Leader Election using Modified Heap Tree Method

Dinesh Kumar Yadav
Deptt. of IT
IPEC,Ghaziabad

C. S. lamba
Deptt.of CS
RIET,Jaipur

Shweta Shukla
Deptt.of CS
RIET,Jaipur

ABSTRACT

In distributed system field, there are many challenges, and one of them is leader election. It is really tough task to find suitable and efficient algorithms for leader election. The main role of an elected leader is that it performs a centralized coordination after being selected and manages the use of a shared resource in an optimal manner. Whenever a failure occurs the new leader is elected by nodes using various algorithms so that nodes can continue working. In this paper, the proposal is a new approach, the improved heap tree mechanism for electing the coordinator. The higher efficiency and better performance in the presented algorithms with respect to the existing algorithms is validated through results

General Terms

Algorithm, Heap tree, MAX-HEAPIFY.

Keywords

Modified Heap tree, Modified Heap tree method

1. INTRODUCTION

A Leader is a member of n nodes that all other nodes acknowledge as being distinguished to perform some special task [1]. The Leader election problem is the problem of choosing a leader from a given set of candidates. Each node has a unique id. A distributed system is collection of autonomous computing nodes which can communicate with each other and which cooperate on a common goal or task [2].

A leader performs a centralized coordination after being selected [1]. This may be necessary if some problems there a completely distributed solution in either not available or offers less attractive performance whenever some failure occurs it is necessary for the nodes to adapt new condition so that they can continue working. This requires some kind of reorganization. Leaders are elected to manage the reorganization.

In any leader election algorithm, a leader is usually chosen based on some criterion such as choosing the node with the largest identifier [3]. Once the leader is elected, the nodes reach a certain state known as *terminated* state. In leader election algorithms, such states are partitioned into *elected states* and *non-elected states* [4].When a node enters either state, it always remains in that state. Every leader election algorithm must be satisfied by the safety and liveness condition for an execution to be admissible [5]. The liveness condition states that every node will eventually enter an elected state or a non-elected state. The safety condition for leader election requires that only a single node can enter the elected state and eventually, become the leader of the distributed system.

Several leader election algorithms such as the Bully algorithm [2], Ring algorithm [6], Chang and Roberts' algorithm [7], Peterson's algorithm [8], LeLann's algorithm [9], and Franklin's algorithm [10] have been proposed over the years. These algorithms, however, require nodes to be directly involved in leader election. Information is exchanged between nodes by transmitting messages to one another until an agreement is reached. Once a decision is made, a node is elected as the leader and all the other nodes will acknowledge the role of that node as the leader.

In this paper there is a proposal of modified heap tree method in electing the leader nodes by imposing lesser complexity in terms of time and message passing comparing to earlier proposed heap tree method for leader election [3]. Also there is a comparison in proposed algorithms with the existing algorithms.

2. RELATED WORK

There are lot of work already have done in the field of leader election. Several leader election algorithms such as the Bully algorithm [2], Ring algorithm [6], Chang and Roberts'algorithm [7], Peterson's algorithm [8], LeLann's algorithm [9], and Franklin's algorithm [10] have been proposed over the years. These algorithms, however, require nodes to be directly involved in leader election. Information is exchanged between nodes by transmitting messages to one another until an agreement is reached. Once a decision is made, a node is elected as the leader and all the other nodes will acknowledge the role of that node as the leader.

There is also a new approach has been proposed for leader election using heap tree method [3]. In this approach formal heap tree method is used. Here MAX-HEAPIFY () procedure is used which runs in O(log(n)) is the key of maintaining MAX-HEAP property.

3. PROPOSED ALGORITHM FOR LEADER ELECTION USING IMPROVED HEAP TREE METHOD

In this section, there is a description of a modified heap tree-based algorithm for leader election. In this approach, each node of the tree corresponds to an element of the array that stores the value in the node. The tree is completely filled on all levels except possibly the lowest, which is filled from the left up to a point. An array A that represents a heap is an object with two attributes: length[A] and heap-size[A], which are the number of elements in the array and in the heap stored within array A, respectively. Although A[1..length[A]] may be valid, no element past A[heap-size[A]] is an element of the heap, where heap-size[A] ≤ length[A]

The root of the tree is A[1], and given the index i of a node, the indices of its parent PARENT(i), left child LEFT(i), and right child RIGHT(i) can be computed easily. Based upon the type of heap being used, the values in the nodes satisfy a modified heap property.

The MAX-HEAPIFY() procedure, which runs in O(log n) time, is the key to maintaining the max-heap property. The BUILD-MAX-HEAP() procedure, which runs in linear time, produces a max-heap from an unordered input array. The MAX-HEAP-INSERT(), HEAP-EXTRACT-MAX(), HEAP-INCREASE-KEY() and HEAP-MAXIMUM() procedures, which run in O(log n) time, allow the heap to be used as a priority queue

A new variant of Heap Sort is modified heap sort [11]. Basic idea of new algorithm is similar to classical Heap sort algorithm but it builds heap in another way. This new algorithm requires nlogn-0.788928n comparisons for worst case and nlogn-n comparisons in average case. This algorithm uses only one comparison at each node. With one comparison we can decide which child of node contains larger element. This child is directly promoted to its parent position In this way algorithm walks down the path until a leaf is reached.

In this way we can reduce one comparison at each step, because there is no need of comparison between children and it is well known that LEFT(i) ≥ RIGHT(i) so left child will be directly promoted to its parent position.

In this structure when the root is deleted from the tree, It is clear that the leader has crashed. As shown in Figure 3, when a node realizes that the leader has crashed, it sends the election message to its father. This message traverses up to the children of the deleted root where the left or right children of the deleted root whosoever receive the election message no need of comparing their IDs with each other to determine the new leader because In Modified Heap Structure left child ID is greater than or equal to right child ID. So any child either left or right whosoever receive election messages elect, left child of deleted father as a leader. So In this way we can minimized and optimized the execution time as well as message passing as compared to normal heap tree method

It is not necessary for all nodes to start sending their own IDs or election message in the tree. The election message sent by the node reaches its direct father in the tree and at this moment, the receiving node analyzes this message to determine whether it is a duplicate message or not. If duplicate, it is dropped by that node, otherwise it is sent to the next father in the tree. By doing so, the leader can be selected in less than O(log n) time at the expense of a comparably reduced number of messages. In this method, each node should save the information of its father, left and right children, and its sibling. This approach requires a same memory space 4n equal to the heap tree method.

Table1: AVERAGE SORTING TIME (IN MSEC) OF ALGORITHMS ON RANDOM DATA AVERAGED 50 RUNS

No.of data item----->	**1000**	**5000**	**10000**	**50000**	**100000**
Heap sort	.0003	.0045	.0033	.0848	.0895
Modified heap sort	.0003	.0057	.013	.0454	.051

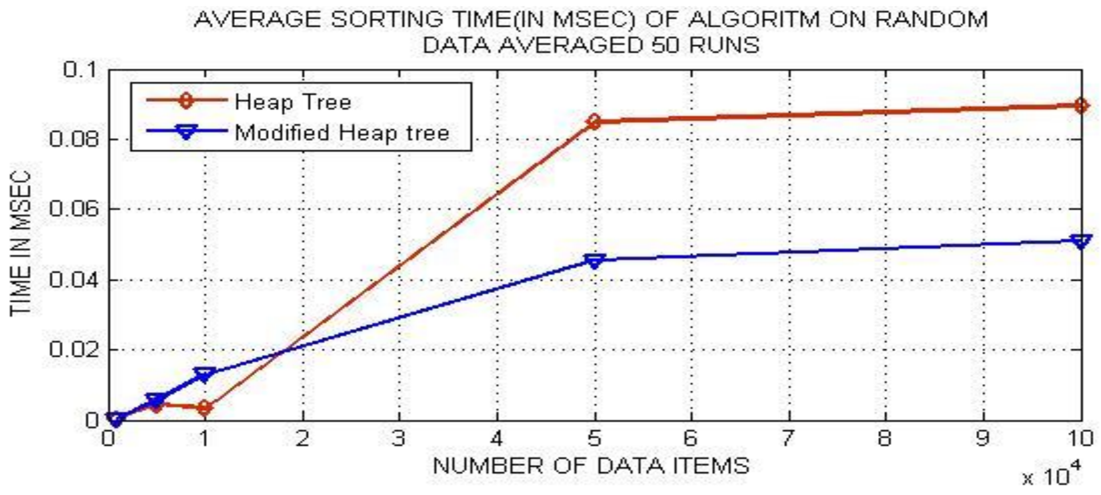

Fig1: Average Sorting Time (In MSEC) Of Algorithms On Random Data Averaged 50 Runs

The above graph shows that modified heap sort algorithm performs better than heap sort algorithm for larger data items [12] and requires nlogn-0.788928n comparisons for worst case and nlogn-n comparisons in average case [11] if it uses Gonnet and Munro's [13] fastest algorithm for building heaps. This algorithm uses only one comparison at each node but normal HEAPSORT needs 2nlogn comparisons.

4. ANALYSIS OF MODIFIED HEAP TREE METHOD FOR LEADER ELECTION

Let's take an example of modified heap tree method of leader election, number of nodes=20.At some time leader crashed and node A[8] =17 came to know about that after that it sends an election message to its father A[4]=23 then A[4] send it to its father. A[2]=27 which is the child of crashed father. In modified heap tree method in this case only two messages will be required for leader election because when child of crashed father received election message from its child it does not send election message or does not compare the ID with its sibling it just elect left child of crashed father as a leader and broadcast this

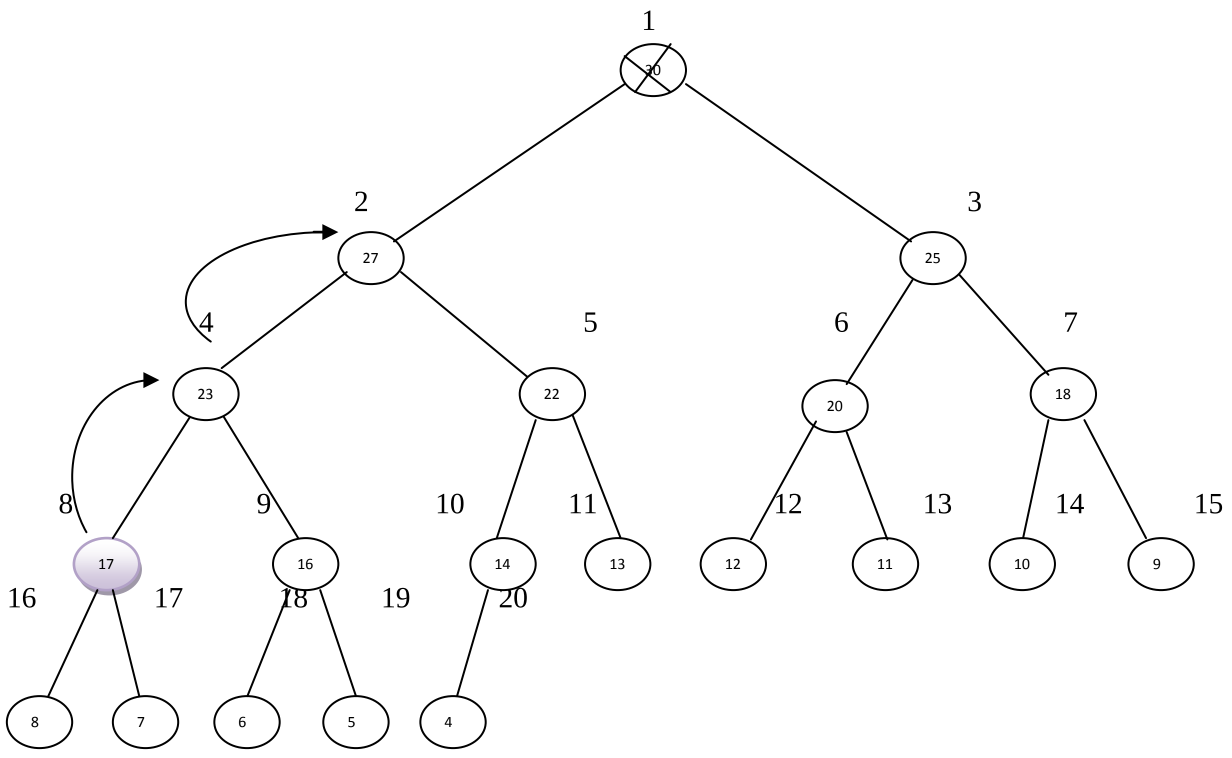

Fig 2: Leader Election using modified heap tree method when leader has crashed

selection message. So total message send in this case =2 for leader election.

In the case of heap tree method

When A[8]=17, came to know about leader failure it sends a message to its father A[4]=23 and A[4] send it to its father A[2]=25 which is the child of crashed leader after that A[2] sends message to A[3] and both compare their IDs and node with larger ID promote as a leader and after that selection message broadcast. In this way in Heap tree method total message send = 3 and one comparison extra as compared to modified heap tree method.

In this way time as well as message can be reduced by using modified heap tree method in place of heap tree method.

Maximum Number of message sent at the time of leader election

At any time T leader election starts and we assume that every node send a message to its father for leader election except root. So if total number of nodes =n and no nodes send any duplicate method then maximum number of message sent at the time of leader election will be (n-1).

So M_{max} = n-1 where M_{max} is the maximum number of message sent at the time of leader election

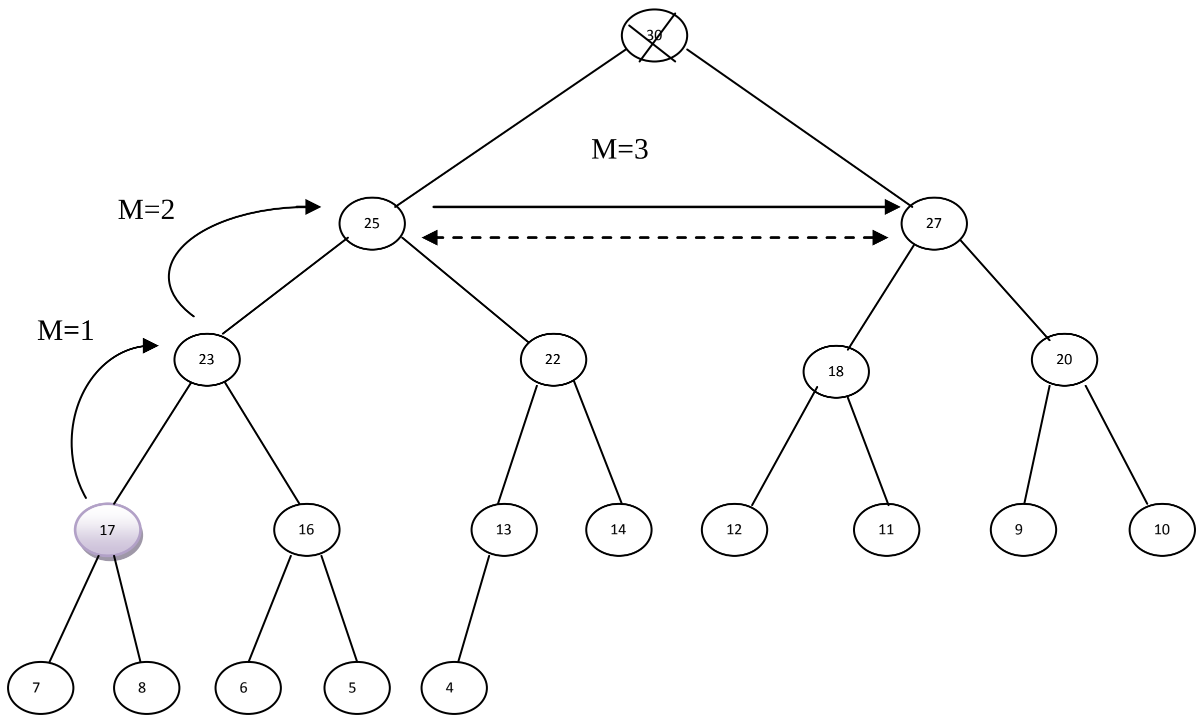

Fig 3: Leader Election using heap tree method when leader has crashed

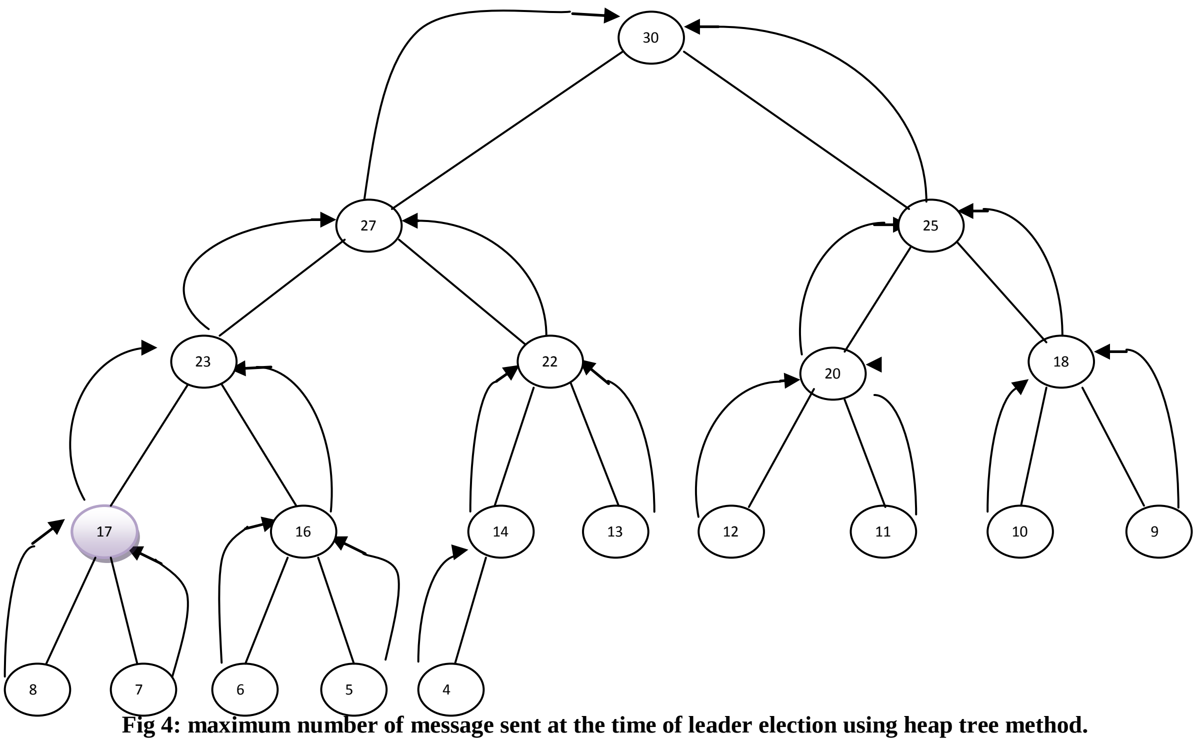

Fig 4: maximum number of message sent at the time of leader election using heap tree method.

Table1: AVERAGE SORTING TIME (IN MSEC) OF ALGORITHMS ON RANDOM DATA AVERAGED 50 RUNS

Method	**Total memory needed**	**Order**	**Minimum message**	**Maximum message**	**Approximate no of message when leader is crashed**
Modified Max Heap	4n	log(n)	log(n)	n-1	$\sum_{i=1}^{k}\lfloor\log(Ci)\rfloor-[\sum_{i,j=1;i\neq j}^{k}\lfloor\log(\max(Ai\cap Bj))\rfloor]-1;$ $Ai=\{f(Ci)=[Ci,f(\lfloor Ci/2\rfloor)]\mid 0<Ci\}and$ $Bj=\{f(Cj)=[Cj,f(\lfloor Cj/2\rfloor)]\mid 0<Cj\}$
Max Heap	4n	log(n)	log(n)	log(n)+(n-1)	$\sum_{i=1}^{k}\lfloor\log(Ci)\rfloor-[\sum_{i,j=1;i\neq j}^{k}\lfloor\log(\max(Ai\cap Bj))\rfloor];$ $Ai=\{f(Ci)=[Ci,f(\lfloor Ci/2\rfloor)]\mid 0<Ci\}and$ $Bj=\{f(Cj)=[Cj,f(\lfloor Cj/2\rfloor)]\mid 0<Cj\}$
Bully	n^2	n^2	2n-2	n^2	$Ni=(n-i+1)(n-i)+n-1$
Ring	n^2	n^2	n	n^2	$\sum_{i=1}^{n}(n-i)=1/2[(n-i)(n-i+1)]$

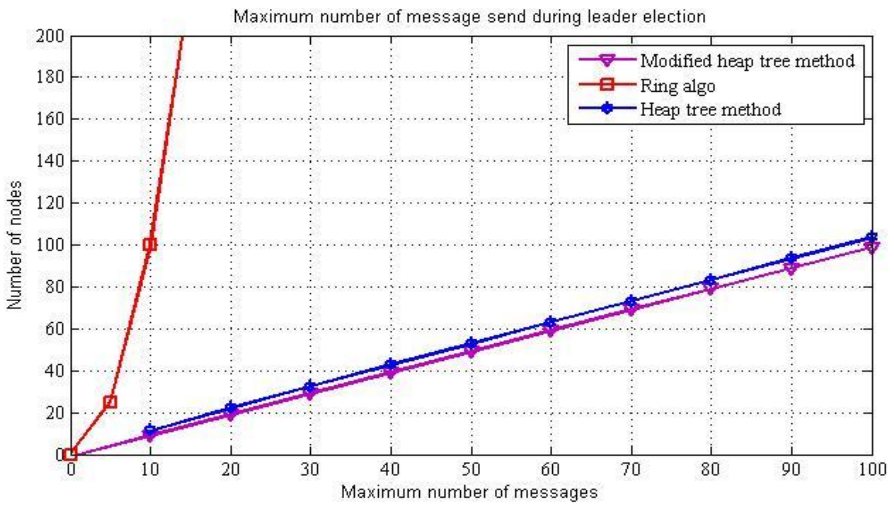

Fig 5: Leader Election using heap tree method when leader has crashed

5. CONCLUSION AND FUTURE WORK

Selection of leader election algorithm in a distributed system plays a vital role in the performance of the system and there should be a proper tradeoff between time and message complexities. The proposed algorithm in this paper is an attempt to improve leader election algorithm (using heap tree method) by using modified heap tree method. Lesser complexity and less number of message send in modified heap tree method shows that it will perform better than earlier proposed algorithm. It has already proved that modified heap tree perform better than heap tree for large number of data items. By using proposed algorithm proper balance between time and message can be obtained. Proposed algorithm sends n-1 maximum message which is quite less from earlier proposed algorithms.

In proposed method Build heap procedure can be further optimized so that it takes lesser time and this will improve the performance of proposed algorithm of leader election. In future we tend to adopt this approach in ad hoc and sensor environment.

6. REFERENCES

[1] Princy Francis and Sanjeev Saxena ,IEEE conference ,1998,Optimal Distributed Leader Election Algorithm For Synchronous complete Network.

[2] H. Gracia-Molina, IEEE Trans. on Computers, vol. C-31, no. 1, Jan. 1982 "Elections in a distributed computing system"

[3] Mohammad Reza EffatParvar , Nasser Yazdani, Mehdi EffatParvar , Aresh Dadlani and Ahmad Khonsari,IEEE conference,2010, Improved Algorithms for Leader Election in Distributed Systems .

[4] E. Korach, S. Moran, and S. Zaks, in Proc. 3rd ACM Symp. on Principles of Distributed Computing, Vancouver, Canada, pp. 199-207,Aug. 1984, "Tight lower and upper bounds for some distributed algorithms for a complete network of processors".

[5] P. M. B. Vitanyi, "Distributed election in an Archimedean ring of processors", USA, pp. 542-547, 1984,in Proc. 16th ACM Symp. on Theory of Computing, Washington.

[6] N. Fredrickson and N. Lynch,Journal of ACM, vol. 34, no. 1, pp. 98-115", 1987 "Electing a leader in a synchronous ring"

[7] E. Chang and R. Roberts, Communications of the ACM, vol. 22, no. 5, pp.281-283, May 1979 "An improved algorithm for decentralized extrema-finding in circular configurations of processes".

[8] G. L. Peterson, ACM Trans. Programming Languages and Systems, pp. 758-762, Oct. 1982 "An O(n log n) unidirectional algorithm for the circular extrema problem".

[9] G. LeLann, Information Processing Letters, pp. 155-160, 1977 "Distributed systems - towards a formal approach".

[10] W. R. Franklin, Communication of the ACM, pp. 336-337, 1982 "On an improved algorithm for decentralized extrema finding in circular configurations of processors".

[11] Xio Dong Wang, Ying Jie Wu, Journal of Computer Science and Technology. 22(6): 898-903 An improved heap sort algorithm with nlogn –0.788928n comparisons in worst case.

[12] Vandana Sharma, Parvinder S. Sandhu, Satwinder Singh, and Baljit Saini, World Academy of Science, Engineering and Technology 42, 2008,Analysis of Modified Heap Sort Algorithm on Different Environment.

[13] Gonnet G H, Munro J I, 1986, 15(6): 964-971, Heaps on Heaps. SIAM Journal on Computing.

[14] McDiarmid C J H, Journal of Algorithms, 1989, 10(3): 352~365,Reed B A. Building Heaps Fast.