Lower Bound Theory

Lower Bound Theory Concept is based upon the calculation of minimum time that is required to execute an algorithm is known as a lower bound theory or Base Bound Theory.

Lower Bound Theory uses a number of methods/techniques to find out the lower bound.

Concept/Aim: The main aim is to calculate a minimum number of comparisons required to execute an algorithm.

The lower bound theory is the method that has been utilized to establish the given algorithm in the most efficient way which is possible.

This is done by discovering a function **g (n)** that is a lower bound on the time that any algorithm must take to solve the given problem.

Now if we have an algorithm whose computing time is the same order as **g(n)** , then we know that asymptotically we cannot do better.

If **f (n)** is the time for some algorithm, then we write **f (n) = Ω (g (n))** to mean that **g (n)** is the **lower bound of f (n)** .

This equation can be formally written, if there exists positive constants **c** and **n0** such that **|f (n)| >= c|g (n)|** for all **n > n0**.

In addition for developing lower bounds within the constant factor, we are more conscious of the fact to determine more exact bounds whenever this is possible.

Deriving good **lower bounds** is more challenging than arrange efficient algorithms. This happens because a

lower bound states a fact about all possible algorithms for solving a problem.

Generally, we cannot enumerate and analyze all these algorithms, so lower bound proofs are often hard to obtain.

Techniques:

The techniques which are used by lower Bound Theory are:

1. Comparisons Trees.
2. Oracle and adversary argument
3. State Space Method

1. Comparison trees:

In a comparison sort, we use only comparisons between elements to gain order information about an input sequence (a1; a2.....an).

Given a_i,a_j from (a_1, a_2.....a_n)We Perform One of the Comparisons

- $a_i < a_j$ less than
- $a_i \leq a_j$ less than or equal to
- $a_i > a_j$ greater than
- $a_i \geq a_j$ greater than or equal to
- $a_i = a_j$ equal to

To determine their relative order, if we assume all elements are distinct, then we just need to consider $a_i \leq a_j$ '=' is excluded &, ≥,≤,>,< are equivalent.

Consider sorting three numbers a1, a2, and a3. There are 3! = 6 possible combinations:

1. (a1, a2, a3), (a1, a3, a2),
2. (a2, a1, a3), (a2, a3, a1)
3. (a3, a1, a2), (a3, a2, a1)

The Comparison based algorithm defines a decision tree.

Decision Tree: A decision tree is a full binary tree that shows the comparisons between elements that are

executed by an appropriate sorting algorithm operating on an input of a given size. Control, data movement, and all other conditions of the algorithm are ignored.

In a decision tree, there will be an array of length n.

So, total leaves will be n! (I.e. total number of comparisons)

If tree height is h, then surely

$n! \leq 2^n$ (tree will be binary)

Taking an Example of comparing a1, a2, and a3.

Left subtree will be true condition i.e. $a_i \leq a_j$

Right subtree will be false condition i.e. $a_i > a_j$

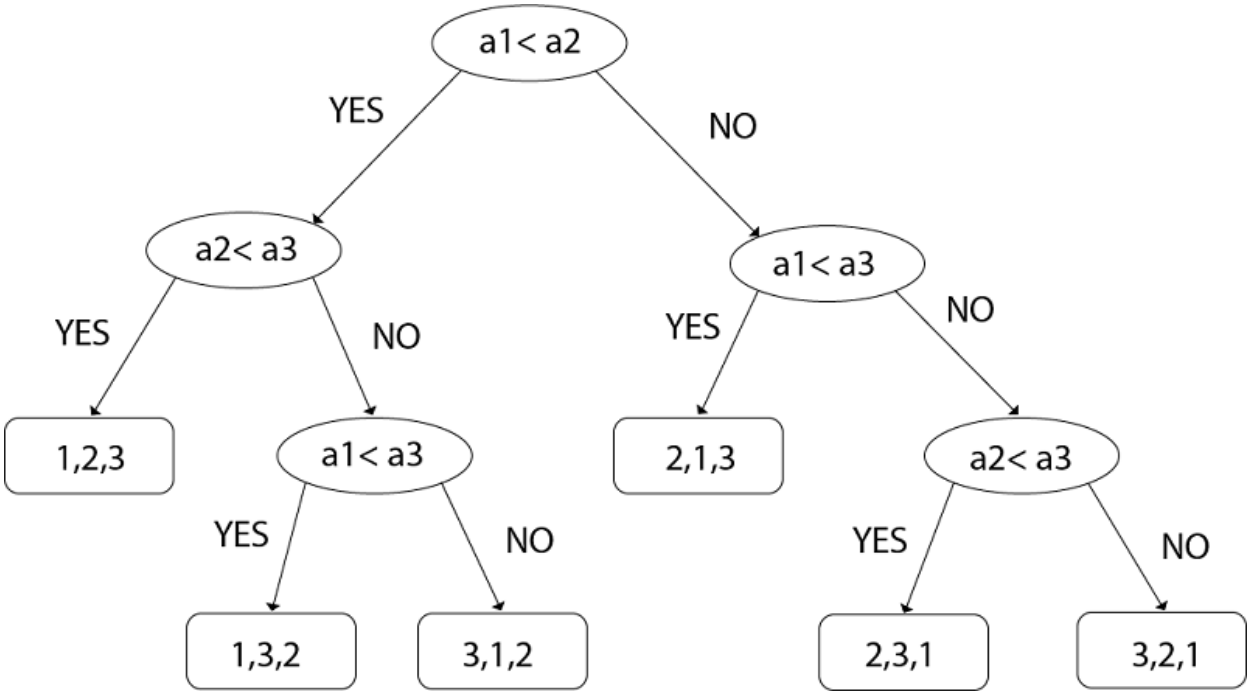


Fig: Decision Tree

So from above, we got

$n! \leq 2^n$

Comparison tree for Binary Search:

Example: Suppose we have a list of items according to the following Position:

1. 1,2,3,4,5,6,7,8,9,10,11,12,13,14

Mid = $\left[\left(\frac{1+14}{2}\right)\right] = \frac{15}{2} = 7.5 = \mathbf{7}$

Note: Choose the greatest integer

1, 2, 3, 4, 5, 6	8, 9, 10, 11, 12, 13, 14
Mid=$\left(\frac{1+6}{2}\right)$=**3**	Mid =$\left(\frac{8+14}{2}\right)$ = **11**

1, 2	4, 5, 6	8, 9, 10	12, 13, 14
Mid=$\left(\frac{1+2}{2}\right)$=**1**	Mid=$\left(\frac{4+6}{2}\right)$=**5**	Mid= $\left(\frac{8+10}{2}\right)$=**9**	Mid=$\left(\frac{12+14}{2}\right)$=**13**

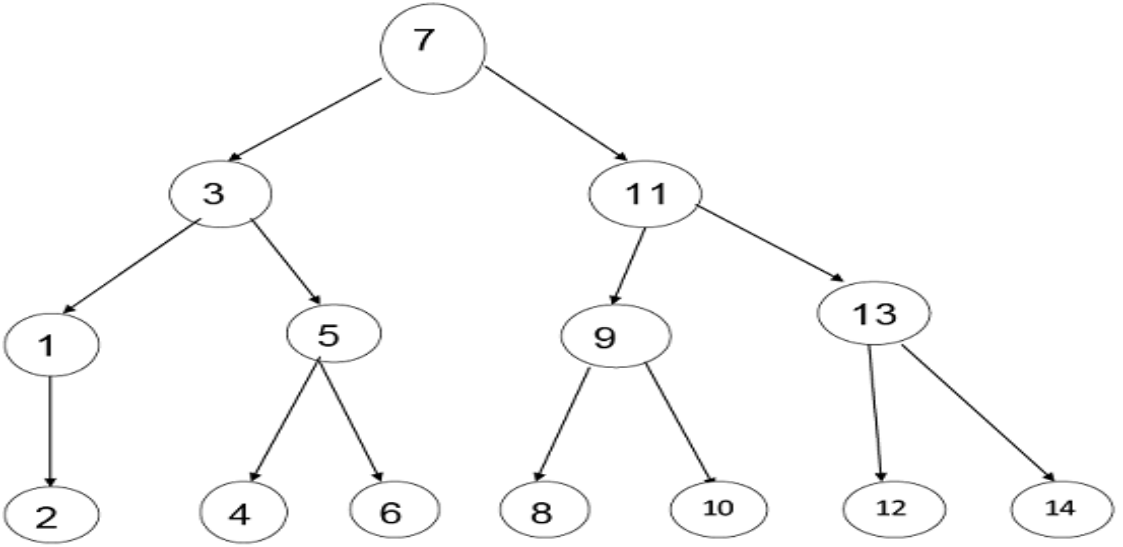
And the last midpoint is:

2, 4, 6, 8, 10, 12, 14

Thus, we will consider all the midpoints and we will make a tree of it by having stepwise midpoints.

The Bold letters are Mid-Points Here

According to Mid-Point, the tree will be:



Step1: Maximum number of nodes up to k level of the internal node is 2^k-1

For Example

2^k-1

2^3-1= 8-1=7

Where k = level=3

Step2: Maximum number of internal nodes in the comparisons tree is n!

Note: Here Internal Nodes are Leaves.