I J C T A, 9(14) 2016, pp. 6675-6682

The Mode from a Sequence of Numbers

Reshma Viswanath* **and R Nandakumar***

ABSTRACT

The mode of a sequence of numbers is defined as the most frequently occurring number in the sequence. A naïve algorithm with two nested loops finds the mode with the complexity of O (n^2). We can also find the mode by sorting the elements and making one pass through the sorted sequence – a complexity of O (n log n). In this paper, we study the so-called Tournament Method and describe a variation of it that finds the mode of n numbers with O (n) complexity, if it is guaranteed that the mode has frequency at least (n/2) +1 time. We present a recursive enhancement of the tournament method for the case where the mode repeats (n/a) +1 times where a is an integer greater than 2.

Keywords: Mode; Tournament method; Median and selection

1. INTRODUCTION

As noted above, the naive method of listing all elements and their frequencies to find the mode is inefficient. First sorting the array and then traversing thru the sorted array is more efficient but still takes O (n log n) comparisons.

If we are guaranteed that the mode in the given sequence of n numbers is repeating at least (n/2) +1 time, then a Sieve method (median and selection method) can be applied. The median and selection method is used to find the rank of any given number in a given sequence .We note that if it is sure that any number occurs more than (n/2) +1 time, then the median and mode will be the same. This algorithm has a complexity of O (n) because there is a recursive algorithm to find the median of n numbers in O (n) time [1].

We now briefly survey the tournament data structure and show how it can be used to find the mode of an elements in O(n) time if we have the guarantee that frequency of mode is equal or greater than (n/2) +1.

The tournament method follows a tree structure. A comparison based tournament is conducted among the elements. Tournament tree is in form of min (max) heap is a complete binary tree. Each external node represents a player and internal node represents winner. In a tournament tree all internal node contains winner and each leaf node contains one player. We can find the 'winner' (the smallest or largest element) in n-1 comparisons as shown in the figure below.

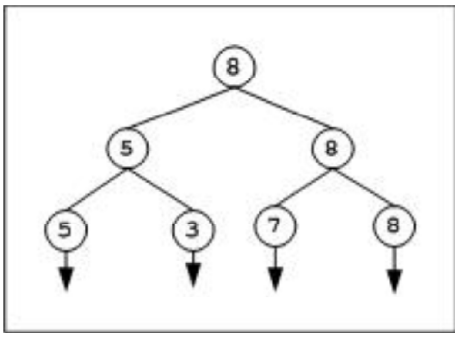


Figure 1

* Department of CS & IT, Amrita School of Arts and Sciences Kochi, Amrita Vishwa Vidyapeetham, India, *E-mails: reshma0219@gmail.com; nandacumar@gmail.com*

Note 1: Since the second best player can only be defeated by the overall winner, it has to be one of the opponents of the eventual winner; so we could conduct another tournament among only those elements which were compared with the winner and find second best player in (n + log2n – 2) comparisons. For this, every winner in each level will need to keep the information about the players he defeated.

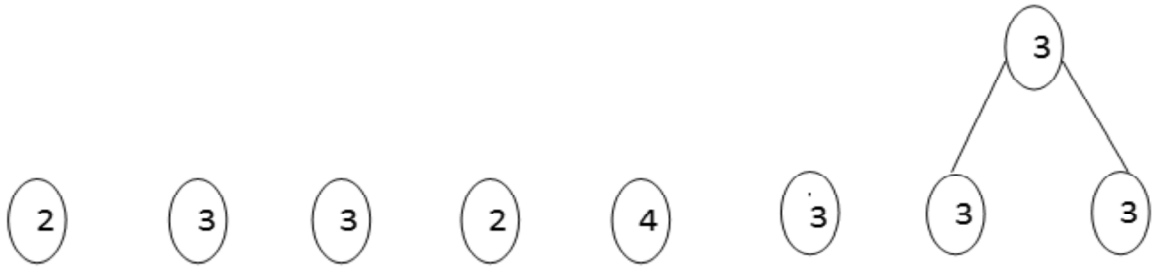
Note 2: Figure 1also reveals that the tournament tree (winner tree) has a max (heap) property [2]. By using the tournament, we could sort an array of integers in O (n log n) time. This method consists of forming the tournament and treating the tournament as a heap to perform heap extractions until the tournament is empty. The numbers extracted will be in sorted order. We note that the tournament for n numbers will have 2n elements due to repetitions (for example, the winner will recur as many times as there are levels in the tournament) and during extractions, we need to disregard repeated occurrences of the same element to get the sorted order. Each extraction from the tournament, viewed as a heap will take only O (log n) time.

2. PROPOSED SYSTEM FOR FINDING MODE

We first describe a tournament based method to find the mode among n elements when we have a guarantee that that mode is repeating (n/2) +1 time. Ii the tournament method only the winner is promoted to next hierarchy; in our modified method, if the two contesting participants have same value, then the value is selected to next level and if the two are different, neither is promoted. The eventual winner of this modified tournament is the mode. We illustrate the method and describe some crucial details with examples below,

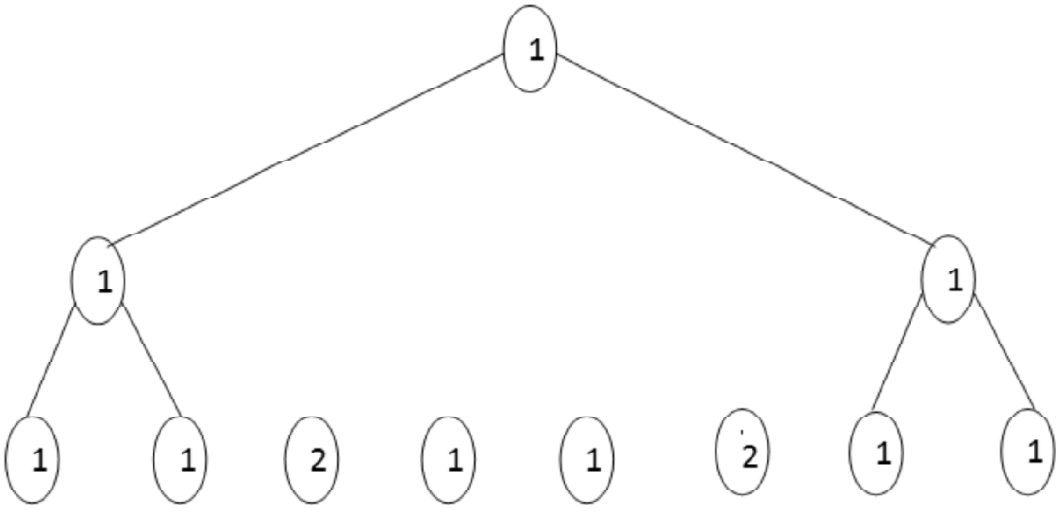
1. Input sequence: -2, 3, 3, 2, 4, 3, 3, 3

Pair numbers, each containing 2 numbers as (2, 3) (3, 2) (4, 3) (3, 3). Only if an element is same as the element it is compared with, promoted to next level.



That is (2≠3), (3≠2) and (4≠3). So only 3 is promoted to next level so mode is 3.

2. Sequence: - 1, 1, 2, 1, 1, 2, 1, 1. The mode, 1 is arrived at as follows.



Note: If the n is an odd number or the number of elements in any stage is odd, we could qualify the promotion rule in 2 ways: We could

1. promote the remaining unpaired number after all the members are paired
2. Discard the remaining unpaired number.

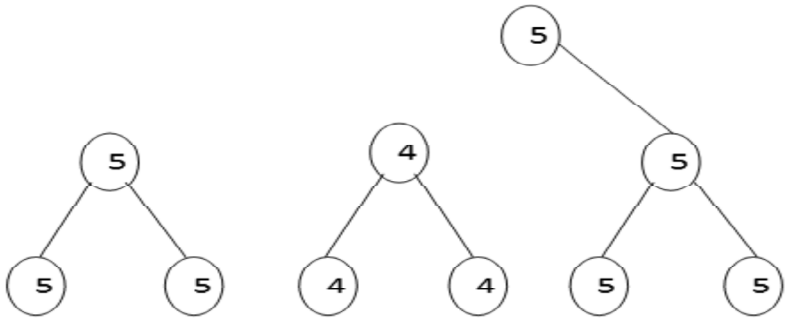
Observation: Either we will get same answer from both promotion methods or we get answer only from one method. It is seen that the two promotion rules above cannot produce two different answers for the same input sequence. So if we get an answer by using, rule 1 (say), there is no need to do the second method.

Example 1: Input sequence: 5, 5, 4, 4, 5, and 5

a. By promoting the remaining unpaired number

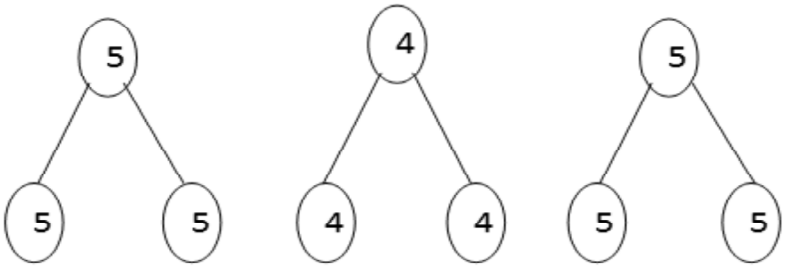
(5, 5) (4, 4) (5, 5)

Here 5, 4 and 5 is selected to next level (5, 4) (5)



(5≠4) and here 5 is the odd one and 5 is promoted to the next level.

b. Discarding the remaining unpaired number:



(5, 5) (4, 4) (5, 5). Here 5, 4 and 5 is selected to next level: (5, 4) (5).

(5≠4) and 5 the remaining number, is ignored. So here we will not get a result.

So from the two methods we get the 5 as the mode, which is indeed the correct answer.

Example 2:

Sequence: - 5, 5, 5, 5, 4, and 4

a. promoting the remaining unpaired number

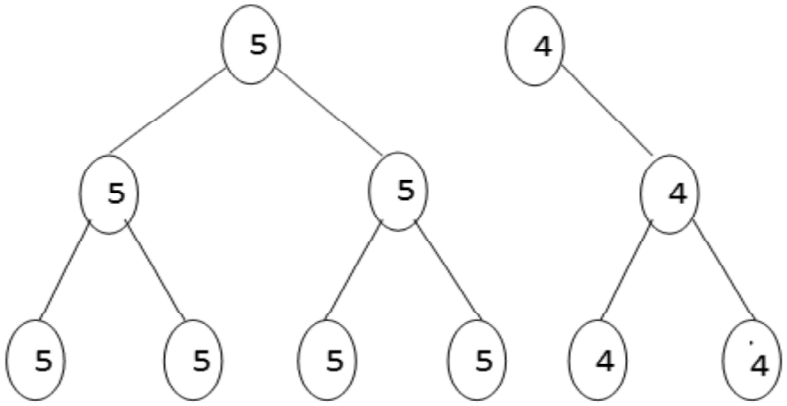
(5, 5) (5, 5) (4, 4)

Here 5, 5 and 4 is selected to next level

(5, 5) (4)

(5=5) and here 4 is the odd one and 5 and 4 is promoted to the next level.

(5, 4)



(5≠4). No result

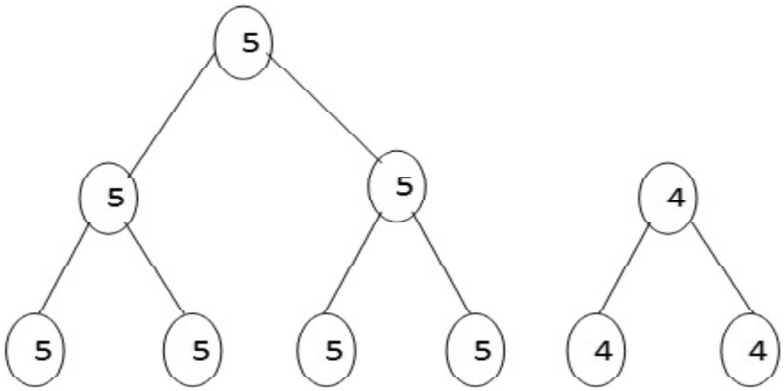
b. discarding the remaining unpaired number

(5, 5) (5, 5) (4, 4)

Here 5, 5 and 4 is selected to next level

(5, 5) (4)

(5=5) and here 4 is the odd one, is ignored.



So here we get 5 as the mode. So one of the two methods gives the correct mode

Example 3:-1, 1, 1, 1,5,5,4,4,1,1

a. By promoting the remaining unpaired number.

(1, 1)(1, 1) (5, 5)(4, 4)(1, 1)

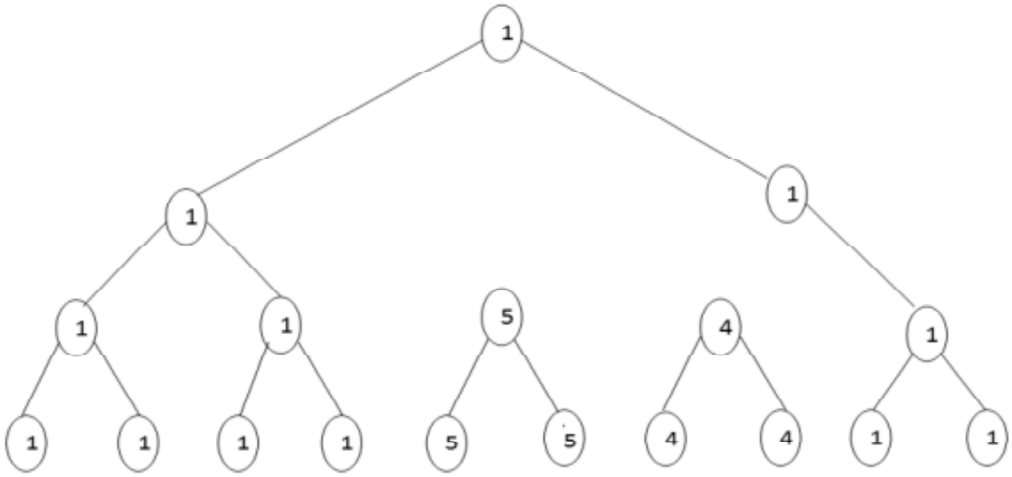
(1=1) (1=1)(5=5)(4=4) and (1=1)

So 1, 1,5,4,1 are promoted to the next stage.

(1, 1)(5, 4)(1) (1) is the odd one.

(1=1). 1 is promoted to next level and odd one (1) is also promoted to the next level.

Again (1=1)



Here 1 is the mode.

b. Discarding the remaining unpaired number

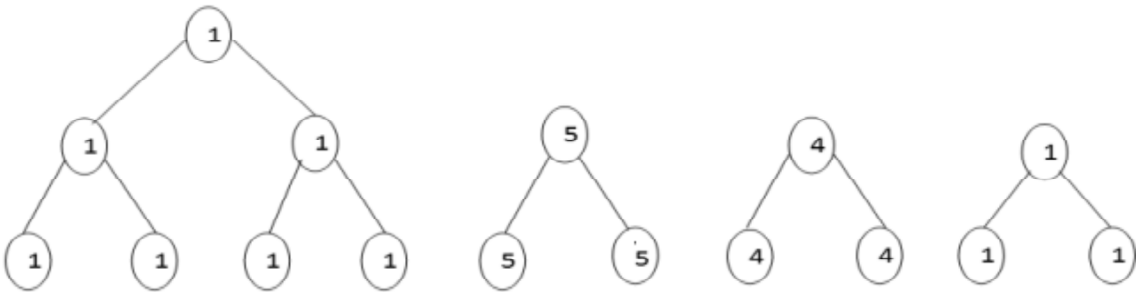
(1, 1)(1, 1) (5, 5)(4, 4) and (1, 1)

(1=1) (1=1)(5=5)(4=4)(1=1)

So 1, 1,5,4,1 are promoted to the next stage.

(1, 1)(5, 4)(1) (1) is the odd one. Discard it.

(1=1). 1 is promoted to next level and 1 is returned.



So both promotion rules give 1 as the mode of the sequence.

Remark: For finding the mode if its frequency is guaranteed to be > n/2, we thus have two O (n) algorithms – the one based on finding the median and the above tournament based method. However, we note the presence of large constant factors of the order of 20 in the complexity of the former method(see [1]); moreover, the tournament method will need n-1 comparisons only as an upper bound and in most cases, the number of comparisons will be even less. So, the above method is relatively more efficient.

Note: If the mode has a frequency less than n/2+1, this tournament based method, if applied directly, could give a wrong answer. Example: for the sequence with 8 elements {4, 4, 5, 6, 5, 9, 5, 3}, the mode is 5 with frequency 3 which is less than n/2+1. The tournament method gives the wrong answer 4.

Complexity: Only n comparisons are needed in the above method. So using tournament for minimum frequency of mode= (n/2) +1 has O (n) complexity.

Generalization to Lower Frequencies of Mode

We first consider the case where the guaranteed frequency of the mode is only (n/4) +1. We first apply the sieve (median and selection) method to find the median of the sequence [1].Then we can use this median as

the pivot element and partition the entire elements in the sequence into two. One partition will contain all the elements less than or equal to the pivot element and the other partition will contain all the elements greater than the pivot. Then we apply our modified tournament method to both these partitions to find modes of both as in the n/2+1 case. We then find and compare the frequencies of the median of the full array and the modes of both partitions. The number that has highest frequencies among these 3 numbers is the mode of the full array.

Note: It might be the case that the mode of only one of the array partitions has frequency at least n/4+1. The mode of the other partition could be less than this value. In that case, the tournament method might give the wrong value of the mode for that partition. But this causes no problems. Indeed, the frequency of this 'wrong mode' will be less than the frequency of the correct mode which will be found by the tournament from the other partition.

Example1,

The sequence contain 25 elements and it is guaranteed that the mode repeats at least (n/4) +1 time, Here n=25 and n/4=6.25 so the mode should be repeat at least (n/4) +1 =7 times.

Example: The elements are 2, 7, 8, 2, 9, 13, 2, 14, 15, 2, 4, 5, 6, 2, 16, 17, 2, 18, 1, 2, 3, 10, 2, 11, 12

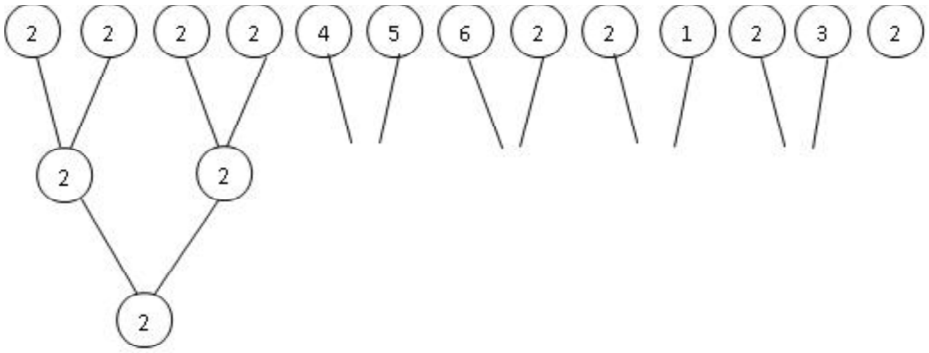
We have the guarantee that the mode repeats at least (n/4) +1 time, then we have to find the median of the sequence by using the median and selection method(sieve technique)[1]. We get 6 as the median of the sequence. Now using this median we partition the entire sequence into two. One partition will contain all the elements less than or equal to the median and the other will contain the elements greater than the median. So now each partition will contain at most n/2 elements. The partitions are,

Partition 1: elements less than or equal to 6 (2,2,2,2,4,5,6,2,2,1,2,3,2)

Partition 2: elements greater than 6(7,8,9,13,14,15,16,17,18,10,11,12)

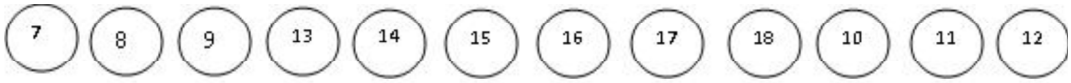
Now we can apply the tournament method directly to these partitions

Partition 1



Mode of this partition = 2.

Partition 2:

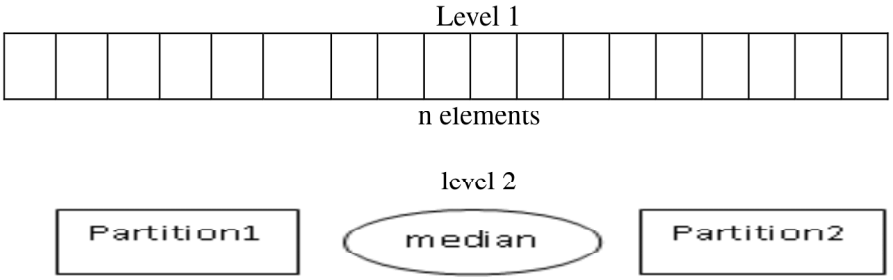


Here no matches found so none of these numbers are promoted to next level.

So now we have to take frequency of the median and the frequency of the result occurring from the tournament method. So here 6 is the median and the frequency is 1and the result occurring from the tournament is 2 and the frequency is 8. So 2 returned as the mode of the sequence.

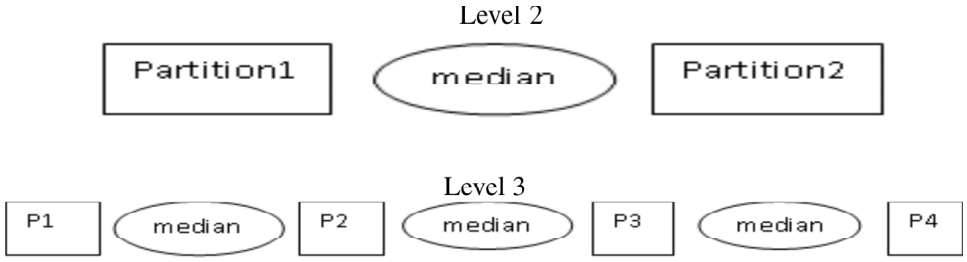
In cases where mode repeats less than (n/4) +1 time, we apply the partition with median step recursively. For example, if the min frequency is guaranteed to be n/8+1, one more level of recursion (a total of 3 levels) will yield the mode. The following figures fully illustrate this method pictorially.

If mode is guaranteed to have frequency at least (n/4) +1,



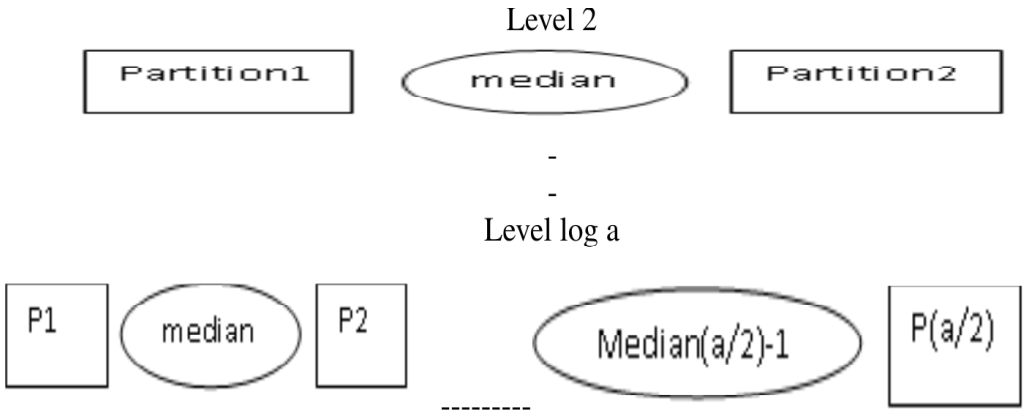
(n elements, 2 partition,1 median and 2 levels)

If mode is guaranteed to have frequency at least (n/8) +1, we progress as follows:



(3 levels, 4 partitions(each partition have at least n/4 elements),3medians)

If mode is guaranteed to have frequency at least n/a+1, we have:



(a/2 partitions each containing n/a elements and (a/2)-1 medians in (log a)th level)

Complexity Analysis: Finding the median of n numbers can be done by the sieve method with only O (n) complexity [1]. As we saw earlier, the tournament method when the frequency of the mode is guaranteed to be at least n/2+1 has only O (n) complexity. For the case where the min frequency is n/4+1, the algorithm has two levels and it is easy to see that each level has O(n) complexity, thus giving a total complexity again of only O(n). The general case of min frequency of mode given by n/a+1, has log a levels with O(n) complexity at each level, hence total complexity is O(n log a), a substantial improvement over O(n log n) if a is small compared to n.

CONCLUSION

We have thus formulated an algorithm to find the mode that improves upon O(n log n) complexity if there is some guarantee that the minimum frequency of the mode is at least n/a+1 where a is small. If the mode has very low frequency, it appears that the sorting based method (O (n log n)) will be difficult to improve upon.

REFERENCES

[1] David M Mount, “Design and Analysis of computer Algorithms”, Page Number: 103-108.

[2] https://en.wikipedia.org/wiki/Heapsorts