Available online at www.sciencedirect.com
SCIENCE DIRECT®
Information Processing Letters 99 (2006) 19–26
Information Processing Letters
www.elsevier.com/locate/ipl

On the cost of searching signature trees

Yangjun Chen [1]

Department of Applied Computer Science, University of Winnipeg, Winnipeg, Manitoba, Canada R3B 2E9

Received 31 August 2005; received in revised form 14 October 2005; accepted 17 October 2005

Available online 11 April 2006

Communicated by L. Boasson

Abstract

A precise analysis of the retrieval of signature trees is presented. A signature tree is a data structure constructed over a signature file to speed up searching all those signatures, which match a given query signature. The methods used include a detailed study of probabilistic analysis in conjunction with suitable contour integration of complex variabled functions.

Keywords: Data structures; Index; Signature files; Signature identifiers; Signature trees; Probabilistic analysis; Contour integration

1. Introduction

The signature file method has been widely advocated as an efficient index schema to handle large volumes of textual databases and recently extended to support a wide range of applications, such as multi-media, hypertext systems [11], relational and object-oriented databases [4,13,15,16], as well as data mining [1]. In comparison with the other index structures, it has mainly the following advantages:

- it can be used to efficiently evaluate set-oriented queries;
- it can handle insertion and update operations easily.

Intuitively, a signature file can be considered as a set of bit strings, called *signatures*. A typical query processing with the signature file is as follows: when a query is given, a query signature (a bit string) is formed from the query values. Then each signature in the signature file is examined over the query signature. If a signature in the file matches the query signature, the corresponding data object becomes a candidate that may satisfy the query. Such an object is called a drop. The next step of the query processing is the false drop resolution. Each drop is accessed and examined whether it actually satisfies the query condition. Drops that fail the test are called false drops while the qualified data objects are called actual drops. In general, for each query processed, the entire signature file needs to be searched [9,10]. Consequently, the signature file method involves high processing and I/O cost. This problem is mitigated by partitioning a signature file, by introducing auxiliary data structure, as well as by exploiting parallel computer architectures [7]. Recently, a new data structure, called a *signature tree* [5], is proposed to get rid of useless signature comparisons. However, no precise analysis has been given to show the benefits brought by it. In this paper, we give a probabilistic analysis of this problem and

E-mail address: ychen2@uwinnipeg.ca (Y. Chen).

[1] The author is supported by NSERC 239074-01 (242523) (Natural Sciences and Engineering Council of Canada).

0020-0190/$ – see front matter
doi:10.1016/j.ipl.2005.10.015

object:	John	12345678	professor

attribute signature:		
John		010 000 100 110
12345678		100 010 010 100
professor	∨	010 100 011 000
object signature (OS)		110 110 111 110

queries:	query signatures:	matching results:
John	010 000 100 110	match with OS
Paul	011 000 100 100	no match with OS
11223344	110 100 100 000	false drop

Fig. 1. Signature generation and comparison.

show that for a file containing n signatures, only $n^{1-b/k}$ signatures need to be checked by using the signature tree built over it, where k and b represent the length of the query signature and its weight (number of 1s appearing in the signature), respectively.

The remainder of the paper is organized as follows. In Section 2, we show what is a signature file and what is a signature tree. In Section 3, we analyze the average number of nodes checked during a signature tree searching, Section 4 is a short conclusion.

2. Signature files and signature trees

In this section, we give a brief description of signature files and signature trees to provide a discussion background.

2.1. Signature files

Signature files are based on the inexact filter. They provide a quick test, which discards many of the nonqualifying elements. But the qualifying elements definitely pass the test although some elements which actually do not satisfy the search requirement may also pass it accidentally, i.e., there may exist "false hits" or "false drops" [9,10]. In an object-oriented database, for instance, an object is represented by a set of attribute values. The signature of an attribute value is a hash-coded bit string of length k with b bit set to "1". As an example, assume that we have an attribute value "professor". Its signature can be constructed as follows. In terms of [3], the letter triplets in a word (or an attribute value) are the best choice for information carrying text segments in the construction of the signature for that word. So we decompose "professor" into a series of triplets: "pro", "rof", "ofe", "fes", "ess", and "sor". Using a hash function *hash*, we will map a triplet to an integer p indicating that the pth bit in the string will be set to 1. For example, assume that we have $hash(\text{pro}) = 2$, $hash(\text{rof}) = 4$, $hash(\text{ofe}) = 8$, and $hash(\text{fes}) = 9$. Then, we will establish a bit string: 010 100 011 000 for "professor" as its word signature (see [8] for a detailed discussion).

An object signature is formed by superimposing the signatures for all its attribute values. (By 'superimposing', we mean a bit-wise OR operation.) Object signatures of a class will be stored sequentially in a file, called a *signature file*. Fig. 1 depicts the signature generation and comparison process of an object having three attribute values: "John", "12345678", and "professor".

When a query arrives, the object signatures are scanned and many nonqualifying objects are discarded. The rest are either checked (so that the "false drops" are discarded) or they are returned to the user as they are. Concretely, a query specifying certain values to be searched for will be transformed into a query signature s_q in the same way as for attribute values. The query signature is then compared to every object signature in the signature file. Three possible outcomes of the comparison are exemplified in Fig. 1:

(1) the object matches the query; that is, for every bit set in s_q, the corresponding bit in the object signature s is also set (i.e., $s \wedge s_q = s_q$) and the object contains really the query word;
(2) the object does not match the query (i.e., $s \wedge s_q \neq s_q$); and
(3) the signature comparison indicates a match but the object in fact does not match the search criteria (false drop).

In order to eliminate false drops, the object must be examined after the object signature signifies a successful match.

In addition, we can see that the signature matching is a kind of inexact matching. That is, s_q matches a signature s if for any bit set to 1 in s_q, the corresponding bit in s is also set to 1. However, for any bit set to 0 in s_q, it does not matter whether the corresponding bit in s is set to 1 or 0.

The purpose of using a signature file is to screen out most of the nonqualifying objects. A signature failing to match the query signature guarantees that the corresponding object can be ignored. Therefore, unnecessary object access is prevented.

signature file:	OIDs:
1 0 1 0 1 0 0 1	o_1
0 1 1 0 0 0 1 1	o_2
0 0 1 0 1 1 0 1	o_3
1 1 1 0 1 0 0 0	o_4
0 0 1 1 1 0 0 1	o_5
1 1 1 0 0 0 1 0	o_6
0 1 0 1 0 0 1 1	o_7
0 1 0 1 0 1 1 0	o_8

Fig. 2. Illustration of sequential.

To determine the size of a signature file, we use the following formula [3]:

$$k \times \ln 2 = b \times D,$$

where D is the average size of a block. (In a relational or an object-oriented database, D can be considered to be the average number of attributes in a tuple or in an object.)

In a signature file, a set of signatures is sequentially stored, which is easy to implement and requires low storage space and low update cost. However, when a query is given, a full scan of the signature file is required. Therefore, it is generally slow in retrieval. Fig. 2 is a quite simple signature file. If more than one objects share a same signature, that signature will be associated with the identifiers of all those objects.

2.2. Signature trees

In [5], a new method was proposed to organize signature files to speed up a signature file scanning. Using this method, a tree over a signature file S, called a signature tree, is constructed with the following properties.

(1) Each node v is associated with a number to tell which bit in s_q to check when v is encountered during the tree searching.
(2) For each node, its left outgoing edge is labeled with 0 and its right outgoing edge is labeled with 1.
(3) Each path from the root to a leaf represents a signature identifier that uniquely identifies a signature in S just as a *position identifier* used to identify a substring [2]. A signature identifier is defined as follows. Let $S = s_1 s_2 \ldots s_n$ denote a signature file. Let $s_i[j]$ represent the jth bit in s_i. The signature identifier for an s_i is a sequence of pairs: $(j_1, s_i[j_1])(j_2, s_i[j_2]) \ldots (j_h, s_i[j_h])$ $(1 \leqslant j_l \leqslant k$; denoted $s_i(j_1, \ldots, j_h))$ such that for any $l \neq i$ $(1 \leqslant l \leqslant n)$ we have $s_i(j_1, \ldots, j_h) \neq s_l(j_1, \ldots, j_n)$.

Example 1. In Fig. 3(b), we show a signature tree for the signature file shown in Fig. 3(a). In this signature tree, each edge is labeled with 0 or 1 and each leaf node is a pointer to a signature in the signature file. In addition, each internal node is associated with a positive integer (which is used to tell how many bits to skip when searching). Consider the path going through the nodes marked 1, 7 and 4. If this path is searched for locating all those signatures that match the query signature s, three bits of s: $s[1]$, $s[7]$ and $s[4]$ must be checked during the searching process. If $s[4] = 1$, the search will go to the right child of the node marked "4". This child node is marked with 5 and then the 5th bit of s: $s[5]$ will be checked.

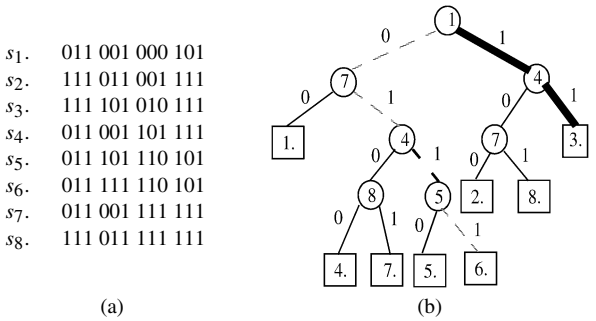

Fig. 3. A signature tree.

See the path consisting of the dashed edges in Fig. 3(b), which corresponds to the identifier of s_6: $s_6(1, 7, 4, 5) = (1, 0)(7, 1)(4, 1)(5, 1)$. Similarly, the identifier of s_3 is $s_3(1, 4) = (1, 1)(4, 1)$ (see the path consisting of the thick edges in Fig. 3(b)).

According to the construction of signature trees, the searching of a signature tree can be defined as follows. Let s_q be a query signature. The ith position of s_q is denoted as $s_q[i]$. During the traversal of a signature tree, the inexact matching is defined as follows:

(i) Let v be the node encountered and $s_q[i]$ be the position to be checked.
(ii) If $s_q[i] = 1$, we move to the right child of v.
(iii) If $s_q[i] = 0$, both the right and left child of v will be visited.

In fact, this definition just corresponds to the signature matching criterion.

Example 2. Consider the signature file and the signature tree shown in Fig. 3 once again.

Assume $s_q = 000\,100\,100\,000$. Then, only part of the signature tree (marked with thick edges in Fig. 4) will be searched. On reaching a leaf node, the signature pointed by the leaf node will be checked against s_q. Obviously, this process is much more efficient than a sequential searching. For this example, only 42 bits are checked

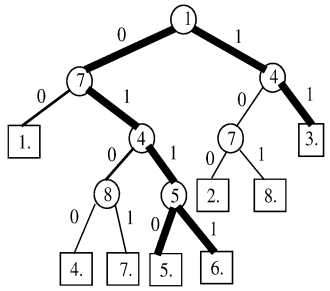

Fig. 4. Signature tree search.

(6 bits during the tree search and 36 bits during the signature checking). But by the scanning of the signature file, 96 bits will be checked.

In the following, we will give a probabilistic analysis to estimate the average number of bits to be checked during a signature tree traversal.

3. Average number of checked bits

Denote by t a signature tree, in which the edges are labeled with 0, or 1. Let $s = s[1]s[2]\dots s[k]$ be a query signature, where $s[i] \in \{0, 1\}$. Then, 1 in s matches only 1 in t while 0 in s matches both 0 and 1 in t. We use $c_s(t)$ to represent the cost of searching t against s. In addition, we use $s', s'', s''', \dots$ to designate the patterns obtained by circularly shifting the bits of s to the left by $1, 2, 3, \dots$ positions. Obviously, if the first bit of s is 0, we have, for the expected cost of a random string s,

$$c_s(t) = 1 + c_{s'}(t_1) + c_{s'}(t_2), \tag{1}$$

where t_1 and t_2 represent the two subtrees of the root of t. See Fig. 5 for illustration.

It is because in this case, the search has to proceed in parallel along the two subtrees with s changing cyclically to s'.

If, contrariwise, the first bit in s is 1, we find

$$c_s(t) = 1 + c_{s'}(t_2) \tag{2}$$

since in this case the search proceeds only in t_2.

Given n ($n \geqslant 2$) random nodes in t, the probability that

$$|t_1| = p, \qquad |t_2| = n - p \tag{3}$$

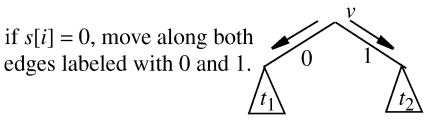

Fig. 5. Illustration for signature tree searching.

is given by the Bernoulli probabilities

$$\binom{n}{p}\left(\frac{1}{2}\right)^p\left(\frac{1}{2}\right)^{n-p} = \frac{1}{2^n}\binom{n}{p}. \tag{4}$$

Let $c_{s,n}$ denote the expected cost of searching a tree of size n against s. We have the following recurrences

if s starts with 0,

$$c_{s,n} = 1 + \frac{2}{2^n}\sum_p \binom{n}{p} c_{s',p}, \quad n \geqslant 2; \tag{5}$$

if s starts with 1,

$$c_{s,n} = 1 + \frac{1}{2^n}\sum_p \binom{n}{p} c_{s',p}, \quad n \geqslant 2. \tag{6}$$

Let $\lambda_i = 1$ if ith bit in s is 1, and $\lambda_i = 2$ if ith bit in s is 0. The above recurrence can be rewritten as follows

$$c_{s,n} = 1 + \frac{\lambda_1}{2^n}\sum_p \binom{n}{p} c_{s',p} - \delta_{n,0} - \delta_{n,1}, \tag{7}$$

where $\delta_{n,j}$ $(j = 0, 1)$ is equal to 1 if $n = j$; otherwise equal to 0.

Proposition 1. *The exponential generating function of the average cost $c_{s,n}$*

$$C_s(z) = \sum_{n\geqslant 0} c_{s,n}\frac{z^n}{n!} \tag{8}$$

satisfies the relation

$$C_s(z) = \lambda_1 e^{z/2} C_{s'}\left(\frac{z}{2}\right) + e^z - 1 - z. \tag{9}$$

Proof. In terms of Eq. (6), $C_s(z)$ can be rewritten as follows

$$\begin{aligned} C_s(z) &= \sum_{n\geqslant 0}\left(1 + \lambda_1\left(\frac{1}{2}\right)^n \sum_p \binom{n}{p} c_{s',p} - \delta_{n,0} - \delta_{n,1}\right)\frac{z^n}{n!} \\ &= \sum_{n\geqslant 0}\frac{z^n}{n!} + \sum_p \lambda_1\left(\frac{1}{2}\right)^n \sum_{n\geqslant 0}\binom{n}{p} c_{s',p}\frac{z^n}{n!} - \sum_{n\geqslant 0}\delta_{n,0}\frac{z^n}{n!} - \sum_{n\geqslant 0}\delta_{n,1}\frac{z^n}{n!} \\ &= e^z + \lambda_1 \sum_p \frac{(z/2)^p}{p!}\sum_{n\geqslant p} c_{s',p}\frac{(z/2)^{n-p}}{(n-p)!} - 1 - z \\ &= \lambda_1 e^{z/2} C_{s'}\left(\frac{z}{2}\right) + e^z - 1 - z. \end{aligned} \tag{10}$$

In the same way, we will get $C_{s'}(z), C_{s''}(z), \dots$, and so on. Concretely, we will have the following equations:

$$
\begin{aligned}
C_s(z) &= \lambda_1 e^{z/2} C_{s'}\left(\frac{z}{2}\right) + e^z - 1 - z,\\
C_{s'}(z) &= \lambda_2 e^{z/2} C_{s''}\left(\frac{z}{2}\right) + e^z - 1 - z,\\
&\cdots\\
C_{s^{(k-1)}}(z) &= \lambda_k e^{z/2} C_s\left(\frac{z}{2}\right) + e^z - 1 - z.
\end{aligned}
\tag{11}
$$

These equations can be solved by successive transportation. For instance, when we transport the expression of $C_{s'}(z)$ given by the second equation in (11), we have

$$
\begin{aligned}
C_s(z) &= a(z) + \lambda_1 e^{z/2} a\left(\frac{z}{2}\right)\\
&\quad + \lambda_1\lambda_2 e^{z/2} e^{z/2^2} C_{s''}\left(\frac{z}{2^2}\right),
\end{aligned}
\tag{12}
$$

where $a(z) = e^z - 1 - z$.

In a next step, we transport $C_{s'''}$ into the equation given in (12). This kind of transformation continues until the relation is only on C_s itself. Then, we have

$$
\begin{aligned}
C_s(z) &= \lambda_1\lambda_2\ldots\lambda_k \exp\left[z\left(1-\frac{1}{2^k}\right)\right] C_s\left(\frac{z}{2^k}\right)\\
&\quad + \sum_{j=0}^{k-1} \lambda_1\lambda_2\ldots\lambda_j \exp\left[z\left(1-\frac{1}{2^j}\right)\right]\\
&\quad \times \left(\exp\left(\frac{z}{2^j}\right) - 1 - \frac{z}{2^j}\right)\\
&= 2^{k-b}\exp\left[z\left(1-\frac{1}{2^k}\right)\right] C_s\left(\frac{z}{2^k}\right)\\
&\quad + \sum_{j=0}^{k-1} \lambda_1\lambda_2\ldots\lambda_j \exp\left[z\left(1-\frac{1}{2^j}\right)\right]\\
&\quad \times \left(\exp\left(\frac{z}{2^j}\right) - 1 - \frac{z}{2^j}\right),
\end{aligned}
\tag{13}
$$

where b is the number of 1s in s.

Let

$$
\alpha = 2^{k-b}, \qquad \beta = 1 - \frac{1}{2^k}, \qquad \lambda = \frac{1}{2^k} \quad \text{and}
$$

$$
\begin{aligned}
A(z) &= \sum_{j=0}^{k-1} \lambda_1\lambda_2\ldots\lambda_j \exp\left[z\left(1-\frac{1}{2^j}\right)\right]\\
&\quad \times \left(\exp\left(\frac{z}{2^j}\right) - 1 - \frac{z}{2^j}\right).
\end{aligned}
$$

We have

$$
C_s(z) = \alpha e^{\beta z} C_s(\lambda z) + A(z). \tag{14}
$$

This equation can be solved by iteration as discussed above:

$$
\begin{aligned}
C_s(z) &= \sum_{j=0}^{\infty} \alpha^j \exp\left(\beta\frac{1-\lambda^j}{1-\lambda} z\right) A(\lambda^j z)\\
&= \sum_{j=0}^{\infty} 2^{j(k-b)} \sum_{h=0}^{k-1} \lambda_1\lambda_2\ldots\lambda_h \Bigg[\exp(z)\\
&\quad - \exp\left(z\left(1-\frac{1}{2^h 2^{kj}}\right)\right)\left(1+\frac{z}{2^h 2^{kj}}\right)\Bigg].
\end{aligned}
\tag{15}
$$

Using *Taylor* formula to expand $\exp(z)$ and $\exp(z(1-\frac{1}{2^h 2^{kj}}))(1+\frac{z}{2^h 2^{kj}})$ in $C_s(z)$ given by the above sum, and then extract the *Taylor* coefficients, we get

$$
c_{s,n} = \sum_{h=0}^{k-1} \lambda_1\lambda_2\ldots\lambda_h \sum_{j\geqslant 0} 2^{j(k-b)} D_{jh}(n), \tag{16}
$$

where $D_{00}(n) = 1$ and for $j > 0$ and $h > 0$,

$$
\begin{aligned}
D_{jh}(n) &= 1 - (1 - 2^{-kj-h})^n\\
&\quad - x 2^{-kj-h}(1 - 2^{-kj-h})^{n-1}.
\end{aligned}
\tag{17}
$$

To estimate $c_{s,n}$, we resort to the complex analysis, which shows that $c_{s,n} \sim n^{1-b/k}$. If $\frac{b}{k} = \frac{1}{2}$, we have

$$
c_{s,n} = \mathrm{O}(n^{0.5}). \tag{18}
$$

In Appendix A, we will discuss how to evaluate $c_{s,n}$ given by (16) by using contour integration of complex variabled functions in great detail.

4. Conclusion

In this paper, the performance of signature trees is analyzed. By a probabilistic analysis, together with the contour integration of complex variabled functions, we show that the average number of the checked nodes in a signature tree is bounded by $\mathrm{O}(n^{1-b/k})$, where n is the number of signatures in the signature file, over which the signature tree is established, k is the length of a query signature and b is the number of 1s appearing in it.

Appendix A

In this appendix, we show how to evaluate $c_{s,n}$.

First, we define

$$
\phi(x) = \sum_{h=0}^{k-1} \lambda_1\lambda_2\ldots\lambda_h \sum_{j\geqslant 0} 2^{j(k-b)} D_{jh}(x) \quad (x \geqslant 0). \tag{A.1}
$$

Then, we perform the following computations to evaluate $\phi(x)$:

(1) Define the Mellin transformation of $\phi(x)$ [6, p. 453]:

$$\phi^*(\sigma) = \int_0^\infty \phi(x) x^{\sigma-1}\,\mathrm{d}x. \tag{A.2}$$

(2) Derive an expression for $\phi^*(\sigma)$, which reveals some of its singularities.
(3) Evaluate the reversal Mellin transformation

$$\phi(x) = \frac{1}{2\mathrm{i}\pi}\int_{c-\mathrm{i}\infty}^{c+\mathrm{i}\infty} \phi^*(\sigma)x^{-\sigma}\,\mathrm{d}\sigma,$$
$$-1 < c < -\left(1-\frac{b}{k}\right). \tag{A.3}$$

The integral (A.3) is evaluated by using Cauchy's theorem as a sum of *residues* to the right of the vertical line $\{c + \mathrm{i}y \mid y \in \Re\}$, where $\Re$ represents the set of all real numbers. This computation method was first proposed in [12]. The following is just an extended explanation of it.

Remember that $D_{jh}(x) = 1 - (1 - 2^{-kj-h})^x - x2^{-kj-h}(1-2^{-kj-h})^{x-1}$. We rewrite it under the form

$$D_{jh}(x) = 1 - \mathrm{e}^{-x\alpha_{jh}} - \beta_{jh}x\mathrm{e}^{-x\alpha_{jh}} \tag{A.4}$$

with $a_{jh} = -\log(1-2^{-kj-h})$ and $\beta_{jh} = 2^{-kj-h}(1-2^{-kj-h})^{-1}$.

Now we consider the following expansion, which is valid for small values of x:

$$\big(-\log(1-x)\big)^{-\sigma} = x^{-\sigma}\left(1 - \frac{x\sigma}{2} + \mathrm{O}\big(|\sigma|^2x^2\big)\right). \tag{A.5}$$

Let $x = 2^{-kj-h}$. Then, we have (by using the above expansion)

$$\alpha_{jh} = \big(-\log(1-2^{-kj-h})\big)^{-(-1)} \sim (2^{kj+h}). \tag{A.6}$$

In addition, for small values 2^{-kj-h}, we also have

$$\beta_{jh} = 2^{-kj-h}(1-2^{-kj-h})^{-1} = \mathrm{O}(2^{-kj}). \tag{A.7}$$

Following the classical properties of Mellin transformation, we have the following proposition.

Proposition 2. *Denote $D^*_{jh}(\sigma)$ the Mellin transformation of $D_{jh}(x)$. We have*

$$D^*_{jh}(\sigma) = \int_0^\infty D_{jh}(x)x^{\sigma-1}\,\mathrm{d}x$$
$$= -(\alpha_{jh})^{-\sigma}\Gamma(\sigma) - \beta_{jh}(\alpha_{jh})^{-\sigma-1}\sigma\Gamma(\sigma) \tag{A.8}$$

provided $-1 < \mathrm{Re}(\sigma) < 0$, where $\Gamma(\sigma)$ is the Euler Gamma function.

Proof. The following formulas are well known:

$$\int_0^\infty (\mathrm{e}^{-x}-1)x^{\sigma-1}\,\mathrm{d}x = \Gamma(\sigma), \quad -1 < \mathrm{Re}(\sigma) < 0, \tag{A.9}$$

$$\int_0^\infty (x\mathrm{e}^{-x})x^{\sigma-1}\,\mathrm{d}x = \sigma\Gamma(\sigma), \quad -1 < \mathrm{Re}(\sigma), \tag{A.10}$$

$$\int_0^\infty f(ax)x^{\sigma-1}\,\mathrm{d}x = a^{-\sigma}\int_0^\infty f(x)x^{\sigma-1}\,\mathrm{d}x \quad \text{for } a > 0. \tag{A.11}$$

In terms of these formulas, we have

$$D^*_{jh}(\sigma) = \int_0^\infty D_{jh}(x)x^{\sigma-1}\,\mathrm{d}x$$
$$= \int_0^\infty (1-\mathrm{e}^{-x\alpha_{jh}})x^{\sigma-1}\,\mathrm{d}x$$
$$- \int_0^\infty \beta_{jh}x\mathrm{e}^{-x\alpha_{jh}}x^{\sigma-1}\,\mathrm{d}x$$
$$= -(\alpha_{jh})^{-\sigma}\Gamma(\sigma)$$
$$- \beta_{jh}(\alpha_{jh})^{-\sigma-1}\sigma\Gamma(\sigma). \tag{A.12}$$

Now we try to evaluate the following two sums:

$$\omega_h(\sigma) = \sum_{j\geqslant 0} 2^{j(k-b)}(\alpha_{jh})^{-\sigma},$$
$$\upsilon_h(\sigma) = \sum_{j\geqslant 0} 2^{j(k-b)}\beta_{jh}(\alpha_{jh})^{-\sigma-1}. \tag{A.13}$$

From (A.6) and (A.7), we can see that the two sums given by (A.13) are uniformly and absolutely convergent when σ is in the following stripe:

$$\textit{Stripe}: \quad -1 < \mathrm{Re}(\sigma) < -\left(1-\frac{b}{k}\right). \tag{A.14}$$

Furthermore, in terms of (A.6) and (A.7), both $\omega_h(\sigma)$ and $\upsilon_h(\sigma)$ can be approximated by the following sum:

$$\hat{\omega}_h(\sigma) = \sum_{j\geqslant 0} 2^{j(k-b)}(2^{kj+h})^{\sigma}. \tag{A.15}$$

When $\mathrm{Re}(\sigma) < \sigma_0 = -(1-\frac{b}{k})$, this series can be summed exactly:

$$\hat{\omega}_h(\sigma) = 2^{h\sigma}\frac{1}{1-2^{k-b+k\sigma}}. \tag{A.16}$$

Thus, $\phi^*(\sigma)$ is defined in *Stripe* and can be computed as follows:

$$\begin{aligned}
\phi^*(\sigma) &= \int_0^\infty \phi(x)x^{\sigma-1}\,\mathrm{d}x \\
&= \int_0^\infty \Bigg(\sum_{h=0}^{k-1}\lambda_1\lambda_2\ldots\lambda_h \\
&\qquad \times \sum_{j\geqslant 0} 2^{j(k-b)}D_{jh}(x)\Bigg)x^{\sigma-1}\,\mathrm{d}x \\
&= -\sum_{h=0}^{k-1}\lambda_1\lambda_2\ldots\lambda_h\big(\omega_h(\sigma)+\sigma\upsilon_h(\sigma)\big)\Gamma(\sigma) \\
&= -\Gamma(\sigma)(1+\sigma) \\
&\qquad \times \sum_{h=0}^{k-1}\lambda_1\lambda_2\ldots\lambda_h 2^{h\sigma}\frac{1}{1-2^{k-b+k\sigma}}.
\end{aligned} \tag{A.17}$$

From this, we can observe all the singularities (poles), i.e., $\sigma = 0$, at which $\Gamma(\sigma)$ is not defined; and all those values of σ, at which $(1-2^{k(\sigma-\sigma_0)})$ becomes 0:

$$\sigma_j = \sigma_0 + \frac{2\mathrm{i}j\pi}{k\log 2} \quad (j = 0, \pm1, \pm2, \ldots). \tag{A.18}$$

To compute the integral in (A.3), we consider the following integral

$$\phi_N(x) = \frac{1}{2\mathrm{i}\pi}\int_{L_N}\phi^*(\sigma)x^{-\sigma}\,\mathrm{d}\sigma, \tag{A.19}$$

where L_N is a rectangular contour oriented clockwise as shown in Fig. 6.

$$\begin{aligned}
L_N &= L_N^1 + L_N^2 + L_N^3 + L_N^4, \\
L_N^1 &= \left\{c + \mathrm{i}u \;\middle|\; |u| \leqslant \frac{(2N+1)\pi}{k\log 2}\right\}, \\
L_N^2 &= \left\{\nu + \mathrm{i}\frac{(2N+1)\pi}{k\log 2} \;\middle|\; c \leqslant \nu \leqslant \frac{b}{3k}\right\},
\end{aligned}$$

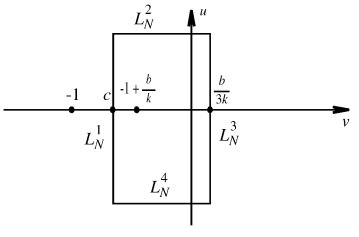

Fig. 6. The rectangular contour L_N.

$$\begin{aligned}
L_N^3 &= \left\{\frac{b}{3k} + \mathrm{i}u \;\middle|\; |u| \leqslant \frac{(2N+1)\pi}{k\log 2}\right\}, \\
L_N^4 &= \left\{\nu - \mathrm{i}\frac{(2N+1)\pi}{k\log 2} \;\middle|\; c \leqslant \nu \leqslant \frac{b}{3k}\right\},
\end{aligned} \tag{A.20}$$

where N is an integer. This contour is of a similar type used in [14, p. 132].

Let ϕ_N^i be the integral along L_N^i $(i = 1, 2, 3, 4)$. Then, $\phi_N(x) = \phi_N^1(x) + \phi_N^2(x) + \phi_N^3(x) + \phi_N^4(x)$. Furthermore, we have the following results:

$$\lim_{N\to\infty}\phi_N^1(x) = \phi(x),$$

$$\lim_{N\to\infty}\phi_N^2(x) = \mathrm{O}(1),$$

$$\big|\phi_N^3(x)\big| \leqslant x^{-b/(3k)}\int_{L_\infty}\big|\phi^*(\sigma)\big|\,\mathrm{d}\sigma = \mathrm{O}(x^{-b/(3k)}),$$

and

$$\lim_{N\to\infty}\phi_N^4(x) = \mathrm{O}(1).$$

Thus, we have

$$\lim_{N\to\infty}\phi_N(x) = \phi(x) + \mathrm{O}(x^{-b/(3k)}). \tag{A.21}$$

On the other hand, $\lim_{N\to\infty}\phi_N(x)$ can be evaluated as the sum of the residues of the integrand, i.e., $\phi^*(\sigma)x^{-\sigma}$, inside L_N. Concretely, we have

$$\begin{aligned}
\lim_{N\to\infty}\phi_N(x) &= -\sum_{\alpha\in\mathrm{Pole}(\phi^*(\sigma))}\big(\phi^*(\sigma)x^{-\sigma},\ \sigma=\alpha\big) \\
&= -\sum_{\alpha\in\mathrm{Pole}(\phi^*(\sigma))}\lim_{\sigma\to\alpha}(\sigma-\alpha)\phi^*(\sigma)x^{-\sigma}.
\end{aligned} \tag{A.22}$$

Within L_∞, $\phi^*(\sigma)$ has the following poles:

$$\alpha = 0, \quad \text{and}$$

$$\alpha = \sigma_j = \sigma_0 + \frac{2\mathrm{i}j\pi}{k\log 2} \quad (j = 0, \pm1, \pm2, \ldots).$$

The contribution of the pole $\alpha = 0$ is $\mathrm{O}(1)$; and the contribution of $\alpha = \sigma_0$ is

$$\begin{aligned}
&\lim_{\sigma\to\sigma_0}(\sigma-\sigma_0)\phi^*(\sigma)x^{-\sigma} \\
&\quad = x^{-\sigma_0}\frac{(1+\sigma_0)\Gamma(\sigma_0)}{k\log 2}\sum_{h=0}^{k-1}\lambda_1\lambda_2\ldots\lambda_h 2^{h\sigma_0}.
\end{aligned} \tag{A.23}$$

Finally, the contribution of each σ_j $(j = \pm1, \pm2, \ldots)$ is

$$\begin{aligned}
&\lim_{\sigma\to\sigma_j}(\sigma-\sigma_j)\phi^*(\sigma)x^{-\sigma} \\
&\quad = x^{-\sigma_0}\exp\left(-\frac{2\mathrm{i}j\pi}{k}\log_2 x\right)(1+\sigma_j)\Gamma(\sigma_j) \\
&\qquad \times \sum_{h=0}^{k-1}\lambda_1\lambda_2\ldots\lambda_h 2^{h\sigma_j}.
\end{aligned} \tag{A.24}$$

So we have

$$\begin{aligned}\lim_{N\to\infty}\phi_N(x) &= x^{-\sigma_0}\frac{(1+\sigma_0)\Gamma(\sigma_0)}{k\log 2}\sum_{h=0}^{k-1}\lambda_1\lambda_2\ldots\lambda_h 2^{h\sigma_0}\\ &\quad+\sum_{j=-\infty}^{-1}x^{-\sigma_0}\exp\left(-\frac{2\mathrm{i}j\pi}{k}\log_2 x\right)\\ &\quad\times(1+\sigma_j)\Gamma(\sigma_j)\sum_{h=0}^{k-1}\lambda_1\lambda_2\ldots\lambda_h\\ &\quad+\sum_{j=1}^{+\infty}x^{-\sigma_0}\exp\left(-\frac{2\mathrm{i}j\pi}{k}\log_2 x\right)\\ &\quad\times(1+\sigma_j)\Gamma(\sigma_j)\sum_{h=0}^{k-1}\lambda_1\lambda_2\ldots\lambda_h\\ &= x^{-\sigma_0}\frac{(1+\sigma_0)\Gamma(\sigma_0)}{k\log 2}\sum_{h=0}^{k-1}\lambda_1\lambda_2\ldots\lambda_h 2^{h\sigma_0}.\end{aligned} \tag{A.25}$$

From this, we know that

$$C_{s,n} = \mathrm{O}(n^{-\sigma_0}) = \mathrm{O}(n^{1-b/k}). \tag{A.26}$$

References

[1] S. Abiteboul, S. Cluet, V. Christophides, T. Milo, G. Moerkotte, J. Simeon, Querying documents in object databases, Internat. J. Digital Libraries 1 (1) (Jan. 1997) 5–19.

[2] A.V. Aho, J.E. Hopcroft, J.D. Ullman, The Design and Analysis of Computer Algorithms, Addison-Wesley Publishing Com., London, 1974.

[3] S. Christodoulakis, C. Faloutsos, Design consideration for a message file server, IEEE Trans. Software Eng. 10 (2) (1984) 201–210.

[4] W.W. Chang, H.J. Schek, A signature access method for the STARBURST database system, in: Proc. 19th VLDB Conf., 1989, pp. 145–153.

[5] Y. Chen, Signature files and signature trees, Inform. Process. Lett. 82 (4) (March 2002) 213–221.

[6] R.V. Churchill, Operational Mathematics, McGraw-Hill Book Company, New York, 1958.

[7] P. Ciaccia, P. Zezula, Declustering of key-based partitioned signature files, ACM Trans. Database Systems 21 (3) (1996) 295–338.

[8] D. Dervos, Y. Manolopoulos, P. Linardis, Comparison of signature file models with superimposed coding, J. Inform. Process. Lett. 65 (1998) 101–106.

[9] C. Faloutsos, Access methods for text, ACM Comput. Surv. 17 (1) (1985) 49–74.

[10] C. Faloutsos, Signature files, in: W.B. Frakes, R. Baeza-Yates (Eds.), Information Retrieval: Data Structures & Algorithms, Prentice-Hall, New Jersey, 1992, pp. 44–65.

[11] C. Faloutsos, R. Lee, C. Plaisant, B. Shneiderman, Incorporating string search in hypertext system: User interface and signature file design issues, HyperMedia 2 (3) (1990) 183–200.

[12] P. Flajolet, C. Puech, Partial Match retrieval of multidimensional data, J. ACM 33 (2) (April 1986) 371–407.

[13] Y. Ishikawa, H. Kitagawa, N. Ohbo, Evaluation of signature files as set access facilities in OODBs, in: Proc. of ACM SIGMOD Int. Conf. on Management of Data, Washington D.C., May 1993, pp. 247–256.

[14] D.E. Knuth, The Art of Computer Programming: Sorting and Searching, Addison-Wesley Publ., London, 1973.

[15] W. Lee, D.L. Lee, Signature file methods for indexing object-oriented database systems, in: Proc. ICIC'92—2nd Int. Conf. on Data and Knowledge Engineering: Theory and Application, Hongkong, Dec. 1992, pp. 616–622.

[16] H.S. Yong, S. Lee, H.J. Kim, Applying signatures for forward traversal query processing in object-oriented databases, in: Proc. of 10th Internat. Conf. on Data Engineering, Houston, TX, Feb. 1994, pp. 518–525.