Vol.2019-AL-173 No.6
2019/5/10

データストリームのための頻出部分文字列発見アルゴリズム

鳥谷部 直弥[1,a)]　古谷 勇[1,b)]　喜田 拓也[1,c)]

概要：データストリームとは，際限なく流れてくるアイテムの列である．中でも，文字をアイテムとするデータストリームを文字列と見なし，そこから頻出な部分文字列を求める問題（頻出部分文字列問題）は基本的かつ重要であるが，これを限られたメモリ領域で厳密に解くことは難しい．本稿では，この頻出部分文字列問題の制約を緩和し，長さ l までの文字列のみを考慮し，出力に偽陽性解の含有を許容した (ϵ, l)-近似頻出文字列問題を定義し，この問題を入力文字ごとにならし $\mathcal{O}(l \log(\frac{1}{\epsilon} \log \epsilon N))$ 更新時間，作業領域を $\mathcal{O}(l \frac{1}{\epsilon} \log \epsilon N)$ 語空間で解くアルゴリズムとして，データ構造 GSTrie-List を用いた pmLC を提案した．加えて，長さ 2 以上の文字列に関する制約をより緩めた問題である (ϵ, η, l)-近似頻出部分文字列問題を定義し，頻出な文字列は頻出な文字から構成される性質を用いることで，$\alpha = \frac{\sum_X f_x^{(N)}}{N}, X = \{x \mid \hat{f}_x^{(N)} > \eta N\}$ に対して，辞書の更新操作を平均 $\mathcal{O}(\min(\frac{\alpha}{1-\alpha}, l) \log \frac{1}{\eta - \epsilon})$ 時間，作業領域を $\mathcal{O}(\frac{1}{\epsilon} + \frac{l}{\eta} \log \eta N)$ 語空間で (ϵ, η, l)-近似頻出アイテム問題を解くことができる混合アルゴリズム KRB-pmLC を提案した．

1. はじめに

オンラインに再現なく流れてくるデータの列のことをデータストリームとよぶ．データストリームに対してそのデータ全てをメモリ上に保持することが難しく，過去に流れてきたデータの一部または全てにアクセスすることはできない．したがって，データストリームに対するアルゴリズムは，データをオンラインに処理する必要があり，実行時間が高速かつ省メモリであることが求められる．

文字を要素とするデータストリームは終わりのない文字列とみなせる．入力された文字列から頻出している部分文字列を抽出することは，例えばデータ圧縮などへの応用があり，重要な問題である．

任意の部分文字列の頻度を求める方法としては，全文検索可能な文字列索引を構築することが挙げられる．入力データに対してオンラインで構築できる文字列索引構造としては接尾辞木 [4] が代表的である．しかしながら，接尾辞木の大きさは入力データ長に対して線形であり，データストリーム長が増えれば比例して増大する．

木の深さに制限を設ける k 切断接尾辞木（k-truncated suffix tree, k-TST）[11] を用いれば，接尾辞木より使用メモリ量を低減できることが知られている．しかしながら，要素の種類数（アルファベットサイズ）を σ とすると，その上限である $O(\sigma^k)$ 領域に達するまで入力データ長に対してほぼ線形に増大するため，データストリームに対して用いる場合には σ や k が大きいと使用メモリ量が莫大なものとなる．

これに対し，本稿では，頻出アイテム発見で用いられる計数アルゴリズムを利用し，頻度が高いと推定される部分文字列のみを保持する手法を提案する．提案手法では，カウンタベースの頻出アイテム発見アルゴリズムによって頻度の高い文字を抽出して管理し，それら頻度の高い文字のみから構成される部分文字列を疎な k 切断接尾辞トライ（sparse and k-truncated suffix trie[*1]）を用いて管理する．そうすることで，頻度が低い文字列による木の増大が抑制され，提案手法のメモリ消費量が $o(N)$ となる．

本稿の構成は次のとおりである．2 節では，準備として，データストリームに対する頻出アイテム問題とそれを解決する三つの代表的な手法について概説する．また，文字列に対する索引データ構造として k-TST についても触れる．3 節では，本稿で取り扱う頻出部分文字列問題について定義し，これに対するナイーブアルゴリズムと提案アルゴリズムについて説明する．最後に 4 節で本稿のまとめと今後の課題について述べる．

2. 準備

2.1 データストリームに対する頻出アイテム問題

記号の有限集合 $\mathcal{A}$ をアルファベット，アルファベットの

[1] 北海道大学 大学院情報科学研究科
Sapporo, Hokkaido 060–0814, Japan
[a)] toriyabe@ist.hokudai.ac.jp
[b)] furuya@ist.hokudai.ac.jp
[c)] kida@ist.hokudai.ac.jp

[*1] 疎な k 切断接尾辞木 [12] のトライ版．

各要素を**アイテム**と呼ぶ．**データストリーム** $\mathcal{S}$ [1] を，際限なく流れてくるアイテムの列と定義する．離散的な時点毎に一つのアイテムが流れてくるものとし，時点 t までのデータストリームを $\mathcal{S}_t = s_1 s_2 \ldots s_t, \forall s_j \in \mathcal{A}$ と書く．

アイテム i がデータストリーム中に出現した回数を i の頻度とよび，時点 t における i の頻度を $f_i^{(t)}$ と記述する．一方，アルゴリズムによる i の頻度の推定値を $\hat{f}_i^{(t)}$ と書く．ここで，ある閾値 ϕ（$0 < \phi < 1$）が与えられた時，アイテム i に対して，$f_i^{(t)} > \phi t$ を満たすならば，そのアイテム i は時点 t で頻出であるという．

2.1.1 問題定義

三種類の頻出アイテム問題を定義する．一つ目は厳密な解を求める問題であり，後の二つは制約を緩めた近似解を求める問題である．

定義 1 ((厳密な) 頻出アイテム問題)．長さ N のデータストリーム $\mathcal{S}_N$ と，閾値 ϕ（$0 < \phi < 1$）が与えられた時，集合 $\mathcal{F} = \{i \in \mathcal{A} \mid f_i^{(N)} > \phi N\}$ を出力する．

データストリーム上で厳密な頻出アイテム問題を解くためには，全てのアイテムの頻度を正確に数え上げる必要があり，現実的ではない．そのため，出力に偽陽性の解を含むことを許容した問題定義を次に示す．

定義 2 (ϕ-頻出アイテム問題)．長さ N のデータストリーム $\mathcal{S}_N$ と，閾値 ϕ（$0 < \phi < 1$）が与えられた時，条件 $\mathcal{F} \supseteq \{i \in \mathcal{A} \mid f_i^{(N)} > \phi N\}$ を満たす集合 $\mathcal{F}$ を出力する．

さらに，出力される偽陽性の解の出現頻度がある値を越えているという保証を加えた問題定義を次に示す．

定義 3 (ϵ-近似頻出アイテム問題)．長さ N のデータストリーム $\mathcal{S}_N$ と，閾値 ϕ, ϵ（$0 < \epsilon < \phi < 1$）が与えられた時，次の二つの条件を満たす集合 $\mathcal{F}$ を出力する．

(1) $\mathcal{F} \supseteq \{i \in \mathcal{A} \mid f_i^{(N)} > \phi N\}$

(2) $\mathcal{F} \cap \{i \in \mathcal{A} \mid f_i^{(N)} \leq (\phi - \epsilon)N\} = \varnothing$

2.1.2 データ構造: Hash-List

カウンタベースの頻出アイテム発見アルゴリズムでは，頻出なアイテムを求めるために，アイテムとその推定頻度の組（カウンタ）を辞書型のデータ構造を用いて管理する．本稿では，ハッシュ表と双方向連結リストを組み合わせた **Hash-List 構造** [6,9] を用いる．

Hash-List 構造では，アイテムの探索，挿入，削除を効率よく行うために，アイテムの値によるハッシュ値で対応するカウンタにアクセスできるハッシュ表を辞書として用いる．加えて，同じ推定頻度を持つカウンタを双方向連結リストで結び，各頻度のリストへアクセスできる頻度バケットを構成する．

頻度バケットは保持している整数値の昇順に並べられており，c をこの構造が保持するアイテムの最大数とすると，Hash-List 構造の作業領域は $\mathcal{O}(c)$ 語領域である．また，各アイテムの挿入と削除，検索操作は，それぞれ $\mathcal{O}(1)$ 平均時間で行うことができる．

2.1.3 Lossy counting (LC)

2002 年に Manku と Motwani ら [8] は，ϵ-近似頻出アイテム問題を解くアルゴリズムである Lossy counting 法（LC）を提案した．その後，2011 年に Liu らのサーベイ論文 [7] において，元の LC に改良を施して出力される偽陽性の解を少なく改良したアルゴリズムが LC として紹介されている．本論文では，その改良型の LC を元の LC と区別するために modified Lossy counting 法（mLC）と呼び，LC と mLC を合わせて LC 系と呼ぶ．

LC 系は以下の二つの操作で構成される．

- 集合 $\mathcal{F}' = \{x \in \mathcal{A} \mid f_x^{(N)} > \epsilon N\}$ を求める．
- 集合 $\mathcal{F} = \{x \in \mathcal{F}' \mid f_x^{(N)} > \phi N\}$ を出力する．

この操作は，次節以降に説明する Space saving や KRB も同様である．

LC 系における解の候補となるアイテムを求め方は以下の通りである．データを大きさ $w = \lceil \frac{1}{\epsilon} \rceil$ のバケットに分けて考え，既にアクセスし終わったバケットの個数を Δ 値として保持する．時点 t における Δ 値は，$\Delta^{(t)} = \lfloor \frac{t}{w} \rfloor$ で表される．辞書で管理するカウンタには，アイテムとその推定頻度に加えて，辞書に入力された時点の Δ 値も保持する．アイテムを受け取った際は，アイテムが辞書内に入っているかどうかを確認し，入っている場合にはそのアイテムの頻度を 1 増やし，入っていない場合には新たなアイテムとして辞書に挿入する．時点 t' において新たなアイテムが辞書に挿入される時，そのアイテムの推定頻度を 1 とし，Δ 値は $\Delta^{(t')}$ とする．ここで，推定頻度と Δ 値の和 $f_i' = \hat{f}_i + \Delta^{(t')}$ を**拡張頻度**と呼ぶことにする．時点 t において，新たなバケット内のアイテムを受け取った時，条件 $\hat{f}_i + \Delta^{(t')} \leq \Delta^{(t)}$ を満たすアイテム，すなわち，拡張頻度が現在の Δ 値より小さいアイテムを辞書から削除する．

次に，LC 系におけるアイテムの頻度と推定頻度，拡張頻度との誤差の範囲に関する補題を示す．

補題 4 ([8])．LC において，アイテム i の頻度を f_i，推定頻度を $\hat{f}_i$ とする．この時，受け取ったアイテムの数 N に対して，不等式 $f_i - \epsilon N \leq \hat{f}_i \leq f_i$ が成り立つ．

補題 5 ([7])．mLC において，アイテム i の頻度を f_i，推定頻度を $\hat{f}_i$ とする．この時，推定頻度と挿入時のデルタ値の和を $f_i' = \hat{f}_i + \Delta^{(t)}$ とすると，受け取ったアイテムの数 N に対して，不等式 $f_i \leq f_i' \leq f_i + \Delta^{(t)}$ が成り立つ．

解の候補から ϵ-近似頻出アイテム問題の解を求めるための補題を以下に示す．

補題 6 ([8])．アイテム i の頻度を f_i，その推定値を e_i とする．パラメータ ϕ, ϵ $(0 < \epsilon < \phi < 1)$ を与えた時，不等式 $f_i - \epsilon N \leq e_i \leq f_i$ が成り立つならば，$\{i \mid e_i > (\phi - \epsilon)N\}$ は，ϵ-近似頻出アイテム問題の解になる．

補題 7 ([7,9])．アイテム i の頻度を f_i，その推定値を e_i

とする．パラメータ ϕ, ϵ $(0 < \epsilon < \phi < 1)$ を与えた時，不等式 $f_i \leq e_i \leq f_i + \epsilon N$ が成り立つならば，$\{i \mid e_i > \phi N\}$ は，ϵ-近似頻出アイテム問題の解になる．

補題 4 から 7 より，次の定理 8 が成り立つ．

定理 8 ([7, 8])．LC 系は，辞書の更新操作をならし $\mathcal{O}(\log \epsilon N)$ 時間，作業領域を $\mathcal{O}(\frac{1}{\epsilon} \log \epsilon N)$ 語領域で ϵ-近似頻出アイテム問題を解く．

2.1.4 Space saving (SS)

2005 年に Metwally ら [9] は，ϵ-近似頻出アイテム問題を解くアルゴリズムである Space saving 法（SS）を提案した．

閾値 ϵ に対して，$k = \lceil \frac{1}{\epsilon} \rceil$ とする．SS では，辞書内で管理しているアイテムの種類数（カウンタの個数）を監視する．新たにアイテムを受け取った時，そのアイテムが辞書に入っているかどうかを確認し，入っている場合はアイテムの頻度を 1 増やす．一方で，そのアイテムが辞書に入っておらず，かつ辞書内のカウンタの個数が k 個未満の場合は，そのアイテムを辞書内に格納する．その際，格納するアイテムの推定頻度は 1 とする．辞書内のカウンタの個数が k 個の場合は，推定頻度が最も小さいアイテムを辞書から削除し，新たなアイテムを削除したアイテムより 1 だけ大きい推定頻度で辞書に格納する．

次に，SS におけるアイテムの頻度と拡張頻度との誤差の範囲に関する補題を示す．

補題 9 ([9])．SS において，アイテム i の頻度を f_i，推定頻度を $\hat{f}_i$ とする．この時，受け取ったアイテムの数 N に対して，不等式 $f_i \leq \hat{f}_i \leq f_i + \epsilon N$ が成り立つ．

補題 7 と 9 より，次の定理 10 が成り立つ．

定理 10 ([9])．SS は，辞書の更新操作を $\mathcal{O}(1)$ 時間，作業領域を $\mathcal{O}(\frac{1}{\epsilon})$ 語領域で ϵ-近似頻出アイテム問題を解く．

2.1.5 KRB

1982 年に Misra と Gires ら [10] は，k-reduced bag という集合の性質を用いることで，オフラインのデータに対して頻出アイテム問題を厳密に解くアルゴリズムを提案した．2000 年代に入り，Misra と Gries のアルゴリズムをデータストリームに応用したアルゴリズム（Frequent）が，Demaine ら [3] と Karp ら [6] によってそれぞれ提案された．

以下に k-reduced bag の定義を示す．

定義 11 (k-reduced bag [10])．多重集合から異なる k 種類のアイテムを削除する操作を k-reduction 操作と呼ぶ．多重集合に対して，k-reduction 操作をできる限り行った結果として残った多重集合を，元の多重集合の k-reduced bag と呼ぶ．

ある多重集合に対する k-reduced bag は，k-reduction 操作によって選ばれるアイテムの組み合わせによって変化するため，一意には定まらないことに注意する．例えば，$k = 3$ とした時，多重集合 $\{1, 1, 1, 1, 2, 2, 2, 3, 3, 4\}$ に対する 3-reduced bag は，$\{1, 1, 1, 2\}$ または $\{1\}$ となる．

Frequent は，この k-reduced bag をオンラインで管理して ϕ-頻出アイテム問題を解く．Frequent では，データストリームからアイテムを受け取った時，(i) そのアイテムが辞書に入っているなら頻度を 1 増やす，(ii) 辞書に入っていなければ新たなアイテムとして挿入する，(iii) 辞書に k 種類のアイテムが入ったならば k-reduction 操作を行う，という処理を行いながら k-reduced bag を保持し続ける．k-reduction 操作によって，アイテムの頻度が減らされるため，Frequent で求めるアイテムの頻度は，正確な頻度ではなく推定値になる．長さ N のデータストリームに対して，k-reduction 操作は $\lfloor \frac{N}{k} \rfloor$ 回しか行うことができないため，$k = \lceil \frac{1}{\phi} \rceil$ とすると k-reduced bag は ϕ-頻出アイテム問題の解となる．

サーベイ論文 [2,7] によると，Frequent は，ϵ-近似頻出アイテム問題を解くアルゴリズムである Lossy counting [8] や，Space saving [9] と比べて，解の適合率が低いという結果が示された．これに対し我々は，解の精度を高めることを目的として Frequent を改善し，k-reduced bag を用いて ϵ-近似頻出アイテム問題を解くアルゴリズム KRB をこれまでに提案している [13]．

KRB では，$k = \lceil \frac{1}{\epsilon} \rceil$ とした k-reduced bag をオンラインで管理し，条件 $\mathcal{F} \supseteq \{i \in \mathcal{A} \mid f_i^{(N)} > \epsilon N\}$ を満たす集合 $\mathcal{F}$ を求める．その際，時点 t での k-reduction 操作の回数 $\Delta^{(t)}$ を監視しつつ，辞書にはアイテムとその拡張頻度を保持する．一方で，k-reduction 操作については，(i) k-reduction 操作時に推定頻度を減らす操作はせず，(ii) $f'_i = \Delta^{t'}$ となるアイテムを辞書から削除する，という二点の変更を行う．

次に，KRB におけるアイテムの頻度と拡張頻度との誤差の範囲に関する補題を示す．

補題 12 ([13])．KRB において，アイテム i の頻度を f_i，推定頻度を $\hat{f}_i$ とする．この時，推定頻度と挿入時の Δ 値の和を $f'_i = \hat{f}_i + \Delta^{(t)}$ とすると，不等式 $f_i \leq f'_i \leq f_i + \Delta^{(t)}$ が成り立つ．

したがって，補題 7 と 12 より次の定理 13 が成り立つ．

定理 13 ([13])．KRB は，辞書の更新操作をならし $\mathcal{O}(1)$ 時間，作業領域を $\mathcal{O}(\frac{1}{\epsilon})$ 語領域で ϵ-近似頻出アイテム問題を解く．

2.1.6 性能比較

表 1 に，Hash-List 構造を用いた場合の，各アルゴリズムの辞書の更新時間と作業領域，頻度の誤差の範囲を示す．ここで，頻度の誤差の範囲 $[a, b]$ は，辞書に含まれるアイテム i の真の頻度を f_i，アルゴリズムによる推定頻度または拡張頻度を e_i とした時，$f_i + a \leq e_i \leq f_i + b$ が成り立つことを示している．辞書の更新時間は，KRB と SS が LC 系と比べて小さく，頻度の誤差の範囲は，KRB と

mLC が $\Delta^{(t)}$ となっており，LC と SS に比べて小さいことがわかる．

2.2 疎な k 切断接尾辞トライ

データストリームを終わりのない文字列とみるとき，アルファベット $\mathcal{A}$ の要素は**文字**と呼ぶほうが都合が良い．以降では，データストリームをアイテムの系列か文字列かどちらかの見方にするかによって，その要素をアイテムもしくは文字と呼ぶことにする．文字列 $\mathcal{T}$ に対し，$\mathcal{T} = xyz$ $(x, y, z \in \mathcal{A}^*)$ であるとき，文字列 x, y, z をそれぞれ，文字列 $\mathcal{T}$ の接頭辞，部分文字列，接尾辞と呼ぶ．文字列 $\mathcal{T}$ の長さを $|\mathcal{T}|$，文字列 $\mathcal{T}$ の全ての部分文字列からなる集合を $Sub(\mathcal{T})$ で表す．

疎な k 切断接尾辞トライ（sparse and k-truncated suffix tree）とは，文字列 $\mathcal{T}$ の全ての接尾辞からなる集合の部分集合を保持する辞書トライを，k より深いノードを削除して得られる木構造である．疎な接尾辞木 [5] に深さ k の制約を付けた疎な k 切断接尾辞木 [12] の，パスを圧縮しない構造と同じものになる．構造としては，長さが k の文字列の集合に対する一般化接尾辞トライ（generalized suffix trie, GSTrie）と変わらない．したがって，以降では，疎な k 切断接尾辞トライによる辞書を単に GSTrie と記述する．

3. 頻出部分文字列問題

本節では，データストリームから頻出な部分文字列を求める問題を定義し，その問題を解く二つのアルゴリズムを提案する．

データストリームにおいて，各時点に流れてくるアイテムを文字と仮定する．この時，各時点に文字が一つずつ流れてくるデータストリームは単に，アクセスできる範囲に制限が存在する条件付きの文字列とみなすことができる．今後，各時点に文字が一つずつ流れてくるデータストリームを単に文字列と記述する．時点 t の文字列を $\mathcal{S}_t = s_1 s_2 \ldots s_t, \forall s_j \in \mathcal{A}$ とかく．

ある部分文字列 x が長さ N の文字列 $\mathcal{S} \in \mathcal{A}^*$ 内に出現した回数のことを頻度とよび，$f_x^{(N)}$ と記述する．ここで，あるパラメータ $\phi (0 < \phi < 1)$ が与えられた時，ある部分文字列 x に対して，$f_x^{(N)} > \phi N$ を満たすならば，その部分文字列 x は頻出であるという．

3.1 問題設定

定義 14 ((厳密な) 頻出部分文字列問題)．長さ N の文字列 $\mathcal{S}_N$ と，閾値 ϕ $(0 < \phi < 1)$ が与えられた時，集合 $\mathcal{F} = \{x \in \mathrm{Sub}(\mathcal{S}_N) \mid f_x^{(N)} > \phi N\}$ を出力する．

文字 a が N 個連なった文字列を $\mathcal{S}_N = a^n$ について考えると，長さ N の文字列中の頻出な部分文字列になりうる文字列の最長の長さは $(1-\phi)N$ であることがわかる．長さが $\mathcal{O}(N)$ の文字列を保持することは，データストリームの仮定より難しい．加えて，データストリーム上で厳密な頻出部分文字列問題を解くためには，出現した全ての部分文字列の頻度を正確に数え上げる必要があり，現実的ではない．従って，偽陽性の解を含むことを許容し，出力される文字列の長さに制約を持たせた問題設定を定義 15 に示す．

定義 15 ((ϵ, l)-近似頻出部分文字列問題)．長さ N の文字列 $\mathcal{S}_N$ と，閾値 ϕ, ϵ $(0 < \epsilon < \phi < 1)$ が与えられた時，条件 (1)(2) を満たす集合 $\mathcal{F}$ を出力する．

(1) $\mathcal{F} \supseteq \{x \in \mathrm{Sub}(\mathcal{S}_N) \mid f_x^{(N)} > \phi N \text{ and } |x| \leq l\}$

(2) $\mathcal{F} \cap \{x \in \mathrm{Sub}(\mathcal{S}_N) \mid f_x^{(N)} \leq (\phi - \epsilon)N \text{ or } |x| > l\} = \varnothing$

頻出部分文字列問題を厳密に解く最も単純なアルゴリズムは，文字列内に存在する全ての部分文字列の出現頻度を正確に数え上げることである．加えて，2.2 節で示したような文字列に対するデータ構造を用いることで，より効率よく解くことができる．しかし，これらのナイーブアルゴリズムや文字列に対するデータ構造は，長さ N の文字列を保持する必要があり，少なくとも $\mathcal{O}(N)$ 語空間が必要である．そのため，データストリームに対して適用することは困難である．したがって，本節の残りで，データストリームに対して頻出部分文字列問題を解くアルゴリズムについて説明する．

3.2 データストリームに対するナイーブアルゴリズム

長さ N の文字列 $\mathcal{S}_N$ が与えられた時，部分文字列をアイテムとみなし，頻出アイテム発見アルゴリズム（mLC, SS, KRB）を用いて，長さ毎に頻出なアイテムを求めることにより，(ϵ, l)-近似頻出部分文字列問題を解くことを考える．

N 個のアイテムを受け取った際の，ϵ-近似頻出アイテム問題の解は以下の条件を満たす．

(1) $\mathcal{F} \supseteq \{i \in \mathcal{A} \mid f_i^{(N)} > \phi N\}$

(2) $\mathcal{F} \cap \{i \in \mathcal{A} \mid f_i^{(N)} \leq (\phi - \epsilon)N\} = \varnothing$

ここで，ϵ-近似頻出アイテム問題の解を用いて，(ϵ, l)-近似頻出部分文字列問題を解くための補題を示す．

補題 16. 長さ N の文字列 $\mathcal{S}_N$ が与えられた時，$N' = N - j + 1$ 個の長さが j の部分文字列のみを入力として，ϵ-近似頻出アイテム問題を解くアルゴリズムの解を $\mathcal{F}_j$ とする．また，長さ 1 から l までの解の和集合を $\mathcal{F}' = \mathcal{F}_1 \cup \mathcal{F}_2 \cup \ldots \cup \mathcal{F}_l$ とする．この時，$\mathcal{F}'$ は，(ϵ, l)-近似頻出部分文字列問題の解になる．

証明. 以下の条件 (1)(2) を満たすことを示す．

(1) $\mathcal{F}' \supseteq \{x \in \mathrm{Sub}(\mathcal{S}_N) \mid f_x^{(N')} > \phi N \text{ and } |x| \leq l\}$

(2) $\mathcal{F}' \cup \{x \in \mathrm{Sub}(\mathcal{S}_N) \mid f_x^{(N')} \leq (\phi - \epsilon)N \text{ or } |x| > l\} = \varnothing$

条件 (1):

$A_j = \{x \in \mathcal{A}^j \mid f_x^{(N')} > \phi N\}$,

$B_j = \{x \in \mathrm{Sub}(\mathcal{S}_N) \mid f_x^{(N')} > \phi N \text{ and } |x| = j\}$ とする．

ある文字列 $x \in A_j$ の時，$|x| = l$ かつ $f_x^{(N')} > \phi N$ である．

表 1 辞書の更新時間と作業領域，頻度の誤差範囲の比較

名前	更新時間	作業領域	頻度の誤差範囲
KRB [鳥谷部&喜田，2019 年]	$\mathcal{O}(1)$ amortized	$\mathcal{O}(\frac{1}{\epsilon})$	$[0, \Delta^{(t)}]$
Lossy counting [Manku&Motwani，2002 年]	$\mathcal{O}(\log \epsilon N)$ amortized	$\mathcal{O}(\frac{1}{\epsilon} \log \epsilon N)$	$[-\epsilon N, 0]$
Modified Lossy counting [Liu ら，2011 年]	$\mathcal{O}(\log \epsilon N)$ amortized	$\mathcal{O}(\frac{1}{\epsilon} \log \epsilon N)$	$[0, \Delta^{(t)}]$
Space saving [Metwally ら，2005 年]	$\mathcal{O}(1)$	$\mathcal{O}(\frac{1}{\epsilon})$	$[0, \epsilon N]$

加えて，$f_x^{(N')} > 0$ より $x \in \mathrm{Sub}(\mathcal{S}_N)$ である．したがって，$x \in A_j \Rightarrow x \in B_j$ が成り立つ．また，$x \in B_j$ の時，$|x| = l$ より，$x \in \mathcal{A}^l$ が成り立つ．したがって，$x \in B_j \Rightarrow x \in A_j$ が成り立つ．よって，$A_j = B_j$ である．ここで，$\mathcal{F}_j \supseteq A_j$ であるので，$\mathcal{F}_j \supseteq B_j$ が成り立つ．

$B_j = \{x \in \mathrm{Sub}(\mathcal{S}_N) \mid f_x^{(N')} > \phi N \text{ and } |x| = j\}$ により．$B_1 \cup B_2 \cup ... \cup B_l = \{x \in \mathrm{Sub}(\mathcal{S}_N) \mid f_x^{(N')} > \phi N \text{ and } |x| \leq l\}$ であるので，$\mathcal{F}' \supseteq \{x \in \mathrm{Sub}(\mathcal{S}_N) \mid f_x^{(N')} > \phi N \text{ and } |x| \leq l\}$ が成り立つ．

条件 (2) も同様に示すことができる． □

$\mathcal{S}_N$ の部分文字列のうち，長さが l の文字列は $N - l + 1 = N'$ 個存在する．補題 16 から，N' 個のアイテムを受け取り，N 個のアイテムに対する ϵ-近似頻出アイテム問題の解を求めるアルゴリズムが存在するならば，そのアルゴリズムを各長さの文字列のみを受け取る問題に適用することで，(ϵ, l)-近似頻出部分文字列問題の解を求めることができる．

補題 17. mLC，SS，KRB は，N' 個のアイテムを受け取って，N 個のアイテムに対する ϵ-近似頻出アイテム問題の解を求めることができない．

証明. N' 個のアイテムに対して，ϵ-近似頻出アイテム問題を解くと，出力の集合は以下の二つの条件を満たす．
(1) $\mathcal{F}' \supseteq \{x \in \mathrm{Sub}(\mathcal{S}_N) \mid f_x^{(N')} > \phi N' \text{ and } |x| \leq l\}$
(2) $\mathcal{F}' \cup \{x \in \mathrm{Sub}(\mathcal{S}_N) \mid f_x^{(N')} \leq (\phi - \epsilon) N' \text{ or } |x| > l\} = \varnothing$
$A = \{x \in \mathrm{Sub}(\mathcal{S}_N) \mid f_x^{(N')} > \phi N' \text{ and } |x| \leq l\}$, $B = \{x \in \mathrm{Sub}(\mathcal{S}_N) \mid f_x^{(N')} > \phi N \text{ and } |x| \leq l\}$ とすると，$A \supseteq B$ であるので，$\mathcal{F}' \supseteq B$ が満たされる．$C = \{x \in \mathrm{Sub}(\mathcal{S}_N) \mid f_x^{(N')} \leq (\phi - \epsilon) N' \text{ or } |x| > l\} = \varnothing$, $D = \{x \in \mathrm{Sub}(\mathcal{S}_N) \mid f_x^{(N')} \leq (\phi - \epsilon) N \text{ or } |x| > l\} = \varnothing$ とすると，$C \subseteq D$ であるので，$\mathcal{F}' \cup D = D \setminus C$ となる．よって，$C \neq D$ の時，$\mathcal{F}' \cup D \neq \varnothing$ である．したがって，mLC, SS，KRB は，N' 個のアイテムを受け取って，N 個のアイテムに対する ϵ-近似頻出アイテム問題の解を求めることができない． □

mLC, SS, KRB において，操作を以下のように工夫したアルゴリズムをそれぞれ mLC', SS', KRB' とする．

- 集合 $\mathcal{F}' = \{x \in \mathcal{A} \mid f_x^{(N')} > \epsilon N'\}$ を求める．
- 集合 $\mathcal{F} = \{x \in \mathcal{F}' \mid f_x^{(N')} > \phi N\}$ を出力する．

補題 18. mLC'，SS'，KRB' は，N' 個のアイテムを受け取って，N 個のアイテムに対する ϵ-近似頻出アイテム問題の解を求めることができる．

証明. mLC', SS', KRB' の出力は，以下の条件を満たす．
(1) $\mathcal{F}' \supseteq \{x \in \mathrm{Sub}(\mathcal{S}_N) \mid f_x^{(N')} > \phi N \text{ and } |x| \leq l\}$
(2) $\mathcal{F}' \cup \{x \in \mathrm{Sub}(\mathcal{S}_N) \mid f_x^{(N')} \leq \phi N - \epsilon N' \text{ or } |x| > l\} = \varnothing$
$C = \{x \in \mathrm{Sub}(\mathcal{S}_N) \mid f_x^{(N')} \leq \phi N - \epsilon N' \text{ or } |x| > l\} = \varnothing$
$D = \{x \in \mathrm{Sub}(\mathcal{S}_N) \mid f_x^{(N')} \leq (\phi - \epsilon) N \text{ or } |x| > l\} = \varnothing$
$C \supseteq D$ であるので，$\mathcal{F}' \cap D = \varnothing$ したがって．mLC'，SS'，KRB' は，N' 個のアイテムを受け取って，N 個のアイテムに対する ϵ-近似頻出アイテム問題の解を求めることができる． □

補題 16 と 18 から mLC'，SS'，KRB' を用いて (ϵ, l)-近似頻出部分文字列問題を解くことができる．しかし，文字の長さが短い順に頻出なアイテムを求めるだけでは，データを l 回走査する必要がある．時点 t に新たな文字を受け取ったとき，文字列全体として，新たに確認する必要がある部分文字列は，文字列 $s_{t-l+1}s_{t-l+2}...s_t$ の l 個の接尾辞である．そこで，文字を一つ受け取るたびに，1 から l までの各長さの頻出な文字列を格納する辞書を全て更新すれば良い．mLC', SS', KRB' を用いて，データストリームに対して (ϵ, l)-近似頻出部分文字列問題を解くアルゴリズムをそれぞれ Parallel mLC (pmLC), Parallel SS (pSS), Parallel KRB (pKRB) とする．アルゴリズム pmLC, pSS, pKRB の擬似コードを以下のアルゴリズム 1 から 3 に記載する．

定理 19. データ構造に Hash-List を用いた pmLC, pSS, pKRB は，辞書の更新操作をならし $\mathcal{O}(l^2 T)$ 時間，作業領域を $\mathcal{O}(l^2 M)$ 語空間で ϵ-近似頻出アイテム問題を解く．ただし，T, M はそれぞれ mLC', SS', KRB' における辞書の更新時間，データ構造の作業領域とする．

証明. 各長さの頻出な部分文字列は．多くても M 個しか存在しない．したがって，各長さの頻出な部分文字列が全て M 個存在すると仮定すると，必要な作業領域は，$M(1 + 2 + ... + l) = \mathcal{O}(l^2 M)$ 語空間である．また，ハッシュ表の衝突を仮定すると，長さ l の文字列が格納されて

Algorithm 1 pmLC

Input: データストリーム $\mathcal{S}_N = s_1...s_N$, 閾値 $\epsilon, \phi (0 < \epsilon < \phi < 1)$
Output: (ϵ, l)-近似頻出部分文字列問題の解

$\mathcal{D}_1, ..., \mathcal{D}_l \leftarrow \emptyset$, $k \leftarrow \lceil \frac{1}{\epsilon} \rceil$, $\Delta_1^{(0)}, ..., \Delta_l^{(0)} \leftarrow 0$.
for each t $(1 \leq t \leq N)$ **do**
  for $i = 1$ to l **do**
    $str = s_{t-i+1}...s_t$
    if $str \in \mathcal{D}_i$ **then**
      $f'_{str} \leftarrow f'_{str} + 1$
    else
      $\mathcal{D}_i \leftarrow \mathcal{D}_i \cup \{str\}, f'_{str} \leftarrow \Delta_i^{(t)} + 1$
    end if
    if $\Delta_i^{(t-1)} \neq \Delta_i^{(t)}$ **then**
      for each $j \in \mathcal{D}_i$ **do**
        if $f'_j < \Delta_i^{(t)}$ **then**
          $\mathcal{D}_i \leftarrow \mathcal{D}_i \setminus \{j\}$
        end if
      end for
    end if
  end for
end for
output the set $\{i | f'_i > \phi N\}$ from $\mathcal{D}_1, ..., \mathcal{D}_l$

Algorithm 2 pSS

Input: データストリーム $\mathcal{S}_N = s_1...s_N$, 閾値 $\epsilon, \phi (0 < \epsilon < \phi < 1)$
Output: (ϵ, l)-近似頻出部分文字列問題の解

$\mathcal{D}_1, ..., \mathcal{D}_l \leftarrow \emptyset$, $k \leftarrow \lceil \frac{1}{\epsilon} \rceil$, $\Delta_1^{(0)}, ..., \Delta_l^{(0)} \leftarrow 0$.
for each t $(1 \leq t \leq N)$ **do**
  for $i = 1$ to l **do**
    $str = s_{t-i+1}...s_t$
    if $str \in \mathcal{D}_i$ **then**
      $\hat{f}_{str} \leftarrow \hat{f}_{str} + 1$
    else if $|\mathcal{D}_i| < k$ **then**
      $\mathcal{D}_i \leftarrow \mathcal{D}_i \cup \{str\}, \hat{f}_{str} \leftarrow 1$
    else
      $j \leftarrow \mathrm{argmin}_{j \in \mathcal{D}_i} \hat{f}_j$
    end if
  end for
end for
output the set $\{i | \hat{f}_i > \phi N\}$ from $\mathcal{D}_1, ..., \mathcal{D}_l$

いるかどうかを確認する必要がある．したがって，辞書の更新時間は平均 $\mathcal{O}(l^2 T)$ である． □

3.3 pmLC + GSTrie-List

データ構造として，ハッシュ表を用いた場合，ハッシュの衝突を考慮すると，文字列の一致を確認する必要が出てくる．加えて，長さが l の文字列をデータ構造内に保持している．そこで，複数の文字列を効率よく扱うデータ構造である一般化接尾辞トライ (Generalizes Suffix Trie; GSTrie) と連結リストを組み合わせた構造である GSTrie-List を用いることで，作業領域の削減を行う．GSTrie は前節で説明したデータ構造 Hash-List におけるハッシュテーブルと同様に文字列にアクセスするための辞書としての役割を持つ．連結リストは辞書内の各文字列の出現頻度を格納する役割を持つ．

Algorithm 3 pKRB

Input: データストリーム $\mathcal{S}_N = s_1...s_N$, 閾値 $\epsilon, \phi (0 < \epsilon < \phi < 1)$
Output: (ϵ, l)-近似頻出部分文字列問題の解

$\mathcal{D}_1, ..., \mathcal{D}_l \leftarrow \emptyset$, $k \leftarrow \lceil \frac{1}{\epsilon} \rceil$, $\Delta_1^{(0)}, ..., \Delta_l^{(0)} \leftarrow 0$.
for each t $(1 \leq t \leq N)$ **do**
  for $i = 1$ to l **do**
    $str = s_{t-i+1}...s_t$
    if $str \in \mathcal{D}_i$ **then**
      $f'_{str} \leftarrow f'_{str} + 1$
    else if $|\mathcal{D}_i| < k - 1$ **then**
      $\mathcal{D}_i \leftarrow \mathcal{D}_i \cup \{str\}, f'_{str} \leftarrow \Delta_i^{(t)} + 1$
    else
      for each $j \in \mathcal{D}_i$ **do**
        $f'_j \leftarrow f'_j - 1$
        if $f'_j = 0$ **then**
          $\mathcal{D}_i \leftarrow \mathcal{D}_i \setminus \{j\}$
        end if
      end for
      $\Delta_i^{(t)} \leftarrow \Delta_i^{(t)} + 1$
    end if
  end for
end for
output the set $\{i | f'_i > \phi N\}$ from $\mathcal{D}_1, ..., \mathcal{D}_l$

GSTrie 上のノードはある文字列に対応しているため，頻出部分文字列問題を解く際に辞書に保持されている文字列の個数とノード数が一致する場合，すなわち，辞書に含まれる全ての文字列の部分文字列も全て，辞書に含まれている場合に効率よく作用する．pKRB と pSS は，辞書からアイテムを削除する操作が起こるタイミングが，文字列の長さ毎に異なる可能性があるため，辞書に含まれる全ての文字列の部分文字列が全て辞書に含まれていない場合がある．一方，pmLC は辞書内からアイテムを削除する操作が起こるタイミングが，全ての長さで同一のため，GSTrie を用いることで辞書内の部分文字列を効率よく扱うことができる．

したがって，以下の定理が示される．

定理 20. データ構造に GSTrie-List を用いた pmLC は，辞書の更新操作をならし $\mathcal{O}(l \log(\frac{1}{\epsilon} \log \epsilon N))$ 時間，作業領域を $\mathcal{O}(l \frac{1}{\epsilon} \log \epsilon N)$ 語空間で (ϵ, l)-近似頻出部分文字列問題を解く．

証明. 辞書に含まれる全ての文字列の部分文字列が全て辞書に含まれている場合，GSTrie G の葉に対応する全ての文字列を入力とする GSTrie G' の各ノードはそれぞれ，辞書内に含まれているある文字列と対応する．すなわち，G と G' は一致する．したがって，GSTrie のノード数は，辞書内に含まれる部分文字列の数と一致し，$\mathcal{O}(l \frac{1}{\epsilon} \log \epsilon N)$ で抑えられる．よって，必要な作業領域についても $\mathcal{O}(l \frac{1}{\epsilon} \log \epsilon N)$ 語空間になる．

時点 t での GStrie の更新の際に，長さが $l-1$ の文字列 $s_{t-l+2}...s_{t-1}$ の接尾辞のうち，辞書に格納されている最長の文字列に対応するノードへのポインタと，接尾辞リンク

を用いることにより，辞書の更新操作を $\mathcal{O}(l\frac{1}{\epsilon}\log\epsilon N)$ 時間で行うことができる．□

3.4 混合手法 KRB-pmLC + GStrie-List

pmLC, pSS, pKRB はそれぞれ辞書に入っていない部分文字列を全て辞書内に格納していた．ここで，頻出な文字列に関して頻出な部分文字列は，頻出な文字のみから構成されるという性質が存在する．したがって，頻出な文字列と長さが 1 の文字列との間の関係を補題 21 に示す．

補題 21. 辞書 $\mathcal{D}$ に含まれる全ての文字列 $\mathcal{S}=s_p...s_q$ に対して，条件 $\forall i\in\{p,...,q\}, s_i\in\mathcal{D}$ が成り立つ．

今後，長さ 1 の部分文字列と文字を同一視する．

本節では，(ϵ,l)-近似頻出部分文字列問題に基づき，長さが 2 以上の部分文字列についての制約を緩めた問題設定を (ϵ,η,l)-近似頻出部分文字列問題を定義する．加えて，頻出アイテム問題を解くアルゴリズムと頻出部分文字列問題を解くアルゴリズムの混合手法を提案する．

定義 22 ((ϵ,η,l)-近似頻出部分文字列問題)．長さ N の文字列 $\mathcal{S}_N$ と，閾値 ϕ,η,ϵ $(0<\epsilon<\eta<\phi<1)$ が与えられた時，条件 (1)(2) を満たす集合 $\mathcal{F}$ を出力する．

(1) $\mathcal{F}\supseteq\{x\in\mathrm{Sub}(\mathcal{S}_N)\mid f_x^{(N)}>\phi N \text{ and } |x|\le l\}$

(2) $\mathcal{F}\cap\{x\in\mathrm{Sub}(\mathcal{S}_N)\mid P(\phi-\epsilon,1) \text{ or } P(\phi-\eta,l) \text{ or } |x|>l\}=\varnothing$

ただし，条件式 $P(a,b)=\{f_x^{(N)}\le a \text{ and } |x|\le b\}$ とする．

混合手法は，頻出な文字を求めるために，頻出アイテム問題を利用する．加えて，頻出な長さが 2 以上の文字列を求めるために，頻出部分文字列問題を用いる．この時，辞書に入れる文字列は，条件 $\{x\in\mathrm{Sub}(\mathcal{S}_N)\mid f_x^{(N)}>\eta N\}$ を満たす文字列に限るものとする．頻出アイテム問題を解くためのアルゴリズムは，mLC, SS, KRB のいずれでも理論上可能であるが，理論計算量と，偽陽性解の範囲を考慮し，KRB を使用する．頻出文字列問題を解くアルゴリズムとしては，入力される文字列の個数に依存しない pmLC を使用する．混合手法のうち，KRB と pmLC を組み合わせたアルゴリズムを KRB-pmLC とよぶ．KRB-pmLC の擬似コードをアルゴリズム 4 に記載する．

定理 23. データ構造に GSTrie-List を用いた KRB-pmLC は, 辞書の更新操作を平均 $\mathcal{O}(\min(\frac{\alpha}{1-\alpha},l)\log\frac{1}{\eta-\epsilon})$ 時間, 作業領域を $\mathcal{O}(\frac{1}{\epsilon}+\frac{l}{\eta}\log\eta N)$ 語空間で (ϵ,η,l)-近似頻出アイテム問題を解く．ただし，$\alpha=\frac{\sum_X f_x^{(N)}}{N}, X=\{x\mid\hat{f}_x^{(N)}>\eta N\}$ とする．

証明. 辞書内に含まれている頻度の推定値が ηN より大きいアイテムの集合を $X=\{x\mid\hat{f}_x^{(N)}>\eta N\}$ とする．この時，$\alpha=\frac{\sum_X f_x^{(N)}}{N}$ は，頻度の推定値が ηN より大きいアイテムの真の頻度の合計から求められる確率を表すパラメータである．ある文字列に対して，頻度の推定値が ηN より高い文字のみで構成されている最長の接尾辞を長さの期待値は，$\frac{\alpha(1-\alpha^l)}{1-\alpha}$ である．したがって，長さが l の文字列に対して，頻度の推定値が ηN より高い文字のみで構成されている最長の接尾辞の長さの期待値は，$\mathcal{O}(\min(\frac{\alpha}{1-\alpha},l))$ である．また，頻度の推定値が ηN より大きい文字の種類数は，多くても $\frac{1}{\eta-\epsilon}$ であるため，辞書の更新時間は，平均 $\mathcal{O}(\min(\frac{\alpha}{1-\alpha},l)\log\frac{1}{\eta-\epsilon})$ 時間である．作業領域は，KRB と pmLC の作業領域を合わせた $\mathcal{O}(\frac{1}{\epsilon}+\frac{l}{\eta}\log\eta N)$ 語空間である．

文字の頻度 f_x^N と KRB-pmLC における推定値 e_x^N との誤差は，KRB に依存するため，各アイテム x が辞書に挿入された時点を t_i とすると，$f_x^N\le e_x^N\le f_x^N+\Delta^{(t_i)}\le\epsilon N$ が成り立つ．長さ 2 以上の部分文字列の頻度 f_x^N と KRB-pmLC

Algorithm 4 KRB-pmLC

Input: データストリーム $\mathcal{S}_N=s_1...s_N$, 閾値 $\epsilon,\phi(0<\epsilon<\phi<1)$
Output: (ϵ,η,l)-近似頻出部分文字列問題の解

$\mathcal{D}_1,...,\mathcal{D}_l\leftarrow\emptyset$, $k\leftarrow\lceil\frac{1}{\epsilon}\rceil$, $\Delta_1^{(0)},...,\Delta_l^{(0)}\leftarrow 0, check\leftarrow\varphi$.
for each t $(1\le t\le N)$ **do**
  if $s_t\in\mathcal{D}_1$ **then**
    $f'_{s_n}\leftarrow f'_{s_n}+1$
  else if $|\mathcal{D}_1|<k-1$ **then**
    $\mathcal{D}_1\leftarrow\mathcal{D}_1\cup\{s_n\}, f'_{s_n}\leftarrow\Delta_1^{(t)}+1$
  else
    for each $j\in\mathcal{D}_1$ **do**
      $f'_j\leftarrow f'_j-1$
      if $f'_j=0$ **then**
        $\mathcal{D}_1\leftarrow\mathcal{D}_1\setminus\{j\}$
      end if
    end for
    $\Delta_1^{(t)}\leftarrow\Delta_1^{(t)}+1$
  end if
  if $f'_{s_t}>\eta t$ **then**
    for $i=2$ to $|check|+1$ **do**
      $str=s_{t-i+1}...s_t$
      if $str\in\mathcal{D}_i$ **then**
        $f'_{str}\leftarrow f'_{str}+1$
      else
        $\mathcal{D}_i\leftarrow\mathcal{D}_i\cup\{str\}, f'_{str}\leftarrow\Delta_i^{(t)}+1$
      end if
      if $\Delta_i^{(t-1)}\neq\Delta_i^{(t)}$ **then**
        for each $j\in\mathcal{D}_i$ **do**
          if $f'_j<\Delta_i^{(t)}$ **then**
            $\mathcal{D}_i\leftarrow\mathcal{D}_i\setminus\{j\}$
          end if
        end for
      end if
    end for
    if $|check|=l$ **then**
      $check\leftarrow s_{t-l+2}...s_t$
    else
      $check\leftarrow check+s_t$
    end if
  else
    $check\leftarrow\varphi$
  end if
end for
output the set $\{i|f'_i>\phi N\}$ from $\mathcal{D}_1,...,\mathcal{D}_l$

における推定値 e_x^N との誤差は，pmLC に依存するため，$f_x^N \leq e_x^N \leq f_x^N + \eta N$ が成り立つ．したがって，$e_x^N > \phi N$ を満たす部分文字列を出力した時，(ϵ, η, l)-近似頻出アイテム問題の解の条件を満たしていることがわかる．

データ構造に GSTrie-List を用いた KRB-pmLCB は，辞書の更新操作を平均 $\mathcal{O}(\min(\frac{\alpha}{1-\alpha}, l) \log \frac{1}{\eta - \epsilon})$ 時間，作業領域は $\mathcal{O}(\frac{1}{\epsilon} + \frac{l}{\eta} \log \eta N)$ 語空間で (ϵ, η, l)-近似頻出アイテム問題を解く． □

4. おわりに

本稿では，文字を要素とするデータストリームを終わりのない文字列とみなし，文字列から頻出な部分文字列を求める問題について考察した．データストリームに対して解くことが可能な問題設定として，(ϵ, l)-近似頻出部分文字列問題を定義し，ナイーブなアルゴリズムとして，データ構造 Hash-List を用いた pmLC, pSS, pKRB の三つのアルゴリズムを説明した．このうち，pmLC は出力される全ての文字列の部分文字列も出力に含まれることが保証されているため，一般化接尾辞トライ (GSTrie) を用いて出力を保持することができる．そこで，GSTrie と連結リストを組み合わせたデータ構造 GSTrie-List を定義し，(ϵ, l)-近似頻出部分文字列問題を辞書の更新操作をならし $\mathcal{O}(l \log(\frac{1}{\epsilon} \log \epsilon N))$ 時間，作業領域を $\mathcal{O}(l \frac{1}{\epsilon} \log \epsilon N)$ 語空間で解く GSTrie を用いたアルゴリズム pmLC を提案した．

加えて，(ϵ, l)-近似頻出部分文字列問題の長さが 2 以上の文字列に関する制約を緩めた問題として (ϵ, η, l)-近似頻出部分文字列問題を定義した．頻出な文字列は，頻出な文字から構成されるという性質を利用し，頻出アイテム問題を解くアルゴリズム KRB を用いて頻出な文字を見つけ，頻出部分文字列問題を解くアルゴリズム pmLC を用いて頻出な文字列を見つける混合アルゴリズム KRB-pmLC を提案した．このアルゴリズムは，データ構造に GSTrie-List を用いることにより，辞書の更新操作を平均 $\mathcal{O}(\min(\frac{\alpha}{1-\alpha}, l) \log \frac{1}{\eta - \epsilon})$ 時間，作業領域を $\mathcal{O}(\frac{1}{\epsilon} + \frac{l}{\eta} \log \eta N)$ 語空間で (ϵ, η, l)-近似頻出アイテム問題を解くことができる．ただし， $\alpha = \frac{\sum_X f_x^{(N)}}{N}, X = \{x \mid \hat{f}_x^{(N)} > \eta N\}$ とする．

今後の課題として，部分文字列問題の設定の制約に関する再考と，混合手法についてのより精密な解析が挙げられる．

参考文献

[1] Babcock, B., Babu, S., Datar, M., Motwani, R. and Widom, J.: Models and issues in data stream systems, *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems(PODS)*, ACM, pp. pp.1–16 (2002).

[2] Cormode, G. and Hadjieleftheriou, M.: Methods for finding frequent items in data streams, *The VLDB Journal*, Vol. 19, No. 1, pp. 3–20 (2010).

[3] Demaine, E. D., López-Ortiz, A. and Munro, J. I.: Frequency estimation of internet packet streams with limited space, *European Symposium on Algorithms (ESA)*, Springer, pp. 348–360 (2002).

[4] Giegerich, R. and Kurtz, S.: From Ukkonen to McCreight and Weiner: A Unifying View of Linear-Time Suffix Tree Construction, *Algorithmica*, Vol. 19, No. 3, pp. 331–353 (1997).

[5] Kärkkäinen, J. and Ukkonen, E.: Sparse suffix trees, *Computing and Combinatorics* (Cai, J.-Y. and Wong, C. K., eds.), Berlin, Heidelberg, Springer Berlin Heidelberg, pp. 219–230 (1996).

[6] Karp, R. M., Shenker, S. and Papadimitriou, C. H.: A simple algorithm for finding frequent elements in streams and bags, *ACM Transactions on Database Systems (TODS)*, Vol. 8, No. 1, pp. 51–55 (2003).

[7] Liu, H., Lin, Y. and Han, J.: Methods for mining frequent items in data streams: an overview, *Knowledge and information systems*, Vol. 26, No. 1, pp. 1–30 (2011).

[8] Manku, G. S. and Motwani, R.: Approximate frequency counts over data streams, *Proceedings of the 28th international conference on Very Large Data Bases*, pp. 346–357 (2002).

[9] Metwally, A., Agrawal, D. and El Abbadi, A.: Efficient computation of frequent and top-k elements in data streams, *International Conference on Database Theory*, Springer, pp. 398–412 (2005).

[10] Misra, J. and Gries, D.: Finding repeated elements, *Science of computer programming*, Vol. 2, No. 2, pp. 143–152 (1982).

[11] Na, J. C., Apostolico, A., Iliopoulos, C. S. and Park, K.: Truncated suffix trees and their application to data compression, *Theoretical Computer Science*, Vol. 304, No. 1, pp. 87–101 (2003).

[12] Uemura, T. and Arimura, H.: Sparse and Truncated Suffix Trees on Variable-Length Codes, *Combinatorial Pattern Matching* (Giancarlo, R. and Manzini, G., eds.), Berlin, Heidelberg, Springer Berlin Heidelberg, pp. 246–260 (2011).

[13] 鳥谷部直弥，喜田拓也：データストリームに対する効率良い頻出アイテム発見アルゴリズム，第 11 回データ工学と情報マネジメントに関するフォーラム (DEIM) (2019).