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

HANADA, Hiroyuki 花田 博幸
（ERATO Minato Project / Hokkaido University）
hana-hiro@live.jp

About me
自己紹介

Hiroyuki HANADA

This is the first participation for ERATO camp.

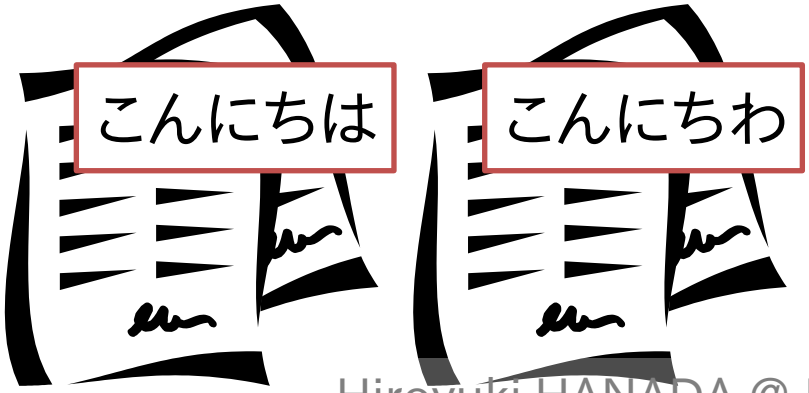
- August 2014 - (current):
  Post-doc researcher, ERATO Minato Project
- May 2014 - July 2014:
  Research assistant, ERATO Minato Project
- until March 2014:
  Doctor course student,
  Laboratory for Pattern Recognition and Machine Learning, Graduate School of Information Science and Technology, Hokkaido University
  (Supervisor: Prof. Mineichi KUDO)

Aim of the research

Aim of the research (1)
研究の目的

- An important task of analyzing string data is to find strings with similar functions
- To achieve this, we consider finding similar strings in character appearances/ordering
  文字の出現/並びとして似ている文字列を見つけたい

[variants in spelling]



[genome analysis]

ATCGCGTATGCGACTCA

ATCAACGCGTATTAAGC

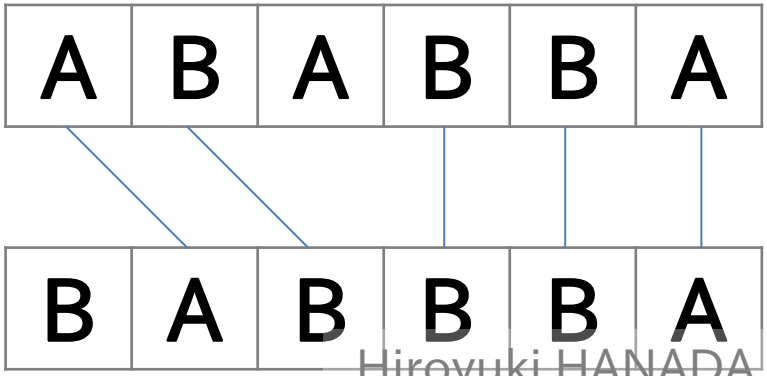
ATCAACGAAGCGTAAGC

Aim of the research (2)
研究の目的

In such applications, alignment-based similarity (e.g. edit distance) is advantageous in the coordination for the tasks but slow

→ To find string similarities with fast computation and similar value to the edit distance

編集距離などアラインメント系の類似度は有用だが遅いので、それと似た傾向を示す距離でより高速に計算できるものを検討



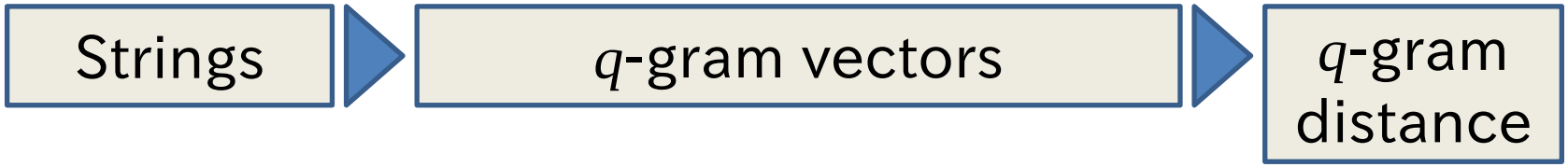
Edit Distance: 2
(#unmatched locations)
Time complexity:
$O(|x||y|)$ ($|\cdot|$: string length)

The q-gram distance and the substring searching

q-gram distance (1)

Distance between two strings defined by the sum of the difference of q-gram appearances

[Ukkonen 1992]



ABABBA

BABBBA

2-gram	#appear
AA	0
AB	2
BA	2
BB	1

2-gram	#appear
AA	0
AB	1
BA	2
BB	2

distance

$= |0 - 0|$

$+|2 - 1|$

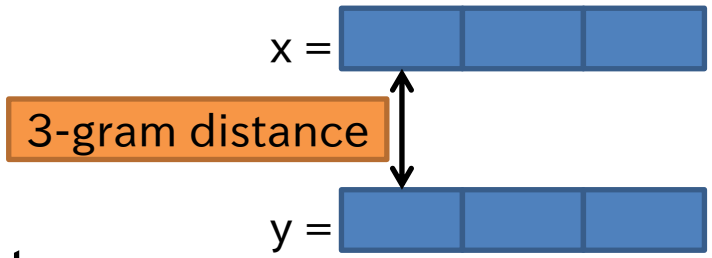
$+|2 - 2|$

$+|1 - 2|$

$= \mathbf{2}$

q-gram distance (2)

- Bounding the value of the edit distance
  編集距離の値の範囲を限定する
  $d_e(x, y) \geq d_q(x, y)/2q$ for any strings x, y
- Approximating the edit distance
  編集距離を近似する
  [Bar-Yossef 2004] Splitting strings into non-overlapping q parts and then sum up the q-gram distance for the parts
  [Sokolov 2007]
  Multiple q and overlapping parts
  - Approximation factor about 2 in experiments [Hanada 2011]



q-gram distance (3)

Related concepts

- Jumbled pattern matching [Eres 2004][Burcsi 2011]
  Searching substrings of the q-gram distance with q=1 and distance 0
  e.g. “abcdefg” includes “cdb” in the 2nd to the 4th characters in the sense of jumbled pattern matching
- String kernel [Leslie 2002][Lodhi 2002]
  Take the inner product rather than L^1-distance for the q-gram vector or another characteristic vector

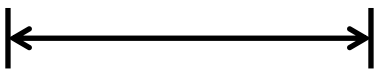
Substring searching by the q-gram distance
q-gram距離基準での部分文字列検索

Given: q, text string t, pattern string p and the upper bound of the distance k

Output: enumerate the locations of substrings of t whose q-gram distance to p is k or less

$p =$ | A | B | B | C | (Example of $q = 1, k = 1$)

$t =$ | A | B | C | A | B | A | B | C | A | B | C |



Examine substrings of t whose lengths are between $|p|-k$ and $|p|+k$. (Here: 3 to 5)
Otherwise distances must be more than k.

Substring searching by the q-gram distance
q-gram距離基準での部分文字列検索

Given: q, text string t, pattern string p and
the upper bound of the distance k
Output: enumerate the locations of substrings of t
whose q-gram distance to p is k or less

$p =$ | A | B | B | C | (Example of $q = 1, k = 1$)

$t =$ | A | B | C | A | B | A | B | C | A | B | C |

1-gram	出現数
A	1
B	1
C	1

Distance to p: 1
→ Output as an answer

Substring searching by the q-gram distance
q-gram距離基準での部分文字列検索

Given: q, text string t, pattern string p and
the upper bound of the distance k
Output: enumerate the locations of substrings of t
whose q-gram distance to p is k or less

$p =$ | A | B | B | C | (Example of $q = 1, k = 1$)

$t =$ | A | B | C | A | B | A | B | C | A | B | C |

1-gram	出現数
A	2
B	1
C	1

Distance to p: 2
→ Not an answer

Substring searching by the q-gram distance
q-gram距離基準での部分文字列検索

Given: q, text string t, pattern string p and
the upper bound of the distance k
Output: enumerate the locations of substrings of t
whose q-gram distance to p is k or less

$p =$ | A | B | B | C | (Example of $q = 1, k = 1$)

$t =$ | A | B | C | A | B | A | B | C | A | B | C |

1-gram	出現数
A	2
B	2
C	1

Distance to p: 1
→ Output as an answer

Time complexities for the q-gram distance
q-gram距離の時間計算量

- Computing the distance between two strings x, y 二つの文字列x, yの間の距離を求める
  - ➢in the ideal $O(|x|+|y|)$ time with the help of a suffix tree [Ukkonen 1992] ※$|\cdot|$: string length
- Finding substring appearances in a string t whose q-gram distance to a string p is less than certain value. 部分文字列を検索する
  - Some open problems
  - Is $O(|t|+|p|)$-time searching possible?
  - Is $o(|t|)$-time searching possible with an index of appropriate time/space?

Outline of results

Substring searching by the q-gram

- in $O(|t|+|p|)$ time
  - Previous: $O(|t|\log|p|+|p|)$ time [Ukkonen 1992]
  - Proposed:
    $O(|t|+|p|)$ (average), $O(|t|\log|p|+|p|)$ (worst)
- in $o(|t|)$ time with an index for t
  - Current status

Substring searching without indices

Substring searching without indices (1)

- There exists $O(|t|\log k + |p|) = O(|t|\log|p| + |p|)$-time algorithm [Ukkonen 1992]
- but no more improvement had been conducted

Why not $k = \Theta(1)$ but $k = \Theta(|p|)$?
⇒ Some applications require $k = \Theta(|p|)$
- The condition "k is **% of the pattern length" is often used
- The task of the nearest neighbor searching requires $k = |p|$ (The maximum q-gram distance)

Substring searching without indices (1)

- There exists $O(|t|\log k + |p|) = O(|t|\log|p| + |p|)$-time algorithm [Ukkonen 1992]
- but no more improvement had been conducted
- Proposed: $O(|t|+|p|)$-time on average in case $q = \Omega(\log_{|\Sigma|}|p|)$ and t is randomly chosen (worst: $O(|t|\log k + |p|)$) [Hanada 2014]
  With Mineichi KUDO and Atsuyoshi NAKAMURA (Hokkaido Univ.)

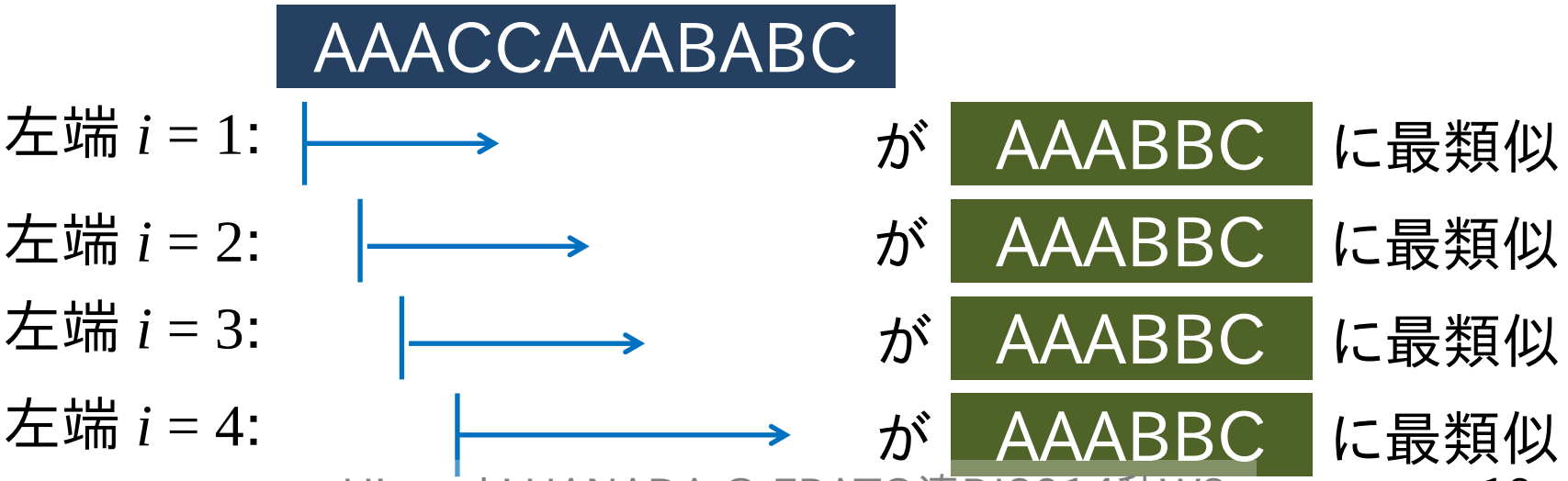
Σ: Alphabet (the set of characters for strings)
"t is randomly chosen" means that: (1) fixing the length of t beforehand and then (2) choosing every character in t i.i.d. from Σ in equal probability

Substring searching without indices (2)

Strategy of (Ukkonen's|proposed) algorithm:

For each left end of t, find the right end of t such that the substring is most similar to p.

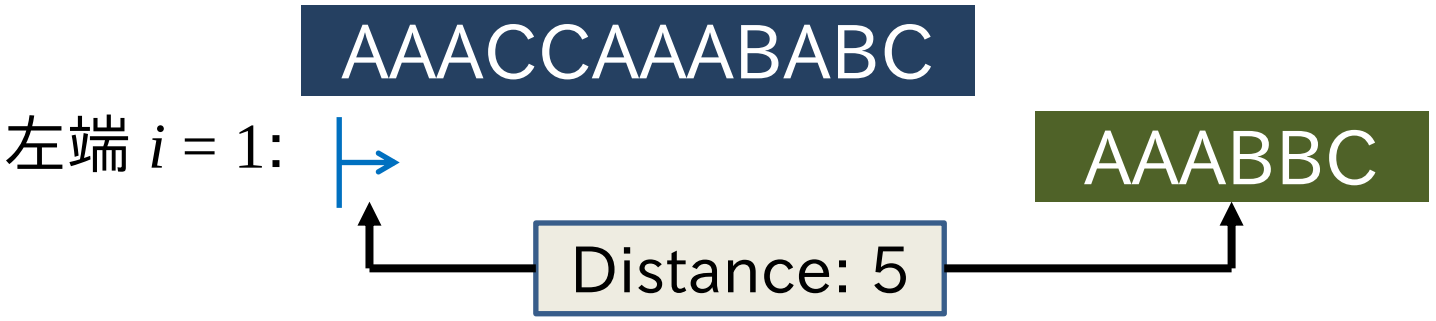
左端の位置ごとに、pとの距離が最小になる右端を得る



Substring searching without indices (2)

Strategy of (Ukkonen's|proposed) algorithm:

For each left end of t, find the right end of t such that the substring is most similar to p.

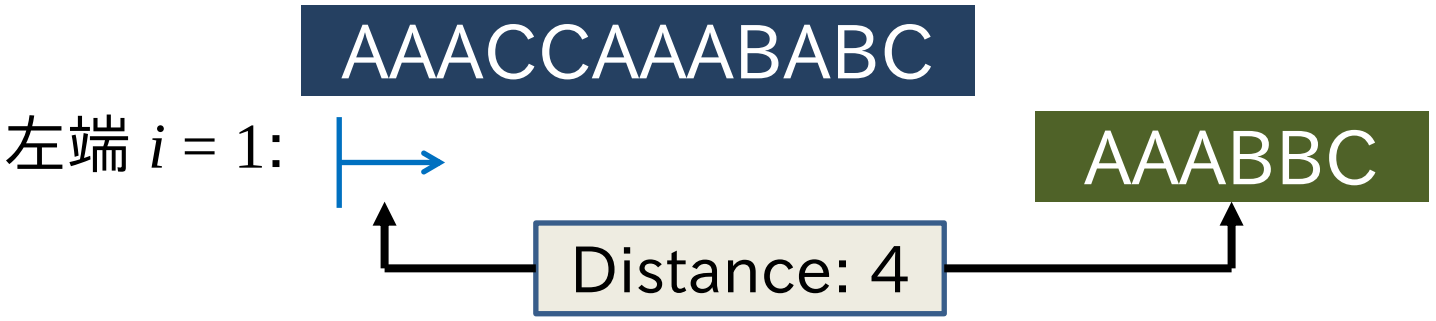


	tの部分文字列のpとの距離 $d_q(p, t[i..j])$					
	$j = 1$	$j = 2$	$j = 3$	$j = 4$	$j = 5$	...
$i = 1$	5					

Substring searching without indices (2)

Strategy of (Ukkonen's|proposed) algorithm:

For each left end of t, find the right end of t such that the substring is most similar to p.

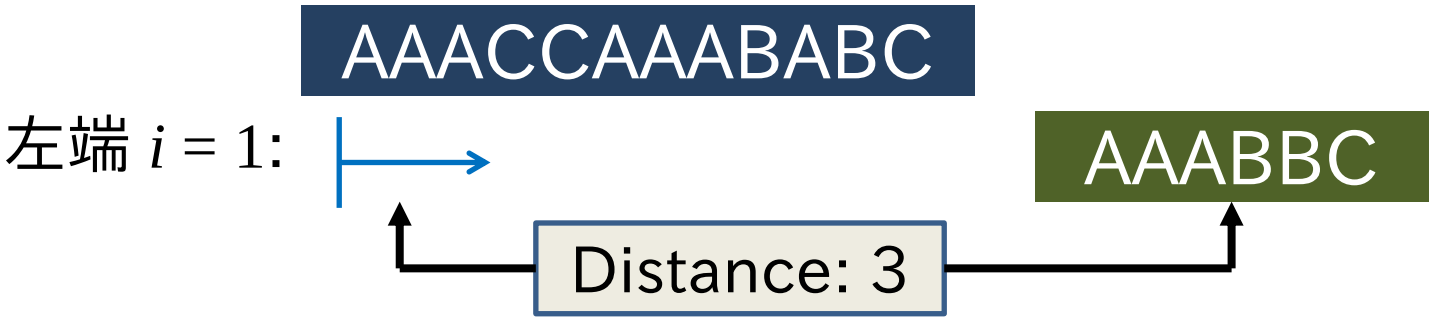


	tの部分文字列のpとの距離 $d_q(p, t[i..j])$					
	$j = 1$	$j = 2$	$j = 3$	$j = 4$	$j = 5$	...
$i = 1$	5	4				

Substring searching without indices (2)

Strategy of (Ukkonen's|proposed) algorithm:

For each left end of t, find the right end of t such that the substring is most similar to p.

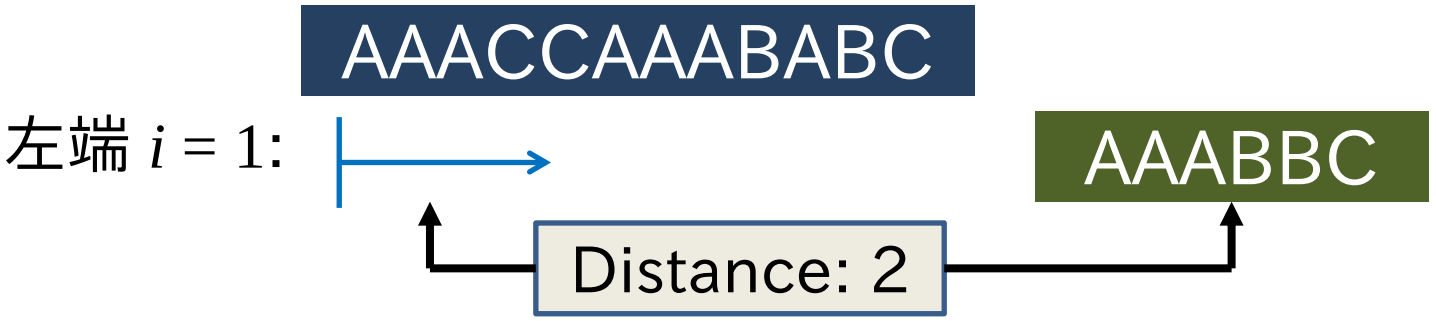


	tの部分文字列のpとの距離 $d_q(p, t[i..j])$					
	$j = 1$	$j = 2$	$j = 3$	$j = 4$	$j = 5$	...
$i = 1$	5	4	3			

Substring searching without indices (2)

Strategy of (Ukkonen's|proposed) algorithm:

For each left end of t, find the right end of t such that the substring is most similar to p.

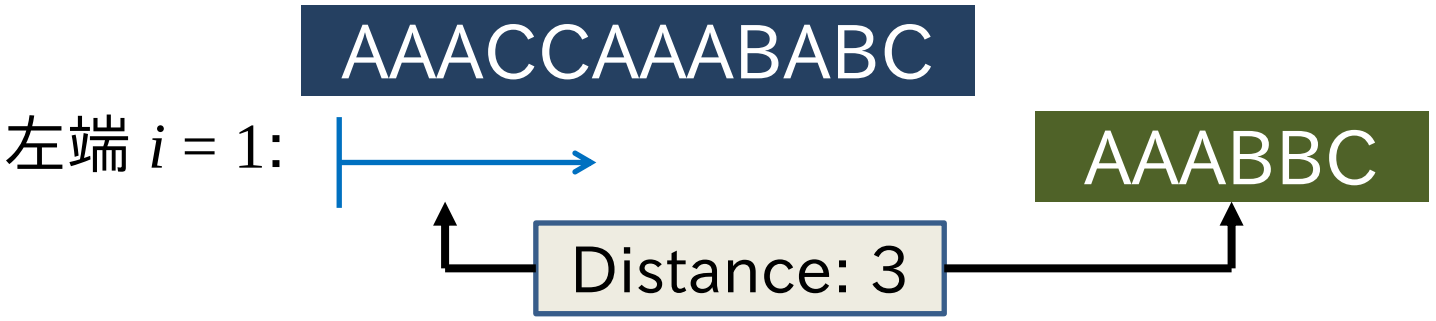


	tの部分文字列のpとの距離 $d_q(p, t[i..j])$					
	$j = 1$	$j = 2$	$j = 3$	$j = 4$	$j = 5$	...
$i = 1$	5	4	3	2		

Substring searching without indices (2)

Strategy of (Ukkonen's|proposed) algorithm:

For each left end of t, find the right end of t such that the substring is most similar to p.



	tの部分文字列のpとの距離 $d_q(p, t[i..j])$					
	$j = 1$	$j = 2$	$j = 3$	$j = 4$	$j = 5$	...
$i = 1$	5	4	3	2	3	

Substring searching without indices (2)

Strategy of (Ukkonen's|proposed) algorithm:

For each left end of t, find the right end of t such that the substring is most similar to p.

	tの部分文字列のpとの距離 $d_q(p, t[i..j])$											
	$j = 1$	2	3	4	5	6	7	8	9	10	11	12
$i = 1$	5	4	3	2	3	4	5	6	5	6	5	6

左端の位置	右端の位置j					
	距離最小	+1	+2	+3	+4	+5
$i = 1$	4	5	6	11	12	

Substring searching without indices (2)

Strategy of (Ukkonen's|proposed) algorithm:

For each left end of t, find the right end of t such that the substring is most similar to p.

	tの部分文字列のpとの距離 $d_q(p, t[i..j])$											
	$j = 1$	2	3	4	5	6	7	8	9	10	11	12
$i = 2$	6	5	4	3	4	3	4	5	4	5	4	5

左端の位置	右端の位置j					
	距離最小	+1	+2	+3	+4	+5
$i = 2$	6	11	12			

Substring searching without indices (2)

Strategy of (Ukkonen's|proposed) algorithm:

For each left end of t, find the right end of t such that the substring is most similar to p.

	tの部分文字列のpとの距離 $d_q(p, t[i..j])$											
	$j = 1$	2	3	4	5	6	7	8	9	10	11	12
$i = 3$		6	5	4	3	4	3	4	3	4	3	4

左端の位置	右端の位置j					
	距離最小	+1	+2	+3	+4	+5
$i = 3$	11	12				

Substring searching without indices (3)

Key of the proposed algorithm:

Updating the candidates of the right ends in $O(1)$-time (average)/$O(\log k)$-time (worst)

左端の位置	右端の位置j					
	距離最小	+1	+2	+3	+4	+5
$i = 1$	4	5	6	11	12	
$i = 2$	6	11	12			
$i = 3$	11	12				

worst: $O(\log k)$ time

worst: $O(\log k)$ time

average: $O(1)$ time

Substring searching without indices (3)

Key of the proposed algorithm:

Updating the candidates of the right ends

in $O(1)$-time (average)/$O(\log k)$-time (worst)

Data structure: linked list + search tree

左端の位置	右端の位置j					
	距離最小	+1	+2	+3	+4	+5
$i = 1$	4	5	6	11	12	

Substring searching without indices (conclusion)

Time complexity for the substring searching by the q-gram distance

- Existing: $O(|t|\log|p| + |p|)$
- Proposed: $O(|t|\log|p| + |p|)$ (worst)
  $O(|t|+|p|)$ (average)

Is $O(|t|+|p|)$-time in the worst case possible?

- This algorithm needs $O(\log k)$ time to find an integer from a set in a search tree. How to improve?

Substring searching with indices

Substring searching with indices (1)

Does there exist an index constructed in $O(|t|)$ time and space and achieves $o(|t|)$-time substring searching by the q-gram distance?

⇒ Hard even if $k = 0$ (probably)

(algorithms for $k = 0$)	Time for searching	Time for indexing	space for index
q-gram vectors of all substrings of t in a kd-Tree	$O(\log\|t\| \cdot \|\Sigma\|^q)$	$O(\|t\|^2\|\Sigma\|^q)$	$O(\|t\|^2\|\Sigma\|^q)$
Store numbers of appear-ances of $\mathcal{P}(\Sigma^q)$ for all substrings of t [Burcsi 2011]	$O(2^{\|\Sigma\|^q})$	$O(\|t\|^2 \cdot 2^{\|\Sigma\|^q})$	$O(\|t\| \cdot 2^{\|\Sigma\|^q})$

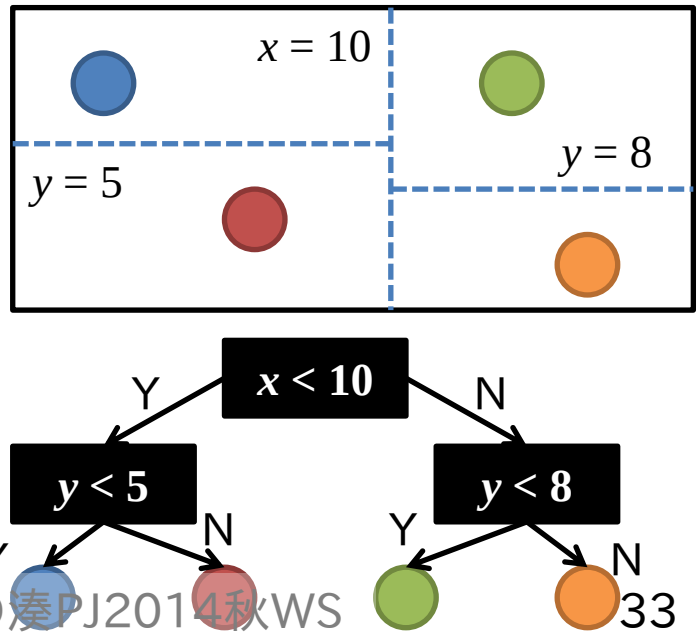
Substring searching with indices (2)

(algorithms for $k = 0$)	Time for searching	Time for indexing	space for index
q-gram vectors of all substrings of t in a kd-Tree	$O(\log\|t\| \cdot \|\Sigma\|^q)$	$O(\|t\|^2\|\Sigma\|^q)$	$O(\|t\|^2\|\Sigma\|^q)$

kd-tree: A search tree for Euclidean spaces [Bentley 1975]

- Examine existence: $O(d\log n)$ time
- Nearest neighbor searching: $O(d\log n) \sim O(dn)$ time

(n: #elements, d: #dimensions)



Substring searching with indices (3)

(algorithms for $k = 0$)	Time for searching	Time for indexing	space for index
Store numbers of appear-ances of $\mathcal{P}(\Sigma^q)$ for all substrings of t [Burcsi 2011]	$O(2^{\lvert\Sigma\rvert^q})$	$O(\lvert t\rvert^2 \cdot 2^{\lvert\Sigma\rvert^q})$	$O(\lvert t\rvert \cdot 2^{\lvert\Sigma\rvert^q})$

An example of the index (table) ($\Sigma=\{a, b\}$, $q=2$)

- Numbers of 2-grams in length-5 substrings in t:
  “aa”: 0 to 2　“ab”: 0 to 2 “ba”: 0 to 2 “bb”: 0 to 1
  “aa” and “ab”: 1 to 3　“aa” and “ba”: 0 to 3
  “aa” and “bb”: 0 to 3
- Numbers of 2-grams in length-6 substrings in t:
  :

※Compute for the half of Σ^q

Substring searching with indices (4)

These algorithms need to enumerate substrings of t and thus needs $O(|t|^2)$ time.

How does it change with more efficient enumeration like the suffix tree? (May be hard?)

(algorithms for $k = 0$)	Time for searching	Time for indexing	space for index
q-gram vectors of all substrings of t in a kd-Tree	$O(\log\|t\| \cdot \|\Sigma\|^q)$	$O(\|t\|^2\|\Sigma\|^q)$	$O(\|t\|^2\|\Sigma\|^q)$
Store numbers of appear-ances of $\mathcal{P}(\Sigma^q)$ for all substrings of t [Burcsi 2011]	$O(2^{\|\Sigma\|^q})$	$O(\|t\|^2 \cdot 2^{\|\Sigma\|^q})$	$O(\|t\| \cdot 2^{\|\Sigma\|^q})$

Conclusion

Fast substring searching by the q-gram distance

【without indices】

- Current: $O(|t|+|p|)$ time on average, $O(|t|\log|p|+|p|)$ time in the worst case
- Problem: Reducing the worst-case complexity

【with indices】

- Hard even if $k = 0$ (the maximum distance)