AXSOS ACADEMY

Problem-Solving Patterns

Sliding window

Smallest Subarray with a given sum

Outline

- Introduce the topic to the academy team including Idea, Problem statement, and solution. **(15 Minutes)**
- Practice a challenge with the team. **(15 Minutes)**
- Take feedback from the team and update later the slides and confluence accordingly. **(10 Minutes)**
- Team to evaluate the session. **(5 Minutes)**
- **Total time: 45 Minutes**

What is sliding window pattern :

Sliding Window Technique:

- is a computational technique that aims to reduce the use of nested loops and replace it with a single loop, thereby reducing the time complexity.

- These problems are painless to solve using a **brute force approach in $O(n^2)$ or $O(n^3)$.** However, the Sliding window technique can **reduce the time complexity to $O(n)$.**

Where we use the Sliding Window:

How to Know, Where we use the Sliding Window?

To know, Where we use the Sliding Window then we remember the following terms which is mentioned below:

- Array
- String
- Sub Array
- Sub String
- Largest Sum
- Maximum Sum
- **Minimum Sum(smallest subarray with given sum)**

Explain the solution:

Problem statement:

Given an array of positive numbers and a positive number ‘S’, find the length of the smallest contiguous subarray whose sum is greater than or equal to ‘S’. Return 0, if no such subarray exists.

Example:

Input: [8, 5, 3, 2, 5, 7], S=10
Output: 2
Explanation: The smallest subarray with a sum great than or equal to ‘10' is [8,5].

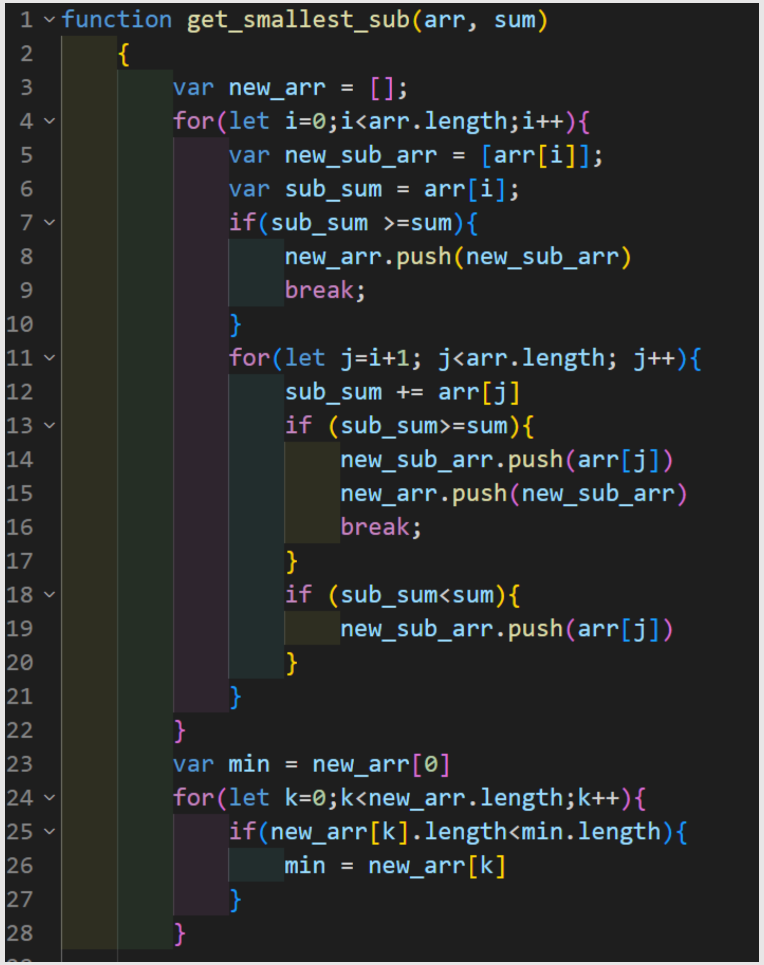
Brute force(naive solution):

Naive approach:

- A simple solution is to use two nested loops.
- The outer loop picks a starting element
- the inner loop considers all elements (on right side of current start) as ending element.
- Whenever sum of elements between current start and end becomes more than the given number, update the result
- if current length is smaller than the smallest length so far.

Brute force(naive solution):

Code implementation:



```
function get_smallest_sub(arr, sum)
    {
        var new_arr = [];
        for(let i=0;i<arr.length;i++){
            var new_sub_arr = [arr[i]];
            var sub_sum = arr[i];
            if(sub_sum >=sum){
                new_arr.push(new_sub_arr)
                break;
            }
            for(let j=i+1; j<arr.length; j++){
                sub_sum += arr[j]
                if (sub_sum>=sum){
                    new_sub_arr.push(arr[j])
                    new_arr.push(new_sub_arr)
                    break;
                }
                if (sub_sum<sum){
                    new_sub_arr.push(arr[j])
                }
            }
        }
        var min = new_arr[0]
        for(let k=0;k<new_arr.length;k++){
            if(new_arr[k].length<min.length){
                min = new_arr[k]
            }
        }
```

Brute force(naive solution):

Time complexity: O(N^2)

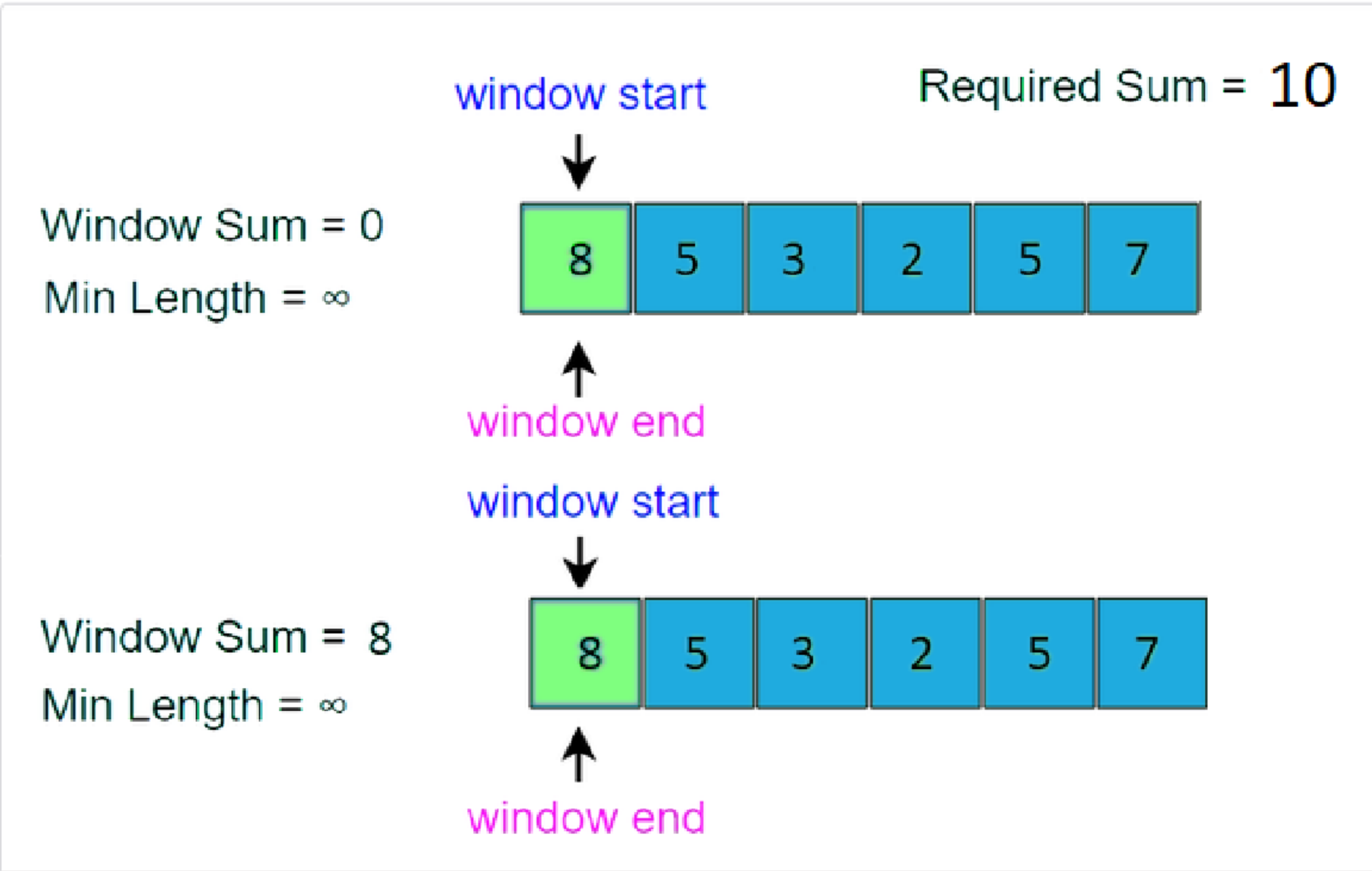
Space complexity: O(1)

Solution:

Here is how we will solve this problem:

- Consider each subarray as a sliding window.
- Start with a sliding window of size 1 (windowStart=0, windowEnd=0).
- Iterate over the array and add elements to the window until we got our condition.
- When reach sum >= K. Remember the length of this window as the smallest window so far.
- Check if the current window length is the smallest so far. If yes, then update the minimum length.
- Iterate to next (windowStart=1, windowEnd=1).
- Repeat last four steps till the end of array

Solution:



Required Sum = 10

window start

Window Sum = 0

Min Length = ∞

8	5	3	2	5	7

window end

window start

Window Sum = 8

Min Length = ∞

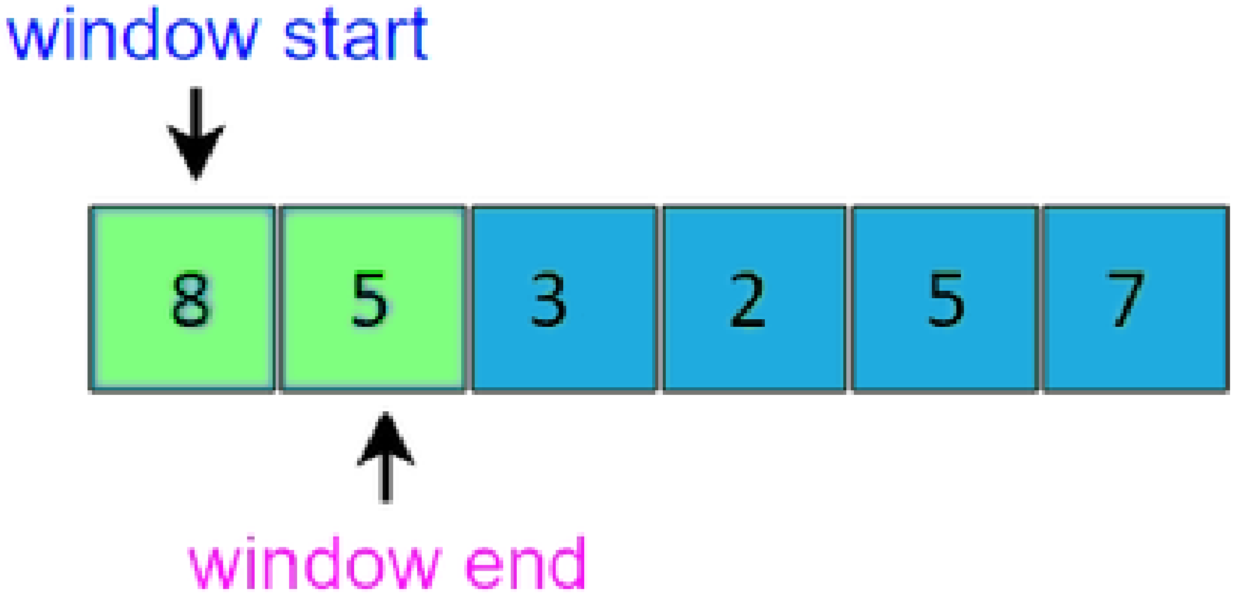
8	5	3	2	5	7

window end

Solution:

Window Sum = 13

Min Length = 2



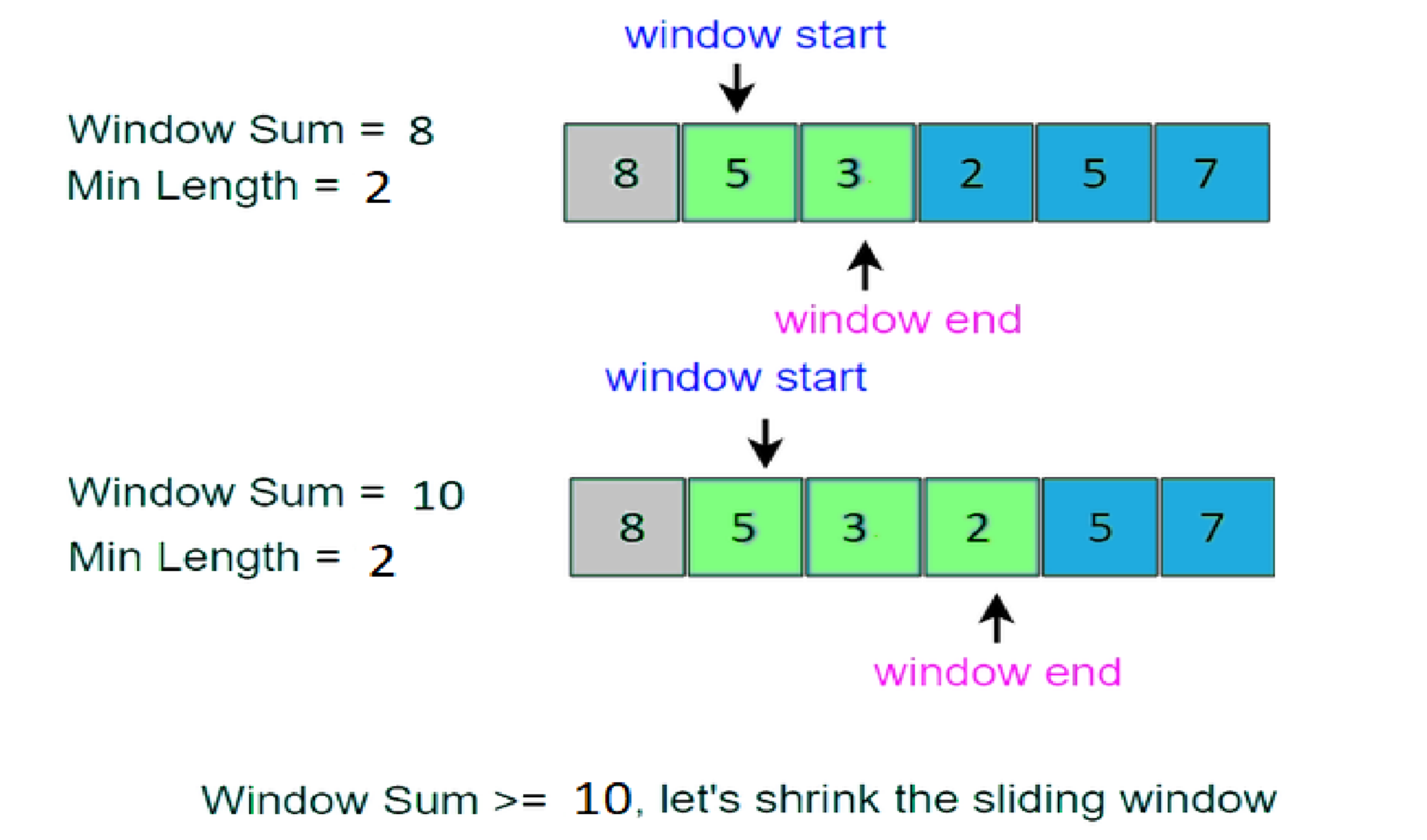
window start

8	5	3	2	5	7

window end

Window Sum >= 10, let's shrink the sliding window

Solution:



Window Sum = 8

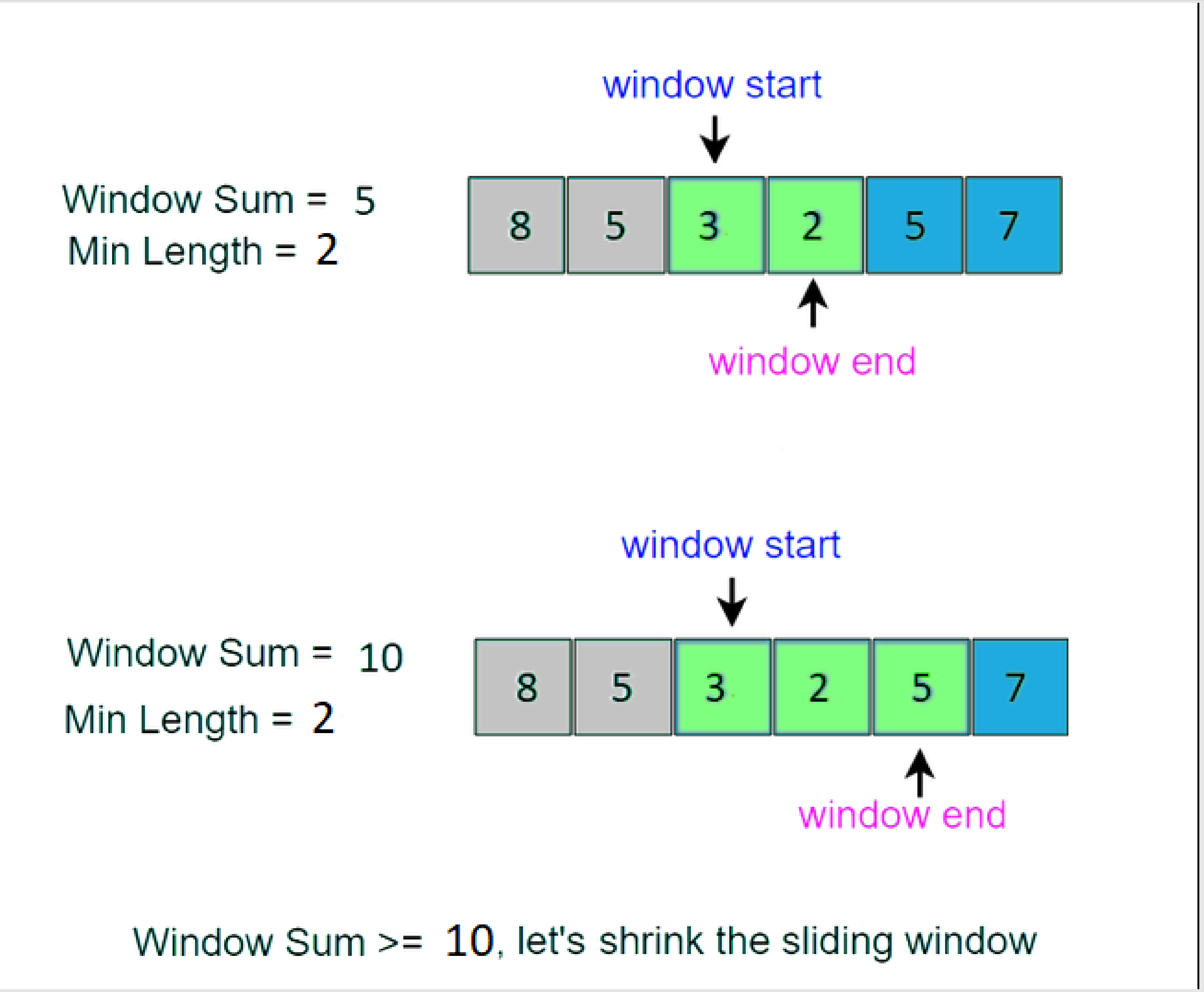
Min Length = 2

Window Sum = 10

Min Length = 2

Window Sum >= 10, let's shrink the sliding window

Solution:



Window Sum = 5
Min Length = 2

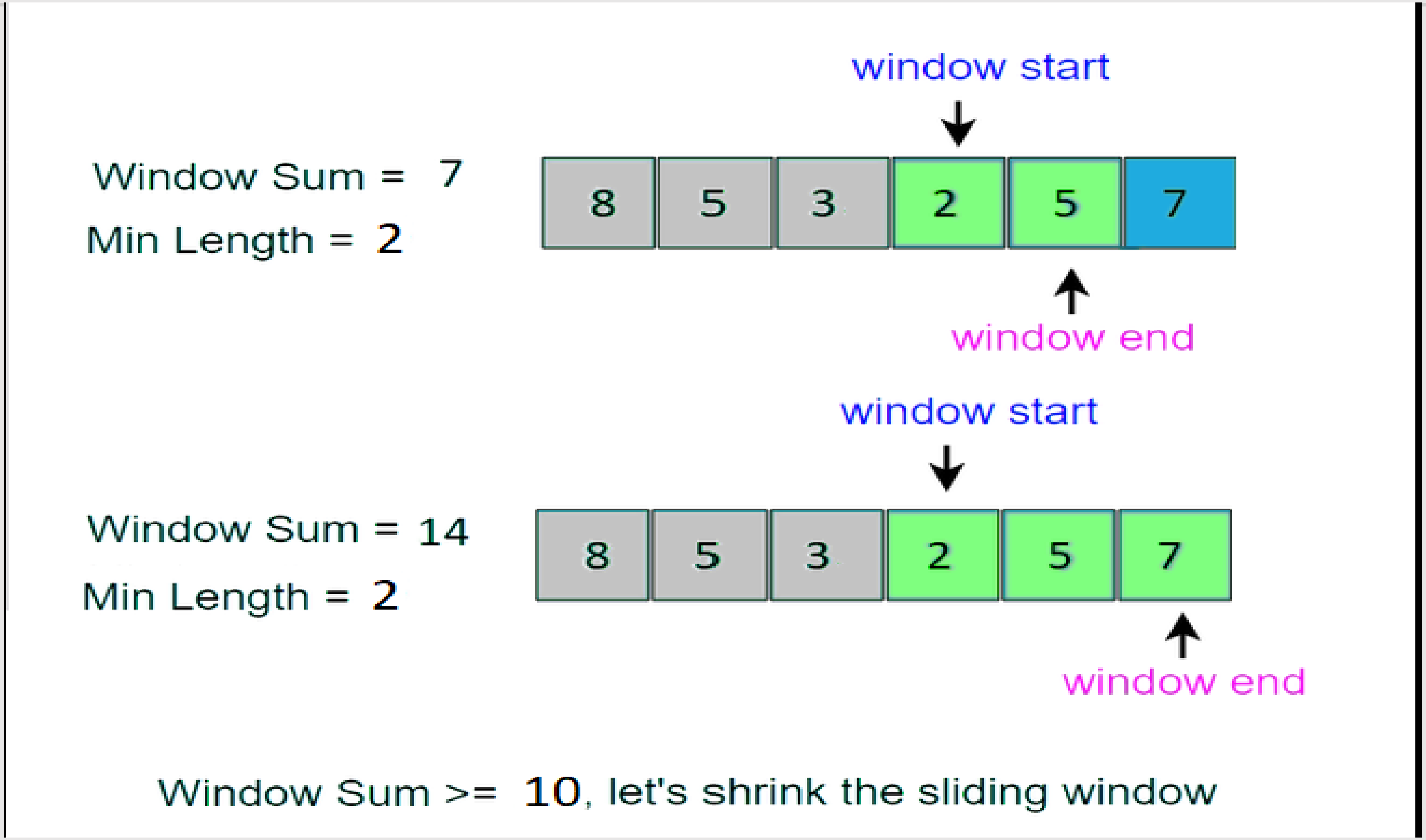
8	5	3	2	5	7

Window Sum = 10
Min Length = 2

8	5	3	2	5	7

Window Sum >= 10, let's shrink the sliding window

Solution:



window start

Window Sum = 7

Min Length = 2

8	5	3	2	5	7

window end

window start

Window Sum = 14

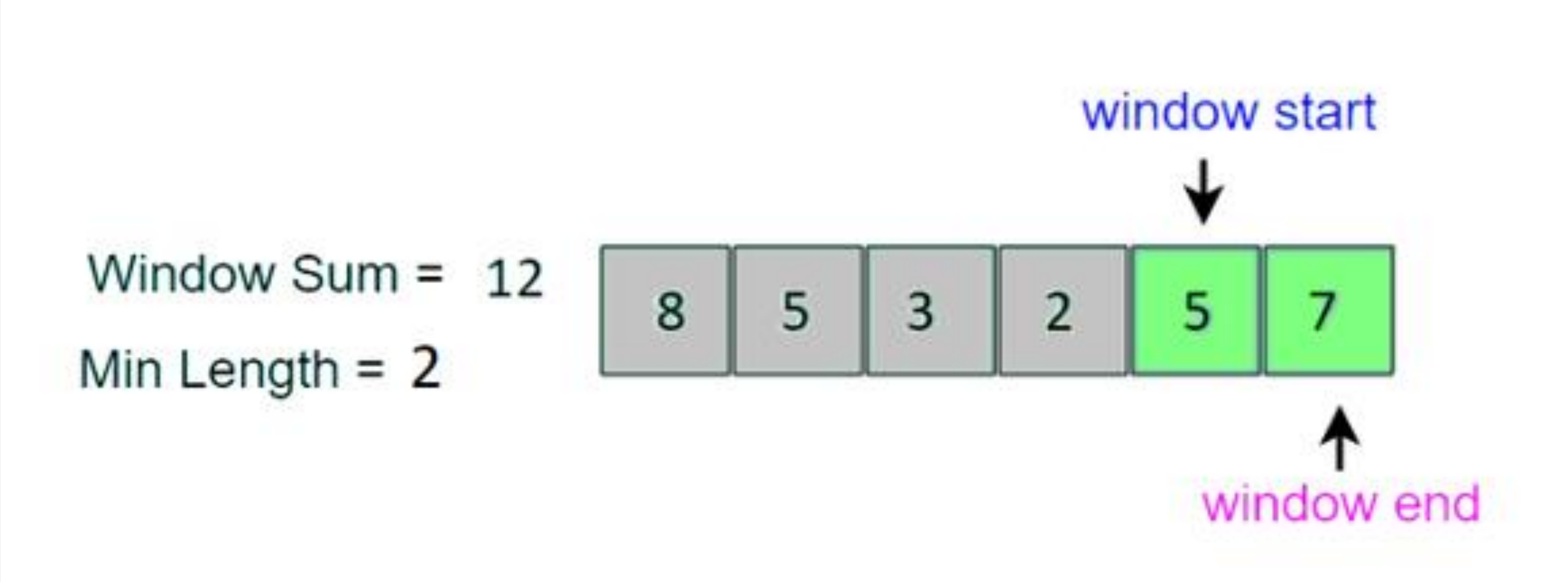
Min Length = 2

8	5	3	2	5	7

window end

Window Sum >= 10, let's shrink the sliding window

Solution:



window start

Window Sum = 12

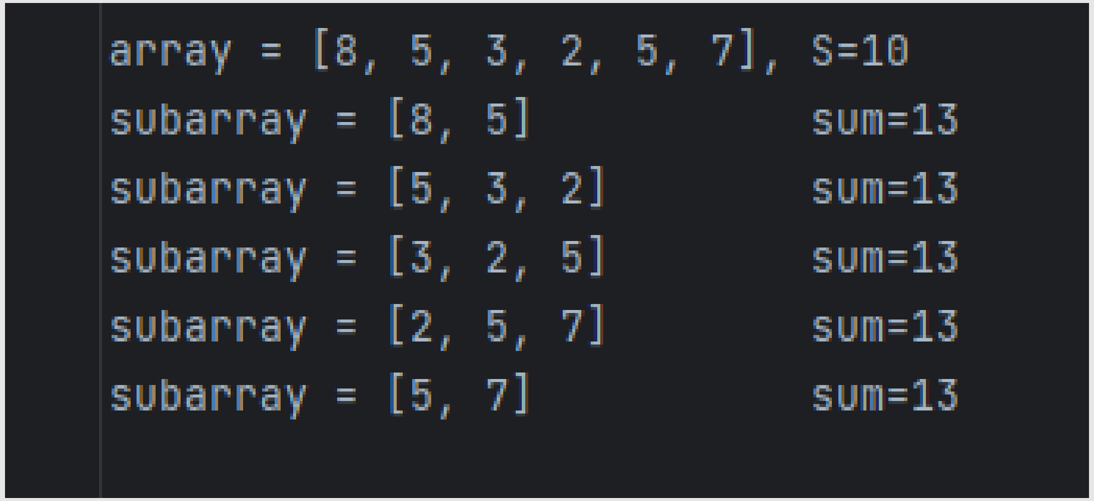
Min Length = 2

8	5	3	2	5	7

window end

Solution:

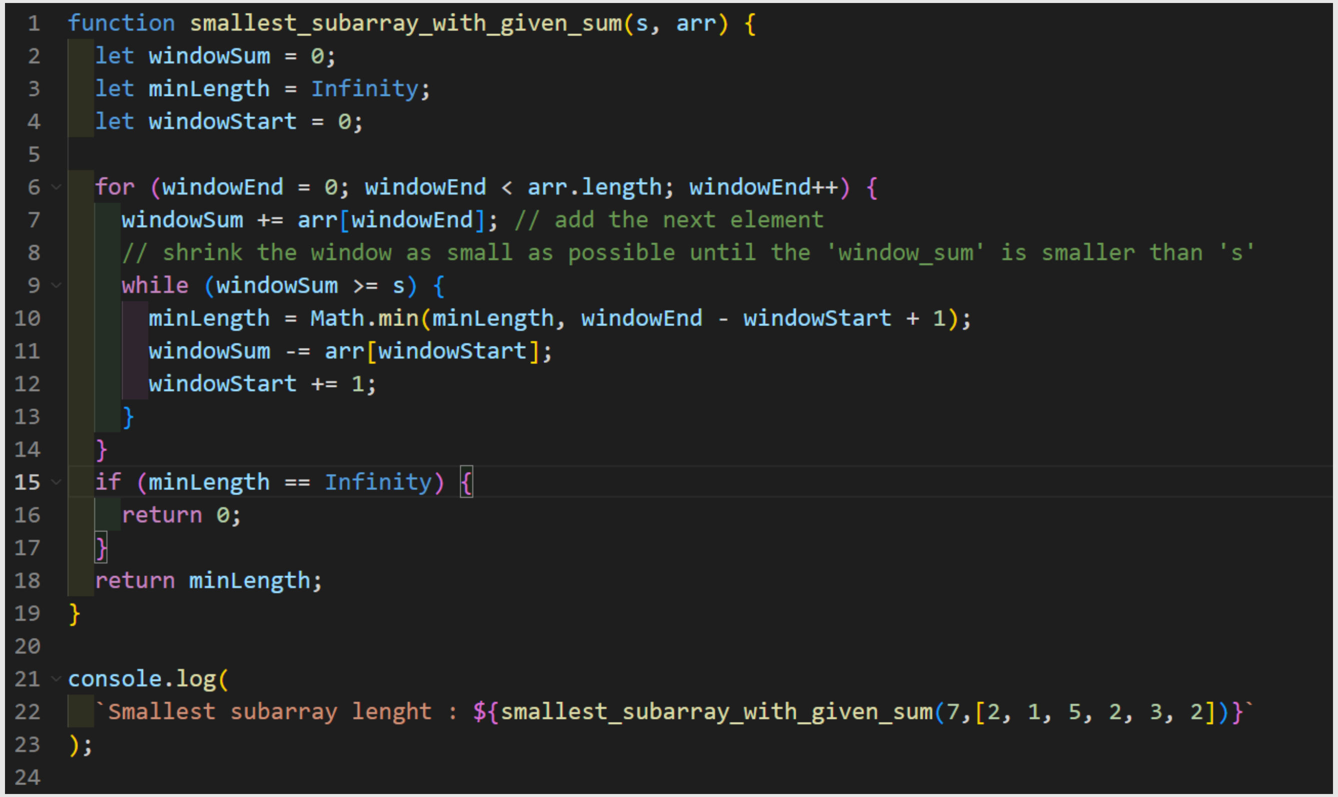
- In the solution virtualization we got four arrays with sum grater than 10 these arrays are:



```
array = [8, 5, 3, 2, 5, 7], S=10
subarray = [8, 5]           sum=13
subarray = [5, 3, 2]        sum=13
subarray = [3, 2, 5]        sum=13
subarray = [2, 5, 7]        sum=13
subarray = [5, 7]           sum=13
```

- The smallest subarray with in the example is third array [5,2] or [5,7] so the output will be 2

Code implementation:

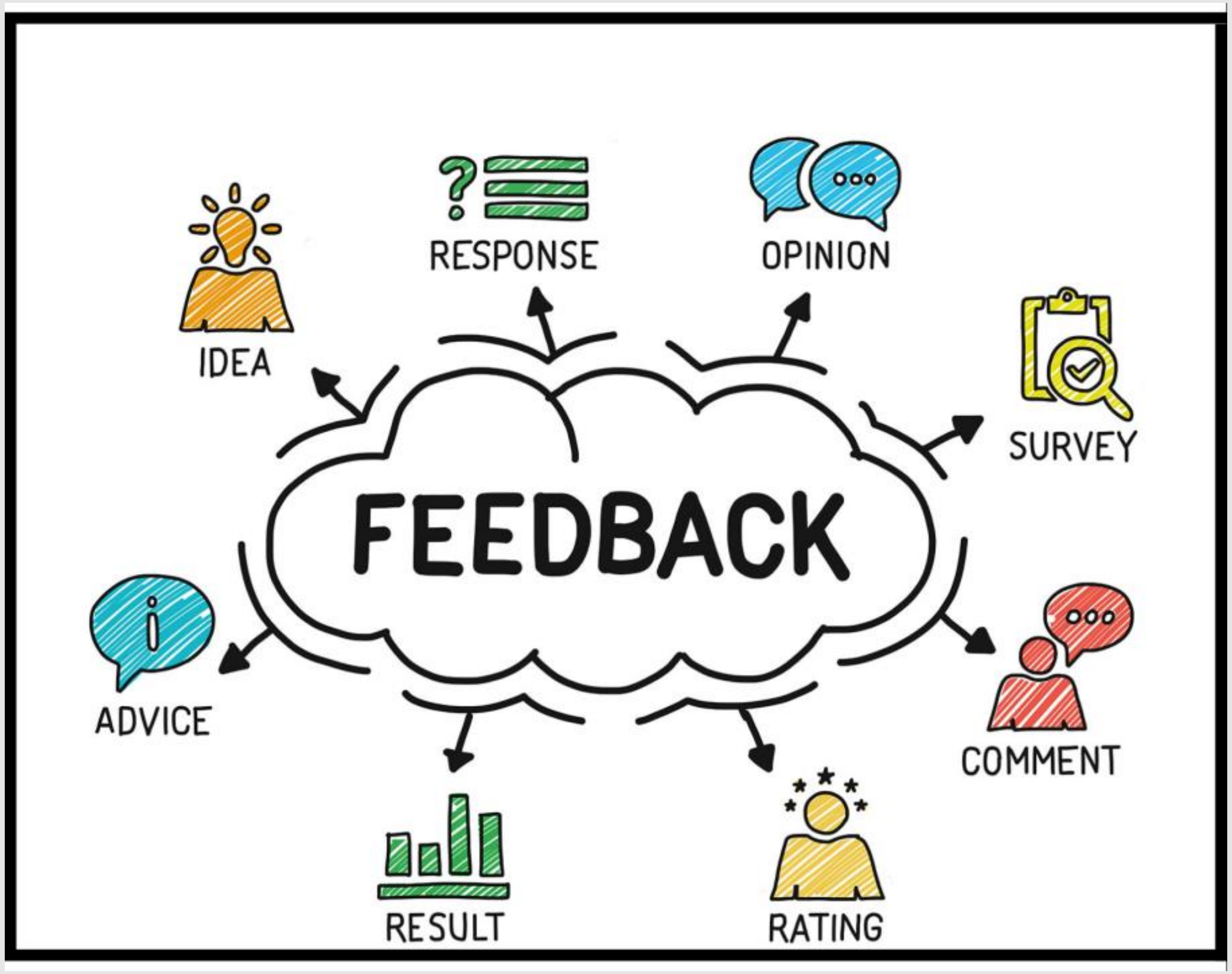


```javascript
function smallest_subarray_with_given_sum(s, arr) {
  let windowSum = 0;
  let minLength = Infinity;
  let windowStart = 0;

  for (windowEnd = 0; windowEnd < arr.length; windowEnd++) {
    windowSum += arr[windowEnd]; // add the next element
    // shrink the window as small as possible until the 'window_sum' is smaller than 's'
    while (windowSum >= s) {
      minLength = Math.min(minLength, windowEnd - windowStart + 1);
      windowSum -= arr[windowStart];
      windowStart += 1;
    }
  }
  if (minLength == Infinity) {
    return 0;
  }
  return minLength;
}

console.log(
  `Smallest subarray lenght : ${smallest_subarray_with_given_sum(7,[2, 1, 5, 2, 3, 2])}`
);

```

Complexity:

- Time complexity: O(N)
- Space complexity: O(1)

Feedback:


FEEDBACK
IDEA
RESPONSE
OPINION
SURVEY
COMMENT
RATING
RESULT
ADVICE

Leetcode questions:

- Minimum size subarray sum
- Minimum window substring

Evaluation

- **Let us evaluate this session by filling out the survey.**
- https://forms.office.com/e/nYjZHFtsPV

- **The aim of the evaluation is to enhance the content.**

Problem Solving Pattern Session evaluation

AXSOS

Zettachring 6 · 70567 Stuttgart
Fon +49 711.901196 0
Fax +49 711.901196 111

Daimlerstraße 17/1 · 72581 Dettingen an der Erms
Fon +49 711.901196 0
Fax +49 711.901196 111

Neuenhofer Str. 11 · 42657 Solingen
Fon +49 212.2245505 0
Fax +49 212.2245505 110

Ougarit Bld. 1st floor · Al Irsal
Street · Al Masayef · Ramallah
Fon +970 2 2988350
Fax +970 2 2988364

19 Hartom Street · Har Hotzvim · Jerusalem
Fon +970 2 2988350
Fax +970 2 2988364

Karatasou 7 · 54627 Thessaloniki
Fon +49 711.901196 0
Fax +49 711.901196 111