بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

In the name of Allah, Most Gracious, Most Merciful

CSE 4303
Data Structure

Topic:Linear Search, Binary Search, Tree Terminologies

Asaduzzaman Herok
Lecturer | CSE | IUT
asaduzzaman34@iut-dhaka.edu

Linear Search

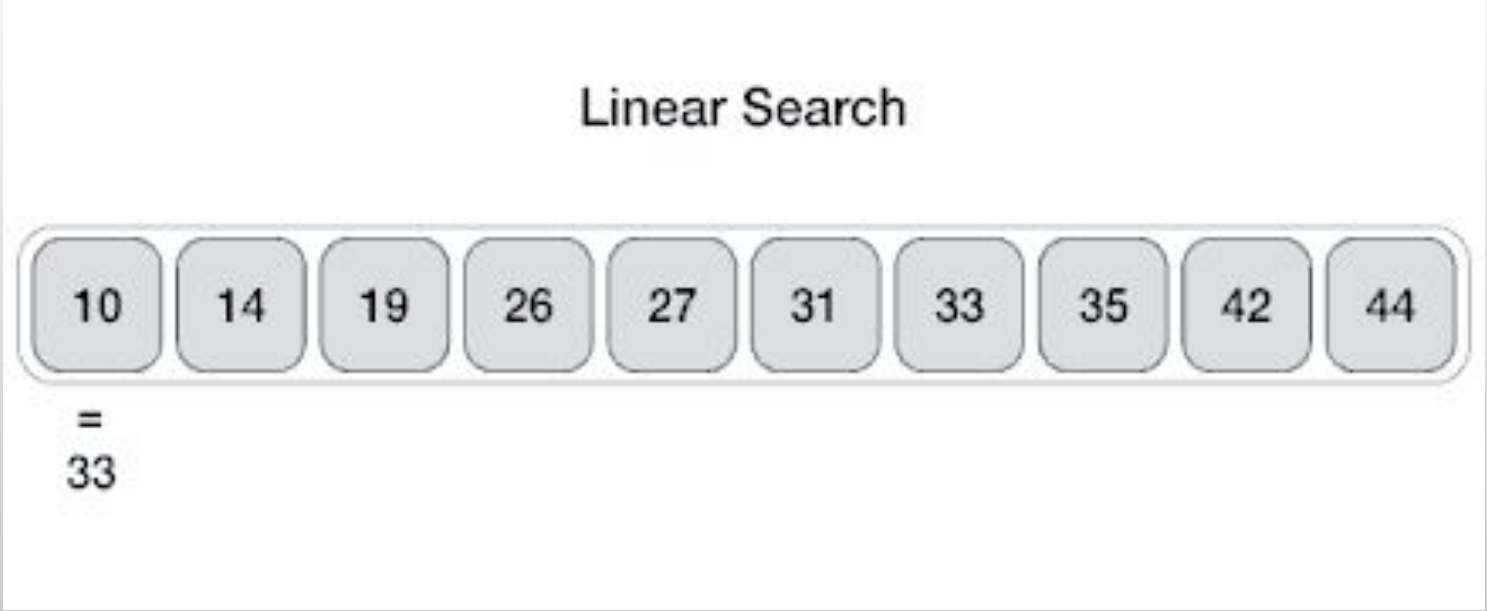
Also called the sequential search

Starting at the first element, this algorithm sequentially steps through an array examining each element until it locates the value it is searching for

```
bool linear_search( Type const &obj, Type *array, int a, int b )
{
        for ( int i = a; i <= b; ++i ) {
                if ( array[i] == obj ) {
                        return true;
                }
        }
        return false;
}
```

```
LINEAR_SEARCH(A, N, VAL)
Step 1: [INITIALIZE] SET POS = -1
Step 2: [INITIALIZE] SET I = 1
Step 3:     Repeat Step 4 while I<=N
Step 4:             IF A[I] = VAL
                            SET POS = I
                            PRINT POS
                            Go to Step 6
                    [END OF IF]
                     SET I = I + 1
            [END OF LOOP]
Step 5: IF POS = -1
        PRINT "VALUE IS NOT PRESENT
        IN THE ARRAY"
        [END OF IF]
Step 6: EXIT
```

Linear Search



Source: Tutorials Point

Complexity of Linear Search Algorithm

Okay, great, now we know how to search.

But how long will our search take?

That's really three separate questions:

- ➔ How long will it take in the best case?
- ➔ How long will it take in the worst case?
- ➔ How long will it take in the average case?

- ➢ Best Case: The target value is in the first element of the array.
  - O(1) constant time.
- ➢ Worst Case: The target value is in the last element of the array.
  - O(n)
- ➢ Average Case: Since the target value can be anywhere in the array, any element of the array is equally likely.
  - So on average, the target value will be in the middle of the array.
  - O(n/2) is O(n)

$$\frac{1+2+\ldots+n}{n} \longrightarrow \frac{1+2+\ldots+n}{n} = \frac{1}{n}\frac{n(n+1)}{2} = \frac{n+1}{2}$$

Linear Search - Tradeoffs

Benefits:

- ➔ Easy algorithm to understand
- ➔ Array can be in any order

Disadvantages:

- ➔ Inefficient (slow): for array of N elements, examines N/2 elements on average for value in array

Why Do We Care About Search Time?

We know that time is money

Binary search

A binary search looks for an item in a list using a divide-and-conquer strategy

Requires array elements to be in order

1) Divides the array into three sections:
   a) middle element
   b) elements on one side of the middle element
   c) elements on the other side of the middle element
2) If the middle element is the correct value, done. Otherwise, go to step 1. using only the half of the array that may contain the correct value.
3) Continue steps 1. and 2. until either the value is found or there are no more elements to examine

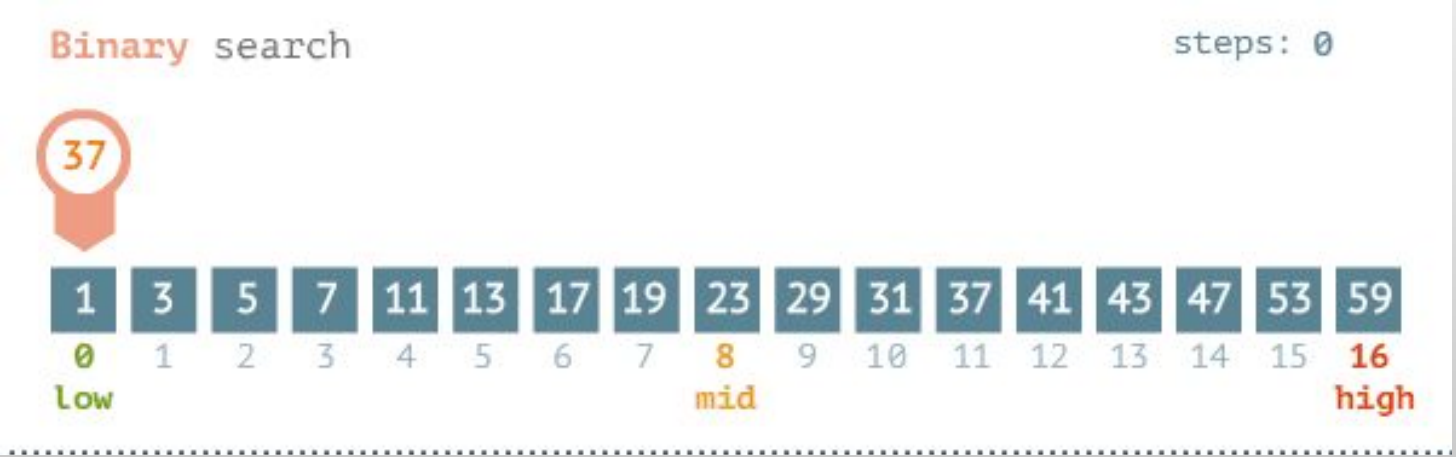
Binary search

```
int binarySearch(int array[], int size, int value)
{
   int first = 0,              // First array element
       last = size - 1,        // Last array element
       middle,                 // Mid point of search
       position = -1;          // Position of search value
   bool found = false;         // Flag

   while (!found && first <= last)
   {
      middle = (first + last) / 2;     // Calculate mid point
      if (array[middle] == value)      // If value is found at mid
      {
         found = true;
         position = middle;
      }
      else if (array[middle] > value)  // If value is in lower half
         last = middle - 1;
      else
         first = middle + 1;           // If value is in upper half
   }
   return position;
}
```

```
BINARY_SEARCH(A, lower_bound, upper_bound, VAL)
Step 1: [INITIALIZE] SET BEG = lower_bound
        END = upper_bound, POS = - 1
Step 2: Repeat Steps 3 and 4 while BEG <= END
Step 3:          SET MID = (BEG + END)/2
Step 4:          IF A[MID] = VAL
                       SET POS = MID
                       PRINT POS
                       Go to Step 6
                 ELSE IF A[MID] > VAL
                       SET END = MID - 1
                 ELSE
                       SET BEG = MID + 1
                 [END OF IF]
        [END OF LOOP]
Step 5: IF POS = -1
            PRINT "VALUE IS NOT PRESENT IN THE ARRAY"
        [END OF IF]
Step 6: EXIT
```

Binary search



Binary Search - Tradeoffs

Benefits:

Much more efficient than linear search. For array of N elements, performs at most log2N comparisons

Disadvantages:

Requires that array elements be sorted

Question:

Should a binary search be called on a very small list?

Run times of searching algorithms

The following table summarizes the run times:

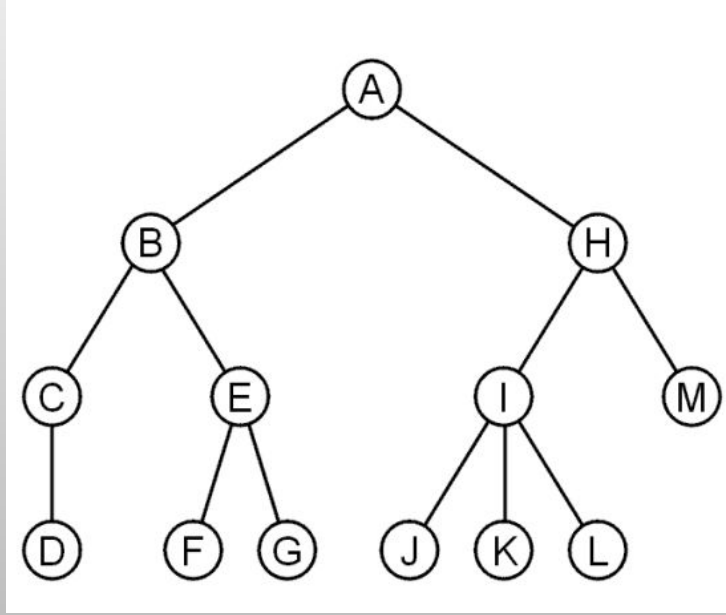
Algorithm	Best Case	Average Case	Worst Case
Linear Search	Ω(1)	O(n)	O(n)
Binary Search	Ω(1)	O(ln(n))	O(ln(n))

Tree

A tree is recursively defined as a set of one or more nodes where one node is designated as the root of the tree and all the remaining nodes can be partitioned into non-empty sets each of which is a sub-tree of the root.

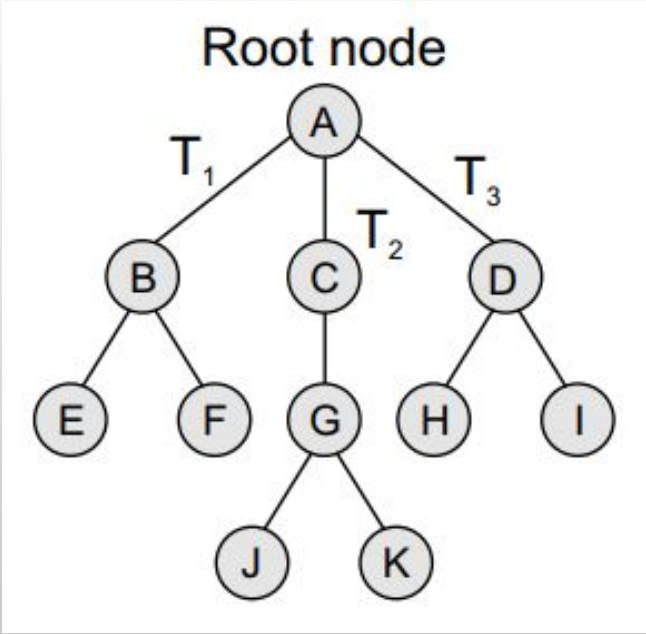
Similar to linked lists:

- ➔ There is a first node, or root
- ➔ Each node has variable number of references to successors
- ➔ Each node, other than the root, has exactly one node pointing to it



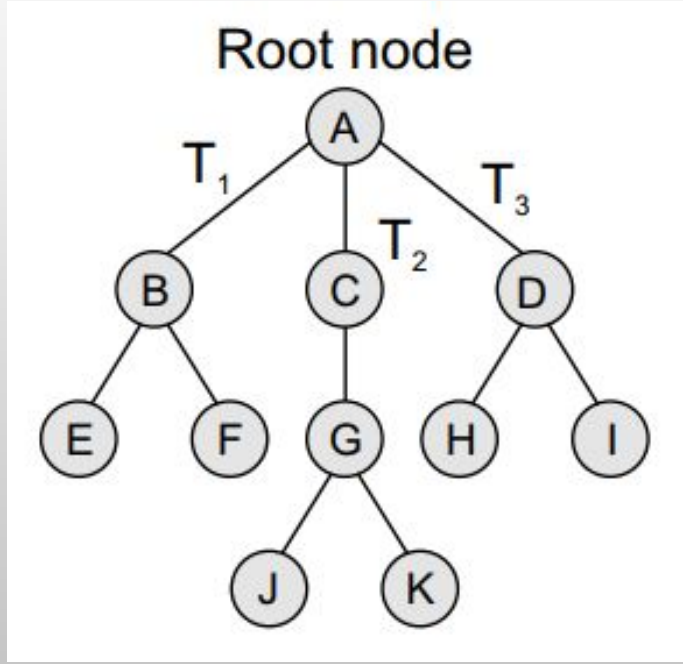
Basic Tree Terminology

- ★ ***Root node:*** The root node R is the topmost node in the tree. If R = NULL, then it means the tree is empty.
- ★ ***Sub-trees:*** If the root node R is not NULL, then the trees T1, T2, and T3 are called the sub-trees of R.
- ★ ***Leaf node:*** A node that has no children is called the leaf node or the terminal node. E, F, J, K, H, I.
- ★ ***Internal nodes:*** All other nodes apart from the leaf nodes are called in internal nodes. A, B, C, D, G.
- ★ ***Path:*** A sequence of consecutive edges is called a path. For example, in the left figure the path from the root node A to node I is given as: A, D, and I. Length is the number of nodes on the path. Path (A, D, I) has length 3.
- ★ ***Ancestor node:*** An ancestor of a node is any predecessor node on the path from root to that node. The root node does not have any ancestors. For example, A, D are the ancestors of node I.



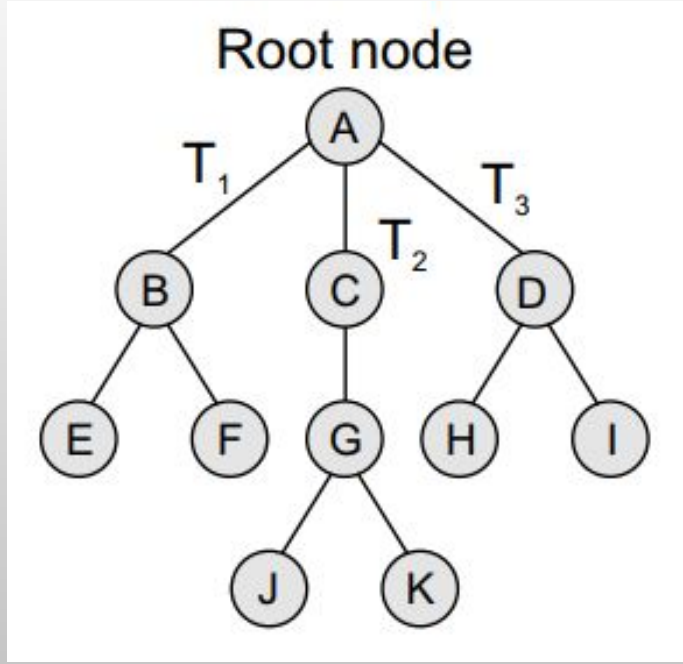
Basic Tree Terminology

- ★ ***Descendant node:*** A descendant node is any successor node on any path from the node to a leaf node. Leaf nodes do not have any descendants. G, J, and K are the descendants of node C.
- ★ ***Level number:*** Every node in the tree is assigned a level number in such a way that the root node is at level 0, children of the root node are at level number 1. Every node is at one level higher than its parent. all child nodes have a level number given by parent's level number + 1.
- ★ ***Degree of a node:*** Is the number of edges that the node is connected to.
- ★ **In-degree:** (Directed graph/tree) the number of incoming connection/edges to a node.
- ★ ***Out-degree:*** (Directed graph/tree) the number of connection/edges leaving that node.



Basic Tree Terminology

- ★ ***Sibling:*** All nodes that are at the same level and share the same parent are called siblings (brothers). E, F are siblings. B, C, D are siblings.
- ★ ***Edge:*** It is the line that connects a node to another node. For any tree (not forest) with n nodes will have has exactly n – 1 edges because every node except the root node is connected to its parent via an edge.
- ★ ***Height of a tree:*** It is the total number of nodes on the path from the root node to the deepest node in the tree. On the left figure the height is 4. A tree with only a root node has a height of 1.



Types of Trees

The followings are some types of trees:

1. General trees
2. Forests
3. Binary trees
4. Binary search trees
5. Expression trees
6. Tournament trees

Acknowledgements

Rafsanjany Kushol
PhD Student, Dept. of Computing Science,
University of Alberta

Sabbir Ahmed
Assistant Professor
Department of CSE, IUT