Fast substring searching by the q-gram distance

q-gram距離基準による類似文字列検索の高速化

Hiroyuki HANADA
(Information Science and Technology, Hokkaido University)
e-mail: hana-hiro@live.jp

Published as:
Hiroyuki Hanada, Mineichi Kudo and Atsuyoshi Nakamura, "Average-case linear-time similar substring searching by the q-gram distance." Theoretical Computer Science, 530 (2014), 23-41.

Similar Substring Searching
類似文字列検索

Reflects real-life situations
実世界を反映するのに役立つ

Strings often represent the same meaning even if they are not exactly the same.
文字列は、完全一致でなくても意味は同じという例は多い。

- Text (mistakes and variants in spelling)
- Signal (noise)
- Genome (individuality, mutation) etc.

In addition, the amount of string data on computers are steadily increasing.
コンピュータで扱える、これらの文字列データが増え続けている。

- Data in the internet
- Development in genome sequencing

→ Want to find or analyze similar strings fast. 文字列を高速かつ、類似まで含め検索・分析する技術の需要が大きい。

q-gram distance
q-gram距離

A string similarity computed fast
高速に計算可能な文字列間距離

The sum of differences (L_1 distance) of q-grams found in two strings
二つの文字列のq-gramの出現数の差(i.e. L_1距離)と定義

- Fast computation: $O(|x|+|y|)$ time ($|\cdot|$: string length)
- Approximates the edit distance [Ukkonen 1992][Bar-Yossef 2004]
- q-grams are informative enough e.g. String kernel [Lodhi 2002][Leslie 2002]

∴ Fit to applications in the left

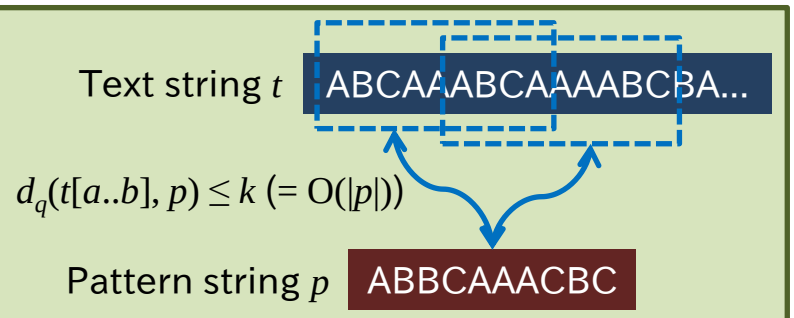
x = "ABABBA"
y = "BABBBA"

2-gram	#appear (x)	#appear (y)	diff.
AA	0	0	0
AB	2	1	1
BA	2	2	0
BB	1	2	1

2-gram distance: 0+1+0+1 = 2

Results and problems in existing research
従来研究の結果と課題

- Single distance computation: $O(|x|+|y|)$ time
  1回の距離計算
- Similar substring searching: $O(|t|\log k+|p|)$ time [Ukkonen 1992]
  類似文字列検索
  → Can it be reduced?



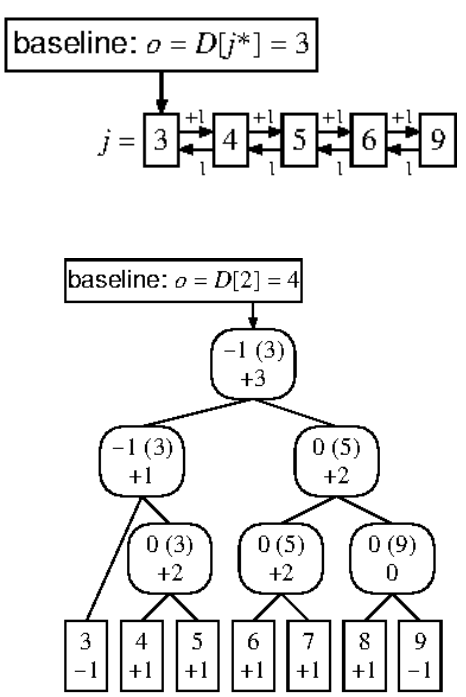
Proposed Method: $O(|t|+|p|)$ time on average with the same worst-case
最悪評価を引き上げずに、平均$O(|t|+|p|)$時間を達成

Improved point
従来からの改善点

Using a linked list instead of a search tree
探索木を連結リストに置き換え

For each $i \in \{1, 2, ..., |t|\}$: beginning of t, there exists $2k+1$ substring candidates. We have to "select an element to update the distance" and "remove an element" for each i.

	Existing	Proposed
Selection/Distance update (for $O(\|t\|/\log k)$ times on average, for $O(\|t\|)$ times in the worst case)	$O(\log k)$ time	$O(\log k)$ time
Removal (for $O(\|t\|)$ times)	$O(\log k)$ time	Avg.: $O(1)$ time Worst: $O(\log k)$ time



Proposed:
Linked list of size $O(k)$ with index.
Selection - $O(\log k)$ time[1]
Removal - $O(1)$ time (avg.), $O(\log k)$ time (worst)[2]

Existing:
Search tree of size $O(k)$.
Selection - $O(\log k)$ time
Removal - $O(\log k)$ time

1. Thanks to the index ($O(k)$ if no index!)
2. To update the index, $O(1)$ time is not always enough.