**Binary Tree Traversals (Depth-First and Breadth-First):**

```
class Node {
  constructor(val) {
    this.val = val;
    this.left = null;
    this.right = null;
  }
}

const a = new Node("a");
const b = new Node("b");
const c = new Node("c");
const d = new Node("d");
const e = new Node("e");
const f = new Node("f");

a.left = b;
a.right = c;
b.left = d;
b.right = e;
c.right = f;

//Depth First Traversal
//TC: O(n) SC: O(n)

const depthFirstTraversalStack = (root) => {
  //Based on Stack Implementation - Push Right Push Left
  if (!root) return [];
  let stack = [root];
  let result = [];
  while (stack.length > 0) {
    let current = stack.pop();
    result.push(current.val);
    if (current.right) {
      stack.push(current.right);
    }
    if (current.left) {
      stack.push(current.left);
    }
  }
  return result;
};

console.log(depthFirstTraversalStack(a));



const depthFirstTraversalRecurssive = (root) => {
    //Based on Recurssive Implementation - Push Right Push Left
    if (!root) return [];
    let stack = [root];
    return [root.val, ...depthFirstTraversalRecurssive(root.left),
...depthFirstTraversalRecurssive(root.right)]
  };

  console.log(depthFirstTraversalRecurssive(a));

  const breadthFirstTraversal = (root) => {
    //Based on Stack Implementation - Push Right Push Left
    if (!root) return [];
    let stack = [root];
    let result = [];
    while (stack.length > 0) {
      let current = stack.shift();
      result.push(current.val);
      if (current.left) {
        stack.push(current.left);
      }
      if (current.right) {
        stack.push(current.right);
      }

    }
    return result;
  };

  console.log(breadthFirstTraversal(a));
```

**Counting Pairs with given sum in Array:**

```
class Solution {
  getPairsCount(arr, k) {
    //O(n^2)
    let count = 0;
    for (let i = 0; i < arr.length; i++) {
      for (let j = i; j < arr.length; j++) {
        if (arr[i] + arr[j] === k) {
          count++;
        }
      }
    }
    return count;
  }
}

const pairSum = new Solution();
console.log(
  "<======>",
  pairSum.getPairsCount([5, 0, -10, 1, 2, 4, 0, 15, -6], 6)
);

function getCountPairs(arr, k) {
  let m = new Map();
  let count = 0;
  for (let i = 0; i < arr.length; i++) {
    if (m.has(k - arr[i])) {
      count += m.get(k - arr[i]);
    }
    m.set(arr[i], (m.get(arr[i]) || 0) + 1);
  }
  return count;
}

console.log(
  "getCountPairs<======>",
  getCountPairs([5, 0, -10, 1, 2, 4, 0, 15, -6], 6)
);
console.log("getCountPairs<======>", getCountPairs([1, 1, 1, 1], 2));
```