Binary Search - Recursive

```
else if (search < array[mid])
              index = recurs ive Bin ary (array,
first, mid - 1, search);
else
              index = recurs ive Bin ary (array,
mid + 1, last, search);
```

Binary Search - Iterative(In order)

```
int start = 0;
int end = array.l ength - 1;
while( start <= end) {
 int mid = start + ((end - start) / 2);
 if ( target == array[ mid]) {
            return s;
- - - - Iterative - - - - - - - - - - - - Recursive
- - - - - - - - - -
 } else if (target < array[ mid]) {
               end = mid - 1; | method (array,
start, MID - 1, target)
 } else {
                  start = mid + 1; | method -
(array, MID + 1, end, target)
             }
}
```

Count word occurence

```
{
    String text = "Turn right then left, then
right again;

    Map <St ring, Intege r> map = new HashMa p<>();
    String[ ] words = text.s pli t(" ");
    for(int i = 0; i < words.l en gth(); i++) {
        String key = words[i];
       if (map.c ont ain sKe y(key))
                map.pu t(key, 1);
       else {
               int value = map.ge t(key);
                val ue++;
                map.pu t(key, value);
       }
```

Count word occurence (cont)

```
map.fo rEa ch(k, v) -> sysout(k + " " + v);
```

1. set text in string
2. create treemap to hold words as key & value (k, v)
3. create string array to split text
4. for loop to go through word array and add into key string
5. if map contains key(the word) then put (key, 1) into map
6. go onto next word, if word is already added to map, then increase value >> (key, value + 1)
7. continue through text
8. display for each map (key + final value) in alpha order

ans:
(AGAIN, 1)
(LEFT, 1)
(RIGHT, 2)
(THEN, 2)
(TURN, 1)

DFS + BFS

DFS (O(H))	O(E+V)
BFS (O(L))	O(E+V)

V & E- # Vertices & Edges
H - max height of tree
L - max # nodes in a single level