Arrays Challenge-First Repeating Element
(Amazon, Oracle)

Problem

Given an array **arr**[] of size **N**. The task is to find the first repeating element in an array of integers, i.e., an element that occurs more than once and whose index of first occurrence is smallest.

Constraints

1 <= N <= 10^6
0 <= Ai <= 10^6

Example

Input**:**
7
1 5 3 4 3 5 6

Output**:**
2

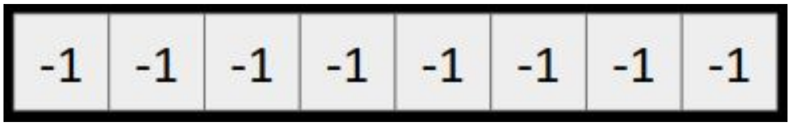
Explanation**:**
5 is appearing twice and its first appearance is at index 2 which is less than 3 whose first occurring index is 3.

Solution

Base idea: To check if an element is repeating, we maintain an array idx[], which stores the first occurrence (index) of a particular element a[i].

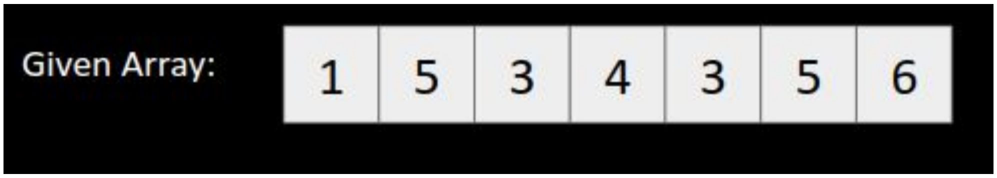
Steps

1. Initialise the idx[] with -1, and minidx with INT_MAX.



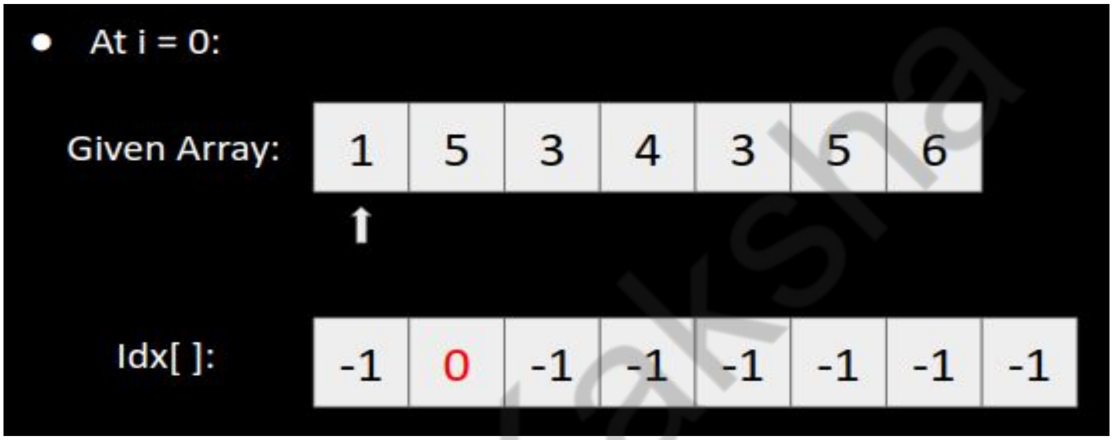
-1	-1	-1	-1	-1	-1	-1	-1

2. Keep updating idx[], while traversing the given array.



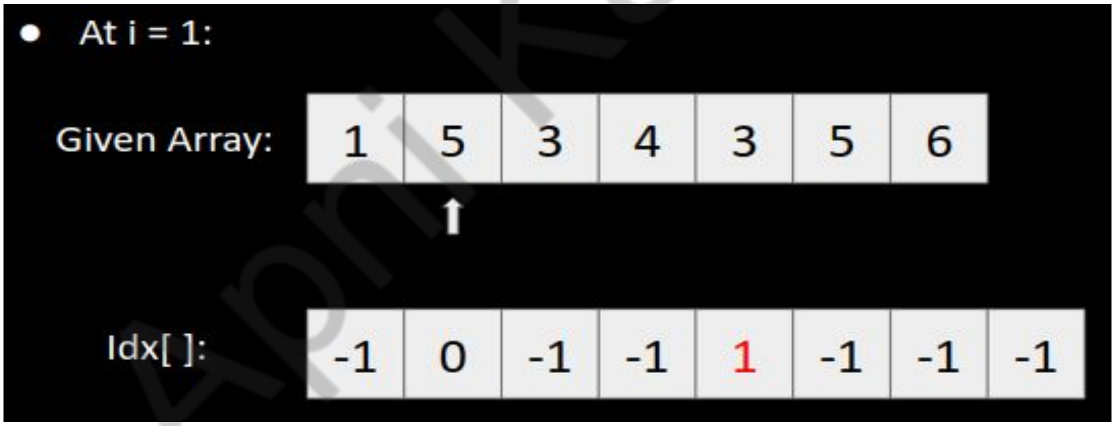
Given Array:	1	5	3	4	3	5	6

Iterations



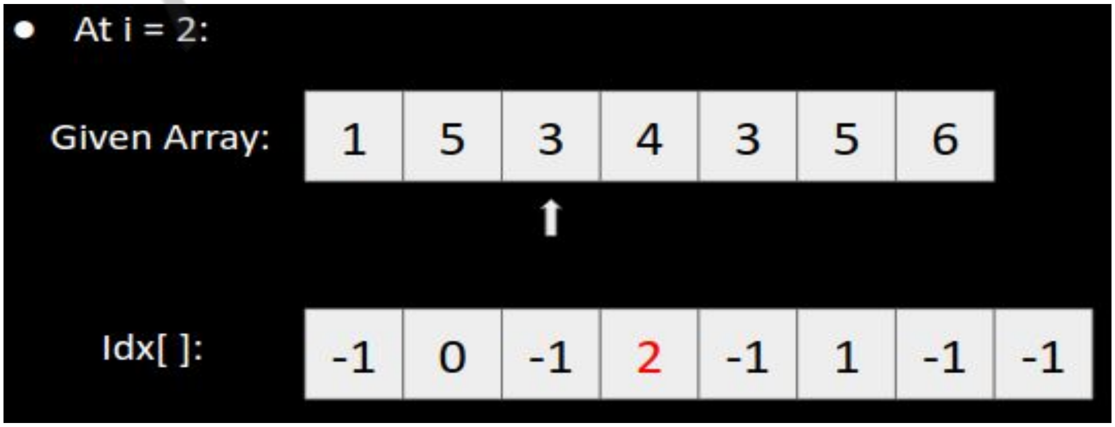
- At i = 0:

Given Array:	1	5	3	4	3	5	6	
	⬆							
Idx[ ]:	-1	0	-1	-1	-1	-1	-1	-1



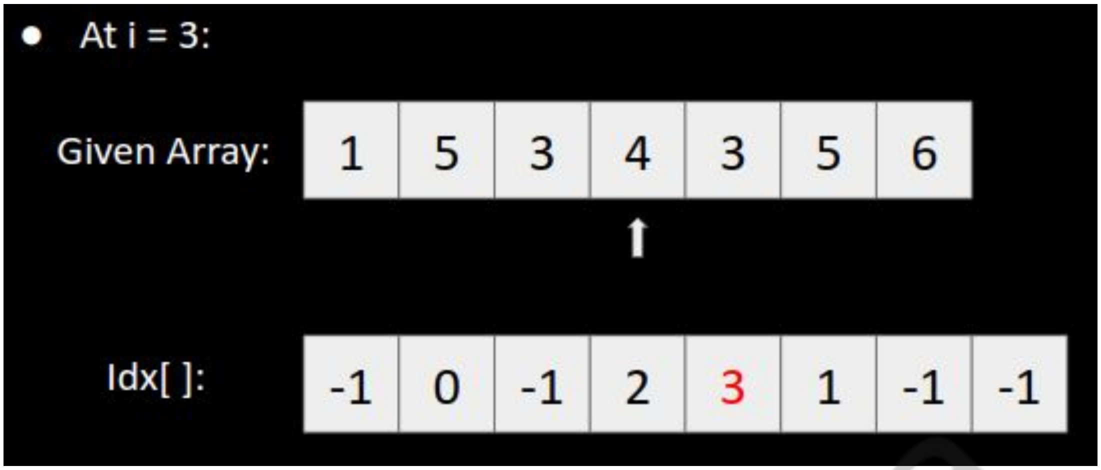
- At i = 1:

Given Array:	1	5	3	4	3	5	6	
		⬆						
Idx[ ]:	-1	0	-1	-1	1	-1	-1	-1



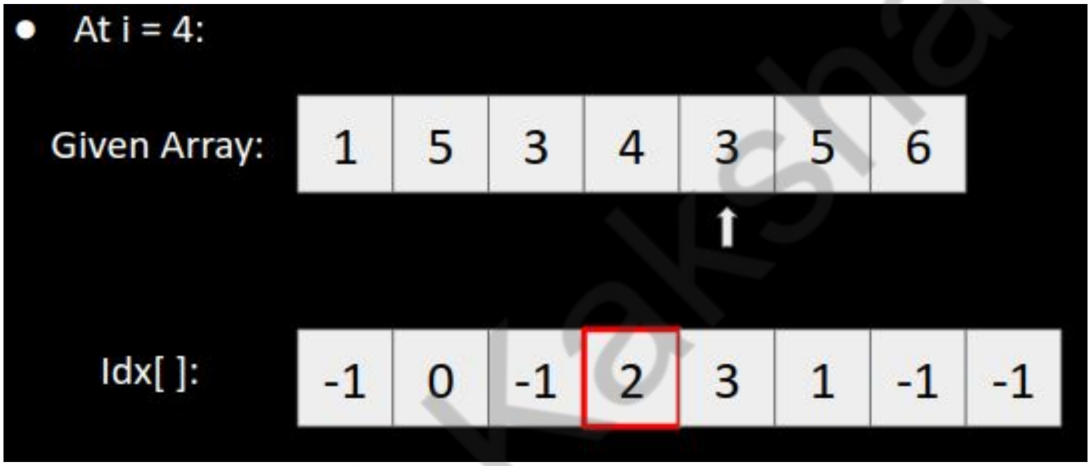
- At i = 2:

Given Array:	1	5	3	4	3	5	6	
			⬆					
Idx[ ]:	-1	0	-1	2	-1	1	-1	-1



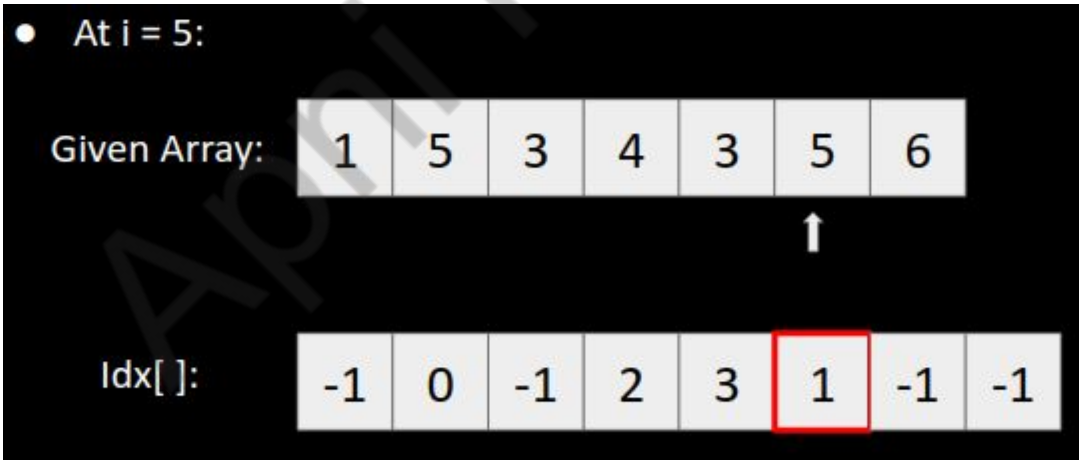
- At i = 3:

Given Array:	1	5	3	4	3	5	6	
				↑				
Idx[ ]:	-1	0	-1	2	3	1	-1	-1



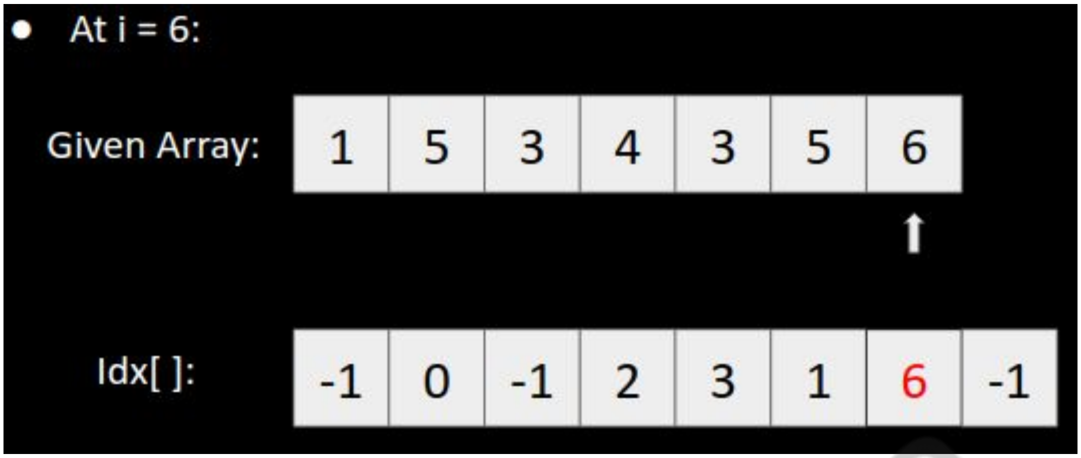
- At i = 4:

Given Array:	1	5	3	4	3	5	6	
					↑			
Idx[ ]:	-1	0	-1	2	3	1	-1	-1



- At i = 5:

Given Array:	1	5	3	4	3	5	6	
						↑		
Idx[ ]:	-1	0	-1	2	3	1	-1	-1



- At i = 6:

Given Array:	1	5	3	4	3	5	6	
							⬆	
Idx[ ]:	-1	0	-1	2	3	1	6	-1

Arrays Challenge - Smallest Positive Missing Number (Amazon, Samsung, Snapdeal, Accolite)

Problem

Find the smallest positive missing number in the given array.

Example: [0, -10, 1, 3, -20], Ans = 2

Intuition

If in O(1), we can tell if an element is present in an array, then our task will be simpler.

For that, we will maintain a Check[ ] array, that will if an element x is present in the array or not.

It will be of boolean type as we only need to check the presence or absence of the number.

Steps to Solve:

1. Build the Check[ ] array initialized with False at all indices.
2. By iterating over the array and marking non-negative a[i] as true i.e.

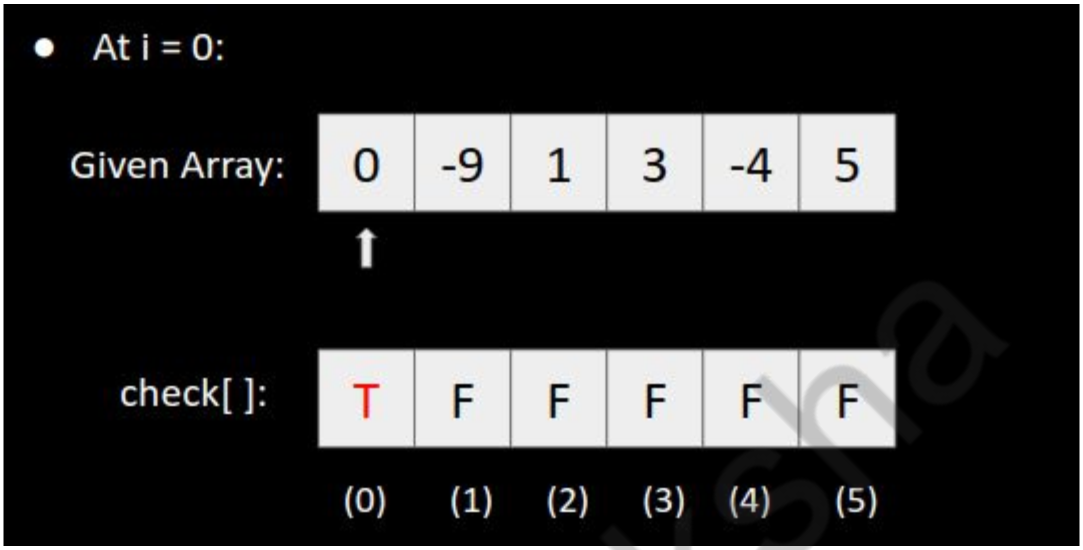
   ```
   if(a[i] >= 0)
     check[a[i]] = True
   ```

3. Iterate in the Check[ ] from i=1, BREAK the loop when you find check[i] = False and store that i in the ans variable.
4. Output the ans.

Example:

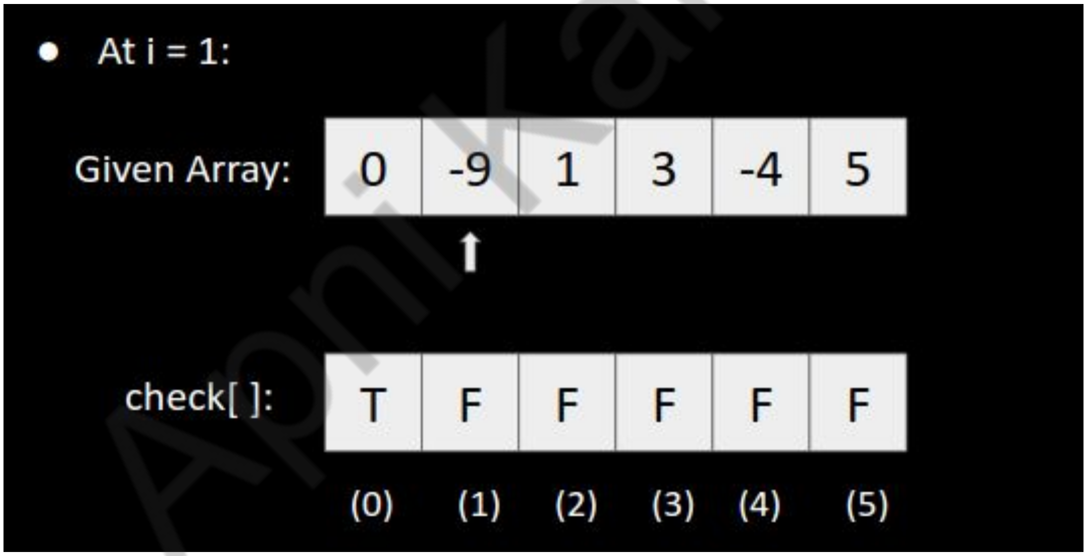
Given Array: [0, -9, 1, 3, -4, 5]

Iterations



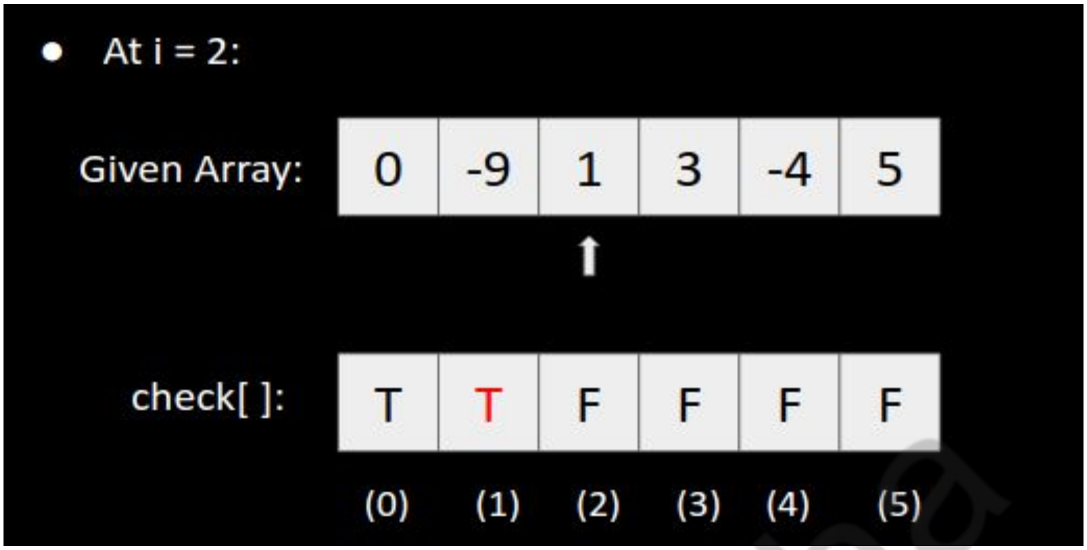
- At i = 0:

Given Array:	0	-9	1	3	-4	5
	↑					
check[ ]:	T	F	F	F	F	F
	(0)	(1)	(2)	(3)	(4)	(5)



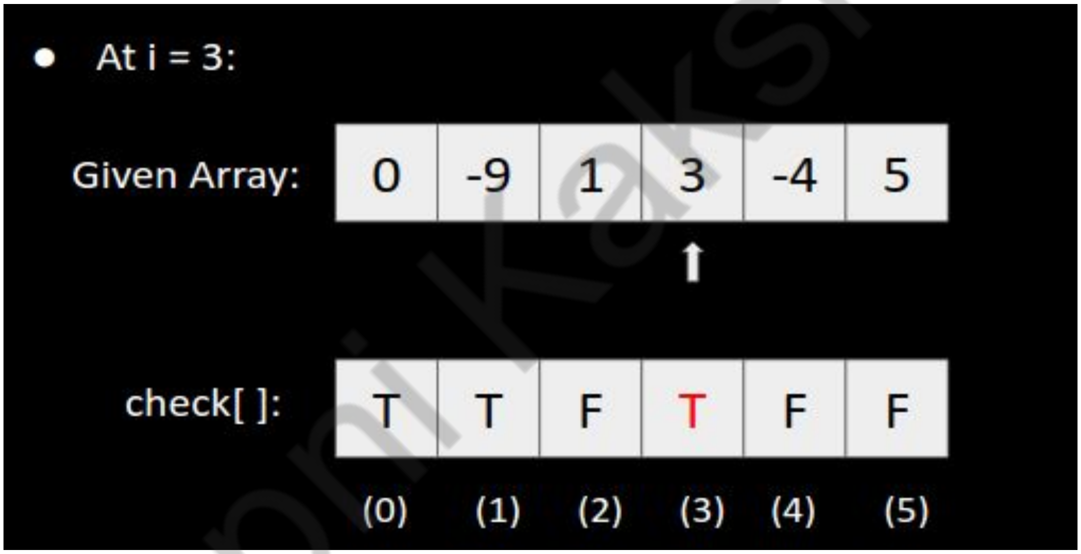
- At i = 1:

Given Array:	0	-9	1	3	-4	5
		↑				
check[ ]:	T	F	F	F	F	F
	(0)	(1)	(2)	(3)	(4)	(5)



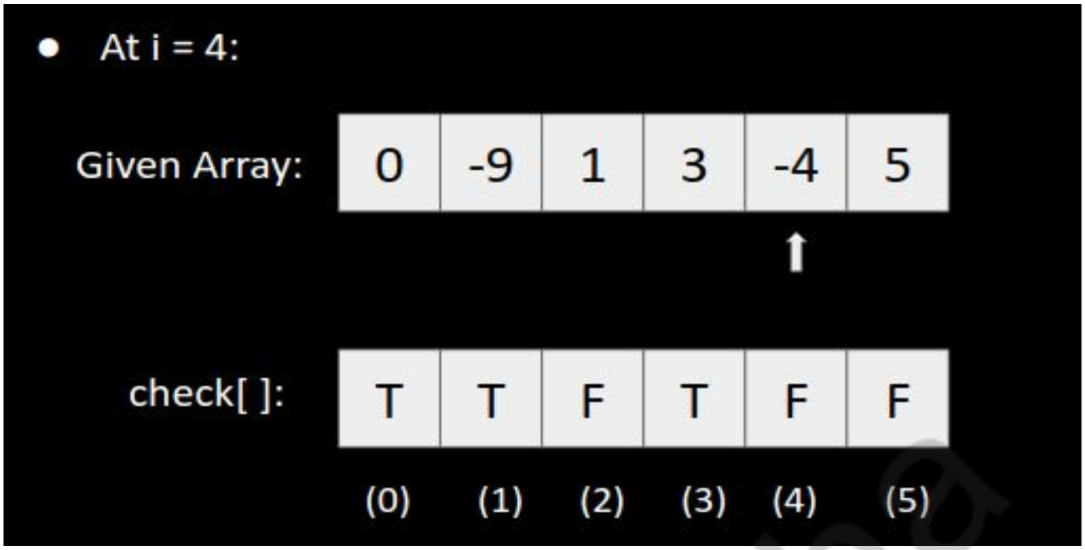
- At i = 2:

Given Array:	0	-9	1	3	-4	5
			↑			
check[ ]:	T	T	F	F	F	F
	(0)	(1)	(2)	(3)	(4)	(5)



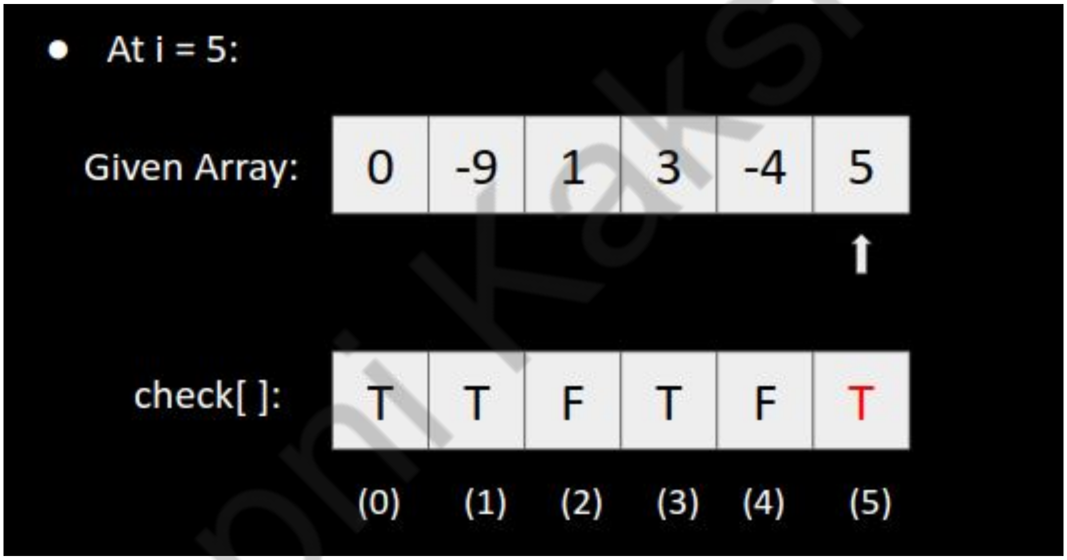
- At i = 3:

Given Array:	0	-9	1	3	-4	5
				↑		
check[ ]:	T	T	F	T	F	F
	(0)	(1)	(2)	(3)	(4)	(5)



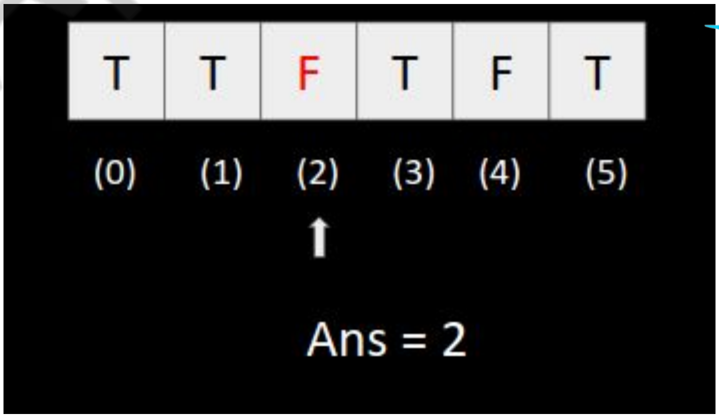
- At i = 4:

Given Array:	0	-9	1	3	-4	5
					⬆	
check[ ]:	T	T	F	T	F	F
	(0)	(1)	(2)	(3)	(4)	(5)



- At i = 5:

Given Array:	0	-9	1	3	-4	5
						⬆
check[ ]:	T	T	F	T	F	T
	(0)	(1)	(2)	(3)	(4)	(5)



T	T	F	T	F	T
(0)	(1)	(2)	(3)	(4)	(5)
		⬆			

Ans = 2

Arrays Challenge-Subarray with given sum (Google, Amazon, Facebook, Visa)

Problem

Given an unsorted array **A** of size **N** of non-negative integers, find a continuous subarray which adds to a given number **S.**

Constraints

$1 <= N <= 10^5$

$0 <= Ai <= 10^{10}$

Example

Input**:**

N = 5, S = 12

A[] = {1,2,3,7,5}

Output**:** 2 4

Explanation**:** The sum of elements from 2nd position to 4th position is 12.

Solution

Brute Force Solution

- Find sum of all possible subarrays. If any of the sum equates to **S,** output the starting and ending index of the subarray.

  Time Complexity : **O(n^2)**

Optimized Approach

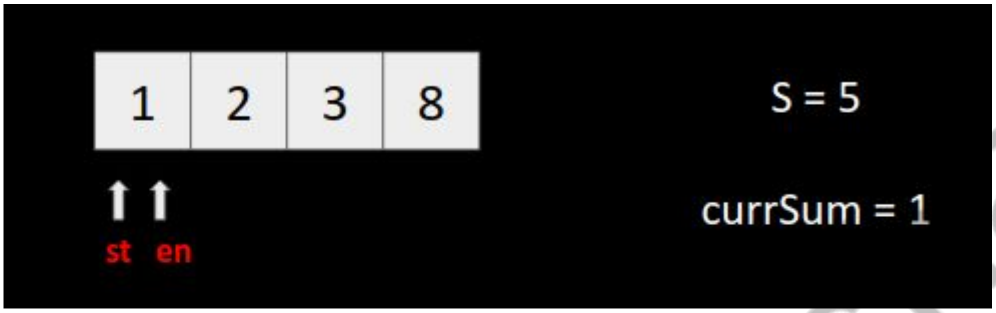
Steps:

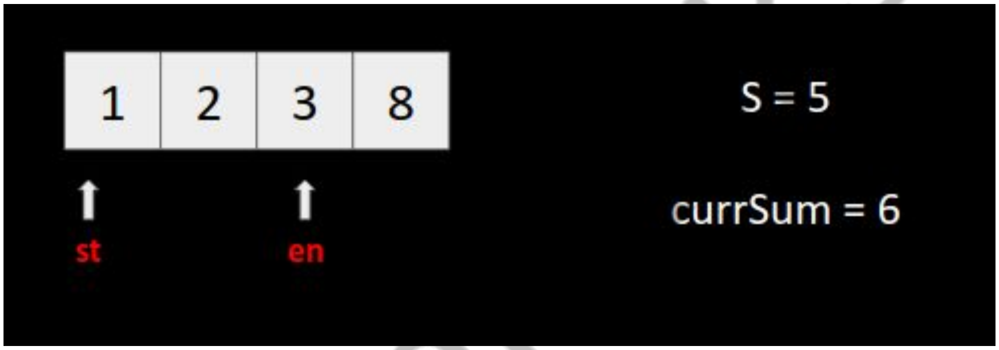
1. Keep the pointers st and en, and a variable currSum that stores the sum from st to en.
2. initialize st = 0, en = 0
3. Increment en till currSum + a[en + 1] > S
4. When 3rd condition occurs, start increasing st until currSum <= S.

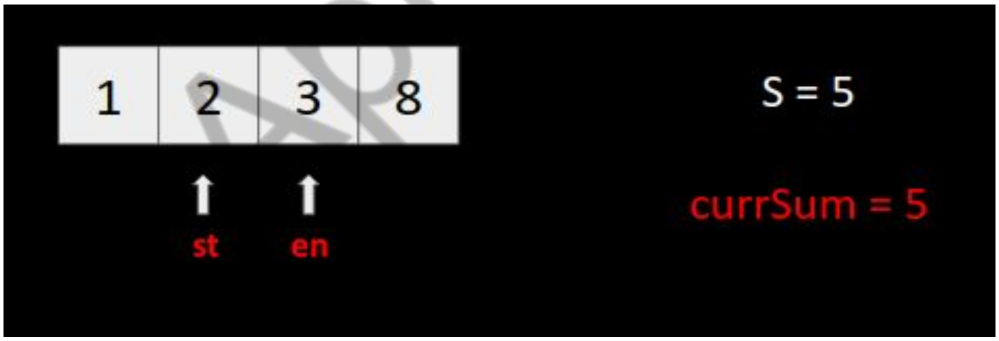
5. Whenever the condition (currSum = S) is satisfied, store st and en and BREAK from the loop.

Time Complexity: **O(n)**

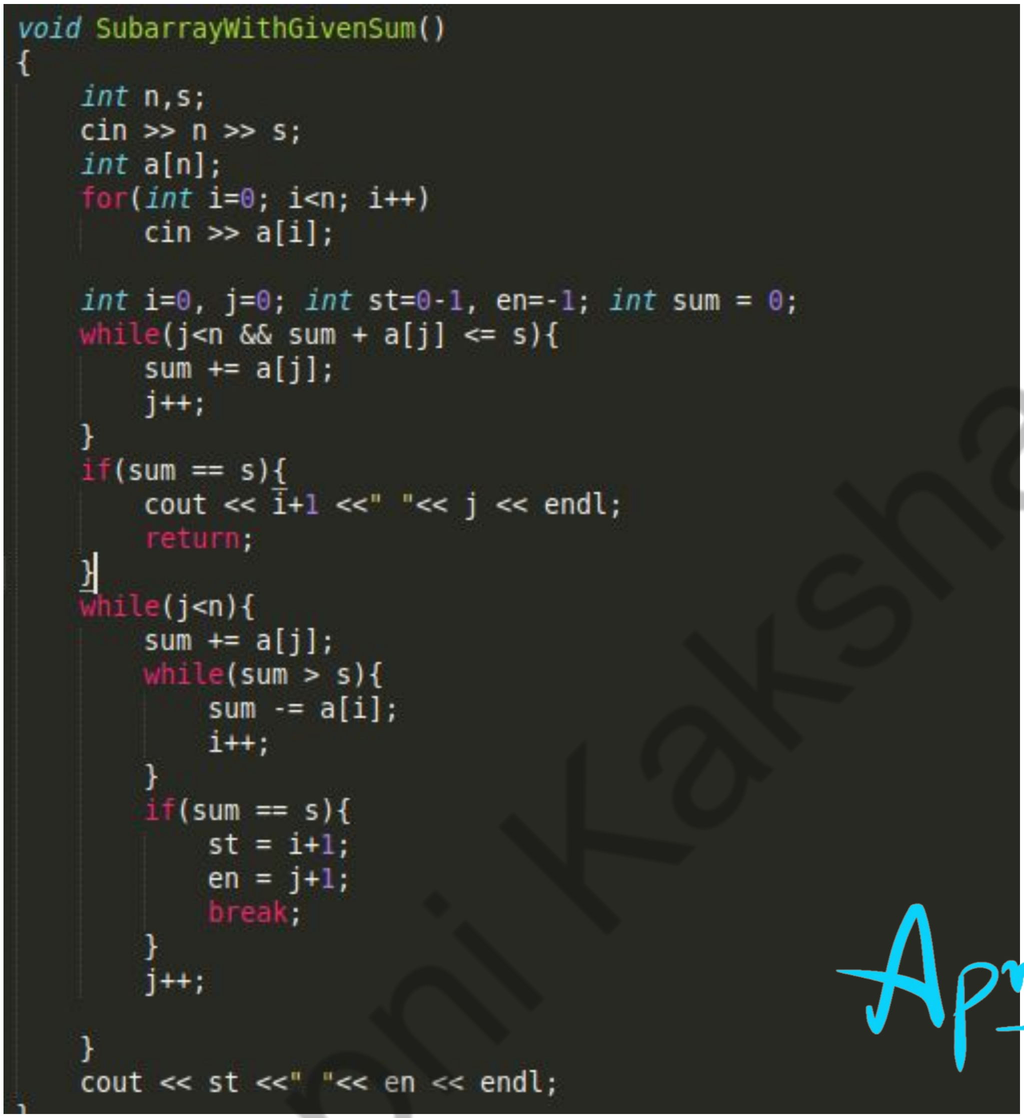
Iterations







Code:



```
void SubarrayWithGivenSum()
{
    int n,s;
    cin >> n >> s;
    int a[n];
    for(int i=0; i<n; i++)
        cin >> a[i];

    int i=0, j=0; int st=0-1, en=-1; int sum = 0;
    while(j<n && sum + a[j] <= s){
        sum += a[j];
        j++;
    }
    if(sum == s){
        cout << i+1 <<" "<< j << endl;
        return;
    }
    while(j<n){
        sum += a[j];
        while(sum > s){
            sum -= a[i];
            i++;
        }
        if(sum == s){
            st = i+1;
            en = j+1;
            break;
        }
        j++;

    }
    cout << st <<" "<< en << endl;
}
```