function longestsubstring(s){

let start = 0;

let end = 0;

let maxlength = 0;

const sets = new Set();

while(end < s.length){

if(!sets.has(s[end])){

sets.add(s[end]);

end++

maxlength = Math.max(maxlength,sets.size)

}else{

sets.delete(s[start]);

start++

}

}

return maxlength

}

console.log(longestsubstring('abcbacbb'))

Largest Subarray with sum k

function longSubArray(arr,n,k){

let sum = 0;

let length =0

let hashObj = {}

for(let i=0;i<n;i++){

sum= sum+arr[i];

if(sum === k){

length = Math.max(length,i+1)

}

if(!hashObj[sum]){

hashObj[sum]=i

}

let rem = sum-k;

//console.log(hashObj[rem] +'+'+ i)

if(hashObj[rem] !== undefined){

length = Math.max(i-hashObj[rem],length)

}

}

return length

}

console.log(longSubArray([10,5,2,6,1,1],6,15)) //5

MAX SUB ARRAY:

**Naïve approach O(n^2)**

let a = [2, 4, 6, 1, 7, 8, 4];

let max = 0;

const k =3;

for(let i=0;i<=a.length-3;i++){

let arr = [];

let sum = 0;

for(let j=i;j<i+k;j++){

sum += a[j]

arr.push(a[j])

}

max = Math.max(sum,max);

console.log(arr);

}

console.log(max);

**SLIDING WINDOW APPROACH:O(n) all SubArrays**

const a = [1, 2, 3, 4, 5];

const k = 3;

let subarr = [];

let result = []

for(let i=0;i<k;i++){

subarr.push(a[i])

}

result.push(subarr)

console.log(subarr);

for(let i=k;i<a.length;i++){

subarr.shift();

subarr.push(a[i])

result.push(subarr)

console.log(subarr);

}

**SLIDING WINDOW APPROACH:O(n) with max sum and subarray**

const a = [1, 2, 3, 4, -6];

const k = 3;

let subarr = [];

let result = []

let maxSum = 0;

for(let i=0;i<k;i++){

subarr.push(a[i]);

maxSum += a[i]

}

let windowSum = maxSum ;

result.push(subarr)

console.log(subarr);

for(let i=k;i<a.length;i++){

windowSum = windowSum-a[i-k]+a[i];

maxSum = Math.max(maxSum,windowSum)

subarr.shift();

subarr.push(a[i])

result.push(subarr)

console.log(subarr);

}

console.log(maxSum )

**SLIDING WINDOW APPROACH MAX Sum Subarray**

const a = [1, 2, 3, 4, -6];

let k=3;

let maxSum = 0;

for(let i=0;i<k;i++){

maxSum += a[i]

}

let wSum = maxSum

for(let i=k;i<a.length;i++){

wSum = wSum-a[i-k]+a[i];

maxSum = Math.max(maxSum,wSum);

}

console.log(maxSum); //9

**kadane’s approach for max sum subarray:**

let a = [5,6,-3,7,-13,8,-2,5,-6,7,-11,3,10,-10,-6,-10,7,2]  
  
let max = a[0];  
let sum = a[0];  
  
for(let i=1;i<a.length;i++){  
 if(sum >=0){  
 sum = sum+a[i]  
 }else{  
 sum = a[i]  
 }  
 // if(sum> max){  
 // max = sum;  
 // }

max = Math.max(max,sum)  
   
}  
  
console.log(max) //16

**Rotate an array:**

let a = [1,2,3,4,5];

let k = 3;

function rotate(arr,s,e){

let temp;

while(s<e){

// temp = arr[s];

// arr[s] = arr[e];

//arr[e] = temp;

[a[end],a[start]]=[a[start],a[end]]

s++;

e--;

}

return arr

}

if(k<0){

k = k+a.length

}else{

k = k%a.length

}

console.log(rotate(a,0,k-1))

console.log(rotate(a,k,a.length-1))

console.log(rotate(a,0,a.length-1))

// 51234 -1

// 12345 0

// 23451 1

// 34512 2

// 45123 3

// 51234 4

**COMPRESSED STRING:**

let str = "aaabbca";

//op:a3b2c1

let obj ={};

for(let i=0;i<str.length;i++){

if(obj.hasOwnProperty(str[i])){

obj[str[i]] = obj[str[i]]+1

}else{

obj[str[i]] = 1

}

}

let compressed = '';

Object.keys(obj).forEach((e,i)=>{

compressed += e+obj[e]

})

console.log(compressed);;

**Group Anagrams**

let arr = ["eat", "tea", "tan", "ate", "nat", "bat"];

//[["eat", "tea", "ate"], ["tan", "nat"], ["bat"]]

let obj = {}

arr.forEach((ele,i)=>{

let sortedWord = ele.split('').sort().join('');

if(obj.hasOwnProperty(sortedWord)){

obj[sortedWord].push(ele)

}else{

obj[sortedWord] = [ele]

}

})

console.log(Object.values(obj));

function countSubstring(str,substr){

let count = 0;

let position = 0;

// console.log(str.indexOf(substr,position))

while((position = str.indexOf(substr,position)) !== -1){

console.log(position)

count++;

position += substr.length;

}

return count;

}

console.log(countSubstring('geeksforgeeksgeeks','geeks'));

function dynamicSortByNested(arr, property, order = "asc") {

return arr.sort((a, b) => {

const dir = order === "desc" ? -1 : 1;

// Access nested properties dynamically

const aValue = property.split('.').reduce((obj, key) => obj?.[key], a);

const bValue = property.split('.').reduce((obj, key) => obj?.[key], b);

if (aValue < bValue) return -1 \* dir;

if (aValue > bValue) return 1 \* dir;

return 0;

});

}

const nestedArray = [

{ name: "Alice", details: { age: 25 } },

{ name: "Bob", details: { age: 30 } },

{ name: "Charlie", details: { age: 20 } }

];

const sortedNestedArray = dynamicSortByNested(nestedArray, "details.age", "asc");

console.log(sortedNestedArray);