

Arrays:Subarrays

A subarray is a contiguous portion of an array.

Concept:

```
function getAllSubarrays(arr) {  
    const subarrays = [];  
    const length = arr.length;  
  
    for (let start = 0; start < length; start++) {  
        for (let end = start + 1; end <= length; end++) {  
            subarrays.push(arr.slice(start, end));  
        }  
    }  
  
    return subarrays;  
}
```

// Example usage:

```
const array = [1, 2, 3];  
const result = getAllSubarrays(array);  
console.log(result);
```

1. Max Consecutive Ones:

Level:Easy

Problem Link: <https://leetcode.com/problems/max-consecutive-ones/description/>

Solution1:'

Code:

```
var findMaxConsecutiveOnes = function(nums) {  
    let maxCount=0;  
    let count=0;  
    for (var i=0;i<nums.length;i++) {  
        if (nums[i]==1) {  
            count++;  
        }  
        else if (nums[i]==0) {  
            count=0;  
        }  
        maxCount=Math.max(maxCount, count);  
    }  
    return maxCount;  
};
```

Time Complexity:O(n)

Space Complexity: $O(1)$

2. Max Sum Subarray of Size K

Level: medium

Problem Link: <https://www.geeksforgeeks.org/problems/max-sum-subarray-of-size-k5313/1>

Solution1:

Code:

```
class Solution {
    maximumSumSubarray(K, Arr, N) {
        let i=0;
        let j=0;
        let maxSum=0;
        let sum=0;
        while(j<N){
            sum+=Arr[j];
            if(j-i+1==K){
                maxSum=Math.max(maxSum,sum);
                sum-=Arr[i];
                i++;
            }
            j++;
        }
        return maxSum;
    }
}
```

Time Complexity: $O(n)$

Space Complexity: $O(1)$

3. First Negative in Every Window of size k:

Level: Medium

Problem Link:

<https://www.geeksforgeeks.org/problems/first-negative-integer-in-every-window-of-size-k3345/1>

Solution1:

Code:

```
printFirstNegativeInteger(N, K, Arr) {
    let i=0;
    let j=0;
    var array=[];
    var ansArray=[]
    while(j<N){
        if(Arr[j]<0){
            array.push(Arr[j]);
        }
        if(j-i+1==K){
            if(array.length===0){
                ansArray.push(0);
            }
        }
    }
}
```

```

    }
    else{
        ansArray.push(array[0]);
        if(Arr[i]==array[0]){
            array.shift();
        }
    }
    i++;
}
j++;
}
return ansArray;
}
}

```

Time Complexity: $O(n)$

Space Complexity: (k) for storing all negatives in window/subarray of size k

4. Sum of all subarrays of size k

<https://www.geeksforgeeks.org/sum-of-all-subarrays-of-size-k/>