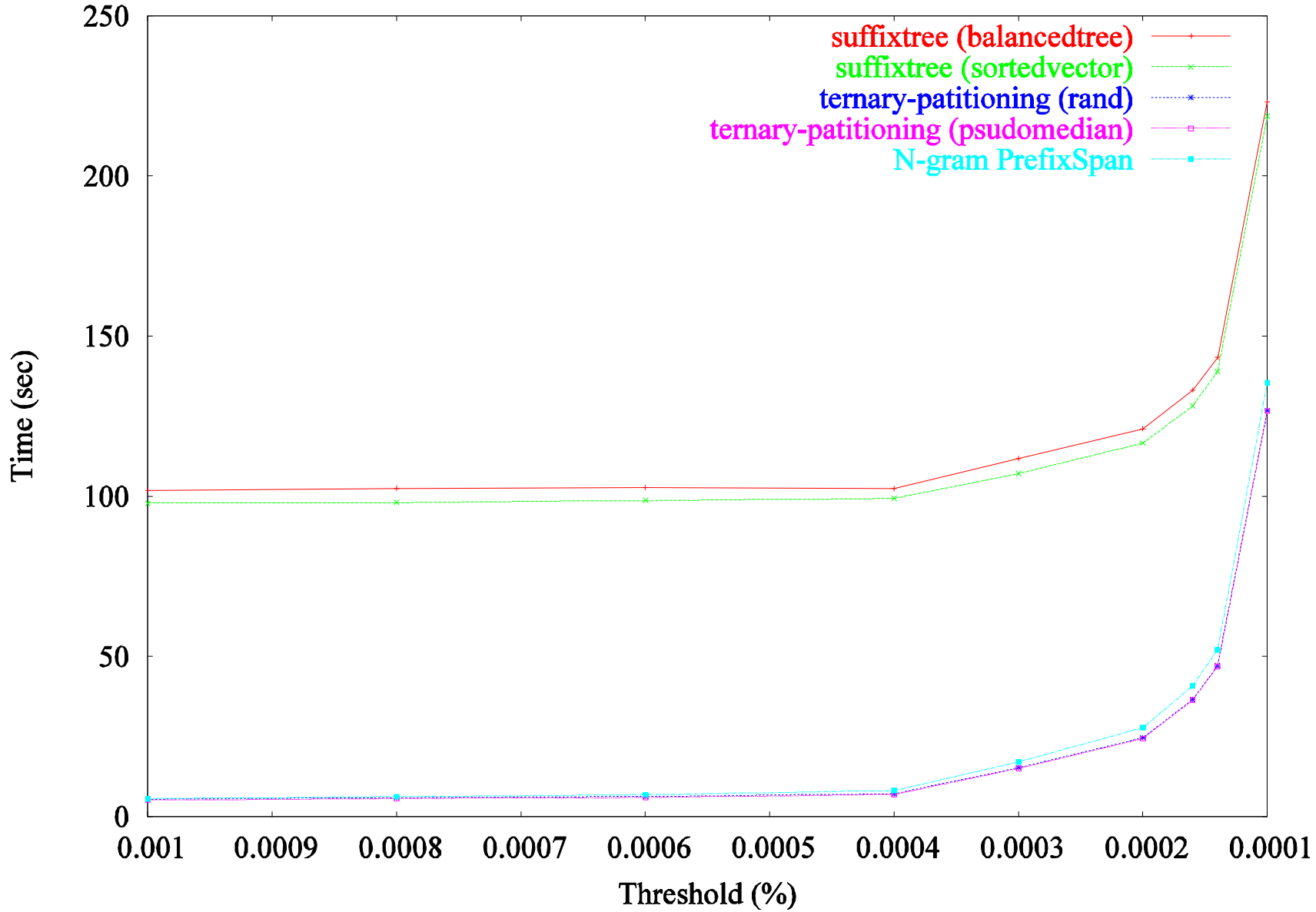
比較アルゴリズムの実装(C++)

- 提案手法:
  - 選択肢:基準値の選択
  - ternary-partitioning (rand) : ランダムに選択
  - ternary-partitioning (pseudo-median):
    3点または9点サンプリングし中央値を選ぶ
- 接尾辞木:
  - 選択肢:子ノードへのリンクのデータ構造
  - suffixtree(balancedtree): バランス木（STL map）
  - suffixtree(sortedvector): ソート済み配列(AssocVector)
- N-gram Prefixspan

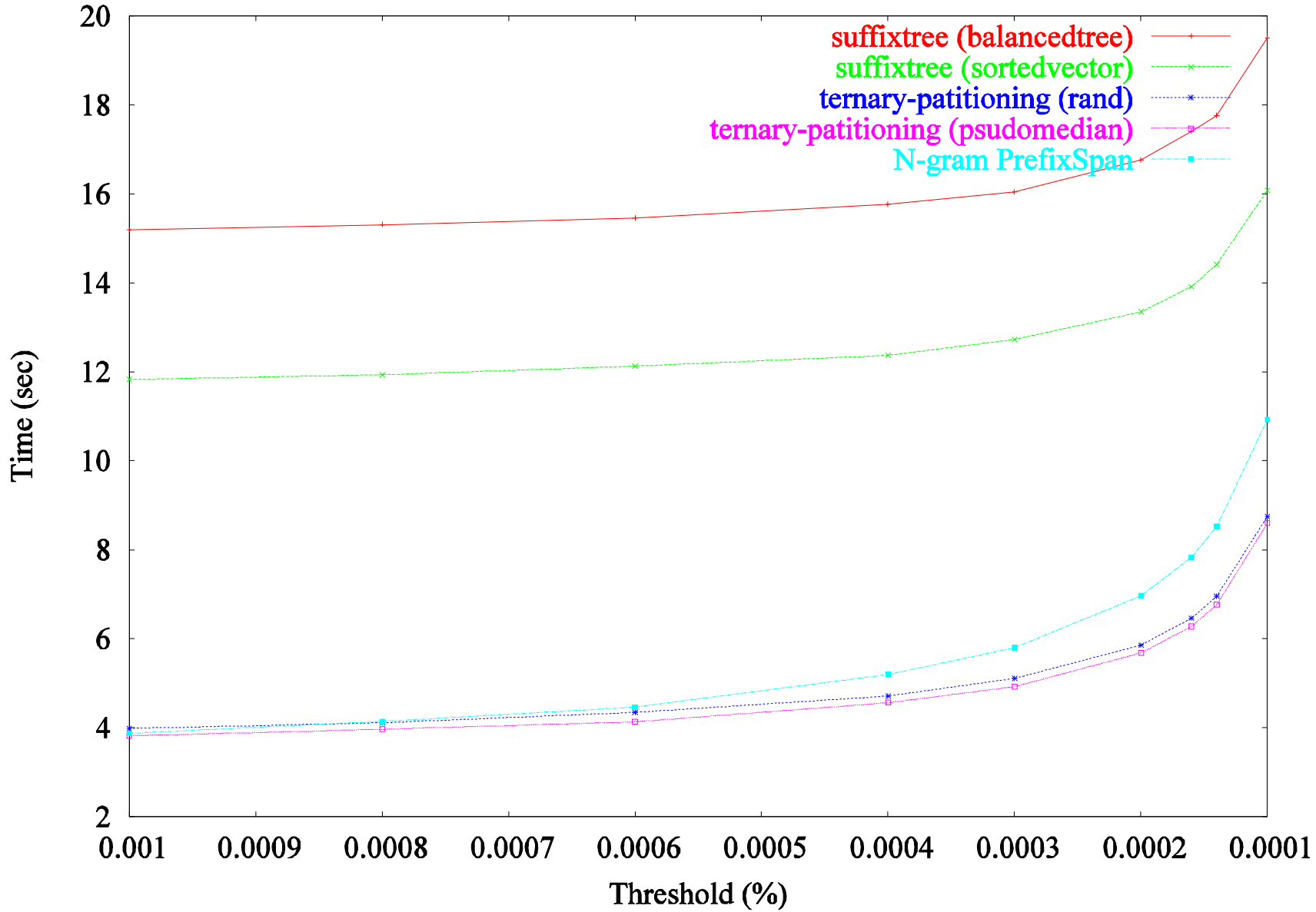
実験1: 入力サイズ, Newsgroups



実験2: 閾値, Newsgroups(5MB)



実験2: 閾値, DNA配列 (4.4MB)



メモリ使用量

- Newsgroups(5MB)

アルゴリズム	メモリ使用量
ternary-partitioning (both)	26 MB
N-gram PrefixSpan	50 MB
suffixtree (sortedvector)	391 MB
suffixtree (balancedtree)	693 MB

接尾辞木との比較 (実験まとめ)

- 両実験とも提案手法が良い性能
- 接尾辞木のメモリ使用量は非常に多くスケールしない
  - 各ノードに文字種分の子ノードへのリンクが必要

    →O(N|Σ|)

    Nは入力文字列サイズ

    |Σ|は文字種サイズ

N-gram PrefixSpan との比較（実験まとめ）

アルゴリズム	探索木	整列法
提案手法	3分木	Multikey Quicksort
N-gram PrefixSpan	トライ	基数ソート

- N-gram PrefixSpan
  - データサイズを増やしたときに提案手法より良い性能

  ←探索空間浅い
- 提案手法
  - メモリ使用量が約半分

  ←領域の分割に余計なメモリを使用しない
- 閾値を小さい時、提案手法が性能が良い

まとめ

- 可変長N-gramのカウントを頻出部分文字列問題として定義した。
- 3分割法によるアルゴリズムを提案
- 計算機実験より
  - 接尾辞木による方法より優位
  - 半分のメモリ使用量で、N-gram PrefixSpanと同等の性能