Computer Engineering and Intelligent Systems
ISSN 2222-1719 (Paper) ISSN 2222-2863 (Online)
Vol.6, No.5, 2015

Skip Ring/Circular Skip List: Circular Linked List Based New Data Structure

Mustafa AKSU*
Kahramanmaraş Vocational High School, Sütçü İmam University, Kahramanmaras, Turkey
* E-mail of the corresponding author: m.aksu@ksu.edu.tr

Ali KARCI
Faculty of Engineering, Department of Computer Engineering, İnönü University, Malatya, Turkey
ali.karci@inonu.edu.tr

Abstract
A linked list is a data structure consisting of a group of nodes which together represent a sequence. Linked lists are used in skip list data structures. They consist of a layered structure and all nodes are in the bottom layer. These nodes are reduced to half towards upper layers and thus a pyramid-like structure is formed, which facilitates search, insertion and removal operations. A circular linked list is a type of linked list in which the last node of the list points back to the first node. Our new data structure, skip ring, is created with the help of circular linked list and skip list data structures. In circular linked list, operations are performed on a single round robin list. However, our new data structure consists of circular link lists formed in layers which are linked in a conical way. Time complexity of search, insertion and deletion equals to O (lg N) in an N-element skip ring data structure. Therefore, skip ring data structure is employed more effectively (O(lg N)) in circumstances where circular linked lists (O(N)) are used.
Keywords: Skip Ring, Circular Skip List, Circular Linked List, Skip List, Data Structure.

1. Introduction

Various disciplines in computer sciences benefit from data structures directly or indirectly. Different data structures are used as solutions to various problems. New data structures are sometimes required due to the limitations such as processing, time and hardware or inefficiency of current data structures. Sometimes, a data structure is preferred over another one because of its processing speed. New data structures emerged because dynamic and static structures are required [1].

Taking these factors into consideration, it is evident that new data structures and algorithms will continue to emerge [2].

A circular linked list is a type of linked list in which the last node of the list points back to the first node. In single or double linked list the last node contains a NULL pointer since there is no next node, whereas in a circular linked list the "next" pointer of the last node contains the address of the first node (Figure 1). Therefore it is called a circular linked list. A circular linked list has a "start" node, but no "end" node. In a Circular Linked List all the nodes are linked in continuous circle (Figure 1).

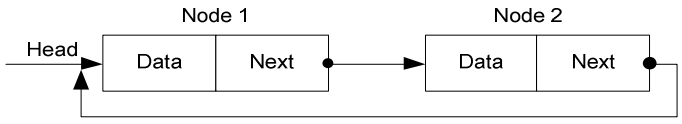

Figure 1. Circular Linked list structure

It can be both singly or doubly linked list. In a circular linked list elements can be added to the back of the list and removed from the front in constant time. Both types of circularly -linked lists benefit from the ability to traverse the full list beginning at any given node. This avoids the necessity of storing first node and last node, but we need a special representation for the empty list, such as a last node variable which points to some node in the list or is null if it's empty. This representation significantly simplifies adding and removing nodes with a non-empty list, but empty lists are then a special case. Circular linked lists are most useful for describing naturally circular structures, and have the advantage of being able to traverse the list starting at any point. They also allow quick access to the first and last records through a single pointer [3].

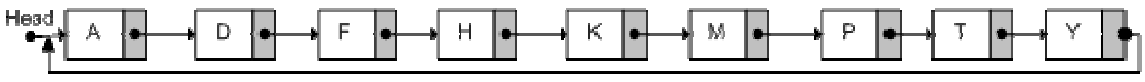

Figure 2. Circular Linked List

2. Skip List

Skip list data structure, which was introduced by Pugh [4,5,6] is a data structure alternative to binary tree search

structure. Linked lists are used in skip list data structure and it is aimed to facilitate searching, insertion and deletion through placing elements in a pyramid-like order at different levels. In this data structure, elements are placed at different levels randomly. First, all nodes are placed at level 0 and, starting from left row and skipping each 2^ith node (i=0,..,MaxLevel (15)), pointers representing each level are created towards the top. The list at level 0 is the linked list at the bottom in skip list data structure and encompasses all nodes. Each list from bottom to the top is arranged as an index of the previous list [1,7].

Search, insertion and delete algorithms of nodes in skip list data structure is discussed in article written by Pugh [4,5]. In addition, several studies have been conducted so far on the improvement and analysis of skip list data structure algorithms. These studies are about level optimization in skip list data structure [1], effects of P threshold values in creation of random level and to the performance of skip list data structure [2], a simple optimistic skip list algorithm [8], analysis of an optimized search algorithm for skip lists [9], skip lists and probabilistic analysis of algorithms [10], deterministic skip lists [11], concurrent maintenance of skip lists [5].

Various data structures and algorithms were also created apart from skip list data structure such as skip graphs [12], tiara (peer-to-peer network maintenance algorithm) [13] and corona [14].

Time complexity is O(N) for search, insertion and deletion processes when linked and ordered lists are used. On the other hand, the time complexity in which these processes are performed is O(lg N) in skip list data structure [6].

In a search algorithm, a node is searched from upper levels to lower levels. During insertion, first, the node to be inserted is searched. If not found, new value is inserted to the matching location starting from a random level and pointers and lists are updated. The process is repeated for other levels where a node is to be inserted. Search is performed from the top level to lower levels for removal operations. The node is deleted when found and pointers and lists are updated. The process is repeated other levels where the node is available [2].
A group of data consisting of {A,C,F,H,K,M,P,T,V,X,Z} elements as a skip list is shown in Figure 3.

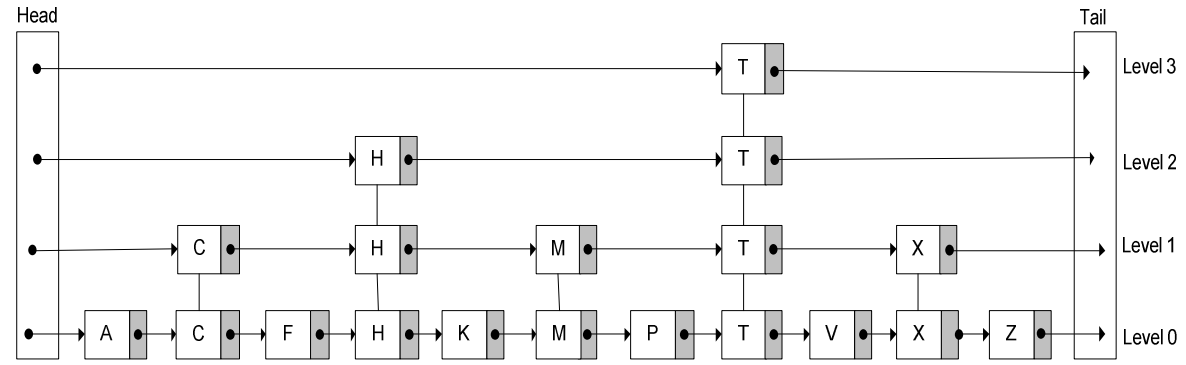

Figure 3. Skip list

3. Skip Ring (Circular Skip List)

As described above, by using the circular linked list and skip list data structures, a new data structure called skip ring (Circular skip list) was developed. Operations are performed only on a single ordered list in circular linked list (Figure 2). However, in our method, nodes are searched, inserted and deleted on circular linked lists (Figure 4) which are linked to each other in levels that are indexes of each other.

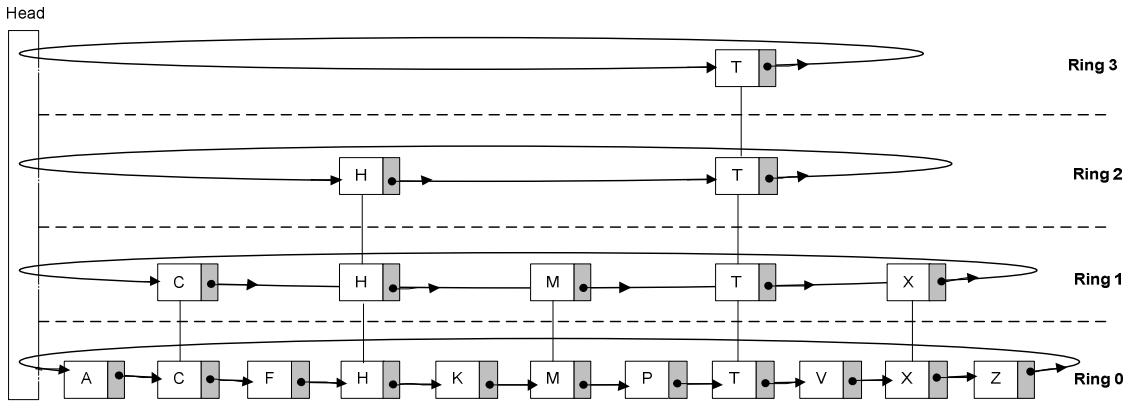

Figure 4. Skip ring

In this new conical data structure, the relationships are defined as Circular linked list 0 = Ring 0,

Circular linked list 1 = Ring 1, ... , Circular linked list (l)=Ring (l) (Figure 4). In skip ring data structure each ring is a sub-sequence of the previous one, Ring 0⊇ Ring 1⊇ ... ⊇ Ring l. Ring 0 is the ring at the bottom level in skip ring data structure and encompasses all elements. Each ring from bottom to the top is lined as an index of previous ring. In addition, skip ring data structure is similar to skip list data structure. In contrast to skip list data structure, head and tail is not required together and only head is enough in skip ring data structure. Our new data structure is created by removing the tail from skip list data structure. In this way the last element (tail) of circular linked list is linked in a way that it points back to the first element (head) (Figure 4), and this process is performed for entire levels.

When Rings in skip ring data structure are created (Ring 0, Ring 1,.., Ring l), levels are created randomly. Let us say that the number of ordered nodes in our skip ring data structure is N. Ring 0 consists of these entire N ordered nodes (Figure 4-Ring 0).

Because N ordered elements are included, search at Ring 0 level is performed in O(N) time complexity. Ring 1 is created if every other element of the list at Ring 0 also has an extra link to the element two ahead of it (Figure 4 – Ring 1). Since the maximum number of elements at Ring 1 level equals $\lceil \frac{N}{2} \rceil + 1$, search is performed within O(N) time complexity. Ring 2 is created if every forth element of the list at Ring 0 also has an extra link to the element four ahead of it (Figure 4-Ring 2). Since the maximum number of elements at Ring 2 level equals $\lceil \frac{N}{4} \rceil + 2$, search is performed within O(N) time complexity. When each 2^ith node (i=0,..,MaxLevel(15 or 31)) is linked to the following 2^ith node via a pointer in this way, since the maximum number of elements at Ring i level equals $\left\lceil \frac{N}{2^i} \right\rceil + i$, time complexity to reach a node in a search equals O(lg N) at maximum.

Similar to skip list data structure searching, insertion and deletion [4,6] can also be performed in our skip ring data structure.

3.1. Node Search

In our skip ring data structure, as described in the previous section, each 2^ith (i=0,..,MaxLevel(15 or 31)) node is linked to the following 2^ith node via a pointer. Thus, a conical structure is created, which allows reaching the required node in a time complexity of O(lg N) at maximum. Search is initiated in the ring at the top level and continues towards rings at lower levels.

Steps to be followed for search in our new data structure are as follows:

- ✓ Determine the “key” value to be searched
- ✓ Check if the skip ring data structure is free or not
- ✓ If not free search the “key” value from top level to lower ones
- ✓ If the “key” is found return the value if not return false

The steps to find ‘K’ node in a skip ring data structure consisting of {A, C, F, H, K, P, T} elements are shown in Figure 5.

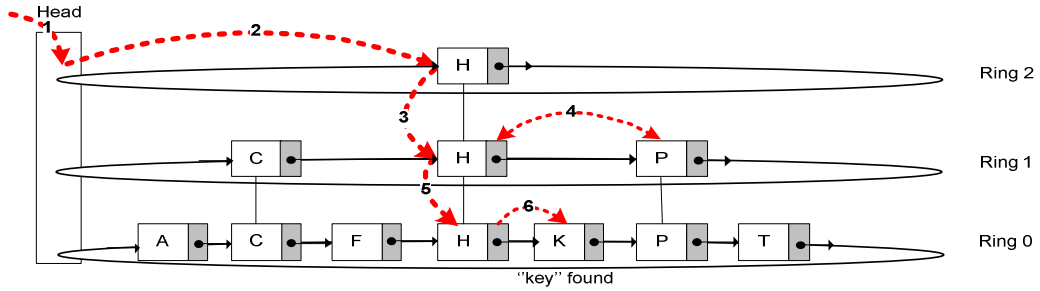

Figure 5. Search node in skip ring

Because it is similar to skip list data structure, search in skip ring data structure can be performed through modifying the algorithm used in skip list.

3.2. Node Insertion

It is required to find the position in order to insert a new node, which requires searching. It is possible to reach a node in a time complexity of O(lg N) at maximum, which is the time complexity for node insertion.

While constructing skip ring data structure from nodes, nodes are placed at random levels. The random_level() algorithm (**Algorithm 1**) creates a random level between 0,..,MaxLevel(15 or 31) to form up levels to insert nodes.

Algorithm 1 { Random level }

```
random_level()
  level <- 0;
  frand<-rand()/RAND_MAX {frand value in 0..1}
  while(frand < P) and (level < MaxLevel)    { P = 1/2 }
     level<-level+1;
  return level;
```

The steps must be followed in order to insert a new node:

- ✓ Define the "key" value to be added
- ✓ Search the "key" value starting from the top level to lower levels.
- ✓ If found, return false and exit
- ✓ If not found, create a new node (M), create a level randomly (Algorithm 1), Assign the value of this node as "key"
- ✓ Update the structure in a way that the pointer of node prior to M shows M and pointer of M represent the node next to M.
- ✓ Insert the node for all the necessary levels.

The steps to insert 'M' node (random level=3) in a skip ring data structure consisting of {A, C, F, H, K, P, T} elements are shown in Figure 6.

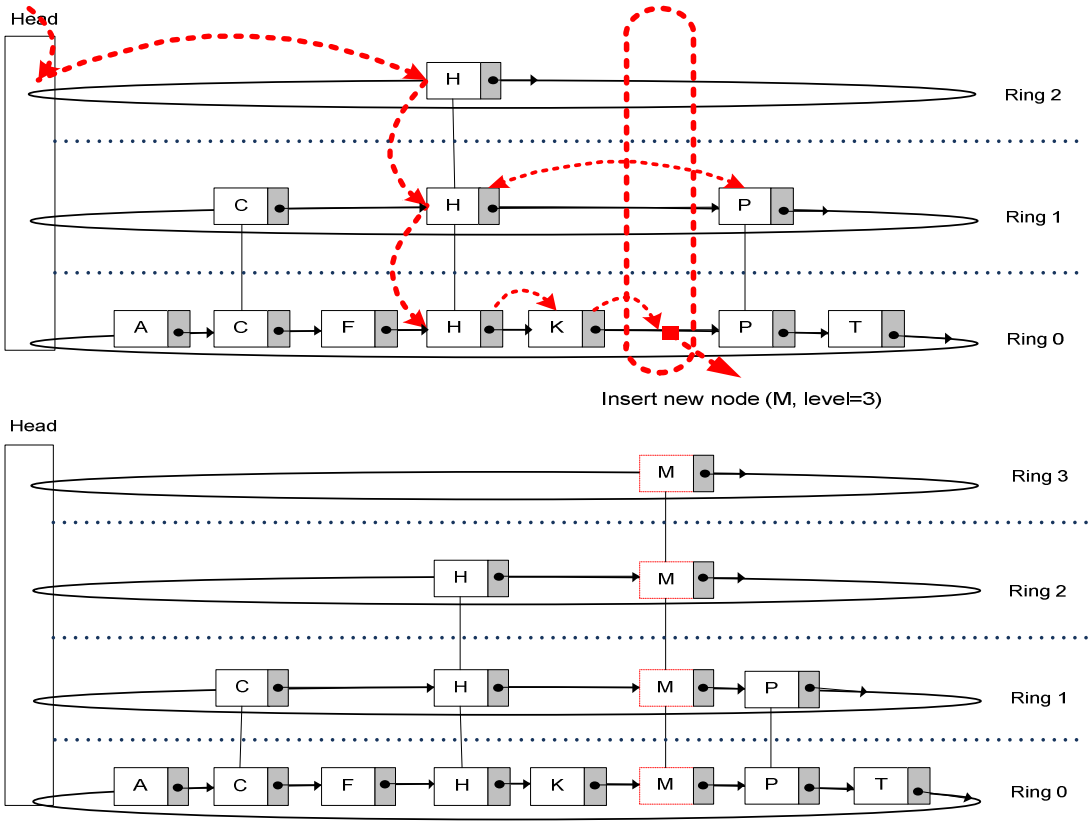

Figure 6. Insert new node and update skip ring

3.3. Node Removal

It is required to search and find a node in order to delete a node. It is possible to reach a node in a time complexity of O(lg N) at maximum, which is the time complexity for node deletion.

Deletion of a node in a skip ring data structure is similar to that of a skip list data structure [4,6]. Similar to search algorithm, the first thing that must be controlled during removal operation is that not a single element can be available in skip ring data structure.

These steps must be followed in order to delete a node:

- ✓ Define the node (M) to be removed
- ✓ Search this node starting from the top level to lower levels
- ✓ If node (M) is found, update the list in a way that pointer of the node prior to M shows the node after M.
- ✓ Remove the node (M) from all levels
- ✓ Remove the node from the memory

3.4. Properties of Skip Ring Data Structure

Skip ring data structure is created through circular linked list and skip list data structures, which requires reflecting the properties of these data structure. It is aimed to focus on the properties of this data structure as a mathematical model and its functions during processes performed in this data structure.

Theorem 1. Given that $0 \le j \le MaxLevel$ and $1 \le k \le MaxIndex_j$, let us say that S(j,k) is skip ring data structure, S(0,k) is the linked list at the bottom of Skip Ring data structure and the index set of nodes at 0 level, $I(S(0,k))=\{i_1,i_2,\ldots,i_k,..,i_{MaxIndex}\}$. If $\frac{i_k}{2^j} \in Z^+$, then $i_k \in I(S(j,k))$ (I(S(j,k)) is the index set for Level j).

Proof. This theorem can be proved through mathematical induction.

Step 1: At Level 0, some nodes may be moved to the next level(Level 1). When N is the number of nodes at Level 0, the index set for nodes to be moved to the next level is {2, 4, …, N} or {1, 3, …, N-1} if N is even and, as a result,

indexes at Level 1 will be as $\left\{\left\lceil\frac{2}{2}\right\rceil,\left\lceil\frac{4}{2}\right\rceil,...,\left\lceil\frac{N}{2}\right\rceil\right\}$=I(S(1,...)) or $\left\{\left\lceil\frac{1}{2}\right\rceil,\left\lceil\frac{3}{2}\right\rceil,...,\left\lceil\frac{N-1}{2}\right\rceil\right\}$=I(S(1,...)),

respectively.

If N is odd, the indexes of nodes to be moved to Level 1 will be as {2,4,…, (N-1)} or {1,3,…,N} and their indexes at Level 1 will be as $\left\{\left\lceil\frac{2}{2}\right\rceil,\left\lceil\frac{4}{2}\right\rceil,...,\left\lceil\frac{N-1}{2}\right\rceil\right\}$ =I(S(1,...)) or $\left\{\left\lceil\frac{1}{2}\right\rceil,\left\lceil\frac{3}{2}\right\rceil,...,\left\lceil\frac{N}{2}\right\rceil\right\}$ =I(S(1,...)),

respectively where $\lceil . \rceil$ function is the ceiling function.

Step 2: Assume that the condition is valid for Level=j=MaxLevel-1.

Step 3: Given that the set of nodes at Level MaxLevel-1 is I(S(MaxLevel-1,…)), assume that the indexes of nodes at this level are $\{1,2,\ldots,MaxIndex_{MaxLevel-1}\}$. If $MaxIndex_{MaxLevel-1}$ is even, the index list of nodes to be moved to MaxLevel are $\{2,4,\ldots,MaxIndex_{MaxLevel-1}\}$= I(S(MaxLevel-1,…)) and the index set at MaxLevel will be as {2/2, 4/2, …, $(MaxIndex_{Maxlevel-1})/2$}=I(S(MaxLevel,…)). If $MaxIndex_{MaxLevel-1}$ is odd, the index list of nodes to be moved to MaxLevel $\{2,4,\ldots,(MaxIndex_{MaxLevel-1}-1)\}$=I(S(MaxLevel-1, …)) and the index set at MaxLevel will be as {2/2, 4/2, …, $(MaxIndex_{Maxlevel-1}-1)/2$}= I(S(MaxLevel,…)).

Theorem 2. Given that $0 \le i \le MaxLevel$ and $1 \le j \le MaxIndex_i$, let us assume that S(i,j) is a skip ring data structure. MaxLevel is the index level at the top level and |S(i,..)| equals the number of nodes at i^{th} level. Given that N is the number of nodes at the bottom level,

$$\sum_{i=1}^{\lceil \lg N \rceil} \sum_{j=1}^{MaxIndex_i} 1 \le N$$

Proof. |S(0,…)|=N and $\lceil . \rceil$ is the ceiling function. Those nodes which have even indexes at Level 0 are moved to the next level and their indexes are halved. In this case, the number of nodes at Level 1 is the half of those at Level 0 or one less than it. This means that the number of nodes at Level 1 is $\left\lceil\frac{|S(0,..)|}{2}\right\rceil$. Then, the number of nodes at Level 2 will be as $\left\lceil\frac{|S(1,..)|}{2}\right\rceil$. As a result, the number of nodes at MaxLevel will be as $\left\lceil\frac{|S(MaxLevel-1,..)|}{2}\right\rceil$. In this case, the number of nodes at all levels except Level 0 is $\sum_{i=1}^{MaxLevel}\left\lceil\frac{|S(i,...)|}{2}\right\rceil$ and

MaxLevel equals $\lceil \lg N \rceil$. In other words, the maximum value of $\sum_{i=1}^{MaxLevel} \left\lceil \frac{|S(i,...)|}{2} \right\rceil$ will be N, which is the exponent of 2. If $N=2^r$ and $r \in Z^+$. In this case, the number of nodes at all levels will be as $|S(1,\ldots)|=2^{r-1}$, $|S(2,\ldots)|=2^{r-2}$, …, $|S(MaxLevel,\ldots)|=1$ and $\lceil \lg N \rceil = r$. Except Level 0, the number of nodes at all other levels will be as

$$\sum_{i=1}^{r-1} 2^i = \frac{1-2^r}{1-2} = 2^{\lceil \lg N \rceil} - 1 = 2^{\lg N} - 1 = N - 1.$$

Theorem 3. When S is a skip ring, the number of nodes in this skip ring will be as $\sum_{i=0}^{\lceil \lg N \rceil} \left\lceil \frac{N}{2^i} \right\rceil = N + \sum_{i=1}^{\lceil \lg N \rceil} \left\lceil \frac{N}{2^i} \right\rceil$ and at level i, it is $\left\lceil \frac{N}{2^i} \right\rceil$.

Proof. Given that $N_0=N$, let us say that N_1 is the number of nodes at Level 1 and it is N_2 at Level 2. The number of nodes at the last level is $N_{\lceil \lg N \rceil}$.

$$N_i = \left\lceil \frac{N_{i-1}}{2} \right\rceil = \begin{cases} \frac{N_{i-1}}{2} & if \quad \frac{N_{i-1}}{2} \in Z^+ \\ \frac{N_{i-1}-1}{2} & if \quad \frac{N_{i-1}-1}{2} \in Z^+ \end{cases}$$

given as a ceiling function, the link between the number of nodes at different levels will be as

$N_1 = \left\lceil \frac{N_0}{2} \right\rceil$, $N_2 = \left\lceil \frac{N_1}{2} \right\rceil$, …, $N_{\lceil \lg N \rceil} = \left\lceil \frac{N_{\lceil \lg N \rceil - 1}}{2} \right\rceil$ which proves our argument.

Theorem 4. When S is a skip ring and $\forall k_1$, k_2 are node indexes and i and j are level indexes, following arguments are valid.

a) If i=j, then $S(i,k_1) \le S(j,k_2)$ for $k_1<k_2$.
b) If i<j and $k_1 \le 2^{j-i}k_2$ and if $S(i,k_1) \le S(j,2^{j-I}k_2)$ or j<i and $k_1 \le 2^{j-i}k_2$, then $S(j,k_1) \le S(j,2^{i-j}k_2)$.

Proof. First, the data structure at each level of skip ring is a linked list. Therefore, step a) is proved because nodes are placed on this linked list in an ascending form.

b) Nodes at the bottom level the indexes of which are even are moved to a level. Their indexes equal to the indexes of nodes at the previous level divided by two. In this case, if i<j and j-i>0, the index of node at level j is divided by 2, j-i times. Therefore, the index of node at the bottom level equals to the index multiplied by 2^{j-i}. If i<j, k_1 index is the 2^{j-i} times as much as k_2 index. In this case, $k_1 \le 2^{j-i}k_2$, due to condition is provided because linked list is in an ordered one $S(i,k_1) \le S(j,2^{j-i}k_2)$. The other condition is the reverse of this.

Theorem 5. When N is the number of nodes at the bottom level (Level 0), the longest path in search process in skip ring data structure will be as $\lceil \lg N \rceil + 1$.

Proof: Under normal circumstances, the number of nodes at the top level is 1. In this case, it is determined whether to move to the next level or not through a single comparison at this level. If the node searched is found at this level, the path comes to an end. If not, it moves to the lower level and the comparison at this level will be as ~1. When it reaches the bottom level in this way, a path corresponding to the level in skip ring data structure is followed, which is defined as $\sim \lceil \lg N \rceil + 1$.

4. Conclusions

Skip list, in which linked lists are used and search, insertion and deletion is performed rapidly, is a fast and dynamic data structure introduced by Pugh. Skip list data structure is important because it reduces O(N), which is the time complexity for search, insertion and deletion processes in linked list data structure, to O(lg N).

Skip ring data structure is created with the help of skip list and circular linked list data structures. Thanks to its conical and layered structure, skip ring data structure presented in this study reduces O(N), which the time complexity for search, insertion and deletion processes in circular link list data structure, to O(lg N) time complexity. Therefore, it can be used anywhere both data structures are used.

Circular linked list data structure is used for various purposes requiring circular processing. Similarly, our skip ring data structure can be used more efficiently in those circumstances where circular processing is required (O(lg N)). For instance, skip ring data structure is suitable for memory management, task management

and process scheduling in operating systems.

In conclusion, just as skip list reduces O(N), which is the time complexity for search, insertion and deletion processing linked list data structure, to O(lg N), our data structure, skip ring, reduces time complexity from O(N) to O(lg N) in circular linked list. This enables to save remarkable time when larger sets of data are taken into consideration.

References

[1]. Aksu, M., Karcı, A., Yılmaz, Ş. (2013), "Level optimization in Skip List data structure", ISITIES2013 (1ST International Symposium on Innovative Technologies in Engineering and Science), 389-396.
[2]. Aksu, M., Karcı, A., Yılmaz, Ş. (2013), "Effects of P Threshold Values in Creation of Random Level and to the Performance of Skip List Data Structure", Bitlis Eren University Journal of Science, 2(2), 148-153.
[3]. Shaffer, C. A. (2011), "Data Structures & Algorithm Analysis in C++", Dover Publications, 95-226.
[4]. Pugh, W. (1989), "A Skip List Cookbook", UMIACS–TR–89–72.1, Dept. of Computer Science, University of Maryland, College Park.
[5]. Pugh, W. (1989), "Concurrent Maintenance of Skip Lists", TR–2222, Dept. of Computer Science, University of Maryland, College Park.
[6]. Pugh, W. (1990), "Skip Lists: A Probabilistic Alternative to Balanced Trees", Communications of the ACM, 33(6), 668–676
[7]. Goodrich, M. T., Tamassia, R. (2014), Algorithm Design and Applications, Wiley, America, 557-562.
[8]. Herlihy, M., Lev, Y., Luchangco, V., Shavit, N. (2007), "A Simple Optimistic Skiplist Algorithm", SIROCCO, 124-138.
[9]. Kirschenhofer, P., Martinez, C., Prodinger, H. (1995), "Analysis of an optimized search algorithm for skip lists", Theoretical Computer Science, 144; 199-220.
[10]. Papadakis, T. (1993), "Skip lists and probabilistic analysis of algorithms", Ph.D. Thesis, University of Waterloo, Tech. Report CS-93-28.
[11]. Munro, J.I., Papadakis, T., Poblete, P.V. (1992), "Deterministic Skip Lists", Proceeding SODA '92 Proceedings of the third annual ACM-SIAM symposium on Discrete algorithms, 367-375.
[12]. Aspnes J., Shah, G. (2003), "Skip graphs", 14th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 384–393.
[13]. Clouser, T., Nesterenko, M., Scheideler, C. (2012), "Tiara: A self-stabilizing deterministic skip list and skip graph", Theoretical Computer Science, Volume 428, 18–35.
[14]. Nor, R.M., Nesterenko, M., Scheideler, C. (2013), "Corona: A Stabilizing Deterministic Message-Passing Skip List", Theoretical Computer Science, Volume 512, 119–129.

The IISTE is a pioneer in the Open-Access hosting service and academic event management. The aim of the firm is Accelerating Global Knowledge Sharing.

More information about the firm can be found on the homepage:
http://www.iiste.org

CALL FOR JOURNAL PAPERS

There are more than 30 peer-reviewed academic journals hosted under the hosting platform.

Prospective authors of journals can find the submission instruction on the following page: http://www.iiste.org/journals/ All the journals articles are available online to the readers all over the world without financial, legal, or technical barriers other than those inseparable from gaining access to the internet itself. Paper version of the journals is also available upon request of readers and authors.

MORE RESOURCES

Book publication information: http://www.iiste.org/book/

Academic conference: http://www.iiste.org/conference/upcoming-conferences-call-for-paper/

IISTE Knowledge Sharing Partners

EBSCO, Index Copernicus, Ulrich's Periodicals Directory, JournalTOCS, PKP Open Archives Harvester, Bielefeld Academic Search Engine, Elektronische Zeitschriftenbibliothek EZB, Open J-Gate, OCLC WorldCat, Universe Digtial Library , NewJour, Google Scholar