Journal of Techniques, ISSN: 2708-8383, Vol. 5, No. 2, 2023, Pages 114-122
DOI: https://doi.org/10.51173/jt.v5i2.948

JOURNAL OF TECHNIQUES

Journal homepage: *http://journal.mtu.edu.iq*

RESEARCH ARTICLE - ENGINEERING

Generating Encrypted Document Index Structure Using Tree Browser

Doaa N. Mhawi[1*], Haider W. Oleiwi[2], Heba L. Al-Taie[3]

[1] Computer science department, Technical Institute for Administration, Middle Technical University, Baghdad, Iraq

[2] Department of Electronic and Electrical Engineering, Brunel University London, London, Uxbridge, UK

[3] Information and Communication Technologies Center, Ministry of Construction and Housing, Iraq

[*] Corresponding author E-mail: dododuaaenteesha@mtu.edu.iq

Article Info.

Article history:

Received
14 September 2022

Accepted
02 November 2022

Publishing
30 June 2023

Abstract

The document indexing process aims to store documents in a manner that facilitates the process of retrieving specific documents efficiently in terms of accuracy and time complexity. Many information retrieval systems encounter security issues and execution time to retrieve relevant documents. In addition, these systems lead to ample storage. Therefore, it requires combining confidentiality with the indexed document, and a separate process is performed to encrypt the documents. Hence, a new indexing structure named tree browser (TB) was proposed in this paper to be applied to index files of the large document set in an encrypted manner. This method represents the keywords in a variable-length binary format before being stored in the index. This binary format provides additional encryption to the information stored and reduces the index size. The proposed method (TB) is applied to the WebKB dataset. This dataset is related to web page documents (semi-structured documents). The experimental results demonstrated that the storage size is reduced by using TB-tree to 48.5 MB, while the traditional index is 307 MB.

Keywords: B±Tree; Encryption; Information Retrieval; Inverted Index; TB-Tree.

1. Introduction

Notably, tremendous growth in service-demanding users and devices is expected in the next era of communication systems that will be globally covered. Unprecedented technologies and applications accompany the rapid leap in the communications field with new features, i.e., the data-hungry traffic, unified ubiquitous systems, and revolutionary search processes [1–5] generating huge document databases that include a huge collection of documents from a variety of sources, for instance, research papers, articles, news, digital libraries, books, messages, e-mails, and web pages. Using text databases to expedite the execution of critical tasks has been a part of many people's and organizations' everyday routines throughout the years. As a result, data organization is required to conduct data update and query operations effectively; thus, indexing is one of the methods utilized. Different indexing techniques are being explored for the contents [6–10]. When it comes to database performance and data security, indexing techniques may be critical [11–15]. In contrast, Information Retrieval (IR) indexes have a variety of drawbacks, including huge index sizes, the ability to do an output search and potential security risks. This necessitates the development of index structures that can access important data quickly while using little storage capacity to index huge amounts of data. Balanced-tree (B-tree) structure is a widely used indexing structure. The indexing method is simple, takes less time to build, and results in a lower index size, which has led to its widespread adoption as a starting structure by many academics. The B-trees-based indexes can deal with huge datasets and respond to queries in near-linear time with little input/output (I/O) overhead. The query cost is reduced when a B tree is used as the size of the tree. [10] When used to index trajectory data B tree indexing shows its resilience and versatility [9, 10].

Following the discovery of these faults in several techniques by a group of academics, they commenced developing this framework, using a Tree-Browser (TB). This article presents a novel index structure that may be used to build and create inverted indexes for a variety of information retrieval systems such as those used by search engines.

The primary contributions of this paper are as follows:

- Proposing a new indexing technique called the tree browser (TB) to be applied to large IRS inverted files.
- Developing a system that combines indexing and encryption in a single step.
- Reducing the amount of storage required for the produced document index.
- Developing an index to support single keyword and phrase searches.

Nomenclature & Symbols			
TB Tree	Tree Browser	DOC	Document
IR	Information Retrieval	P	Pointer
LW	Longest Word	SOAS	Semantic Oriented Adaptive Search
GA	Genetic Algorithm	K	CONSTANT
EM	External Memory	Log	Logarithm
VSM	Vector Space Model	SKEY	Search Key
LSI	Latent Semantic Index	WDL	Word Length
N	Constant	EM	External Memory
VSM	Vector Space Model	N/A	Not Available
IRS	Information Retrieval System		

The remainder of the paper is divided into the following sections: section 2 describes the related work, while Section 3 includes similar attempts that included different B-Tree and B±Tree variants.

The proposed structure is based on the B± tree, which is described in more detail in Section 4. The TB-tree structure and associated functionality, which includes searching, updating, and insertion into the proposed TB-tree structure, are explained in detail in section 5. Section 6 contains a presentation of the results and analysis. Finally, in section 7, you will find the conclusion and recommendations for further study.

2. Related Work

A. K. Doaa Nteesha Mhawi, studied storage space [12], by modifying the genetic algorithm (GA) operators, a more efficient inverted index was created, which allowed the retrieval process to be accelerated. In conjunction with the term-proximity fitness function, the author employs a hybrid crossover operator to get the desired result. An accuracy of 89 percent and recall of 84 percent was achieved by the created system was tested on a dataset of 8000 HTML pages with 100MB of storage space. This dataset requires a significant amount of storage space, resulting in a lengthy retrieval time and an extremely large amount of storage space. G. E. Blelloch, J. T. Fineman, P. B. Gibbons, Y. Gu, and J. Shun, in [17] proposed the PRAM model with asymmetric write cost and showed that sorting can be performed in O(n) writes, O(n log n) reads, and logarithmic depth (parallel time). Next, they considered a variant of the External Memory (EM) model that charges $k > 1$ for writing a block of size B to the secondary memory and presented variants of three EM sorting algorithms (multi-way merge sort, sample sort, and heapsort using buffer trees) that asymptotically reduce the number of writes over the original algorithms and perform roughly k block reads for every block write. Finally, they defined a variant of the Ideal-Cache model with asymmetric write costs and presented write-efficient, cache-oblivious parallel algorithms for sorting, FFTs, and matrix multiplication. Adapting prior bounds for work-stealing and parallel-depth-first schedulers to the asymmetric setting yields parallel cache complexity bounds for machines with private caches or shared caches, respectively. G. Ramachandran and K. Selvakumar, [18]. In peer-to-peer networks, the authors provided a content-based search index. This study proposed Semantic Oriented Adaptive Search (SOAS) strategy based on semantic content. It is a multi-layer architecture model which utilizes a dynamic caching technique to achieve effective search on the unstructured P2P networks. It is a scheme constructed as a two-tier P2P network with ultra-peers of high connectivity based on a power law model. This approach extensively used Vector Space Model (VSM) and Latent Semantic Index (LSI) to derive local indices from summarized semantic vectors. Query searching was performed through a round of searches using derived indices from semantic data objects. If one search fails, the next round search is invoked sequentially in the order of local index, cache index, response index, global index, and adaptive search among ultra-peers. R. J. Derscheid, M. C. Rahe, E. R. Burrough, K. J. Schwartz, and B. Arruda, Secondary storage makes advantage of asymmetrical I/O. Writes are costlier than reads in this situation. As a consequence of the alteration of the B tree, an imbalanced B tree was created [19]. They developed a management algorithm that eliminated the need for written documentation via meticulous unbalancing and rebalancing procedures. It is possible to decrease I/O expenses by 30% while simultaneously increasing performance with this technique by changing the parameter settings. Although this method utilizes more nodes than the B tree, the creator points out. This has a detrimental effect on performance in memory-constrained settings. R. Jin, H. J. Cho, S. W. Lee, and T. S. Chung, Another indexing technique that makes use of the B tree is the lazy-split B tree, which was developed by [20] and introduced in the literature. It employs three strategies; the first is to divide nodes lazily to split the smallest number of possible nodes. The second, modify-two-nodes, only update the parents and nodes, thus decreasing the number of writes nodes. The third is to combine utilizing a lazy-coalesce technique to reduce the amount of merging and distribution. When these three methods are used together, the write speed of flash memory and the use of buffer space increase. L. Yang, M. Di, X. Huang, and F. Duan [21] developed an index structure, mapping high-dimensional data to single-dimensional values to overcome the large amount of dimensional data to manage. When the block distance method and the B+-tree structure were combined, the resulting approach was known as the Block B-Tree technique. A high-dimensional data set was translated into single-dimensional key values using the Block distance method; however, the resulting key values were maintained using the compact B+-tree. The use of the Euclidean distance similarity search was made possible by this method. Y. Chen et al., Both the B+-tree and the B-tree were used by [22] in the process of insertion and deletion to the flash memory of their system. It combined Linear Pipelining, Write Once Partitioning, and Background Linear Merging to create an embedded search engine for securing tokens used in the case of managing and securing documents in the personal cloud context. F. Boucenna, O. Nouali, S. Kechid, and M. Tahar Kechadi, The inverted index created by [23] is yet another superior inverted index utilized in information retrieval. With the help of this technique, the document indexing phase of an information retrieval system (IRS) may be defined. It is necessary to build a document collection. The search space is condensed as a result of the score produced by IR Probabilistic. This approach accomplished two objectives; the first is to decrease the amount of storage space available while the second is to reduce the amount of processing time available. D. A. Bonilla, A. Pérez-Idárraga, A. Odriozola-Martínez, and R. B. Kreider in [24] proposed a secure inverted index constructed using a novel index-building technique. The approach used two techniques; the first is the employment of fictitious documents and the second is homomorphic encryption. Both methods, on the other hand, have significant disadvantages. During the search process, the first method generates a large number of false positive results, which is undesirable. A further disadvantage of the second method is that it generates very huge cipher texts that were hundreds of times larger than their plaintext equivalents. To overcome these two disadvantages, the double score weighting method is used in conjunction with a compressed database of encrypted scores. To the best of the authors' knowledge, the proposed method of Generating an Encrypted Document Index Structure Using a Tree Browser is the best, overcoming other related works.

This method represents the keywords in a variable-length binary format before being stored in the index. Moreover, it provides additional encryption to the information stored to reduce index size and fast retrieval.

3. Feature of B±Tree Searching

Frequently, B+-Trees are used in applications that deal with a significant quantity of information such as file systems and database indexes. Compared to the balanced tree (B-tree), it includes just keys in each node, rather than (key-value), and it has an extra level with connected leaves at the button rather than (key-value).

The method of searching in the B-tree is straightforward. It takes a reference to the root as input node x of a sub-tree and accepts a key k to be searched in that sub-tree for the search of B-Tree. This results in a top-level call of the type B-Tree-Search (root(T), k), where B-Tree-Search returns the ordered pair (y, i) as a result of the search. The pair consists of a node y and an index i in such a way that keys(y) = k if k is found in the B-tree; else, keys(y) = NIL is returned from the function. B-Tree includes extra restrictions (shown in Fig. 1) to guarantee that the tree is always balanced.

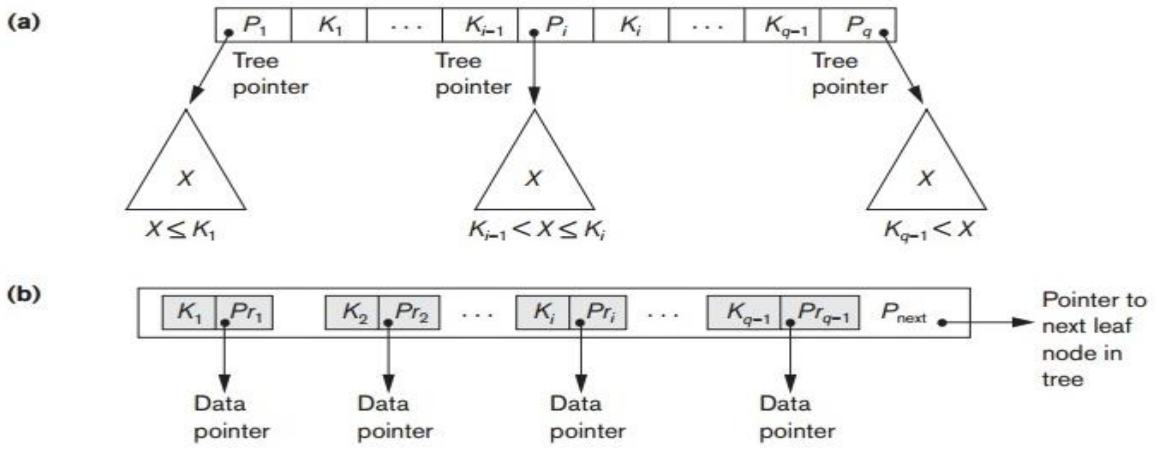


Fig. 1. B-Tree constraints (a. represents internal node while b. represents leaf node with search values and data pointer)

The steps to search for a record by using an access structure on a key field are as follows [25]:

- The form of each internal node in a tree is represented as follows:

<P1, <K1, Pr1>, P2, represented Kq-1, Prq-1>, Pq>

Where, q <= p

P represents the tree pointer.

Pr represents the data pointer.

- Values of the key are:

(Ki……………Kq-1),

These keys are ordered within each node.

- The value of x in the sub-tree is pointed by pi:

For (1< i < q, Ki-1 < x < Ki),

For (i=1, x < Ki),

For (1=q, Ki-1 < x)

- Each node has at most p tree pointers.
- Each node has at least ⌈p/2⌉ three-pointers. However, the root node has at least two tree pointers unless it only considers the node in the tree.
- A node has q-1 search key field values and hence q-1 data pointers when it comes within q tree pointers, (q<=p).
- At the same stage, all leaf nodes are present. Except for the nodes with Pi null tree pointers, the inner nodes have the same structure as the leaf nodes.

Order 3 of the B-Tree (shown in Fig. 2) is accessible via the use of a key field structure. This results in the values being unique and the pointer pointing to the block cluster when employing the B-tree in a field that does not include a key value.

B±trees represent the entire range of values in a tree with each subinterval considered to be an internal node. The root of the B±trees represents the entire range of values in the tree. If k indicates the length of the longest word in the tree, then we are searching for the leaf that contains the value k in some way. An internal B±Tree node has at most d ≤ b offspring, each of whom represents a distinct sub-interval of the interval represented by the node. The appropriate node is chosen by searching for the key values of the node in question. As shown in Fig. 4, this tree does not have a complete index page and one of the data pages has empty slots. In addition, one of the data pages has empty spaces [20], [26–28].

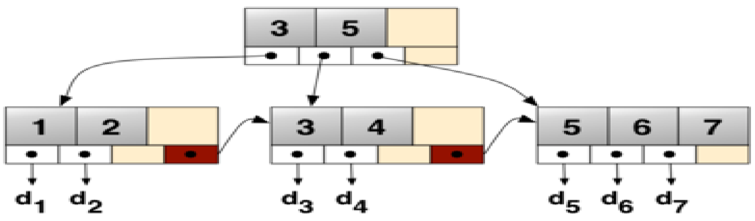


Fig. 2. The Link list structure of B tree

4. The Proposed TB-Tree Structure

The TB tree is a multi-tiered tree with many branches. It comprises an amazing balanced tree with a publishing-ready file for distribution. It has many of the same features as the B±tree structure that was previously suggested. The TB tree's root page layout is similar to that of the B+ tree, with the addition of a field for storing the largest word length in the TB tree's root page layout. Keeping track of the keys and the sequence in which they are kept is critical; thus, this section must be filled out correctly [15], [29].

4.1. TB Tree properties

There are different structures and properties for the leaf and non-leaf pages, besides, the variable length in their records.

4.1.1. The leaf pages' properties in the TB tree

If the M2 considers the maximum number of records on the leaf index page. Then;

- There are at least $\frac{M_2}{2}$ records for all tree-leaf nodes.
- There is a unified distance from the root to all leaf nodes which is made it appear at the same level.
- To record the posting file, there is a pointer located for each entry (word key) on the leaf page and another pointer for each leaf page to the right sibling page.

4.1.2. The non-leaf pages' properties in the TB tree

If the M1 is considered to be the maximum number of records on the non-leaf index page. Then,

- There are at least two children in the root.
- There are $\frac{M_1}{2}$ children in the tree nodes except the tree root.

4.1.3. Posting File in TB tree

In the postings file, several format records are presented as follows {document-no, a pointer to document-no, and positions}. Also, a linked list of records for each word has been used to show the file that the word belongs to and the word's location in that file.

4.2. B. Keys in TB tree

In the proposed work, the key is associated with each word which is identified uniquely and efficiently, resulting in accessing every single word as in the case of the B±tree. The unique keys in TB-tree are defined as Firstly, the TB-tree English language is used and the number between 1 and 26 is assigned to each character in ascending order as shown in Table 1, which represents the character number of mappings. Secondly, concatenating the associated numbers of the characters of the word to construct the word key K. An example of a situation for the word "orange" is "15 18 01 14 07 05". The binary value of this number using a minimum number of bits is (01111 10010 00001 01110 00111 00101); it is stored in a text field of the record within the index. However, this requires additional processing to sort the words in ascending order in the TB tree. Equations (1) and (2) are used to compute the search key SKEY.

$$n = LW - WDL \tag{1}$$

$$SKEY = K^{*}(10^{n})^{2} \tag{2}$$

Where n represents the difference between the longest word (LW) stored in the root node and the length of the word (WDL), i.e., the character number of the searched word. This difference is important to find the value of the skey.

Table 1. Character-number mapping

A	B	C	D	E	F	G	H	I	J	K	L	M
1	2	3	4	5	6	7	8	9	10	11	12	13
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
14	15	16	17	18	19	20	21	22	23	24	25	26

Skey in (1) will have the same key value for the same words with different variations such as apple, apples, etc. To differentiate them, the second term o2 ($(10^{n})^{2}$) is used, where additional zeros are added for ease of searching while maintaining the original word in the tree.

5. Insertion and Searching in TB-Tree

This section explains with examples the process of insertion of words into TB-Tree, then explains the two types of search: single keyword and phrase query.

5.1. Insertion algorithm

By Algorithm 1, each word is transformed into a numerical representation before being included in the database. It is first transformed into a number using the mapping provided in Table 1, then translated into a binary number using the mapping shown in Table 2. If encryption is required, the encryption algorithm is applied to the converted text before it is put into the TB-tree; otherwise, only the converted text is placed into the TB-tree according to Algorithm 1, which is implemented as a function of the conversion algorithm. The works in question are as follows; While the insertion procedure is in progress, a new record is added to a leaf node that is not yet filled. As soon as the leaf node becomes full, it is split into two nodes, which are designated as new and old nodes to preserve the data. However, if an overflow occurs during the insertion process, the data will be reallocated across sibling pages before the splitting process is performed to minimize the number of pages that need to be divided.

```
Algorithm 1: Preprocessing (converting to decimal and binary form) and Encryption.
Begin
Initialization step:
▪   Word_in_decimal = 0, Word_in_binary [ ].
    LOOP: For each C in W Do   // C is a character, and W is the word.
     Get the corresponding numerical value of C as per Table 2.
      Add to the Word_in_decimal. Convert the word stored in Word_in_decimal into binary then add to Word_in_binary.
        If Encryption is required then
▪         Encrypt (Word_in_binary)
▪       End if
▪   END LOOP
▪   Return Word_in_binary [W].
▪   End
```

5.2. Single keyword query

The search method for locating a single word with an exact key value in a B±tree employs approaches similar to those employed in other search algorithms. The search key (Skey), on the other hand, is concerned with the (K) key, which is kept in the tree. The search process begins at the root node and works its way down to the remaining nodes. The non-leaf nodes may be used to connect the search to the leaf nodes. By closing the Postings File, which will be open throughout this process, if the required word (or the numerical value of the word) is found in the leaf node, the search process is directed to the requested documents that contain the desired word. Using the example of 20 where the inputs are the root page R and the keyword Key-Value K; the output is a list of documents that include the word K, the method of running a single keyword query is described. Consider the following example to get a better understanding of this method.

Table 2. The Document indexing number with word position example

Documents Number	**Word position in the document**				
	1	**2**	**3**	**4**	**5**
1	apple	banana	carrot	apple	lemon
2	apple	banana	date	banana	mango
3	Cherry	endive	fig	grape	melon
4	nut	orange	peas	fig	

Example 1: Based on the set of documents listed in Table 2, suppose that there is a query for finding the word “fig" in the desired documents. However, the search key K= 060907, where 06, 09, and 07 are the mapping values off, i and g respectively according to Table 1, the word length WDL =3, the length of the longest word stored in the tree LW = 6, n = LW-WDL = 6 - 3 = 3, thus:

$$\text{Skey} = \text{K}*(10^{n})^{2} = 60907*(103)2 = 60907000000$$

Before insertion into the posting file, the Skey is converted into the equivalent binary code to be 00110 01001 00111, where each number in the mapped code is represented by 5 bits.

Example 2: if users want to search for the word “apple”, then its binary representation is 00001 10000 10000 01100 00101. The number of bits in this example is 25-bits, while the maximum number of bits used in these examples (according to the set in Table 2) is 30-bits according to the longest word inserted into the TB tree (LW=6), as shown in Table 3. These bits are split into two parts to produce 15-bits each. This process is done to minimize the required storage and to ease the retrieval of the related documents. The resultant reduced bits are stored in the posting file so that the entered word length to the TB tree is half-length of the original word. This word is finally inserted into the posting file as shown in Figs. 3, 4, and 5.

5.3. Encryption in TB-tree

As illustrated in the previous two examples, the indexed words are stored in binary representations, and they depend on several dynamic parameters that change according to the dataset. These parameters are the longest word length in the set LW and the word length WDL. These values are used to calculate n, which is used to generate the search key Skey as per in (2). The secrecy here is that different variations of a given word such as read, reading, and reader will have the same length and hence the same representation in the index. To differentiate them, the (2) is applied where additional zeros are added for ease of searching while maintaining the original word in the index. Another confidentiality aspect is that it relies on the longest-term length, which varies from set to set.

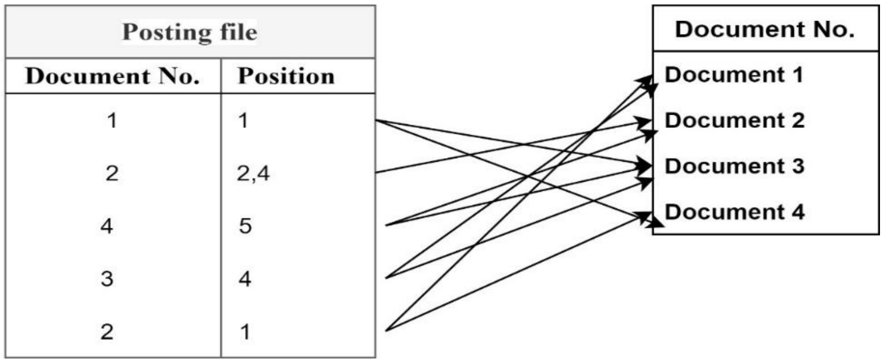
5.4. Phrase query search

In the phrase query search, the B±tree technique is used for descending the tree from the root using the search algorithm. In addition, in the window query on R-tree, more than one sub-tree under a visited node may need to be searched. Therefore, children's nodes may have only a few visits to each tree level. Besides, the query is satisfied by guiding the search to the leaf nodes via the non-leaf index pages. Moreover, the search process is conducted for the desired documents that contain the required word in the sentence by the Postings File. This method is explained where the inputs are TB tree, root node R, linked_List1, and Linked_List2 and the output is all page IDs of the documents containing the words of the searched phrase. Initially, link-list1=link-list2=LWD=0, supposing that the root page R="grape", hence, the R-LONGEST WORD=5.

The next step in the algorithm is to check whether this root is a leaf page or not. If not, search for all children's pages that satisfy the query and contain the words of the sentence while at the same time being at tree-level and leading to the leaf nodes. Now, the two children of this root are carrot and endive. Suppose these children's page-ID (document) =1, so the Linked_List1 will equal one. This operation continues until reaching the leaf page.

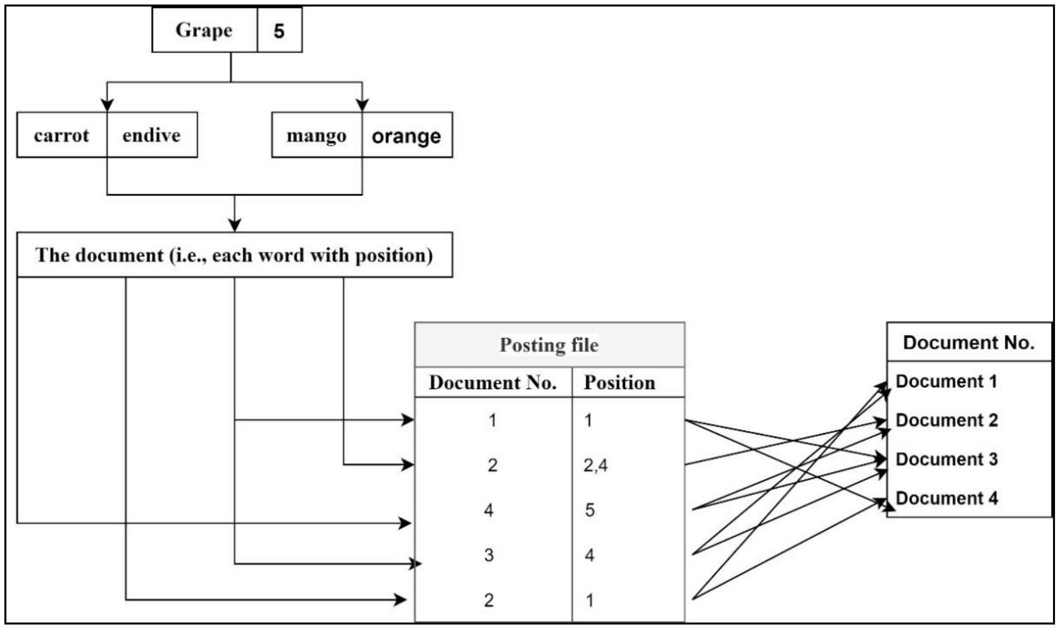
Table 3. The inserted words into TB-tree

Entry Seq.	Word Length	n	$(10^n)^2$	Stored Key	Search Key
date	4	2	10000	4012005	40120050000
Cherry	6	0	1	30805181825	30805181825
endive	6	0	1	51404092205	51404092205
carrot	6	0	1	30118181520	30118181520
apple	5	1	100	116161205	11616120500
lemon	5	1	100	1205131514	120513151400
banana	6	0	1	20114011401	20114011401
grape	5	1	100	718011605	71801160500
melon	5	1	100	1305121514	130512151400
orange	6	0	1	151801140705	151801140705
nut	3	3	1000000	140120	140120000000
peas	4	2	10000	16050119	160501190000
sorrel	6	0	1	191518180512	191518180512
mango	5	1	100	1301140715	130114071500
fig	3	3	1000000	60907	60907000000



Posting file	
Document No.	Position
1	1
2	2,4
4	5
3	4
2	1

Fig. 3. Sampling of the posting file



Posting file	
Document No.	Position
1	1
2	2,4
4	5
3	4
2	1

Fig. 4. Processing of the TB tree inserting word

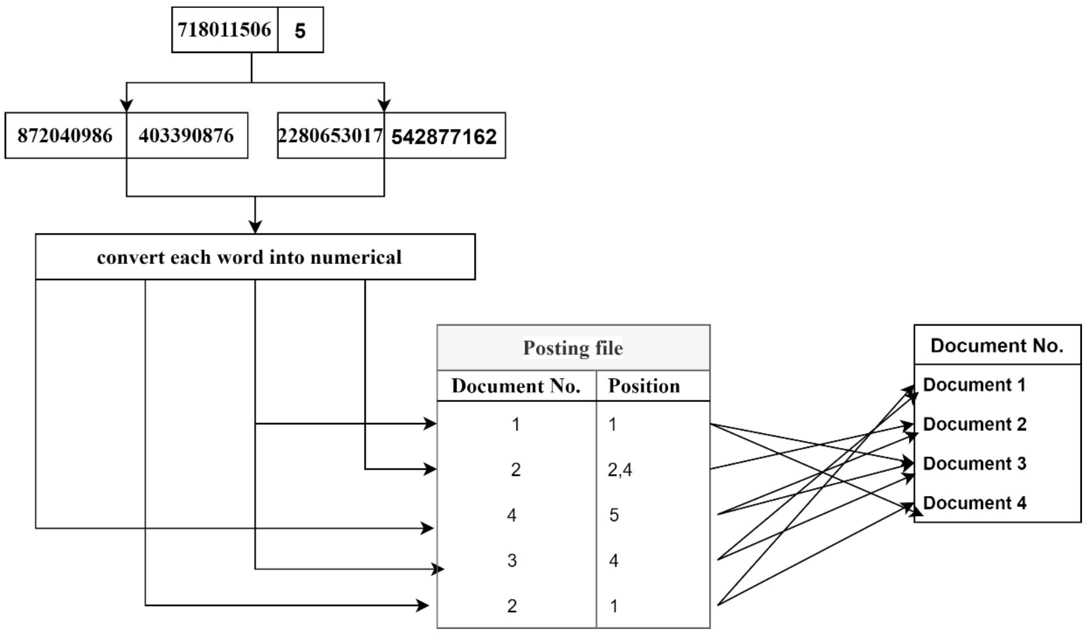


Fig. 1. Numerical meaning of TB-tree terms that were converted to binary before

6. Result and Discussion

This section describes the test environment used to implement this work and the experimental results of the proposed algorithm.

6.1. Test environment

This system is implemented using VisualBasic.net 2019 with Microsoft Office (Access 2019) to store the database. It is executed by using a laptop with a processor having the following characteristics: Intel (R) Core (TM) i7 CPU M 460 2.4 GHz, RAM 8GB operated by Windows 11 with the 64-bit operating system.

6.2. Experimental results

8282 HTML web documents have been used to evaluate the proposed algorithm's performance. Carnegie Mellon University data set (WebKb) (The 4 Universities Data Set, 1998). Many researchers have used this data set. After preprocessing this data set by removing the noisy information (HTML tags and stop words), the unique words are 128213. A subset consisting of 10K words of this dataset is used in the proposed system. This set of 10K is believed to be reasonable compared to the 4.7K of distinct words tested. This subset is indexed using the insertion Algorithm. Each word of the collection is converted into a numerical number then this numerical number is converted into an equivalent binary number according to Algorithm 1.

6.3. Comparison with other related studies

Fig. 6 demonstrates the comparison process of the proposed method to the one proposed and the traditional index applied in terms of storage space. The storage size is reduced by using TB-tree to 48.5 MB, while by using the traditional index the storage size is 307 MB, and the storage size utilized by applying the Enhanced Inverted Index (EII) proposed by 17 is reduced to 99.9 MB. Hence, TB-tree reduces the storage size by 47.5% compared to EII and reduces it by 94.6% compared to the one applied by 22.

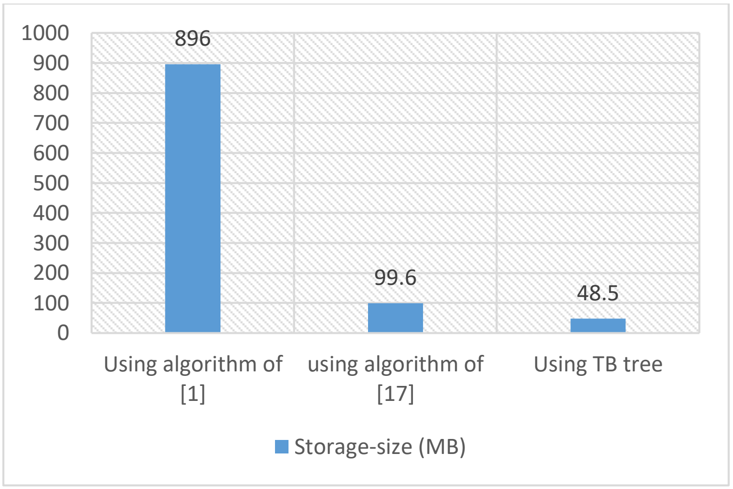


Fig. 6. Storage-size (MB) of indexing algorithms

Fig. 6 demonstrated the differences in the storage space when using traditional, using another prosed related algorithm, and with the proposed TB tree method. It shows that the proposed method is the best when compared to another because it is reducing the storage size to 47.5%. Table 4 depicts the comparison with other related studies.

Table 4. The Result of Comparison with Other Studies

References	Methods	Dataset names	Classification method	Storage Space	Security
[12]	Traditional	Webpages	Traditional algorithm	896	
[10]	EII (Enhanced Inverted Index)	Documents on the web	EII	99.6	
[25]	TB±tree	Some words using as a dataset	TREE		N/A
[12], [32]	Vector Space Method		Vector method	N/A	
Proposed Method	Proposed Method	WebKb dataset	Encrypted TB	48.5%	Secure key

7. Conclusions

This article presented the development of a novel document indexing system named TB tree. In contrast to a technique in the B±tree, single keyword and phrase searches were supported by the system. This paper works on a new algorithm that helps to decrease the index file size by a minimum of 48.5 percent as shown in Fig. 6 where each word in the TB tree is represented by a one-of-a-kind binary integer saved as a binary code. Furthermore, the numerical binary representation was utilized as an encryption of the stored word, providing an extra layer of protection. Applying the suggested method to a bigger dataset (which contains 8282 semi-structure webpages) makes it possible to further generalize the findings obtained thus far. Additionally, a comprehensive test must be carried out to evaluate the time performance of running queries on the newly created index. For future work, it is suggested to use this method with different types of datasets such as documents, text, and images to convert them into a suitable format with a new method of security to make the system more authenticated.

Acknowledgment

The authors would like to thank the Middle Technical University (MTU) for the financial support of this project.

References

[1] H. W. Oleiwi, N. Saeed, and H. S. Al-Raweshidy, "A Cooperative SWIPT-Hybrid-NOMA Pairing Scheme considering SIC imperfection for THz Communications," Proc - 2022 IEEE 4th Glob Power, Energy Commun Conf GPECOM 2022, no. June, pp. 638–643, 2022, doi: 10.1109/GPECOM55404.2022.9815677.
[2] H. W. Oleiwi and H. Al-Raweshidy, "SWIPT-Pairing Mechanism for Channel-Aware Cooperative H-NOMA in 6G Terahertz Communications," Sensors, vol. 22, no. 16, p. 6200, 2022, doi: 10.3390/s22166200.
[3] H. W. Oleiwi and H. Al-Raweshidy, "Cooperative SWIPT THz-NOMA/6G Performance Analysis," Electron, vol. 11, no. 6, 2022, doi: 10.3390/electronics11060873.
[4] H. W. Oleiwi, D. N. Mhawi, and H. Al-Raweshidy, "MLTs-ADCNs: Machine Learning Techniques for Anomaly Detection in Communication Networks," IEEE Access, vol. 10, no. August, pp. 1–1, 2022, doi: 10.1109/access.2022.3201869.
[5] H. W. Oleiwi, N. Saeed, and H. Al-Raweshidy, "Cooperative SWIPT MIMO-NOMA for Reliable THz 6G Communications," Network, vol. 2, no. 2, pp. 257–269, 2022, doi: 10.3390/network2020017.
[6] R. Reinanda, E. Meij, and M. De Rijke, "Knowledge graphs: An information retrieval perspective," Foundations and Trends in Information Retrieval, vol. 14, no. 4. pp. 1–159, 2020, doi: 10.1561/1500000063.
[7] S. Ceri, A. Bozzon, M. Brambilla, E. Della Valle, P. Fraternali, and S. Quarteroni, "An Introduction to Information Retrieval," in Web Information Retrieval, 2013, pp. 3–11.
[8] Z. Wang, K. Banawan, and S. Ulukus, "Private Set Intersection: A Multi-Message Symmetric Private Information Retrieval Perspective," in IEEE Transactions on Information Theory, 2022, vol. 68, no. 3, doi: 10.1109/TIT.2021.3125006.
[9] A. Karim Abdul Hassan and D. Enteesha mhawi, "A Proposed Method for Documents Indexing," Diyala J Pure Sci, vol. 13, no. 2, pp. 43–56, 2017, doi: 10.24237/djps.1302.144a.
[10] A. K. Hassan and D. Enteesha mhawi, "Enhance Inverted Index Using in Information Retrieval," vol. 34, no. 2, 2016.
[11] N. Wang, J. Fu, B. K. Bhargava, and J. Zeng, "Efficient Retrieval Over Documents Encrypted by Attributes in Cloud Computing," IEEE Trans Inf Forensics Secur, vol. 13, no. 10, pp. 2653–2667, 2018, doi: 10.1109/TIFS.2018.2825952.
[12] A. K. Doaa Nteesha Mhawi, "Information Retrieval Using Modified Genetic Algorithm," Al mansour J, no. June, p. 15, 2017, doi: 10.36541/0231-000-027-004.
[13] N. Al-ufoq and N. A. Company, The 3rd International Scientific Conference of Computer Sciences (3SCCS2021), no. January. 2022.
[14] D. N. Mhawi and A. Aldallal, "Advanced Feature-Selection-Based Hybrid Ensemble Learning Algorithms for Network Intrusion Detection Systems," Symmetry (Basel), vol. 14, no. 7, p. 1461, 2022.
[15] H. Oleiwi, N. Saeed, H. Al-Taie, and D. Mhawi, "An Enhanced Interface Selectivity Technique to Improve the QoS for the Multi-homed Node," Eng Technol J, vol. 40, no. 8, pp. 101–109, 2022, doi: 10.30684/etj.2022.133066.1165.
[16] H. Oleiwi, N. Saeed, H. Al-Taie, and doaa nteesha "Evaluation of Differentiated Services Policies in Multihomed Networks Based on an Interface-Selection Mechanism," Sustainability 2022, vol. 14, pages13235. https://doi.org/10.3390/su142013235.
[17] G. E. Blelloch, J. T. Fineman, P. B. Gibbons, Y. Gu, and J. Shun, "Sorting with asymmetric read and write costs," in Annual ACM

Symposium on Parallelism in Algorithms and Architectures, 2015, vol. 2015-June, pp. 1–12, doi: 10.1145/2755573.2755604.
[18] G. Ramachandran and K. Selvakumar, “Dynamic caching for semantic oriented adaptive search in unstructured P2P networks,” Asian J Inf Technol, vol. 13, no. 3, pp. 138–149, 2014.
[19] R. J. Derscheid, M. C. Rahe, E. R. Burrough, K. J. Schwartz, and B. Arruda, “Disease diagnostic coding to facilitate evidence-based medicine: current and future perspectives,” J Vet Diagnostic Investig, vol. 33, no. 3, pp. 419–427, 2021, doi: 10.1177/1040638721999373.
[20] R. Jin, H. J. Cho, S. W. Lee, and T. S. Chung, “Lazy-split B+-tree: A novel B+-tree index scheme for flash-based database systems,” in Design Automation for Embedded Systems, 2013, vol. 17, no. 1, pp. 167–191, doi: 10.1007/s10617-013-9123-4.
[21] L. Yang, M. Di, X. Huang, and F. Duan, “BlockB-Tree: A new index structure combined compact B+-tree with block distance,” in Proceedings - 2015 8th International Congress on Image and Signal Processing, CISP 2015, 2016, pp. 533–538, doi: 10.1109/CISP.2015.7407937.
[22] Y. Chen et al., “An Optimizing and Differentially Private Clustering Algorithm for Mixed Data in SDN-Based Smart Grid,” IEEE Access, vol. 6, 2018.
[23] F. Boucenna, O. Nouali, S. Kechid, and M. Tahar Kechadi, “Secure Inverted Index Based Search over Encrypted Cloud Data with User Access Rights Management,” J Comput Sci Technol, vol. 34, no. 1, pp. 133–154, 2019, doi: 10.1007/s11390-019-1903-2.
[24] D. A. Bonilla, A. Pérez-Idárraga, A. Odriozola-Martínez, and R. B. Kreider, “The 4r’s framework of nutritional strategies for post-exercise recovery: A review with emphasis on new generation of carbohydrates,” International Journal of Environmental Research and Public Health, vol. 18, no. 1. pp. 1–19, 2021, doi: 10.3390/ijerph18010103.
[25] M. Fekihal, I. Jaluta, and D. K. Saini, “TB±tree: Index structure for Information Retrieval Systems,” 2015 2nd Int Conf Comput Sci Comput Eng Soc Media, CSCESM 2015, no. September, pp. 182–186, 2015, doi: 10.1109/CSCESM.2015.7331890.
[26] D. N. Mhawi and S. H. Hashem, “Proposed Hybrid Correlation Feature Selection Forest Panalized Attribute Approach to advance IDSs,” Karbala Int J Mod Sci, vol. 7, no. 4, pp. 405–420, 2021, doi: 10.33640/2405-609X.3166.
[27] M. S. Kadhm, “An Accurate Diabetes Prediction System Based on K-means Clustering and Proposed Classification Approach,” Int J Appl Eng Res, vol. 13, no. 6, pp. 4038–4041, 2018, [Online]. Available: https://www.ripublication.com/ijaer18/ijaerv13n6_118.pdf.
[28] M. S. Kadhm, D. E. Mhawi, and R. M. H. Zaki, “An Accurate Handwritten Digits Recognition system Based on DWT and FCT,” Iraqi J Sci, vol. 58, no. 4B, pp. 2200–2210, 2017, doi: 10.24996/ijs.2017.58.4b.23.
[29] H. W. Oleiwi, H. L. Al-Taie, N. Saeed, and D. N. Mhawi, “A Comparative Investigation on Different QoS Mechanisms in Multi-Homed Networks,” Iraqi J Ind Res, vol. 9, no. 1, pp. 1–11, 2022, doi: 10.53523/ijoirvol9i1id141.
[30] V. Goel, A. K. Ahalawat, and M. N. Gupta, “Distributed air indexing scheme for full-text search on multiple wireless channel,” in Advances in Intelligent Systems and Computing, 2016, vol. 385, pp. 125–135, doi: 10.1007/978-3-319-23258-4_12.
[31] A. Al-Dallal, “Enhancing recall and precision in Web search using advanced fitness function,” in Proceedings of the European, Mediterranean, and Middle Eastern Conference on Information Systems, EMCIS 2012, 2012, pp. 29–41.
[32] D. N. Mhawi, Oleiwi, H.W., Saeed, N.H., Al-Taie, H.L. “An Efficient Information Retrieval System Using Evolutionary Algorithms,”. Network 2022, vol. 2, pp. 583-605., doi.org/10.3390/network2040034.