Day 10

19 November 2022 13:49

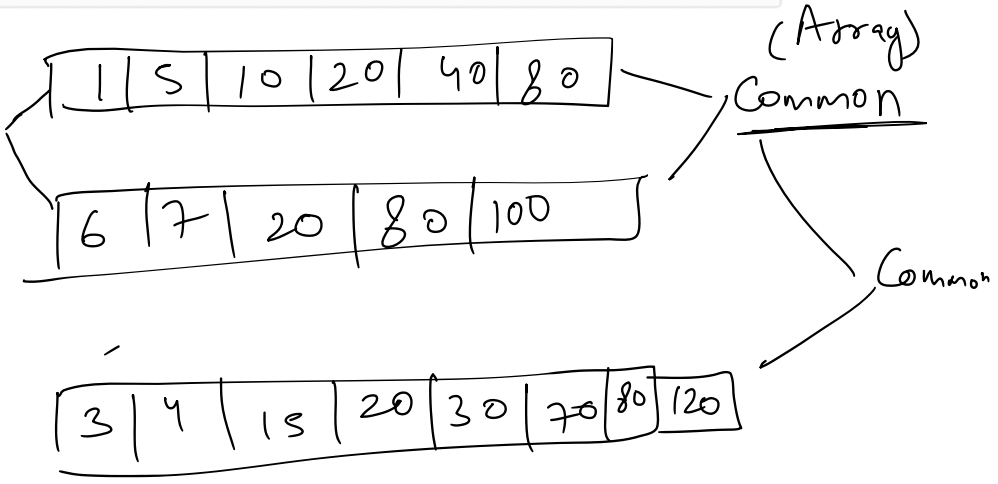
Given three arrays sorted in increasing order. Find the elements that are common in all three arrays.

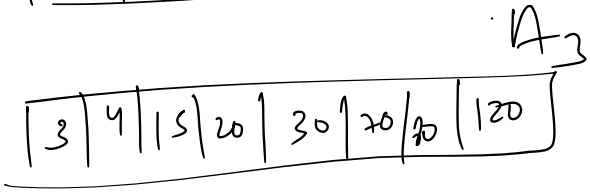
Note: can you take care of the duplicates without using any additional Data Structure?

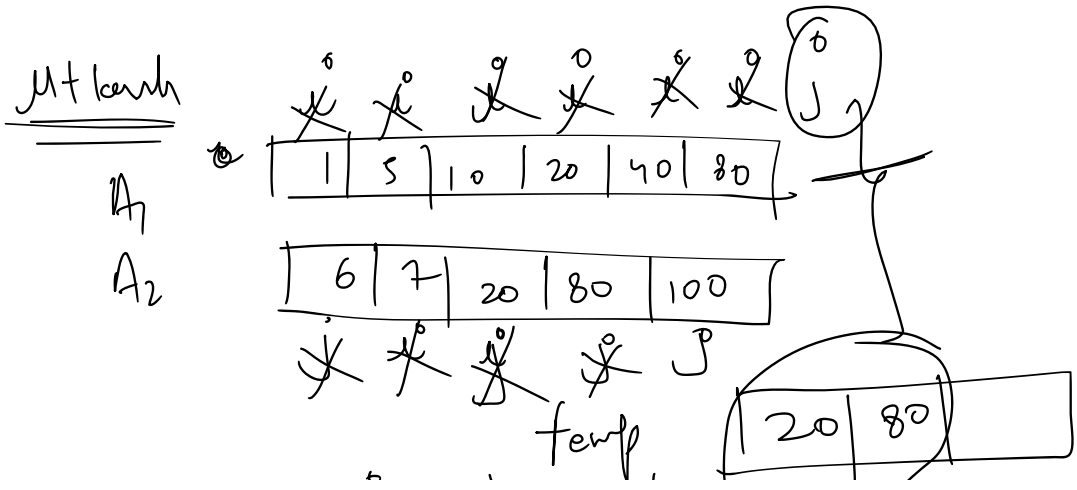
Example 1:

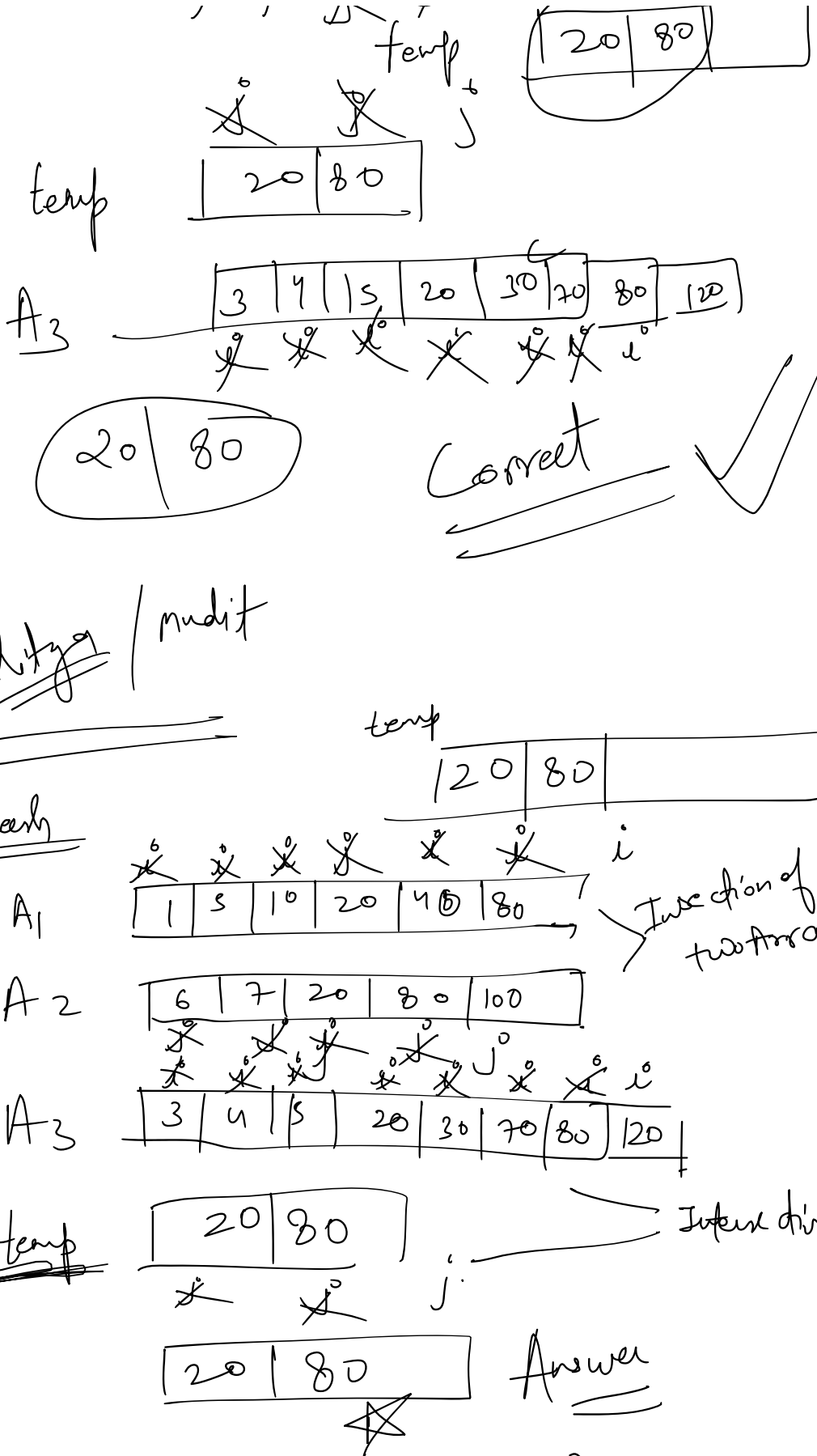
Input:
n1 = 6; A = {1, 5, 10, 20, 40, 80}
n2 = 5; B = {6, 7, 20, 80, 100}
n3 = 8; C = {3, 4, 15, 20, 30, 70, 80, 120}
Output: 20 80
Explanation: 20 and 80 are the only common elements in A, B and C.

20, 80

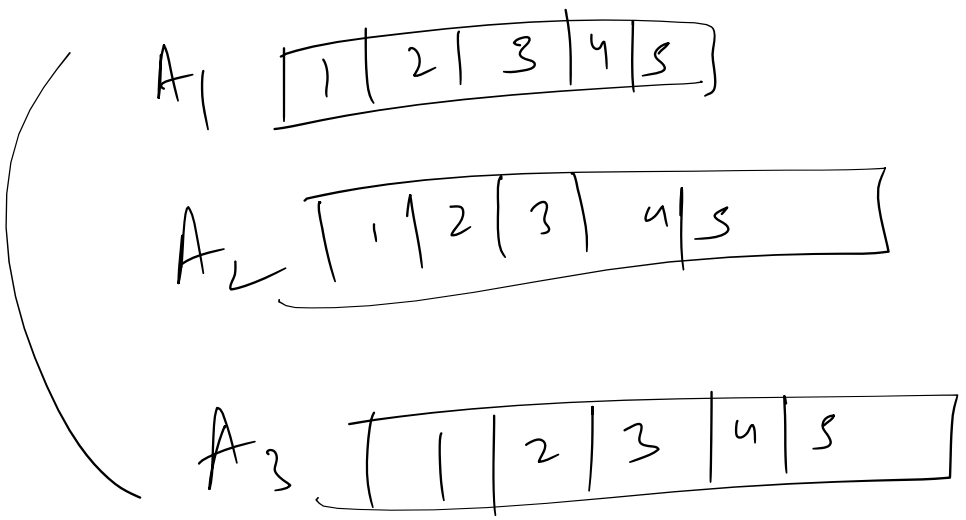




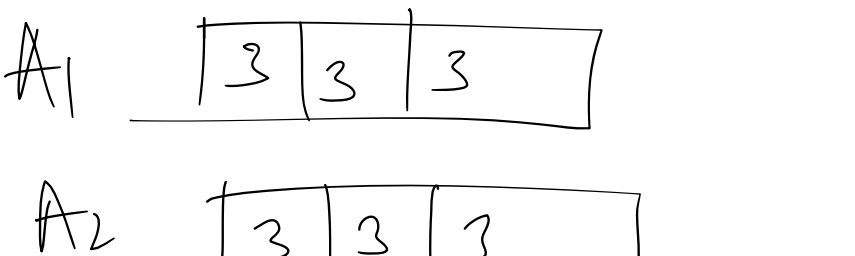



temp
20 80
i j
temp
20 80
A3
3 4 5 20 30 70 80 120
i
20 80
Correct
Aditya / mudit
temp
120 80
Answer
i
A1
1 5 10 20 40 80
Intersection of two Arrays
A2
6 7 20 80 100
j
A3
3 4 5 20 30 70 80 120
i
temp
20 80
i j
Intersection
20 80
Answer
T.C - O(n1 + n2 + n3)
S.C O(n1 + n2 + n3)

S.C O(n1 + n2 + 5)



Ex



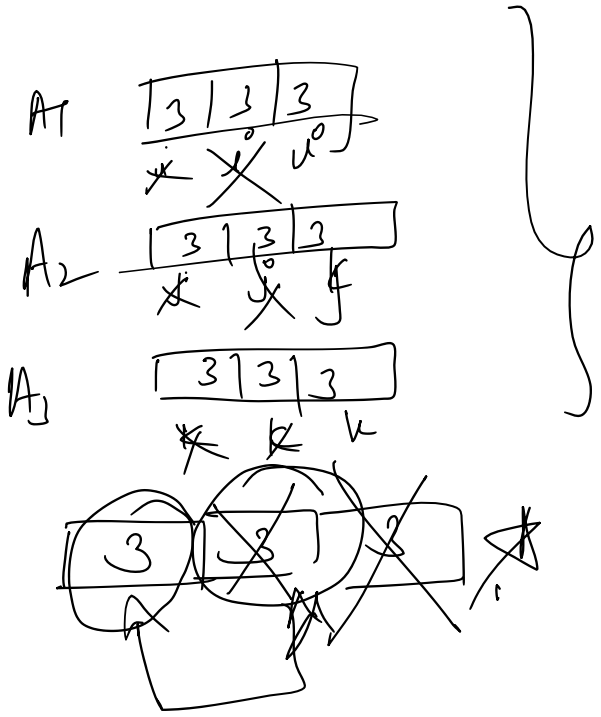
output 3 3 3 ✗

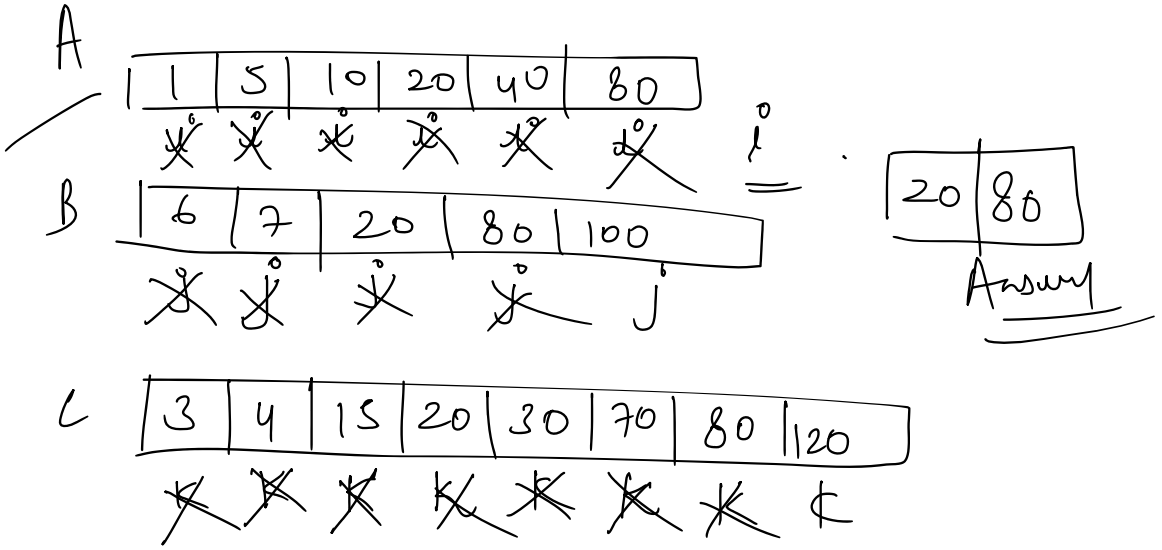
Except Arm

3 set

Time Consuming

vector / Array





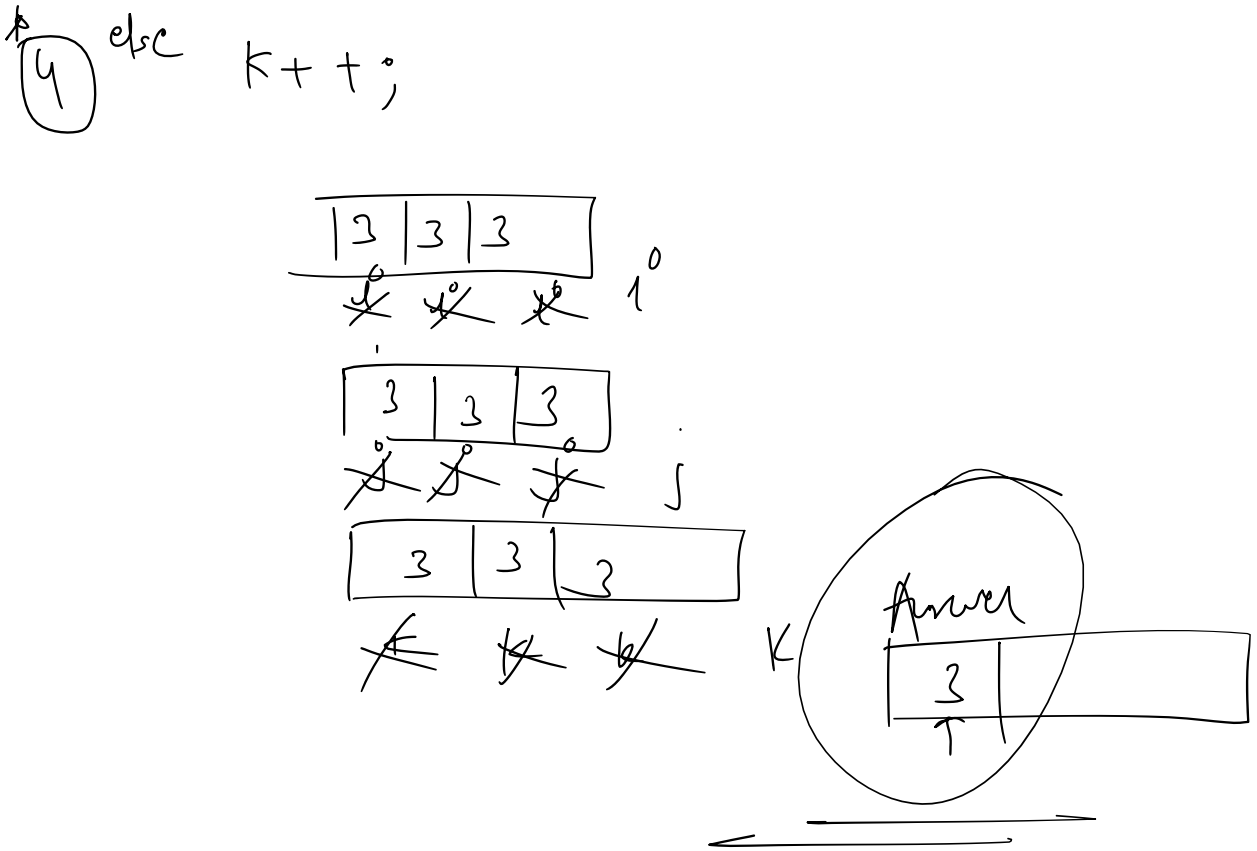
① A[i] == B[j] == C[k] ⟹ ans.push_back(A[i]); i++; j++; k++;

else if

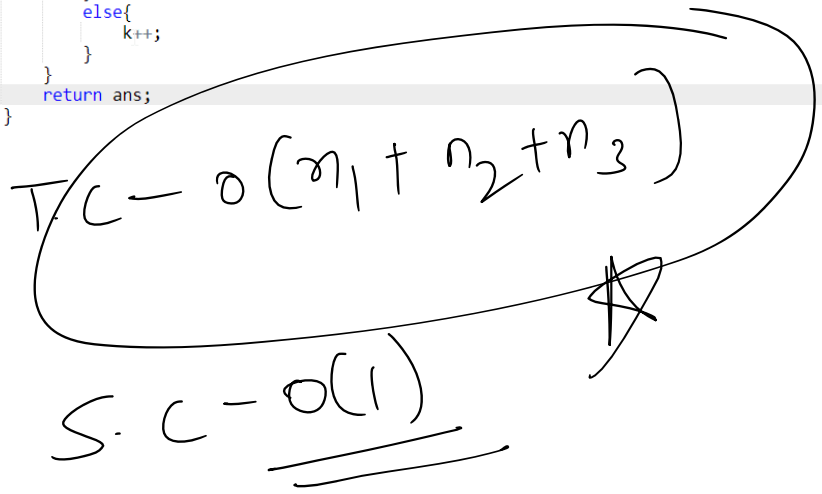
② A[i] < B[j] i++;

else

③ B[j] < C[k] j++;




```
vector <int> commonElements (int A[], int B[], int C[], int n1, int n2, int n3)
 {
    int i = 0,j = 0,k = 0;
    vector<int> ans;
    while(i < n1 && j < n2 && k < n3){
        // case 1
        if(A[i] == B[j] && B[j] == C[k]){
            if(ans.size() != 0){
                if(ans.back() != A[i]){
                    ans.push_back(A[i]);
                }
            }else{
                ans.push_back(A[i]);
            }
            i++;j++;k++;
        } // Case 2
        else if(A[i] < B[j]){
            i++;
        } // case 3
        else if(B[j] < C[k]){
            j++;
        } // Case 4
        else{
            k++;
        }
    }
    return ans;
 }
```



Given an unsorted array **Arr** of **N** positive and negative numbers. Your task is to create an array of alternate positive and negative numbers without changing the relative order of positive and negative numbers.

Note: Array should start with a positive number.

Example 1:

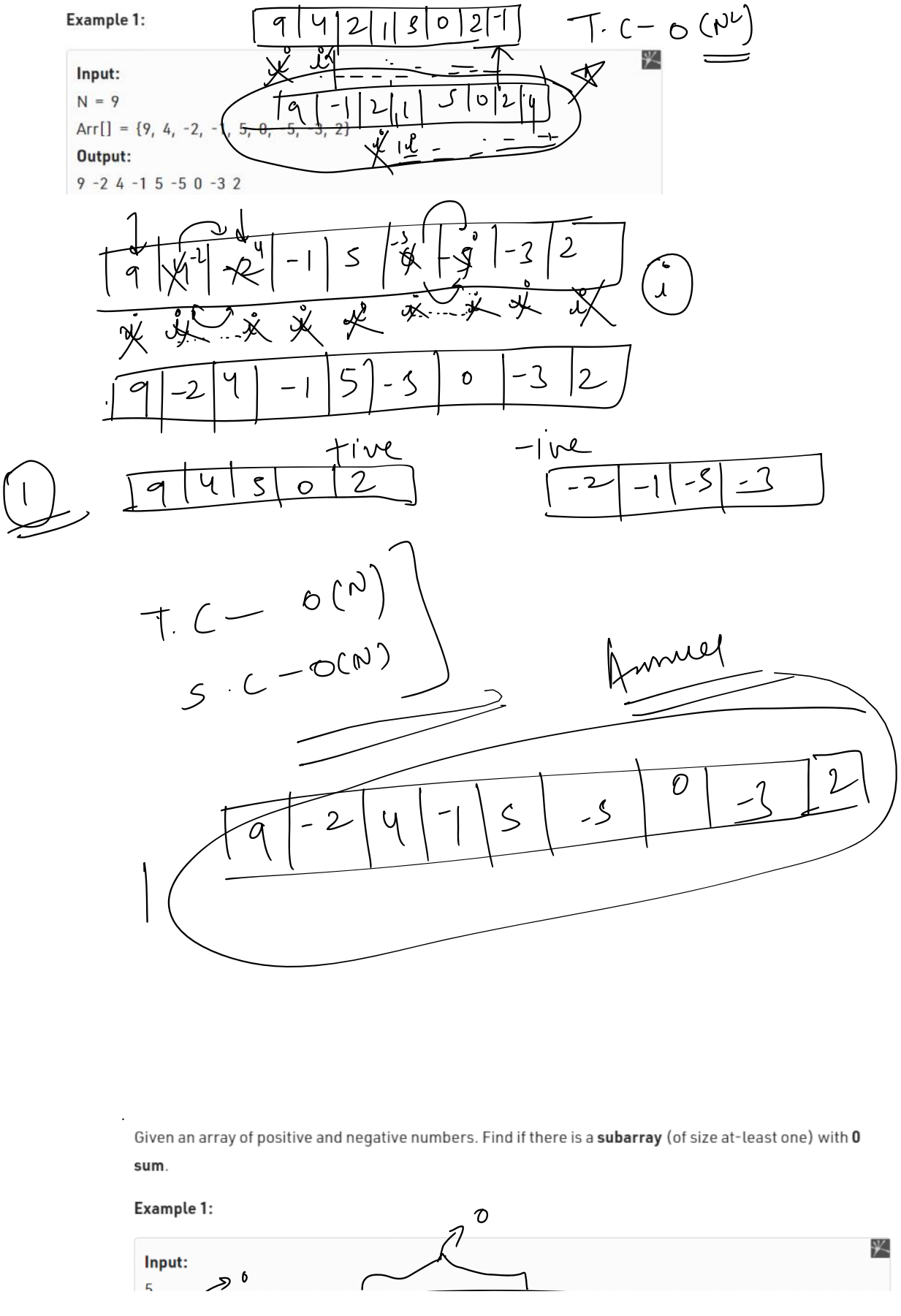
Input:

N = 9

Arr[] = {9, 4, -2, -1, 5, 0, -5, -3, 2}

Output:

9 -2 4 -1 5 -5 0 -3 2



Given an array of positive and negative numbers. Find if there is a **subarray** (of size at-least one) with **0 sum**.

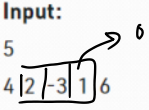
Example 1:

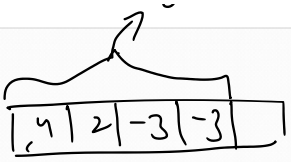
Input:

Input:

5

4 2 -3 1 6





Output:

Yes

Explanation:

2, -3, 1 is the subarray with sum 0.

Possibility

* All subarrays

subarray → 0

True

Naive

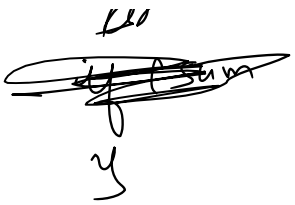
```
for(int i=0; i<n; i++){
    for(int j=i; j<n; j++){
        sum=0
        for(int k=i; k<=j; k++){
            sum += arr[k];
        }
        if(sum==0){
            return true;
        }
    }
}
```

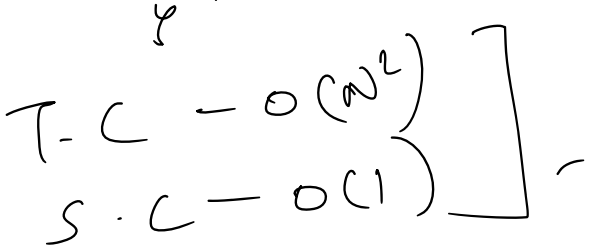
T.C - $O(N^3)$

S.C - $O(1)$

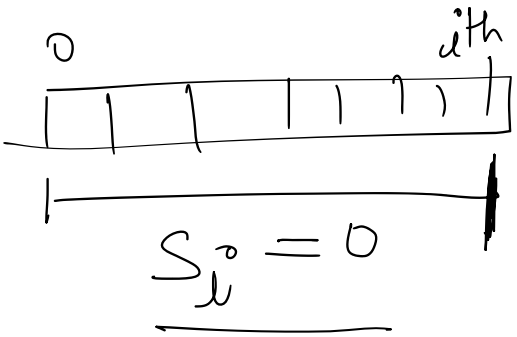
```
for(int i=0; i<n; i++){
    for(int j=i; j<n; j++){
        sum += arr[j]; if(sum==0){
            return true;
        }
```

$O(n^2)$

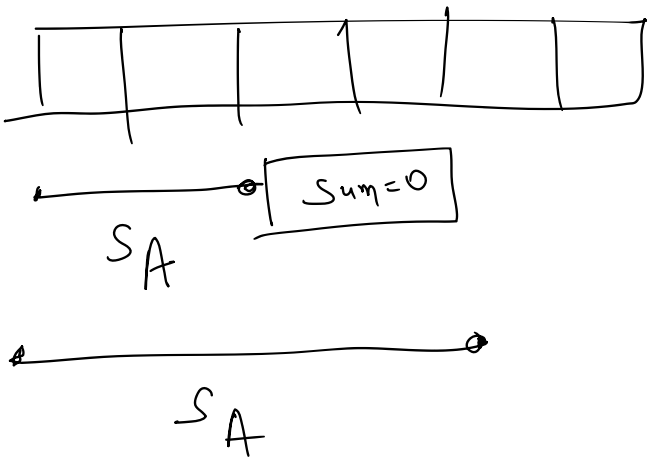


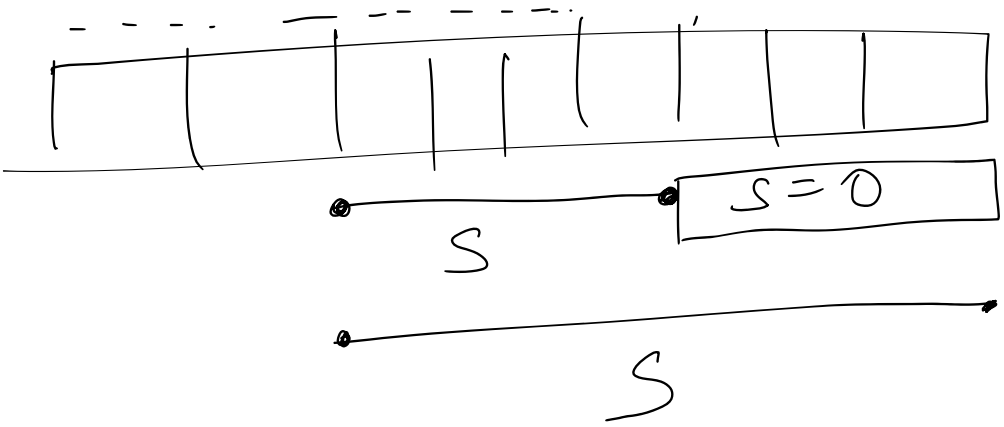


Case1



Case2





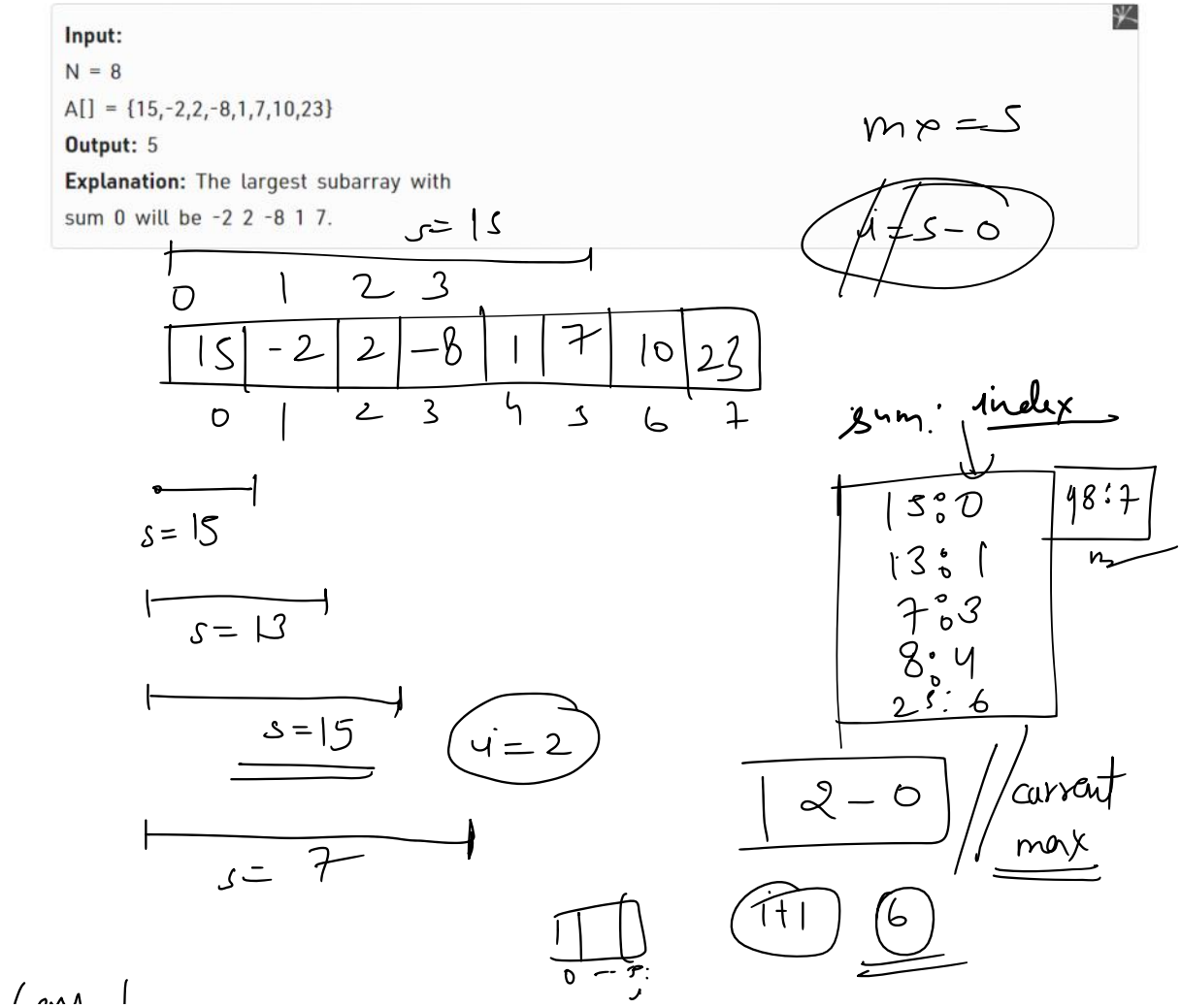
```cpp
bool subArrayExists(int arr[], int n)
{
    // case 1
    int sum = 0;
    unordered_set<int> us;
    for(int i = 0;i < n;i++){
        sum += arr[i];
        // case 1
        if(sum == 0){
            return true;
        }
        // case 2
        if(us.find(sum)!=us.end()){
            return true;
        }
        us.insert(sum);
    }
    return false;
}
```

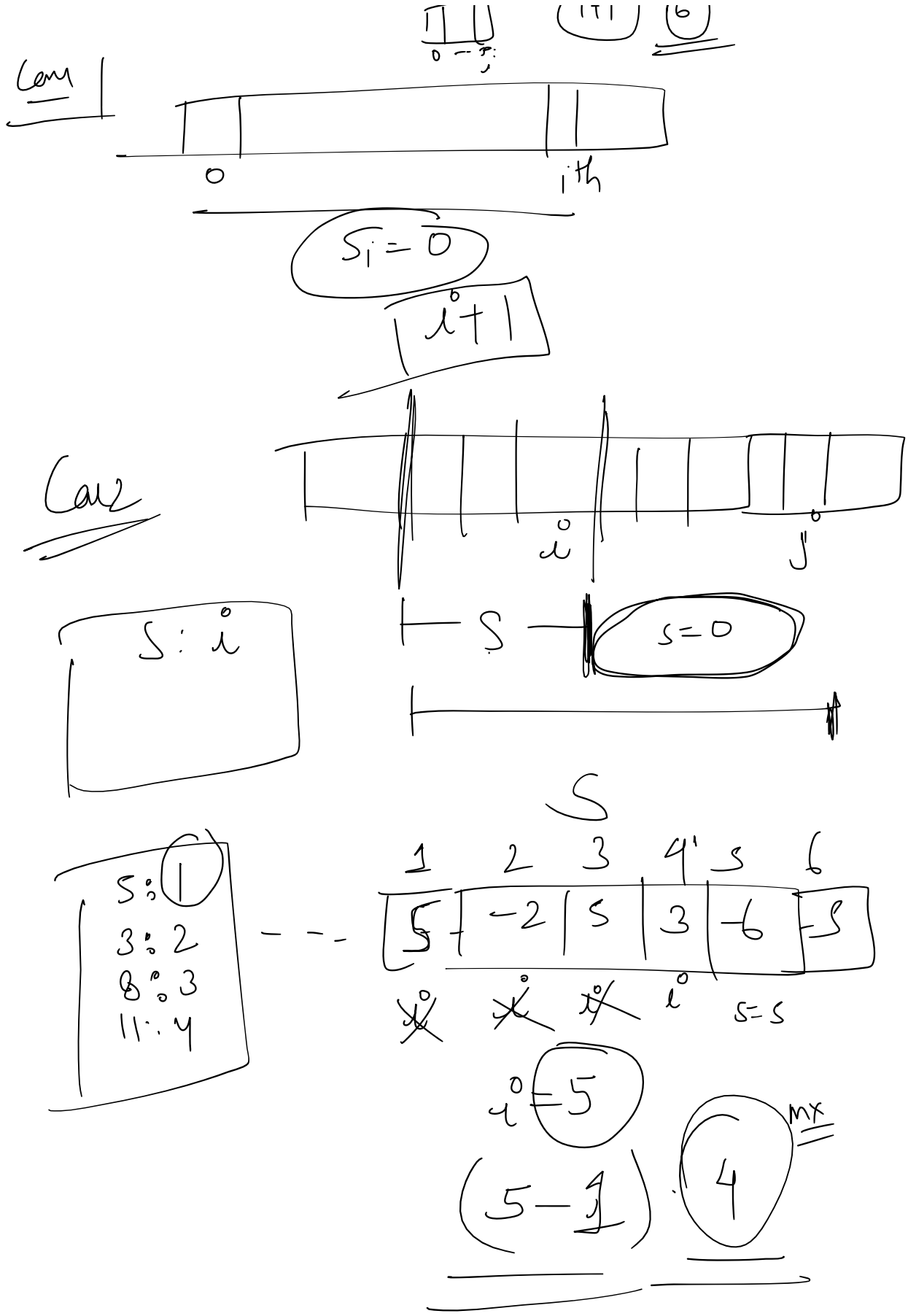
longest Array with zero Sum

Given an array having both positive and negative integers. The task is to compute the length of the largest subarray with sum 0.

Example 1:

Input:
N = 8
A[] = {15,-2,2,-8,1,7,10,23}
Output: 5
Explanation: The largest subarray with sum 0 will be -2 2 -8 1 7.

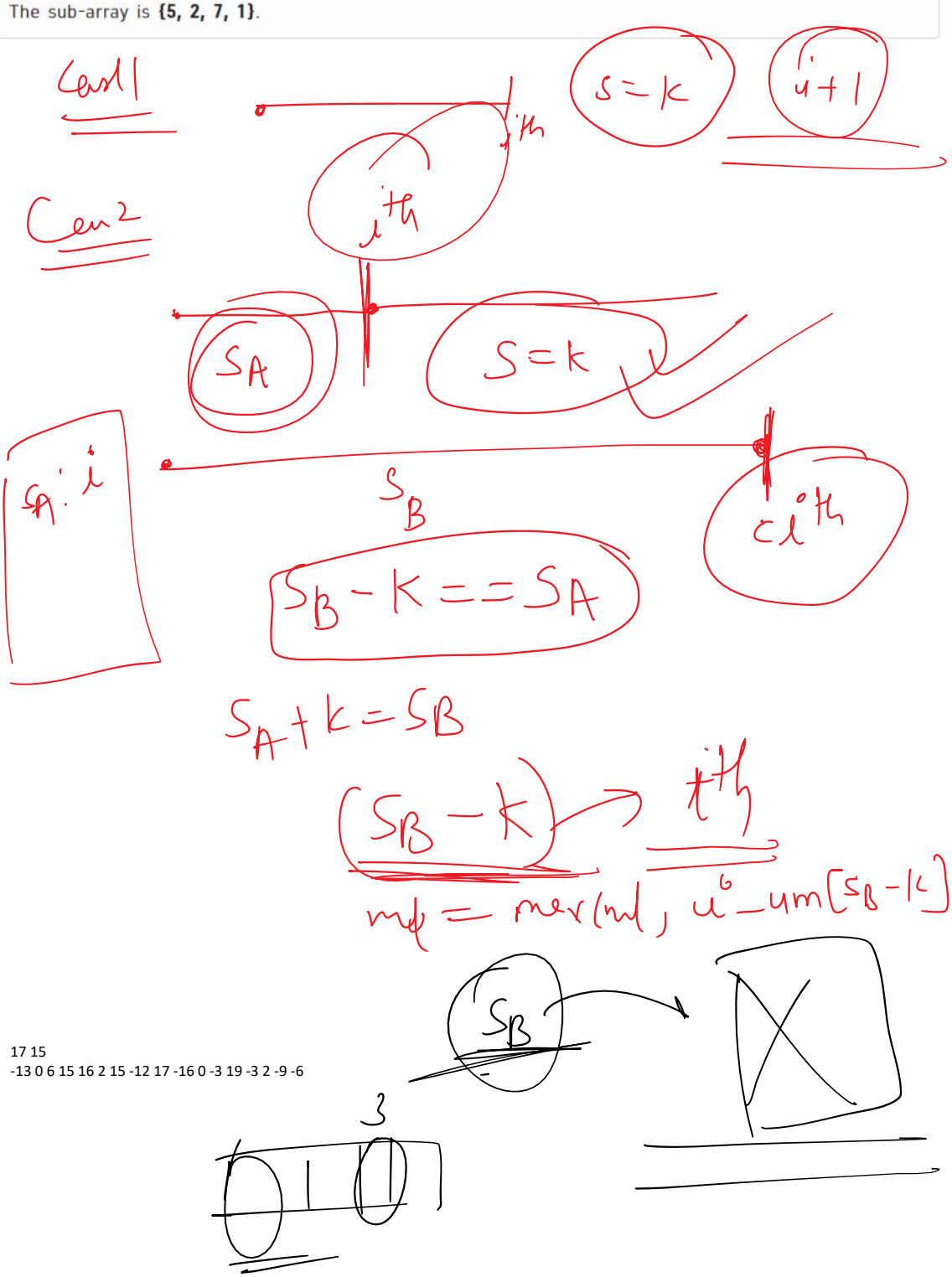



Case 1
0
ith
$S_i = 0$
i+1
Case 2
i
j
S : i
S
S=0
S
1 2 3 4 5 6
5 -2 5 3 -6 -5
i
S=S
i=5
(5-1)
4
MX

Given an array containing **N** integers and an integer **K**., Your task is to find the length of the longest Sub-Array with the sum of the elements equal to the given value **K**.

Example 1:

Input :
A[] = {10, 5, 2, 7, 1, 9}
K = 15
Output : 4
Explanation:
The sub-array is **{5, 2, 7, 1}**.



17 15
-13 0 6 15 16 2 15 -12 17 -16 0 -3 19 -3 2 -9 -6

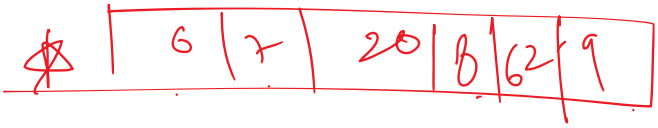
```cpp
int lenOfLongSubarr(int A[],  int n, int k)
{
    int cs = 0,ml = 0;
   unordered_map<int,int> um;
   for(int i = 0;i < n;i++){
       cs += A[i];
       // case 1
       if(cs == k){
           ml = max(ml,i+1);
       }

       // case 2;
       if(um.find(cs-k)!=um.end()){
           ml  = max(ml,i - um[cs-k]);
       }
       if(um.find(cs) == um.end()){
           um[cs] = i;
       }
   }
   return ml;
}
```

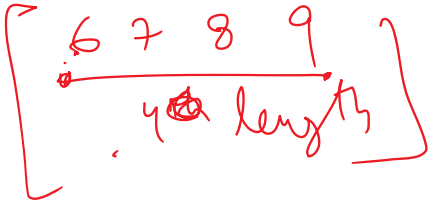
T.C - O(N)
S.C - O(N)

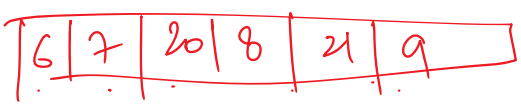
Given an array of positive integers. Find the length of the longest sub-sequence such that elements in the subsequence are consecutive integers, the **consecutive numbers can be in any order.**

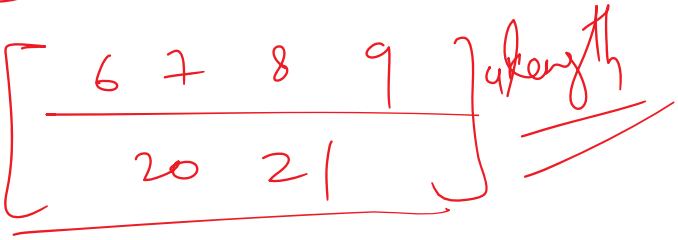
Example 1:

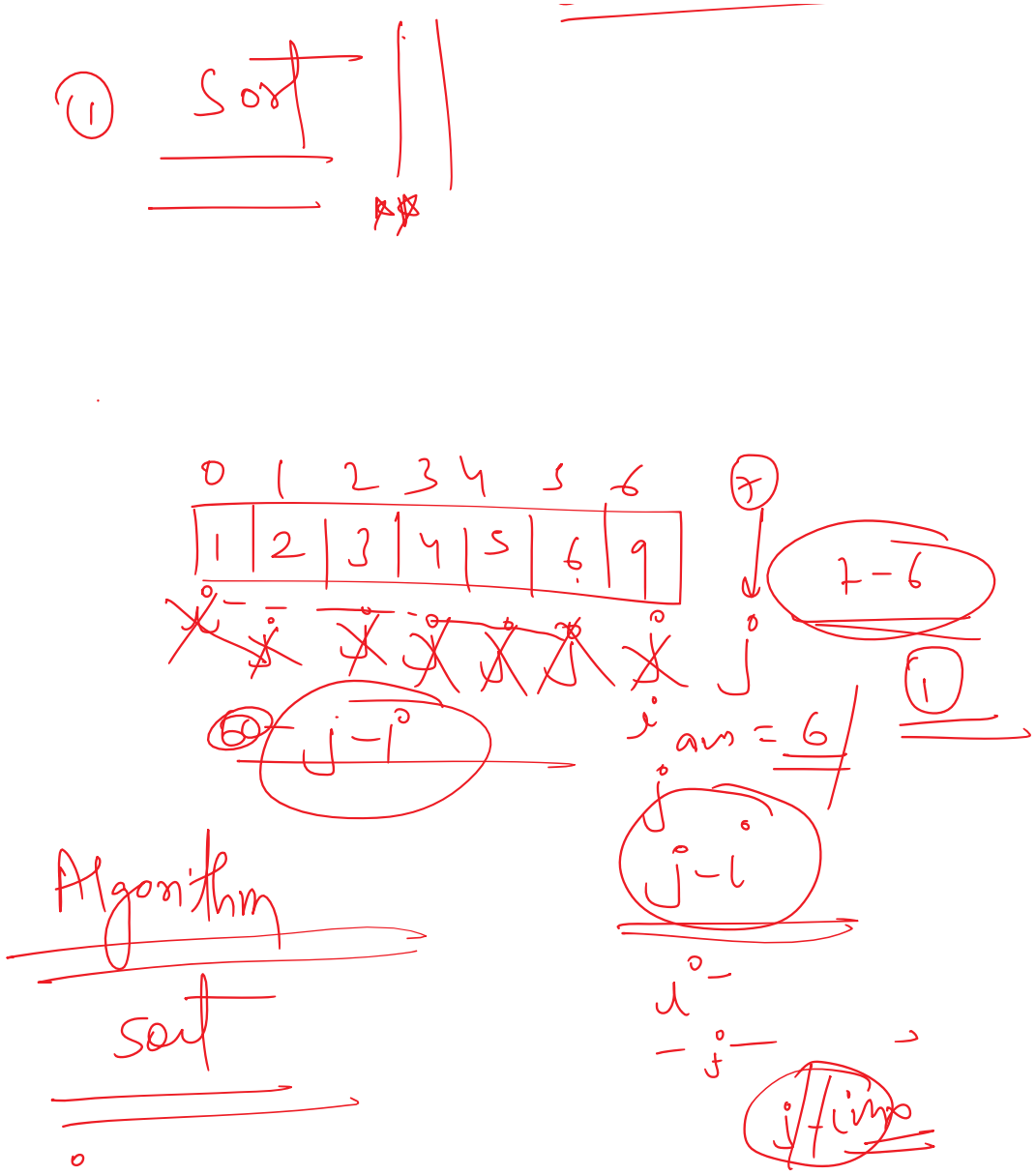
Input:
N = 7
a[] = {2,6,1,9,4,5,3}
Output:
6
Explanation:
The consecutive numbers here are 1, 2, 3, 4, 5, 6. These 6 numbers form the longest consecutive subsequence.





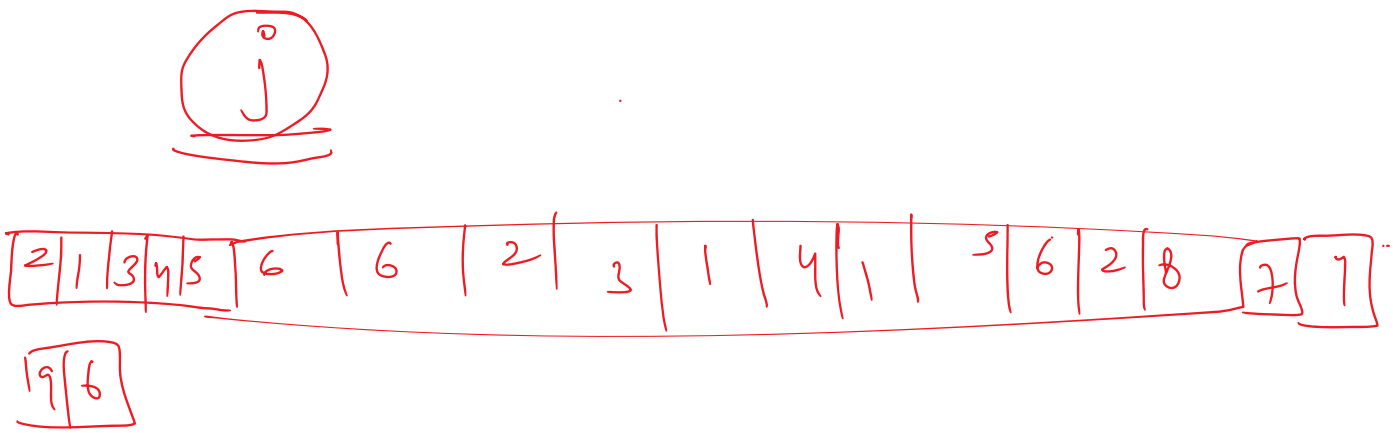




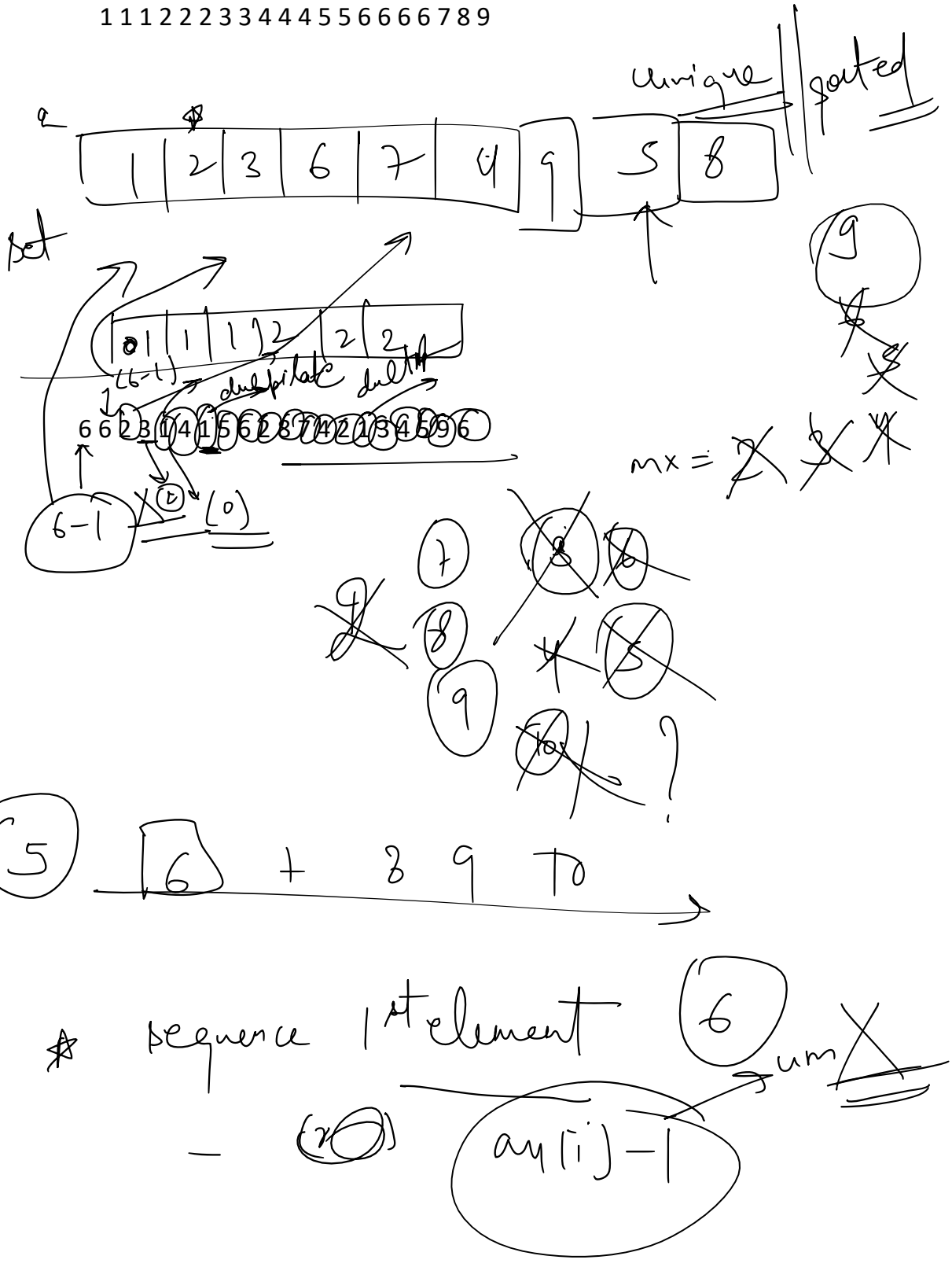


6 6 2 3 4 5 6 2 8 7 4 2 3 4 5 9 6

From <https://practice.geeksforgeeks.org/problems/longest-consecutive-subsequence2449/1>



1 1 1 2 2 2 3 3 4 4 4 5 5 6 6 6 6 7 8 9



```cpp
//Function to return length of longest subsequence of consecutive integers.
int findLongestConseqSubseq(int arr[], int n)
{
   unordered_set<int> s;
   for(int i = 0;i < n;i++){
      s.insert(arr[i]);
   }
   int mx = 0;
   for(int i = 0;i < n;i++){
      // if arr[i]- 1 is not present
      if(s.find(arr[i]-1) == s.end()){
         int c = 1;
         int x = arr[i] + 1;
         // jab tak consecutive sequence mil raha h
         while(s.find(x)!=s.end()){
            c++;
            x++;
         }
         mx = max(mx,c);
      }
   }
   return mx;
}
```